Neural Comput & Applic (2017) 28:4093–4110
DOI 10.1007/s00521-016-2301-0

ORIGINAL ARTICLE

M-estimator-based online sequential extreme learning machine for predicting chaotic time series with outliers

Wei Guo[1,4] · **Tao Xu**[1,2,3] · **Keming Tang**[4]

Received: 9 November 2015 / Accepted: 30 March 2016 / Published online: 13 April 2016

Abstract An M-estimator-based online sequential extreme learning machine (M-OSELM) is proposed to predict chaotic time series with outliers. The M-OSELM develops from the online sequential extreme learning machine (OSELM) algorithm and retains the same excellent sequential learning ability as OSELM, but replaces the conventional least-squares cost function with a robust M-estimator-based cost function to enhance the robustness of the model to outliers. By minimizing the M-estimator-based cost function, the possible outliers are prevented from entering the model's output weights updating scheme. Meanwhile, in the sequential learning process of M-OSELM, a sequential parameter estimation approach based on error sliding window is introduced to estimate the threshold value of the M-estimator function for online outlier detection. Thanks to the built-in median operation and sliding window strategy, this approach is efficient to provide a stable estimator continuously without high computational costs, and then the potential outliers can be effectively detected. Simulation results show that the proposed M-OSELM has an excellent immunity to outliers and can always achieve better performance than its counterparts for prediction of chaotic time series when the training dataset contains outliers, ensuring at the same time all benefits of an online sequential approach.

Keywords Chaotic time series prediction · Extreme learning machine · Online sequential learning · M-estimator · Outliers

✉ Tao Xu
txu@cauc.edu.cn

Wei Guo
weiguo031@163.com

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

[2] School of Computer Science and Technology, Civil Aviation University of China, Tianjin 300300, China

[3] Information Technology Research Base of Civil Aviation Administration of China, Tianjin 300300, China

[4] School of Information Engineering, Yancheng Teachers University, Yancheng 224002, China

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. Particularly in recent years, with the development of computational intelligence technologies, many intelligent prediction models have been proposed and successfully applied in the real-world time series prediction problems, which include rainfall prediction [1, 2], wind power prediction [3], stock prediction [4, 5], traffic flow prediction [6], streamflow prediction [7] and many others.

In real-world applications, the time series data usually arrives successively over time, such as the stream data in the applications of intrusion detection, industrial process monitoring, medical condition monitoring, fault diagnosis, and so on [8]. Traditional batch learning algorithm assumes that all the training data are available completely and learns them all at once, which is apparently not applicable for modeling sequential time series data. In contrary to batch learning algorithm, online sequential learning algorithm can update constantly with respect to the sequentially arriving data and does not require retraining whenever a

new data are received, which supplies an effective way to handle sequential time series data. Moreover, in practice the observed time series often contain outliers. Outliers are isolated observations which appear to deviate significantly from the majority of observations. They may arise due to human errors, instrument degradations, mechanical faults, process disturbances or natural deviations in populations [8, 9]. Outliers in time series may negatively affect model specification, which tend to result in inaccurate prediction models and poor forecasts [10]. Considering the above two points synthetically, a robust online sequential learning algorithm that resists outliers is always preferred in dealing with real-world time series.

To achieve robust models for predicting chaotic time series with outliers, several works have been done. Fu et al. [11] develop a combination model merging the annealing robust fuzzy neural networks and support vector regression to train and predict chaotic time series with outliers, in which an annealing robust learning algorithm is effectively used to overcome outliers. Jeng et al. [12] propose hybrid support vector machines for regression and Gaussian processes for regression to deal with training set with noise and outliers for the chaotic time series systems. The presented support vector machine regression approach can filter out some large noise and outliers in the training set and reduce the affects of them. Li et al. [13] present a robust echo state network to predict the chaotic time series contaminated with outliers. The basic idea of this model is to train echo state network in a Bayesian framework, while replacing the commonly used Gaussian likelihood function with a Laplace one, which is less sensitive to outliers and then can enhance the robustness of the proposed model. In addition, the M-estimator technique is a popular method to solve the problem of robust parameter estimation, which has also been used to combat outliers for chaotic time series prediction [14, 15]. In [14], a support vector echo-state machine is proposed to deal with real-life nonlinear time series and the built-in robust M-estimator loss function such as the Huber loss function makes the proposed model less sensitive to outliers. In [15], the median scale estimator and Welsch M-estimator are employed to eliminate the influences of the noise and the proposed approach is successfully used for noisy chaotic time series prediction. According to the simulation results of these papers, all the above-mentioned prediction models show good immunity to outliers by incorporating different robust approaches, and achieve satisfactory performance for predicting contaminated time series. However, the learning processes of all the models are exclusively of batch learning type, a really practical chaotic time series prediction model with both robustness and sequential learning ability is still vacant, to the authors' knowledge.

Recently, an emerging online sequential learning algorithm for single hidden layer feedforward networks (SLFNs) named online sequential extreme learning machine (OSELM) is proposed by Liang et al. [16]. OSELM originates from the batch extreme learning machine (ELM) [17] algorithm which has been demonstrated to be extremely fast with generalization performance better than other batch learning methods. As a sequential implementation of ELM, OSELM can learn the training data one-by-one or chunk-by-chunk with fixed or varying length, and the output weights of which are constantly updated by adopting a recursive least-squares (LS) algorithm. Apart from selecting the number of hidden nodes, no other control parameters have to be manually chosen. Compared with other popular sequential learning algorithms, OSELM can provide better generalization performance at a much faster learning speed. Depending upon these advantages, OSELM has been widely applied in the field of prediction problems, such as online ship roll motion prediction [18], consumer sentiments prediction [19], time series prediction [20–22], etc. Despite an excellent sequential learning algorithm, OSELM also suffers from a well-known drawback of lacking robustness to outliers like other LS-based algorithms due to the intrinsic vulnerability of the LS to outliers [23], and the performance of which will deteriorate significantly when the observations are corrupted by outliers. Regularization method is an effective way to reduce the adverse effects caused by the perturbation or the multicollinearity, then the regularized ELM (R-ELM) that merges ELM and regularization technique such as ridge regression is proposed to deal with contaminated data [24–26]. Compared with ELM, R-ELM can provide better stability and generalization performance especially when the training set contains noise and outliers. Similarly, in order to improve the generalization ability of the OSELM for noisy data, a regularized OSELM (R-OSELM) based on the bi-objective optimization method is proposed in [27]. The R-OSELM tries to seek the minimization of both the empirical residual and the norm of output weights with Tikhonov regularization, and it tends to yield good generalization models for noisy data. Though both the R-ELM and the R-OSELM have certain immunity to noise and outliers by improving the stability of ELM solution with regularization approach, they are not targeted solutions to solve the outliers problems, and their outliers resistance abilities are far from satisfactory.

In this paper, a novel M-estimator-based online sequential extreme learning machine (M-OSELM) algorithm is proposed to deal with training dataset with outliers for the chaotic time series systems. Our contributions are the following. (1) The proposed M-OSELM inherits the basic idea of OSELM learning in a sequential pattern, but replaces the conventional LS cost function with a robust

M-estimator-based cost function to eliminate the adverse effects of the possible outliers on the output weights updating process. (2) In the sequential learning procedure, an error sliding window-based parameter estimation approach is introduced to estimate the threshold value of the M-estimator function sequentially, and then the potential outliers can be effectively detected online. (3) The outliers resistance capability and ease of use of the M-OSELM are illustrated with both synthetic data and benchmark chaotic time series. (4) Compared with existing robust models which are of batch learning type, the prominent innovation of the proposed M-OSELM is that it is implemented in an online sequential pattern, which is more applicable in practical time series prediction applications.

The rest of the paper is organized as follows. The related works including ELM, R-ELM, OSELM and R-OSELM are revisited in Sect. 2. Section 3 presents the details of the proposed M-OSELM. In Sect. 4, the performance of the proposed M-OSELM is demonstrated by five illustrative examples. Finally, the conclusions and future work are given in Sect. 5.

2 Related works

In this section, we provide a brief review of the ELM, R-ELM, OSELM and R-OSELM.

2.1 ELM

ELM [17] 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

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

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 with respect to input $\boldsymbol{x}_j$.

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

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

The above N equations can be written compactly as:

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{T} \tag{3}$$

where

$$\mathbf{H} = \begin{bmatrix} \boldsymbol{h}_1 \\ \vdots \\ \boldsymbol{h}_N \end{bmatrix} = \begin{bmatrix} G(\boldsymbol{a}_1, b_1, \boldsymbol{x}_1) & \cdots & G(\boldsymbol{a}_n, b_n, \boldsymbol{x}_1) \\ \vdots & \cdots & \vdots \\ G(\boldsymbol{a}_1, b_1, \boldsymbol{x}_N) & \cdots & G(\boldsymbol{a}_n, b_n, \boldsymbol{x}_N) \end{bmatrix}_{N \times n}, \tag{4}$$

$$\boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\beta}_1^T \\ \vdots \\ \boldsymbol{\beta}_n^T \end{bmatrix}_{n \times m} \quad \text{and} \quad \mathbf{T} = \begin{bmatrix} \boldsymbol{t}_1^T \\ \vdots \\ \boldsymbol{t}_N^T \end{bmatrix}_{N \times m}. \tag{5}$$

$\mathbf{H}$ 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 ith column of $\mathbf{H}$ is the ith hidden node's output vector with respect to inputs $\boldsymbol{x}_1$, $\boldsymbol{x}_2$,…, $\boldsymbol{x}_N$.

Huang et al. [28] have proved mathematically that SLFNs with random hidden nodes have the universal approximation capability. As concluded in [28] by Theorem II.1, given any bounded nonconstant piecewise continuous activation function g: $R \to R$ for additive nodes or any integrable piecewise continuous activation function g: $R \to R$ (and $\int_R g(x)\, dx \neq 0$) for RBF nodes, the network sequence $\{f_n\}$ with randomly generated hidden nodes can converge to any continuous target function by only properly adjusting the output weights. Moreover, this Theorem can be further extended from additive or RBF hidden nodes cases to 'generalized' SLFNs [29–31]. With the universal approximation theorem, the hidden nodes of SLFNs can be randomly generated independent of the training data and remain fixed, then the hidden layer output matrix $\mathbf{H}$ for a given training data is a constant matrix. Thus, training an SLFN is simply equivalent to finding a LS solution $\hat{\boldsymbol{\beta}}$ of the linear system $\mathbf{H}\boldsymbol{\beta} = \mathbf{T}$:

$$\left\|\mathbf{H}\hat{\boldsymbol{\beta}} - \mathbf{T}\right\| = \min_{\beta} \|\mathbf{H}\boldsymbol{\beta} - \mathbf{T}\| \tag{6}$$

where $\|\cdot\|$ is a norm in Euclidean space. 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 $\mathbf{H\boldsymbol{\beta}} = \mathbf{T}$, the smallest norm least squares solution of the above linear system is

$$\hat{\boldsymbol{\beta}} = \mathbf{H}^{\dagger}\mathbf{T} \tag{7}$$

where $\mathbf{H}^{\dagger}$ is the Moore–Penrose generalized inverse of matrix $\mathbf{H}$. If $\mathbf{H}^T\mathbf{H}$ is nonsingular, then Eq. (7) can be written as

$$\hat{\boldsymbol{\beta}} = \mathbf{H}^{\dagger}\mathbf{T} = \left(\mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{T}. \tag{8}$$

2.2 R-ELM

ELM has attracted many attentions for its extremely fast training speed and good generalization performance. But it is still based on empirical risk minimization principle [see Eq. (6)] and tends to generate over-fitting models. Consequently, the trained ELM would behave very differently if test data change but slightly away from the training data, and it will become more serious when the training set contains corrupted data such as outliers.

According to the statistical learning theory, a model with good generalization ability should consider not only the empirical risk but also the structural risk and pursue a best tradeoff between the two risks. Based on this idea, a regularized ELM [24, 25] is proposed to seek $\boldsymbol{\beta}$ that minimizes the following cost function:

$$J_R(\boldsymbol{\beta}) = \|\mathbf{H}\boldsymbol{\beta} - \mathbf{T}\|^2 + \lambda\|\boldsymbol{\beta}\|^2, \tag{9}$$

where $\|\mathbf{H}\boldsymbol{\beta} - \mathbf{T}\|^2$ is the sum of squared training errors which can be regarded as empirical risk, $\|\boldsymbol{\beta}\|^2$ is the square of norm of the network output weights vector which represents structural risk, and λ is a positive real value called the regularization parameter to balance the two risks.

The cost function is minimized by differentiating (9) with respect to $\boldsymbol{\beta}$ and setting the results to zero, this yields the following regularization normal equation:

$$\left(\mathbf{H}^T\mathbf{H} + \lambda\mathbf{I}\right)\boldsymbol{\beta} = \mathbf{H}^T\mathbf{T}, \tag{10}$$

where $\mathbf{I}$ is an identity matrix with the same dimensions as $\mathbf{H}^T\mathbf{H}$. The estimator of $\boldsymbol{\beta}$ from Eq. (10) is given by

$$\hat{\boldsymbol{\beta}} = \left(\mathbf{H}^T\mathbf{H} + \lambda\mathbf{I}\right)^{-1}\mathbf{H}^T\mathbf{T}. \tag{11}$$

Compared with ELM, the R-ELM replaces the LS solution [Eq. (8)] with the generalized ridge regression estimator [Eq. (11)], which can provide better stability and generalization ability for noisy data. Moreover, the added regularization item also makes the correlation matrix $\mathbf{H}^T\mathbf{H}$ nonsingular and then the matrix inversion method can be applied directly. A more complete analysis of the R-ELM can be found in [26], where the authors extend such study to generalized SLFNs with different feature mappings as well as kernels.

2.3 OSELM and R-OSELM

As a sequential version of the batch ELM algorithm, the OSELM adopts a recursive way to solve the LS solution, and which may also encounter the ill-posed problems due to the unavoidable presence of noise or outliers. Similar to R-ELM, an improvement of OSELM called regularized OSELM [27] is proposed to improve the stability of OSELM while maintaining the same sequential learning ability as OSELM.

The R-OSELM algorithm uses the same cost function [Eq. (9)] as the R-ELM and aims to seek the optimal regularization solution in a sequential learning fashion. The learning process of R-OSELM consists of an initialization phase and a following sequential learning phase as the same as OSELM, just adding a regularization item to stabilize the initial output weights. The one-by-one R-OSELM is summarized below.

In initialization phase, given an initial training set $\Omega_{k-1} = \{(\boldsymbol{x}_j, t_j) | j = 1, \ldots, k-1\}$, according to Eq. (11), the initial output weights are given by

$$\boldsymbol{\beta}_{k-1} = \mathbf{P}_{k-1}\mathbf{H}_{k-1}^T\mathbf{T}_{k-1} \tag{12}$$

where $\mathbf{P}_{k-1} = \left(\mathbf{H}_{k-1}^T\mathbf{H}_{k-1} + \lambda\mathbf{I}\right)^{-1}$, $\mathbf{H}_{k-1} = \left[\boldsymbol{h}_1^T \ \boldsymbol{h}_2^T \ \cdots \ \boldsymbol{h}_{k-1}^T\right]^T$ and $\mathbf{T}_{k-1} = \left[t_1 \ t_2 \ \cdots \ t_{k-1}\right]^T$.

In the sequential learning phase, the recursive least-squares algorithm is used to constantly update the output weights. Suppose now that we receive another sample $(\boldsymbol{x}_k, t_k)$, the corresponding partial hidden layer output matrix is calculated as $\boldsymbol{h}_k = [G(\boldsymbol{a}_1, b_1, \boldsymbol{x}_k) \ \cdots \ G(\boldsymbol{a}_n, b_n, \boldsymbol{x}_k)]$, then the output weights update equations are determined by

$$\begin{aligned} \mathbf{P}_k &= \mathbf{P}_{k-1} - \frac{\mathbf{P}_{k-1}\boldsymbol{h}_k^T\boldsymbol{h}_k\mathbf{P}_{k-1}}{1 + \boldsymbol{h}_k\mathbf{P}_{k-1}\boldsymbol{h}_k^T}, \\ \boldsymbol{\beta}_k &= \boldsymbol{\beta}_{k-1} + \mathbf{P}_k\boldsymbol{h}_k^T(t_k - \boldsymbol{h}_k\boldsymbol{\beta}_{k-1}). \end{aligned} \tag{13}$$

As seen from Eq. (13), the output weights are updated recursively only based on the newly arrived data, which is discarded immediately as soon as it has been learnt. The above one-by-one R-OSELM algorithm can be easily extended to chunk-by-chunk type. In addition, if the regularization parameter λ in the initial solution [Eq. (12)] equals zero, then R-OSELM becomes the original OSELM.

3 Proposed M-OSELM

In this section, we first present a novel M-estimator-based learning model, next a recursive solution that solves the M-estimator model is derived and concomitantly a sequential parameter estimation approach is introduced to estimate the threshold parameter of the M-estimator function for online outlier detection, finally a robust online sequential learning algorithm named M-OSELM is proposed.

3.1 M-estimator-based learning model

As described in Sect. 2, the learning rules of the ELM and OSELM are based on the LS criterion, which minimizes

the quadratic function of the residual errors. However, the LS criterion is very sensitive to outliers and tends to generate over-fitting models in the presence of outliers. As improvers, the R-ELM and R-OSELM try to seek stable solutions by minimizing both the LS item and the regularization item [Eq. (9)]. Though the obtained regularized solutions are more stable and less sensitive to contaminated data than the original LS solution, they are still far from efficient to eliminate the effects of the outliers. In fact, for a training set with K samples, the cost function in Eq. (9) can be rewritten as

$$J_R(\boldsymbol{\beta}) = \|\mathbf{H}\boldsymbol{\beta} - \mathbf{T}\|^2 + \lambda\|\boldsymbol{\beta}\|^2 = \sum_{i=1}^{k} (e_i)^2 + \lambda\|\boldsymbol{\beta}\|^2, \quad (14)$$

where $e_i = t_i - \boldsymbol{h}_i\boldsymbol{\beta}$ represents the residual error between the desired output and the actual network output. Intuitively, a contaminated training sample such as an outlier will result in a very large estimation error e_i and the squaring operator adopted in LS may magnify it further, then the cost function is biased, and consequently the finally obtained estimator of the output weights will be inaccurate. Based on the above analysis, a possible solution for overcoming outliers would employ a robust criterion instead of the LS method. The M-estimator technique is one of the most popular methods to deal with the noise problems. It uses some gentle cost functions which increase slower than that of LS estimators when the residual error departs from zero, or even keep constant to suppress the response when the residual error goes beyond a threshold. Therefore, the M-estimator-based error function is more robust to outliers than the LS-based error function [15]. Though the M-estimator technique has been used to combat outliers for chaotic time series prediction [14, 15], the learning processes of the prediction models are exclusively of batch learning type, as stated in the introduction.

To provide robust filtering for outliers in a sequential learning fashion, a novel M-estimator-based cost function, instead of Eq. (14), is proposed to train the R-OSELM:

$$J_\rho(\boldsymbol{\beta}_k) \triangleq \sum_{i=1}^{k} \rho(e_i) + \frac{1}{2}\lambda\|\boldsymbol{\beta}_k\|^2 = \sum_{i=1}^{k} \rho(t_i - \boldsymbol{h}_i\boldsymbol{\beta}_k) + \frac{1}{2}\lambda\|\boldsymbol{\beta}_k\|^2 \quad (15)$$

where $\rho(\cdot)$ is an M-estimator function. The purpose of using an M-estimator function $\rho(\cdot)$, instead of the squaring function in Eq. (14), is to limit or even eliminate the adverse effects of the outliers on the cost function. Several commonly used M-estimators such as Huber, Tukey, Cauchy, Welsch, and Geman-McClure can be applied as $\rho(\cdot)$, and they all have similar behaviors [32]. In our study, the following modified Huber function is chosen owing to its good performance and simplicity [33].

$$\rho(e) = \begin{cases} \dfrac{e^2}{2}, & |e| \le \alpha \\ \dfrac{\alpha^2}{2}, & |e| > \alpha \end{cases} \quad (16)$$

where α is the threshold parameter, $|\cdot|$ denotes the absolute value operator. It can be seen that the function $\rho(\cdot)$ is quadratic and it is equivalent with the universal LS function when the absolute value of residual error e is less than or equal to α. On the other hand, for errors whose absolute value greater than α, $\rho(\cdot)$ is equal to a constant, which can be helpful to suppress the large perturbation stemming from the outliers.

3.2 Recursive solution of the M-estimator-based model

In this section, we devote to deriving a recursive solution of the above M-estimator-based learning model under a sequential situation. The optimal output weights for minimizing Eq. (15) can be obtained by setting the first-order partial derivative of $J_\rho(\boldsymbol{\beta}_k)$ with respect to $\boldsymbol{\beta}_k$ to zero. This yields

$$\mathbf{R}_k\boldsymbol{\beta}_k = \mathbf{S}_k \quad (17)$$

where

$$\mathbf{R}_k = \lambda\mathbf{I} + \sum_{i=1}^{k} \varphi(e_i)\boldsymbol{h}_i^T\boldsymbol{h}_i = \mathbf{R}_{k-1} + \varphi(e_k)\boldsymbol{h}_k^T\boldsymbol{h}_k, \quad (18)$$

$$\mathbf{S}_k = \sum_{i=1}^{k} \varphi(e_i)\boldsymbol{h}_i^T t_i = \mathbf{S}_{k-1} + \varphi(e_k)\boldsymbol{h}_k^T t_k, \quad (19)$$

in which $\varphi(e) \triangleq \psi(e)/e$ and $\psi(e) \triangleq \partial\ \rho(e)/\partial\ e$.

Apply the Woodbury formula [34] to Eq. (18) and the Woodbury matrix identity follows

$$\begin{aligned}\mathbf{R}_k^{-1} &= \left(\mathbf{R}_{k-1} + \varphi(e_k)\boldsymbol{h}_k^T\boldsymbol{h}_k\right)^{-1} \\ &= \mathbf{R}_{k-1}^{-1} - \frac{\varphi(e_k)\mathbf{R}_{k-1}^{-1}\boldsymbol{h}_k^T\boldsymbol{h}_k\mathbf{R}_{k-1}^{-1}}{1 + \varphi(e_k)\boldsymbol{h}_k\boldsymbol{R}_{k-1}^{-1}\boldsymbol{h}_k^T}.\end{aligned} \quad (20)$$

For convenience of computation, let $\mathbf{P}_k = \mathbf{R}_k^{-1}$, then

$$\begin{aligned}\mathbf{P}_k &= \mathbf{P}_{k-1} - \frac{\varphi(e_k)\mathbf{P}_{k-1}\boldsymbol{h}_k^T\boldsymbol{h}_k\mathbf{P}_{k-1}}{1 + \varphi(e_k)\boldsymbol{h}_k\mathbf{P}_{k-1}\boldsymbol{h}_k^T} \\ &= \mathbf{P}_{k-1} - \boldsymbol{g}_k\boldsymbol{h}_k\mathbf{P}_{k-1}\end{aligned} \quad (21)$$

where $\boldsymbol{g}(k)$ is the gain vector defined as

$$\boldsymbol{g}_k = \frac{\varphi(e_k)\mathbf{P}_{k-1}\boldsymbol{h}_k^T}{1 + \varphi(e_k)\boldsymbol{h}_k\mathbf{P}_{k-1}\boldsymbol{h}_k^T}. \quad (22)$$

Transforming Eq. (22) into

$$\boldsymbol{g}_k + \varphi(e_k)\boldsymbol{g}_k\boldsymbol{h}_k\mathbf{P}_{k-1}\boldsymbol{h}_k^T = \varphi(e_k)\mathbf{P}_{k-1}\boldsymbol{h}_k^T \quad (23)$$

and subtracting the second term on the left side yields

$$\begin{aligned} \boldsymbol{g}_k &= \varphi(e_k)(\mathbf{P}_{k-1} - \boldsymbol{g}_k \boldsymbol{h}_k \mathbf{P}_{k-1})\boldsymbol{h}_k^T \\ &= \varphi(e_k)\mathbf{P}_k \boldsymbol{h}_k^T . \end{aligned} \tag{24}$$

Substituting Eqs. (19) and (21) into Eq. (17), the update formula for $\boldsymbol{\beta}_k$ is derived as follows:

$$\begin{aligned} \boldsymbol{\beta}_k &= \mathbf{R}_k^{-1}\mathbf{S}_k = \mathbf{P}_k\mathbf{S}_k = \mathbf{P}_k(\mathbf{S}_{k-1} + \varphi(e_k)\mathbf{h}_k^T t_k) \\ &= \mathbf{P}_k\mathbf{S}_{k-1} + \varphi(e_k)\mathbf{P}_k\mathbf{h}_k^T t_k \\ &= (\mathbf{P}_{k-1} - \mathbf{g}_k\mathbf{h}_k\mathbf{P}_{k-1})\mathbf{S}_{k-1} + \mathbf{g}_k t_k \\ &= \mathbf{P}_{k-1}\mathbf{S}_{k-1} - \mathbf{g}_k\mathbf{h}_k\mathbf{P}_{k-1}\mathbf{S}_{k-1} + \mathbf{g}_k t_k \\ &= \mathbf{P}_{k-1}\mathbf{S}_{k-1} + \mathbf{g}_k(t_k - \mathbf{h}_k\mathbf{P}_{k-1}\mathbf{S}_{k-1}) \\ &= \boldsymbol{\beta}_{k-1} + \mathbf{g}_k(t_k - \mathbf{h}_k\boldsymbol{\beta}_{k-1}) \\ &= \boldsymbol{\beta}_{k-1} + \varphi(e_k)\mathbf{P}_k\mathbf{h}_k^T\varepsilon_k . \end{aligned} \tag{25}$$

where $\varepsilon_k = t_k - \boldsymbol{h}_k\boldsymbol{\beta}_{k-1}$ is called the priori error, and $e_k = t_k - \boldsymbol{h}_k\boldsymbol{\beta}_k$ is called the posteriori error. From Eq. (25), the computing of $\boldsymbol{\beta}_k$ relies on the posteriori error e_k, while which is unobtainable without knowing $\boldsymbol{\beta}_k$. Hence, there is a mutual dependence between $\boldsymbol{\beta}_k$ and e_k. Indeed, the priori error ε_k can be viewed as a tentative value of e_k before the updating of $\boldsymbol{\beta}_k$, and it can be used as a good approximation to replace the posteriori error e_k, then by incorporating Eqs. (21) and (25) the equations for updating $\boldsymbol{\beta}_k$ can be finally written as

$$\begin{aligned} \mathbf{P}_k &= \mathbf{P}_{k-1} - \frac{\varphi(\varepsilon_k)\mathbf{P}_{k-1}\boldsymbol{h}_k^T\boldsymbol{h}_k\mathbf{P}_{k-1}}{1 + \varphi(\varepsilon_k)\boldsymbol{h}_k\mathbf{P}_{k-1}\boldsymbol{h}_k^T} \\ \boldsymbol{\beta}_k &= \boldsymbol{\beta}_{k-1} + \varphi(\varepsilon_k)\mathbf{P}_k\boldsymbol{h}_k^T\varepsilon_k \end{aligned} \tag{26}$$

where $\varepsilon_k = t_k - \boldsymbol{h}_k\boldsymbol{\beta}_{k-1}$.

Equation (26) gives the recursive formula for updating $\boldsymbol{\beta}_k$.

Remark 1 From Eq. (26), we can see that the updating procedure of $\boldsymbol{\beta}_k$ is implemented in a sequential pattern. At any time, only the newly arrived data are seen and learned, and which can be discarded without needing to store as soon as the learning procedure for it has been completed.

Remark 2 Since the same modified Huber function [Eq. (16)] is used as the M-estimator function as Chan and Zou [33], the similar efficacy can be attained here. According to the definition of Eq. (16), if $|\varepsilon_k| \le \alpha$, which means no outlier is detected, then $\varphi(\varepsilon_k) = 1$, Eq. (26) becomes identical to Eq. (13) in the R-OSELM algorithm, and the current data will be learned as a useful sample. However, when $|\varepsilon_k| > \alpha$, the input vector $\boldsymbol{h}_k$ and (or) target output t_k are suspected to be contaminated with outliers, then $\varphi(\varepsilon_k) = 0$, from Eq. (26), the $\mathbf{P}_k$ and $\boldsymbol{\beta}_k$ will remain the same as $\mathbf{P}_{k-1}$ and $\boldsymbol{\beta}_{k-1}$, respectively. In other words, when the estimated priori error exceeds the threshold α, the output weights will not update to prevent it from being influenced by the possibly corrupted sample in the presence of outliers. These properties make the finally obtained solution very robust against outliers or even entirely unaffected by outliers as long as the threshold α is chosen properly so as to distinguish whether the current sample is clean or corrupted by outliers.

Remark 3 Though the prior error ε_k is adopted as an estimator of the posteriori error e_k to calculate the output weights, it should be noted that the optimization procedure for minimizing the cost function is actually based on e_k rather than ε_k.

3.3 Sequential parameter estimation for online outlier detection

As mentioned above, in the sequential learning process, the estimation error is compared with the threshold parameter to decide whether an observation should be processed or just discarded as an outlier. That is to say, an online outlier detection procedure is needed. To this end, a correct choice for the threshold parameter α is of great importance, and which can significantly affect the performance of the M-estimator-based learning model. It can be seen that a large value of α does not behave well to against the adverse effects of the outliers. Oppositely, a small value of α is more likely to have good suppression of the outliers, but value of α too small would also result in the wrong filtering to those clean samples. To accurately detect the potential outliers online, a sequential parameter estimation approach is proposed to estimate the threshold parameter α continuously.

Similar to references [33, 35], the estimation error without outliers is assumed, for simplicity, to be normally distributed with mean zero and variance $\hat{\sigma}_k^2$, which is denoted as $e_k \sim N(0, \hat{\sigma}_k^2)$. However, instead of using the obscure complementary error function adopted in [33, 35], we deduce the relationship between α and $\hat{\sigma}_k$ in a direct way. According to the above assumption, it follows that $e_k' = e_k/\hat{\sigma}_k$ subjects to the standard normal distribution, $\varepsilon_k' \sim N(0, 1)$. Then, the probability of $|e_k|$ greater than a given threshold α is $\theta_\alpha = P_r\{|e_k| > \alpha\} = P_r\{|e_k'| > \alpha/\hat{\sigma}_k\}$. According to the properties of the standard normal distribution, we have $P_r\{e_k' < \alpha/\hat{\sigma}_k\} = \int_{-\infty}^{\alpha/\hat{\sigma}_k} e^{-x^2/2}dx/\sqrt{2\pi} = 1 - \theta_\alpha/2$, and it follows that $\alpha/\hat{\sigma}_k = \Phi^{-1}(1 - \theta_\alpha/2)$, where $\Phi^{-1}(\cdot)$ is the fractile of standard normal distribution corresponding to a certain probability. Different values of θ_α will yield different confidence in detecting the outliers. If θ_α is chosen to be 0.01, we can detect and reject the outliers with 99 % confidence, and the corresponding threshold parameter α is determined as

$$\alpha = \Phi^{-1}(0.995)\hat{\sigma}_k \approx 2.576\hat{\sigma}_k \tag{27}$$

where $\hat{\sigma}_k$ is the estimated standard deviation of the estimation error without outliers. A commonly used robust estimation of the standard deviation is given as

$$\hat{\sigma}_k = 1.483\ \mathrm{MAD}(e_k) \tag{28}$$

where MAD(·) is the median of all absolute deviations from the median [36]. Despite a robust estimator, however, its computational complexity is rather high for the considerable amount of median operations required. Besides, references [33, 35] propose a recursive estimator for $\hat{\sigma}_k^2$, but which is inapplicable for our computing environment. In this paper, another robust but much cheaper estimator based on robust regression is adopted to estimate $\hat{\sigma}_k$ continuously:

$$\hat{\sigma}_k = 1.483(1 + 5/(L_w - 1))\sqrt{\mathrm{med}(W_e(k))} \tag{29}$$

where med(·) denotes the sample median operation, $W_e(k) = \left\{e_k^2,\ e_{k-1}^2,\ \cdots,\ e_{k-L_w+1}^2\right\}$ is a sliding window of the posteriori error at time k, L_w is the length of error window and $1.483(1 + 5/(L_w - 1))$ is a finite sample correction factor. More details about robust regression can be found in [36].

Remark 4 The median is one of the most commonly used robust statistics which are largely unaffected by the presence of outliers. Owing to the median operation, Eq. (29) gives rise to a stable estimation of $\hat{\sigma}_k$ under outliers environment.

Remark 5 Instead of using all the residual errors, we use an error sliding window with fixed length to estimate the $\hat{\sigma}_k$ and α sequentially. In another words, when a new error is obtained and added to the error window, the oldest error will be discarded to keep the error window online updating, subsequently the $\hat{\sigma}_k$ and α will also be online updated. The sliding window strategy not only makes the estimation more consistent with the current situation but also prevents the infinite increase of computational burdens, which is much appreciated for sequential learning.

3.4 Summary of the M-OSELM algorithm

Combining the above analysis, a novel robust online sequential extreme learning machine based on M-estimator is proposed, we call it M-OSELM for short. Similar to the generalized OSELM, our M-OSELM algorithm consists of an initial batch learning phase and a following sequential learning phase, which is summarized as follows:

(1) **Initialization phase** Given a initial training subset $\Omega_{k-1} = \{(\boldsymbol{x}_j, t_j)|j = 1, \ldots, k-1\}$, activation function $G(\boldsymbol{a}, b, \boldsymbol{x})$, hidden nodes number n, regularization parameter λ, and error sliding window length L_w,

(a) Randomly assign hidden nodes parameters $(\boldsymbol{a}_i, b_i)$, $i = 1, 2,\ \cdots,\ n$.

(b) Calculate the initial hidden layer output matrix $\mathbf{H}_{k-1}$

$$\mathbf{H}_{k-1} = \begin{bmatrix} G(\boldsymbol{a}_1, b_1, \boldsymbol{x}_1) & \cdots & G(\boldsymbol{a}_n, b_n, \boldsymbol{x}_1) \\ \vdots & \cdots & \vdots \\ G(\boldsymbol{a}_1, b_1, \boldsymbol{x}_{k-1}) & \cdots & G(\boldsymbol{a}_n, b_n, \boldsymbol{x}_{k-1}) \end{bmatrix}_{(k-1)\times n}. \tag{30}$$

(c) Calculate the initial output weights

$$\boldsymbol{\beta}_{k-1} = \mathbf{P}_{k-1}\mathbf{H}_{k-1}^T\mathbf{T}_{k-1} \tag{31}$$

where $\mathbf{P}_{k-1} = \left(\mathbf{H}_{k-1}^T\mathbf{H}_{k-1} + \lambda\mathbf{I}\right)^{-1}$, $\mathbf{T}_{k-1} = [t_1\ t_2\ \cdots\ t_{k-1}]^T$.

(d) Calculate the initial estimation error vector

$$\boldsymbol{e}_{k-1} = \mathbf{T}_{k-1} - \mathbf{H}_{k-1}\boldsymbol{\beta}_{k-1} \tag{32}$$

where $\boldsymbol{e}_{k-1} = [e_1\ e_2\ \cdots\ e_{k-1}]^T$.

(e) Estimate the initial threshold value α

$$\begin{aligned} &L_w = \min(L_w, k-1) \\ &W_e(k-1) = \left\{e_{k-1}^2, e_{k-2}^2, \ldots, e_{k-L_w}^2\right\} \\ &\hat{\sigma}_{k-1} = 1.483(1 + 5/(L_w - 1))\sqrt{\mathrm{med}(W_e(k-1))} \\ &\alpha = 2.576\hat{\sigma}_{k-1}. \end{aligned} \tag{33}$$

(2) **Sequential learning phase** For each arriving observation $(\boldsymbol{x}_k, t_k)$,

(a) Calculate the hidden layer output matrix $\boldsymbol{h}_k$

$$\boldsymbol{h}_k = [G(\boldsymbol{a}_1, b_1, \boldsymbol{x}_k)\ \cdots\ G(\boldsymbol{a}_n, b_n, \boldsymbol{x}_k)]. \tag{34}$$

(b) Calculate the prior error ε_k

$$\varepsilon_k = t_k - \boldsymbol{h}_k\boldsymbol{\beta}_{k-1}. \tag{35}$$

(c) Calculate the 0–1 weight factor $\varphi(\varepsilon_k)$

$$\varphi(\varepsilon_k) = \begin{cases} 1 & if\ |\varepsilon_k| \le \alpha \\ 0 & otherwise \end{cases}. \tag{36}$$

(d) Calculate the output weights $\boldsymbol{\beta}_k$

$$\begin{aligned} &\mathbf{P}_k = \mathbf{P}_{k-1} - \frac{\varphi(\varepsilon_k)\mathbf{P}_{k-1}\boldsymbol{h}_k^T\boldsymbol{h}_k\mathbf{P}_{k-1}}{1 + \varphi(\varepsilon_k)\boldsymbol{h}_k\mathbf{P}_{k-1}\boldsymbol{h}_k^T} \\ &\boldsymbol{\beta}_k = \boldsymbol{\beta}_{k-1} + \varphi(\varepsilon_k)\mathbf{P}_k\boldsymbol{h}_k^T\varepsilon_k. \end{aligned} \tag{37}$$

(e) Calculate the posteriori error e_k and update the threshold value α when $|\varepsilon_k| \le \alpha$

$$\begin{aligned} &e_k = t_k - \boldsymbol{h}_k\boldsymbol{\beta}_k \\ &W_e(k) = \left\{e_k^2, e_{k-1}^2, \ldots, e_{k-L_w+1}^2\right\} \\ &\hat{\sigma}_k = 1.483(1 + 5/(L_w - 1))\sqrt{\mathrm{med}(W_e(k))} \\ &\alpha = 2.576\hat{\sigma}_k. \end{aligned} \tag{38}$$

(f) Set $k = k + 1$. Go to Step (2).

4 Simulation experiments and performance evaluation

In this section, the performance of the proposed M-OSELM is evaluated on one artificial dataset 'SinC' and four benchmark chaotic time series systems, Mackey–Glass, Rossler, Logistic and Henon. The experimental results of the proposed algorithm are also compared with that of the original ELM, R-ELM, OSELM and R-OSELM. All the five algorithms use the same sigmoidal additive activation function $G(\boldsymbol{a}, b, \boldsymbol{x}) = 1/(1 + \exp(-(\boldsymbol{a} \cdot \boldsymbol{x} + b)))$ where the input weights $\boldsymbol{a}$ and the biases b are randomly selected from the range $[-1, 1]$. All the simulations are carried out in MATLAB R2010b environment running on an ordinary PC with 3.4 GHZ CPU and 4 GB RAM. For each problem, the reported experimental results are the average values of 30 independent experimental runs and the performance is measured by the root mean squared error (RMSE) defined as

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

where y_i, $\hat{y}_i$ are the desired value and actual prediction value, respectively. N is the number of the testing examples.

4.1 Function approximation of 'SinC'

In the first experiment, all the five algorithms (ELM, R-ELM, OSELM, R-OSELM and M-OSELM) are carried out to approximate the 'SinC' function which is popularly used to illustrate neural network in the literature

$$y(x) = \begin{cases} \sin x/x, & x \neq 0, \\ 1, & x = 0. \end{cases} \tag{40}$$

A training set (x_i, y_i) and testing set (x_i, y_i) with 5000 samples, respectively, are generated where x_i' S are uniformly randomly distributed on the interval $(-10, 10)$. Random noise distributed in $[-0.2, 0.2]$ are added to all the training samples while testing data remain noise-free. In addition, to examine the robustness of the proposed M-OSELM algorithm, some outliers are added to the training set to replace the corresponding target output y_i at $i = 200, 300, \ldots, 5000$.

In this simulation, 20 hidden nodes are assigned for the five algorithms directly according to the previous literatures [17, 24]. Similar to [24], we estimate the generalized accuracy of R-ELM and R-OSELM using different regularization parameter λ: $\lambda = [10^{-10}, 10^{-9}, \ldots, 10^{4}, 10^{5}]$. Average results of 30 trials of simulations with each λ are obtained and then the best performance is reported in this paper. Similarly, the error window length L of M-OSELM is determined in the same way just setting $L = [10, 20, \ldots, 90, 100]$. Average results of 30 trials of simulations with each L are obtained and then the best performance is reported. Besides, the number of initial training data for OSELM, R-OSELM and M-OSELM are set as 100. To evaluate and compare the robustness of the five algorithms, a contrast test is conducted to learn the 'SinC' with or without outliers for each algorithm, and the testing RMSE and learning time are obtained, respectively. As demonstrated in [37], unlike the ELM algorithm, the OSELM algorithm does not have a structural tolerance property, it is very sensitive and unstable at current experimental situation, and the average testing RMSE of 30 trials is very large and always goes beyond the normal range, so the experimental results of OSELM are omitted in this case, and the results of the other four algorithms are listed in Table 1. It can be seen that when the learners are trained by training data without outliers, the testing performances of the four algorithms are competitive with each other and the testing RMSE of ELM is slightly higher than that of the other three algorithms. However, when the learners are trained by training data contaminated with outliers, the performances of the ELM, R-ELM and R-OSELM degrade significantly, while the performance of the M-OSELM remains the same as that without outliers, that is to say, the outliers do not cause any influence when the M-OSELM algorithm is used. As far as the training time is concerned, we can see from Table 1 that the two batch learning algorithms (ELM and R-ELM) achieve a much faster learning speed than the sequential ones (R-OSELM and M-OSELM), which is consistent with the discoveries of paper [16]. Though there are additional calculations for online outlier detection in the learning process of M-OSELM, the training time of which is just slightly higher than that of the R-OSELM. In a word, compared with R-OSELM, the M-OSELM achieves a great

Table 1 Performance comparisons for learning 'SinC' with or without outliers

Algorithms	Testing RMSE		Training time (s)	
	Without outliers	With outliers	Without outliers	With outliers
ELM	0.00797	0.03195	0.01510	0.01458
R-ELM	0.00772	0.02972	0.01042	0.01094
R-OSELM	0.00763	0.02984	0.49115	0.49167
M-OSELM	0.00761	0.00766	0.55833	0.56771

improvement on robustness with little increase in learning time.

In order to intuitively illustrate the impact of outliers and how the proposed M-OSELM algorithm deals with the outliers effectively, the typical output curves of the R-OSELM and the M-OSELM for approximating 'SinC' with or without outliers are demonstrated in Figs. 1 and 2. As shown in Figs. 1 and 2, the R-OSELM is seriously affected and its actual outputs deviate far from the desired outputs when outliers are added; while for M-OSELM, the actual curves of the network output almost fit the desired curves in both case. It further confirms that the proposed M-OSELM is very robust and almost not affected by outliers.

4.2 Chaotic time series prediction

4.2.1 Dataset description

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:

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = \frac{ax(t-\tau)}{1+x^c(t-\tau)} - bx(t). \tag{41}$$

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

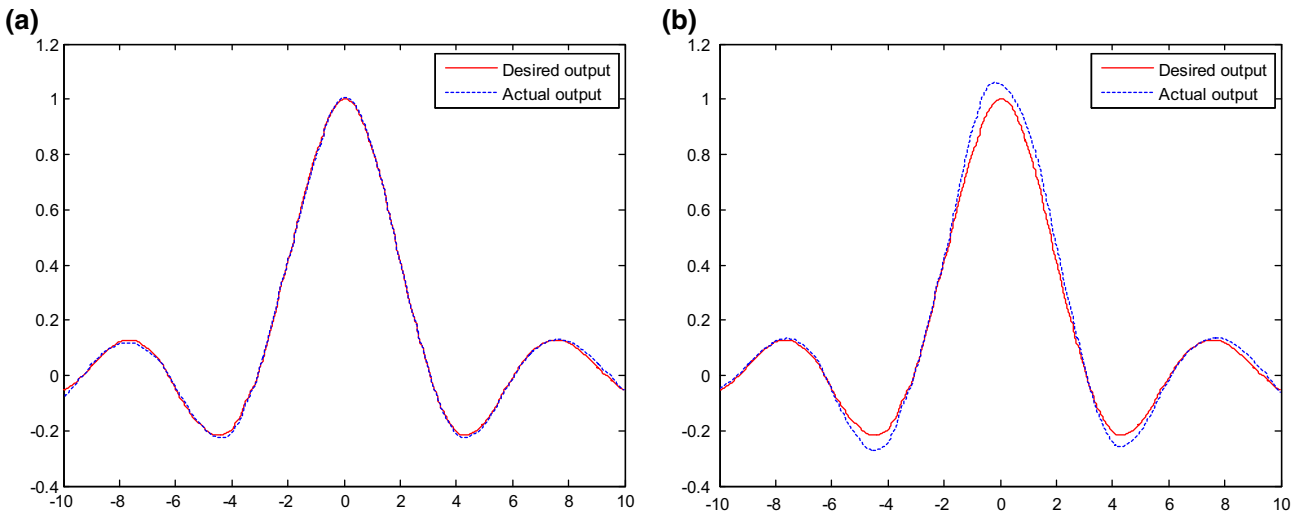

Fig. 1 Function approximation results of R-OSELM. **a** Training set without outliers. **b** Training set with outliers

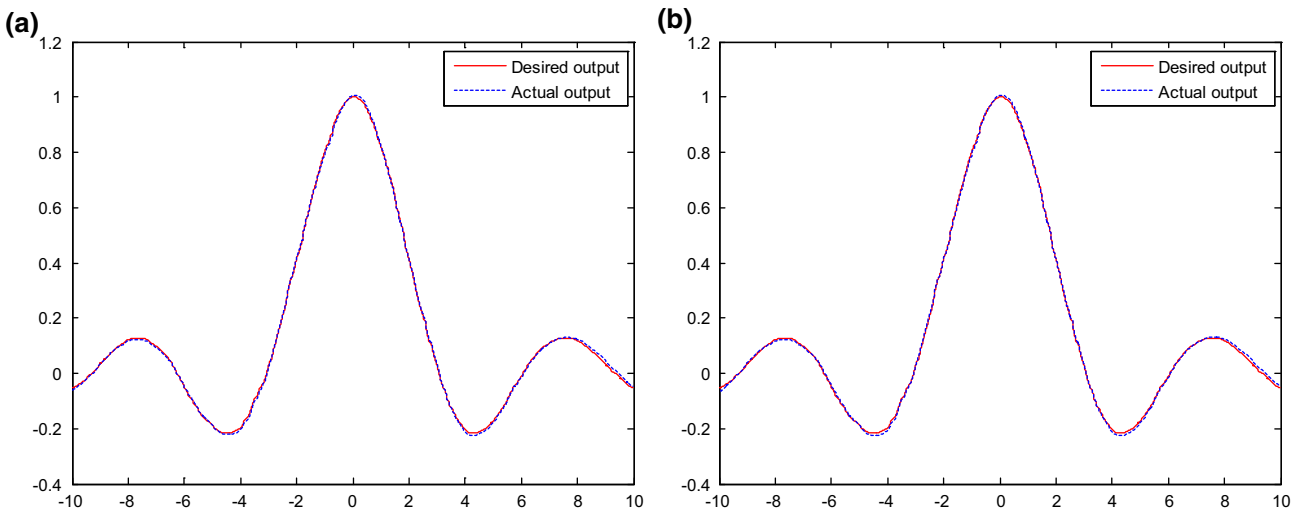

Fig. 2 Function approximation results of M-OSELM. **a** Training set without outliers. **b** Training set with outliers

[13, 38–40]. A chaotic time series samples set with length of 1900 is generated by the fourth-order Runge–Kutta integration of Eq. (41) with initial value $x(0) = 1.2$. To reduce the transient effect, we omit the initial 200 values and only keep the last 1700 values for experiment, in which the first 1000 samples are used for training, and the remaining 700 samples are used for prediction [13]. In order to evaluate the robustness of the proposed method, eight training samples are replaced by eight randomly generated outliers while the testing data remain outliers-free. The embedding dimension and time delay for phase space reconstruction of the original chaotic time series are set as 4 and 6, respectively, as Li et al. [13].

Rossler attractor is one of the most famous chaotic attractor. The sequence of the Rossler chaotic time series is derived from the following differential systems

$$\begin{aligned}\frac{dx(t)}{dt} &= -z(t) - y(t)\\ \frac{dy(t)}{dt} &= x(t) + dy(t)\\ \frac{dz(t)}{dt} &= e + z(t)(x(t) - f)\end{aligned} \tag{42}$$

where d, e, f are the control parameters and the typical values for these parameters are $d = 0.15$, $e = 0.2$ and $f = 10$. In this case, the system is chaotic. The x-coordinate of the Rossler time series is considered for experiment and a time series with a length of 2200 is generated by the fourth-order Runge–Kutta integration of Eq. (42) with step size $h = 0.01$ and initial state $\{x(0), y(0), z(0)\} = \{0.05, 0.05, 0.05\}$. To reduce the transient effect, we omit the initial 200 values and only keep the last 2000 values for experiment, in which the first 1500 samples are used for training, and the remaining 500 samples are used for prediction. For robustness testing, thirteen outliers are added into the training set while the testing data remain outliers-free. The embedding dimension and time delay for phase space reconstruction are chosen as 1 and 5, respectively. Similar to the Mackey–Glass chaotic time series, the above-selected experimental parameters for Rossler chaotic time series are mainly according to Li et al. [13].

The Logistic chaotic map and Henon chaotic map are two other classic chaotic time series, which are described with Eqs. (43) and (44), respectively:

$$x(t+1) = 4x(t)(1 - x(t)); \tag{43}$$

$$\begin{aligned}x(t+1) &= 1 - 1.4x^2(t) + y(t),\\ y(t+1) &= 0.3x(t).\end{aligned} \tag{44}$$

In our simulations, the Logistic chaotic time series with a length of 2200 is generated by Eq. (43) with initial value $x(0) = 0.1$, the Henon chaotic time series with a length of 2200 is generated by Eq. (44) with initial state $\{x(0), y(0)\} = \{0.1, 0.1\}$ and the x-coordinate is considered for experiment. For both Logistic and Henon time series, the initial 200 values are omitted to reduce the transient effect, and the last 2000 values are kept for experiment, in which the first 1500 samples are used for training, and the remaining 500 samples are used for prediction. For robustness testing, 13 outliers are added into the training set while the testing data remain outliers-free. The embedding dimension and time delay for phase space reconstruction are chosen as 4 and 1, respectively, for both the two time series. The experimental parameters for Logistic and Henon chaotic time series are mainly according to Zhang and Wang [41].

4.2.2 Experimental results

The prediction performances of ELM, R-ELM, OSELM, R-OSELM and M-OSELM are compared on the four above-mentioned chaotic time series contaminated with outliers. For each problem, the hidden nodes number n of ELM and OSELM are selected based on the validation method. We gradually increase n by an interval of 10 and the nearly optimal number of hidden node generating the best performance is shown in this paper. As proposed by Deng et al. [24], we estimate the generalized accuracy of R-ELM and R-OSELM using different combination of regularization parameter λ and the hidden nodes number n: $\lambda = [10^{-10}, 10^{-9}, \ldots, 10^{4}, 10^{5}]$ and n is gradually increased by an interval of 10. Average results of 30 trials of simulations with each combination of (n, λ) are obtained and then the best performance is reported in this paper. Similar to R-ELM and R-OSELM, the error window length L and the hidden nodes number n of M-OSELM are determined in the same way just setting $L = [10, 20, \ldots, 90, 100]$. Besides, the number of initial training data for OSELM, R-OSELM and M-OSELM are set as 200 and a small value of regularization parameter $\lambda = 10^{-8}$ is assigned to stabilize the initial solution for M-OSELM.

Table 2 presents the selected parameter values of the five algorithms for Mackey–Glass, Rossler, Logistic and Henon chaotic time series. In Table 2, n denotes the number of hidden nodes of ELM, R-ELM, OSELM, R-OSELM and M-OSELM; λ denotes the regularization parameter of R-ELM and R-OSELM; L denotes the length of error window of M-OSELM.

Table 3 shows the prediction performances of the five methods on the four chaotic time series with outliers. In Table 3, the mean and standard deviations (Dev) of prediction RMSE over 30 independent trials are given. It can be seen that in each simulation the M-OSELM can always achieve the best prediction results and the ELM (or OSELM) gives the worst ones. Though the R-ELM (R-OSELM) obtains better prediction results than the original

Table 2 Parameter values of ELM, R-ELM, OSELM, R-OSELM, M-OSELM for chaotic time series with outliers

Time series	ELM n	R-ELM (n, λ)	OSELM n	R-OSELM (n, λ)	M-OSELM (n, L)
Mackey–Glass	50	$(200, 10^{-5})$	170	$(200, 10^{-5})$	(200, 10)
Rossler	130	$(170, 10^{-5})$	180	$(140, 10^{-5})$	(20, 10)
Logistic	60	$(160, 10^{-5})$	130	$(170, 10^{-5})$	(190, 10)
Henon	60	$(180, 10^{-2})$	50	$(170, 10^{-2})$	(180, 10)

Table 3 Comparisons of prediction performances on chaotic time series with outliers

Time series	ELM		R-ELM		OSELM		R-OSELM		M-OSELM	
	RMSE	Dev	RMSE	Dev	RMSE	Dev	RMSE	Dev	RMSE	Dev
Mackey–Glass	0.01283	7.35 E−04	0.01052	2.85 E−04	0.01329	8.85 E−04	0.01048	3.46 E−04	2.46 E−03	7.56 E−05
Rossler	0.02839	5.54 E−04	0.01467	2.04 E−04	0.02173	2.91 E−04	0.01469	2.25 E−04	2.65 E−03	2.60 E−06
Logistic	0.07094	3.26 E−03	0.06791	3.95 E−04	0.10561	3.20 E−02	0.06776	5.72 E−04	7.08 E−05	2.82 E−05
Henon	0.11104	8.58 E−03	0.08957	3.09 E−03	0.11428	1.33 E−02	0.08976	3.01 E−03	7.69 E−05	2.49 E−05

Table 4 Prediction results of M-OSELM on chaotic time series with or without outliers

Time series	Outliers	RMSE	Dev	(n, L)
Mackey–Glass	With	2.46 E−03	7.56 E−05	(200, 10)
	Without	2.43 E−03	5.92 E−05	(200, 10)
Rossler	With	2.65 E−03	2.60 E−06	(20, 10)
	Without	2.65 E−03	2.04 E−06	(20, 10)
Logistic	With	7.08 E−05	2.82 E−05	(190, 10)
	Without	6.72 E−05	2.53 E−05	(200, 10)
Henon	With	7.69 E−05	2.49 E−05	(180, 10)
	Without	7.95 E−05	2.16 E−05	(180, 10)

ELM (OSELM) by improving the stability of solutions, their performances are still far from well compared with that of the M-OSELM algorithm.

To verify the robustness of the proposed M-OSELM to outliers, another group of contrast experiment is conducted to investigate the prediction performances of the M-OSELM for the four chaotic time series without outliers, and the prediction results are compared with that obtained under outliers environment. The comparison results are listed in Table 4. As shown in Table 4, the prediction results of the M-OSELM in the presence of outliers are almost identical with that without outliers, which confirms that the proposed M-OSELM has an excellent resistance ability against outliers.

In order to intuitively illustrate the impact of outliers on the learning process and the effectiveness of the proposed M-OSELM algorithm to deal with the outliers, we take Mackey–Glass as a representative example and present the typical training result of the M-OSELM on this contaminated time series. For comparison, the training result of the R-OSELM is also given, as shown in Fig. 3. We can see from Fig. 3 that the actually obtained output of the R-OSELM does not fit well with the desired output due to the influence of the outliers, while the M-OSELM gives a very accurate fitting result under the same outliers condition. It indicates that the M-OSELM is a more robust learning algorithm with better immunity to outliers. The trained models are then applied to the testing set for prediction, and the typical prediction results of the R-OSELM and the M-OSELM are demonstrated in Figs. 4 and 5, respectively. Figures 4a and 5a compare the desired output with the actual prediction output of the two algorithms. Similar to the training results, the actual prediction output of the R-OSELM does not fit well with the desired output, while for M-OSELM the two outputs are almost entirely consistent with each other. To go along with this, the corresponding prediction errors between the actual values and desired values of the two algorithms are given in Figs. 4b and 5b, respectively. The results show that the prediction error of the M-OSELM is much smaller than that of the R-OSELM. To sum up, by taking the M-estimator-based cost function, the proposed M-OSELM algorithm can effectively get rid of the influence of the outliers in the learning process and tends to produce an unbiased prediction model, and then an accurate prediction result can be expected. The similar experimental results can be found on the other three chaotic time series.

All the above simulations are performed for one-step-ahead prediction of the chaotic time series with outliers, and the experimental results show that the proposed M-OSELM is very desirable for dealing with the outliers. Additionally, the performance of the M-OSELM has also been tested in the case of multi-step-ahead prediction. Similar to [42], the multi-step-ahead prediction is realized

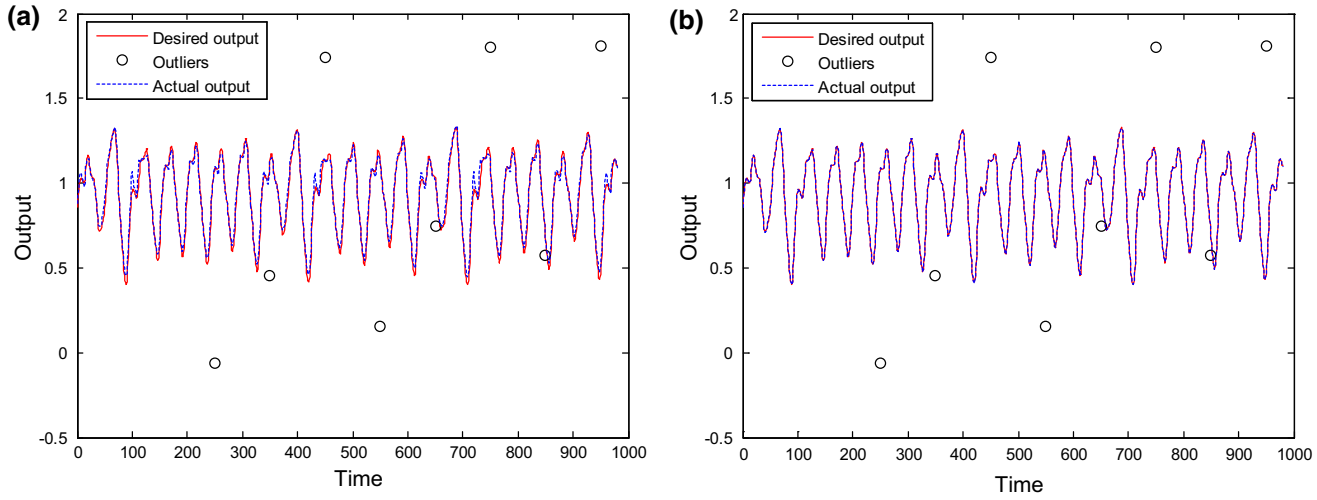

Fig. 3 Training results of Mackey–Glass chaotic time series with outliers. **a** Given by R-OSELM. **b** Given by M-OSELM

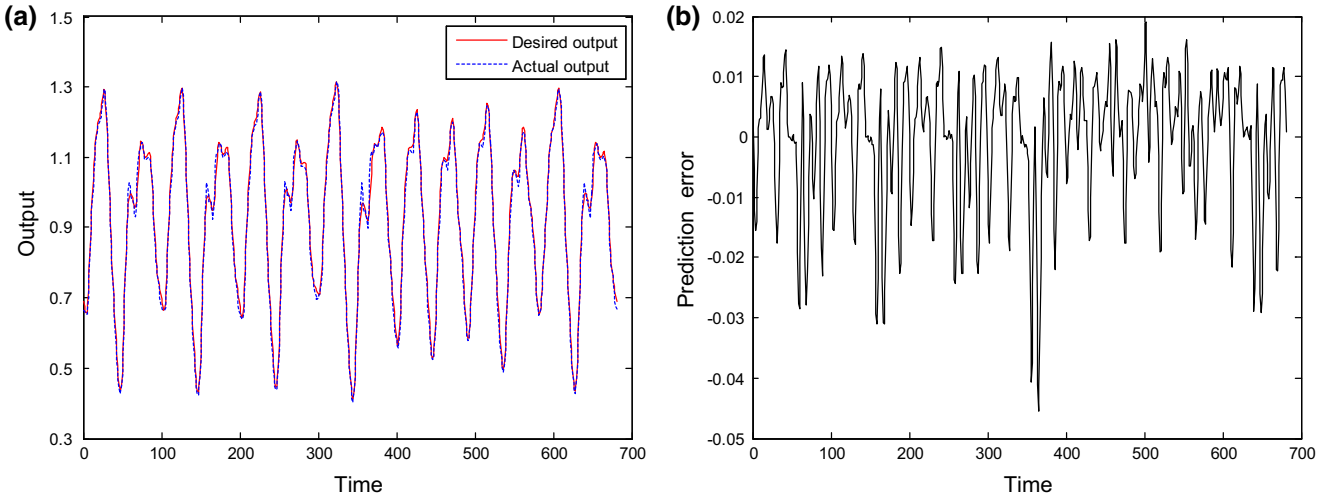

Fig. 4 Prediction results given by R-OSELM on Mackey–Glass chaotic time series with outliers. **a** Desired values and actual values. **b** Prediction errors between the desired values and the actual values

with an iterated strategy by applying the previously trained one-step-ahead prediction model repeatedly. In more detail, for a multi-step-ahead prediction we first perform the first step prediction with the M-OSELM model. Subsequently, the value just predicted is used as part of the input variables for predicting the next step by using the same M-OSELM model. We continue in this way until the entire prediction step is reached. We have carried out the multi-step-ahead prediction on the above four chaotic time series contaminated with outliers and the Mackey–Glass is taken as a representative example to illustrate the prediction results. For comparison, the typical prediction results of the M-OSELM as well as the R-OSELM for performing multi-step-ahead prediction on the Mackey–Glass chaotic time series are given in Fig. 6. From Fig. 6, we can see first that the prediction performances of the R-OSELM and the M-OSELM both degrade gradually with the increment of the prediction step. This is because in the multistep iterated prediction the error present in intermediate forecasts will accumulate continually and it makes the subsequent prediction more and more inaccurate. In addition, comparing the prediction results of the R-OSELM and the M-OSELM, it finds that the R-OSELM obtains acceptable result only within short prediction steps, and the predicted values deviate far from the real values when the prediction step exceeds 60, while the M-OSELM provides a more accurate prediction result and its actual output matches the desired output well even when the prediction step exceeds 200. The essential reason for the different behaviors of the two algorithms in multi-step-ahead prediction still lies in their

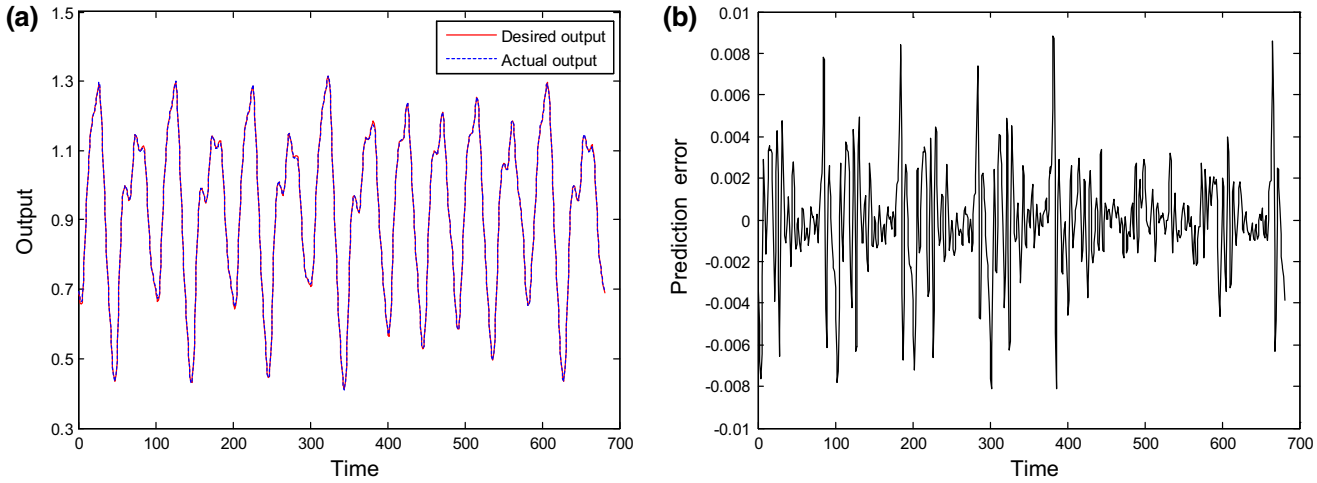

Fig. 5 Prediction results given by M-OSELM on Mackey–Glass chaotic time series with outliers. **a** Desired values and actual values. **b** Prediction errors between the desired values and the actual values

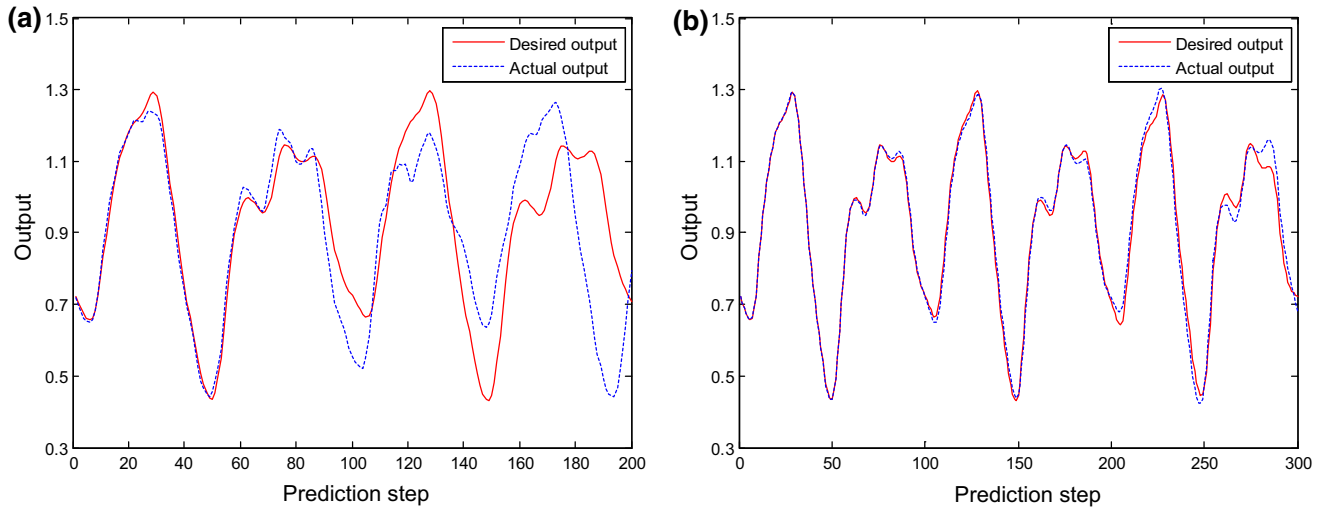

Fig. 6 Multi-step-ahead prediction results of Mackey–Glass chaotic time series with outliers. **a** Given by R-OSELM. **b** Given by M-OSELM

disparate capabilities for resisting outliers. Due to the influence of the outliers, the trained R-OSELM model used for one-step-ahead prediction is biased, then the accumulation error will increase rapidly in the process of multistep iterated prediction, accordingly the prediction performance deteriorates quickly with the increment of the prediction step. In contrast, the M-OSELM has very good immunity to outliers and it tends to produce an unbiased one-step-ahead prediction model, then the accumulation error will increase more slowly in the process of multistep iterated prediction, so a longer predictable step with good prediction accuracy can be achieved. For the other three chaotic time series, though the predictable steps are not the same, the similar experimental conclusions can be obtained.

4.2.3 Comparisons between R-OSELM and M-OSELM

In our simulations, five ELM-based algorithms including the original ELM, R-ELM, OSELM, R-OSELM and the proposed M-OSELM are investigated to evaluate their robustness for modeling a chaotic time series contaminated with outliers. The original ELM and R-ELM are of the batch type, while the OSELM, R-OSELM and M-OSELM are implemented in a sequential pattern. Since this paper mainly focuses on the sequential learning algorithm, and the R-OSELM and the M-OSELM can always provide better results than the original OSELM, then a more comprehensive comparison between R-OSELM and M-OSELM is provided at the end of this part. In this

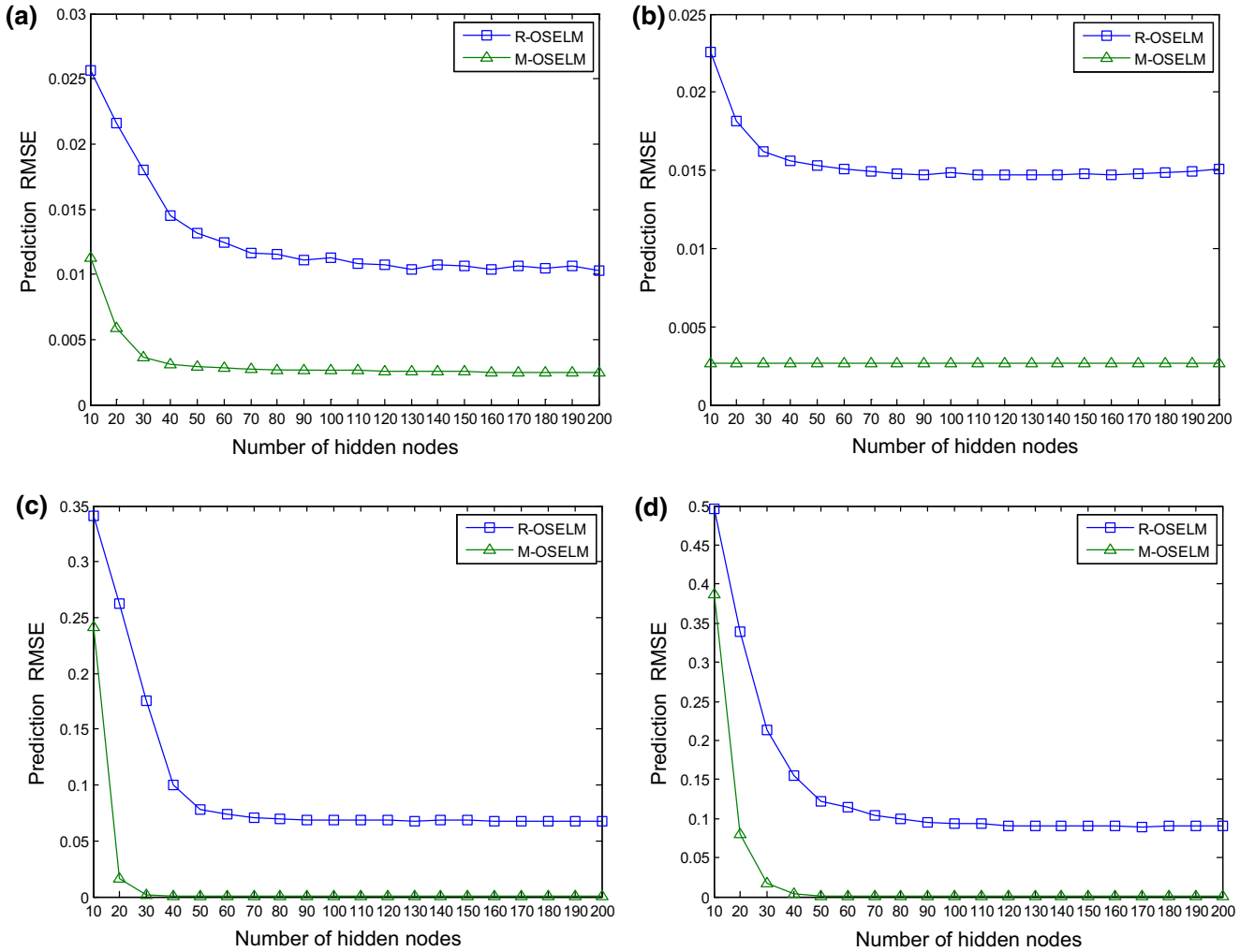

Fig. 7 Prediction performances of R-OSELM and M-OSELM with different values of the hidden nodes number. **a** Mackey–Glass. (The regularization parameter for R-OSELM is chosen as 10^{-5} and the error window length for M-OSELM is chosen as 10.) **b** Rossler. (The regularization parameter for R-OSELM is chosen as 10^{-5} and the error window length for M-OSELM is chosen as 10.) **c** Logistic. (The regularization parameter for R-OSELM is chosen as 10^{-5} and the error window length for M-OSELM is chosen as 10.) **d** Henon. (The regularization parameter for R-OSELM is chosen as 10^{-2} and the error window length for M-OSELM is chosen as 10.)

section, all the simulations are carried out on the Mackey–Glass, Rossler, logistic and Henon chaotic time series with outliers for one-step-ahead prediction.

Since the number of hidden nodes is a common parameter for all the ELM-based learning algorithms, we first compare the prediction performances of R-OSELM and M-OSELM with different values of the hidden nodes number on the four chaotic time series. In this simulation, the best values of regularization parameter and the error window length are chosen for R-OSELM and M-OSELM, respectively, and the prediction results are shown in Fig. 7. It is clear to see from Fig. 7 that the proposed M-OSELM achieves much lower prediction RMSE than the R-OSELM on all the four chaotic time series. Moreover, with the increment of hidden nodes, the prediction errors of the R-OSELM and M-OSELM decrease gradually and then remain stable, and both the two algorithms can provide stable generalization performances on a wide range of number of hidden nodes.

Besides the number of hidden nodes, there is another parameter for R-OSELM and M-OSELM, respectively, that is the regularization parameter for R-OSELM and the length of error window for M-OSELM. In this simulation, we will evaluate the influences of the two parameters on the performances of the corresponding algorithms. For a convictive demonstration, each simulation is repeated under four typical values of hidden nodes number n: $n = 50$, 100, 150, 200, respectively. Figure 8 gives the relationship between the generalization performance of the R-OSELM and its regularization parameter. Figures 9 and 10 show the prediction RMSE and training time of the M-OSELM with different values of the error window

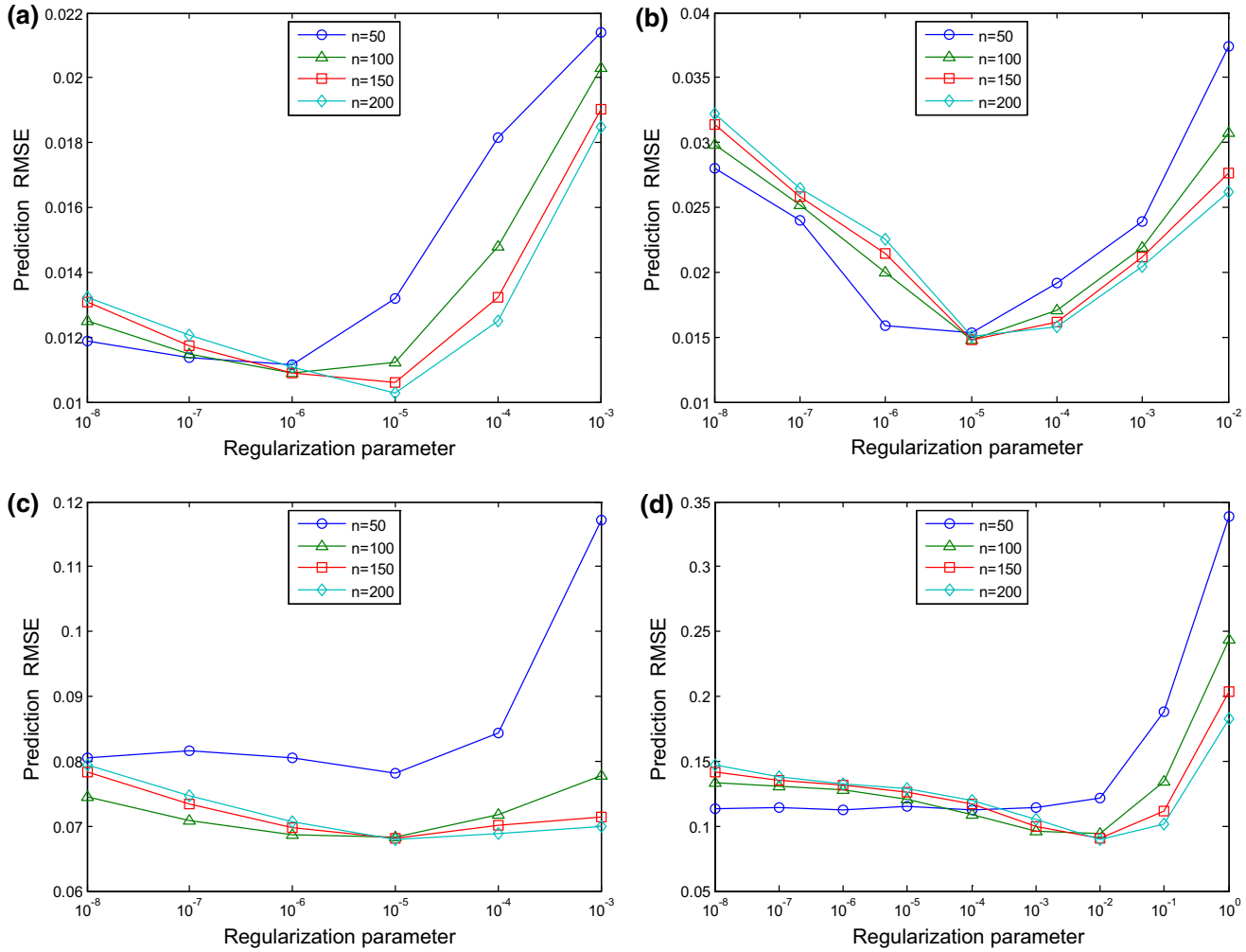

Fig. 8 Prediction performances of R-OSELM with different values of the regularization parameter. **a** Mackey–Glass. **b** Rossler. **c** Logistic. **d** Henon. (n is the number of hidden nodes.)

length, respectively. As shown in Fig. 8, the regularization parameter is an important factor in the R-OSELM and may affect the performance of the prediction model over a large range of values, especially for Mackey–Glass and Rossler chaotic time series, the prediction RMSE will become much worse if the regularization parameter is improperly chosen. In comparison, we can see from Fig. 9 that the prediction results of M-OSELM are much stable with different values of the error window length. More importantly, from the experimental results, the optimal regularization parameters of R-OSELM are various for different time series (10^{-5} for Mackey–Glass, Rossler and logistic, 10^{-2} for Henon), while the optimal value of error window length of M-OSELM is consistently achieved at 10 in all cases. In other words, the best performance of M-OSELM is much stable than that of the R-OSELM, and which can further imply that the parameter selection of M-OSELM will be much easier than that of the R-OSELM algorithm for unknown applications. Similarly, it can be seen from Fig. 10 that the training time of M-OSELM increases little with the increment of the length of error window, and it is mainly determined by the number of hidden nodes. Combining Figs. 9 and 10, we can say that the value of error window length influences very little on both the generalization ability and the training time of the M-OSELM. That is to say, the performance of the M-OSELM is insensitive to the error window length, which can be chosen in a loose way (the values between [10, 50] always satisfy according to our experimental experiences), and this makes the proposed M-OSELM an easy-to-use method in practical applications.

5 Conclusions and future work

In this paper, a robust online sequential learning algorithm named M-OSELM is proposed to train and predict the chaotic time series with outliers. By minimizing the

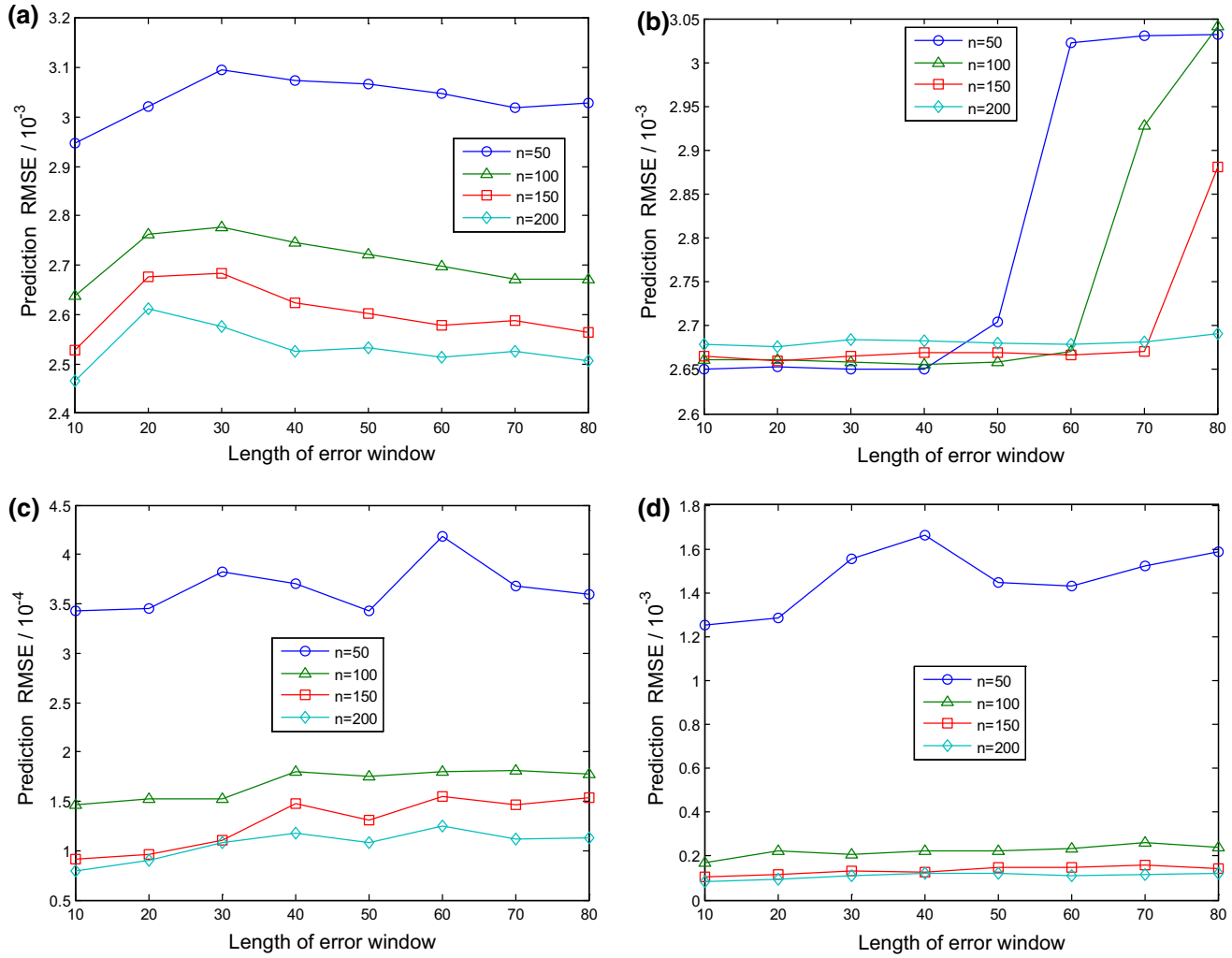

Fig. 9 Prediction performances of M-OSELM with different values of the error window length. **a** Mackey–Glass. **b** Rossler. **c** Logistic. **d** Henon. (n is the number of hidden nodes.)

robust M-estimator-based cost function instead of the conventional LS function, the possible outliers are prevented from entering the model's output weights updating scheme. Meanwhile, the M-OSELM adopts an error sliding window-based parameter estimation approach to estimate the threshold value of the M-estimator function, thanks to the built-in median operation and sliding window strategy, this approach is efficient to sequentially provide a stable estimator of the threshold value, and which can be successfully used to distinguish and detect the potential outliers online. The simulation results also demonstrate that the M-OSELM is very robust and almost not affected by outliers. Compared with other robust models working under batch learning situation, the prominent innovation of this paper is implementing a novel robust learning algorithm in an online sequential pattern. In conclusion, the proposed M-OSELM has a strong robustness to outliers while maintaining good sequential learning ability, which is especially applicable when the training data arrives sequentially and contains outliers, such as the real-world time series prediction applications.

Although the proposed method has shown satisfactory results, it still can most probably be improved. Similar to the generalized OSELM, our M-OSELM algorithm consists of an initial batch learning phase and a following sequential learning phase, and it assumes that all the initial obtained training data are outliers-free. This assumption, however, does not always hold up in practical applications. To solve this problem, the M-estimator technology may also be well incorporated into the batch ELM algorithm to provide a robust solution for the initial batch learning process, and similar works can be found in [14, 15]. As future work, we will consider this problem further and try to provide a more complete solution for the prediction of chaotic time series with outliers.

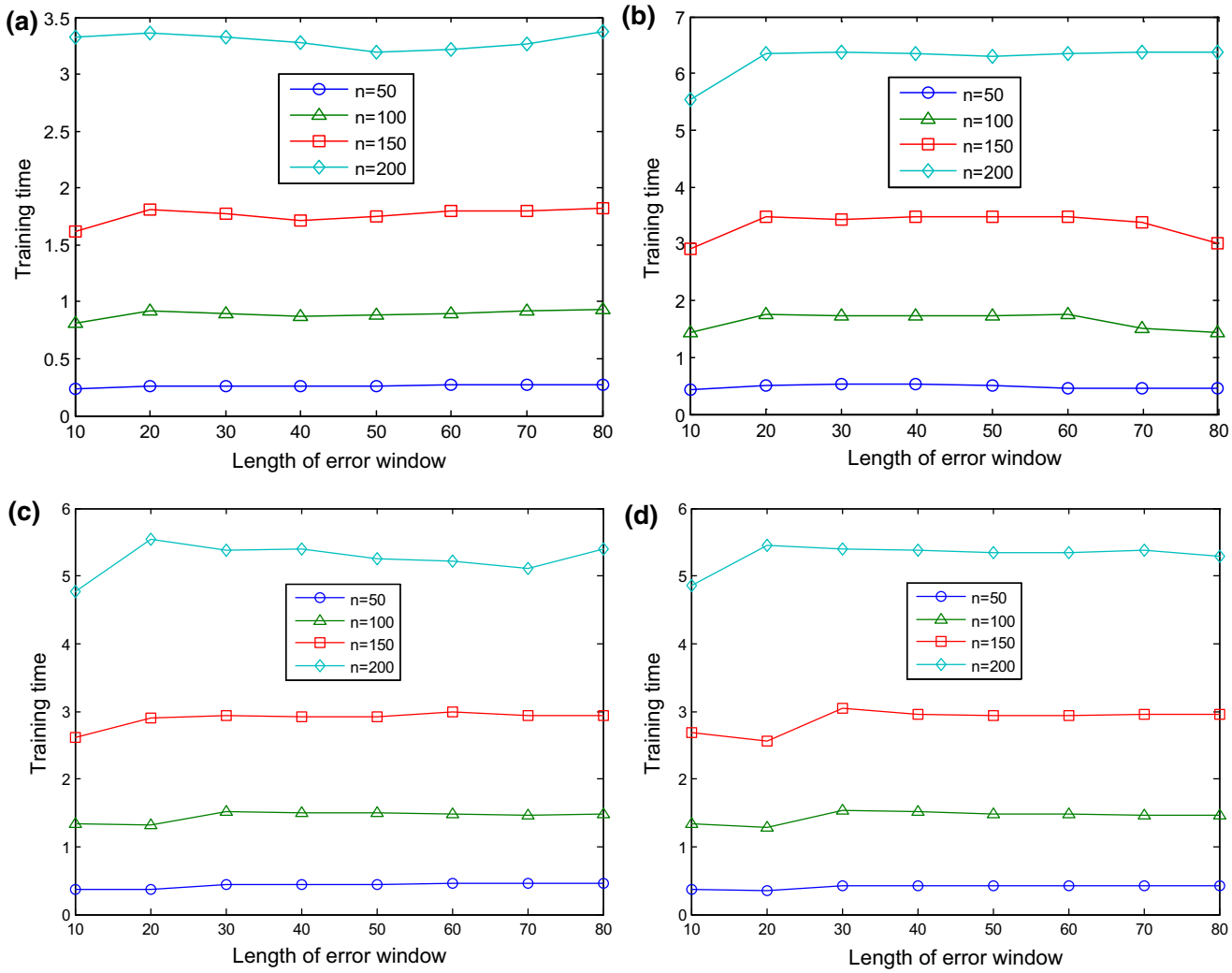

Fig. 10 Training time of M-OSELM with different values of the error window length. **a** Mackey–Glass. **b** Rossler. **c** Logistic. **d** Henon. (*n* is the number of hidden nodes.)

Acknowledgments This work is supported by the National Natural Science Foundation of China (Grant Nos. 61139002, 61379064), the National Key Technology Research and Development Program of the Ministry of Science and Technology of China (Grant No. 2014BAJ04B02), the Natural Science Foundation of Jiangsu Province of China (Grant No. BK2012672), 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. Wu J, Long J, Liu M (2015) Evolving RBF neural networks for rainfall prediction using hybrid particle swarm optimization and genetic algorithm. Neurocomputing 148:136–142
2. Akrami SA, El-Shafie A, Naseri M, Santos CA (2014) Rainfall data analyzing using moving average (MA) model and wavelet multi-resolution intelligent model for noise evaluation to improve the forecasting accuracy. Neural Comput Appl 25(7–8): 1853–1861
3. Bao Y, Wang H, Wang B (2014) Short-term wind power prediction using differential EMD and relevance vector machine. Neural Comput Appl 25(2):283–289
4. Patel J, Shah S, Thakkar P, Kotecha K (2015) Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques. Expert Syst Appl 42(1):259–268
5. Rosillo R, Giner J, de la Fuente D (2014) The effectiveness of the combined use of VIX and support vector machines on the prediction of S&P 500. Neural Comput Appl 25(2):321–332
6. 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
7. Sudheer C, Maheswaran R, Panigrahi BK, Mathur S (2014) A hybrid SVM-PSO model for forecasting monthly streamflow. Neural Comput Appl 24(6):1381–1389
8. Hodge VJ, Austin J (2004) A survey of outlier detection methodologies. Artif Intell Rev 22(2):85–126

9. Maiz CS, Molanes-Lopez EM, Miguez J, Djuric PM (2012) A particle filtering scheme for processing time series corrupted by outliers. IEEE Trans Signal Process 60(9):4611–4627
10. Cai Y, Davies N (2003) A simple diagnostic method of outlier detection for stationary Gaussian time series. J Appl Stat 30(2):205–223
11. Fu Y-Y, Wu C-J, Jeng J-T, Ko C-N (2010) ARFNNs with SVR for prediction of chaotic time series with outliers. Expert Syst Appl 37(6):4441–4451
12. Jeng J-T, Chuang C-C, Tao C-W (2010) Hybrid SVMR-GPR for modeling of chaotic time series systems with noise and outliers. Neurocomputing 73(10–12):1686–1693
13. Li D, Han M, Wang J (2012) Chaotic time series prediction based on a novel robust echo state network. IEEE Trans Neural Netw Learn Syst 23(5):787–799
14. Shi Z, Han M (2007) Support vector echo-state machine for chaotic time-series prediction. IEEE Trans Neural Netw 18(2):359–372
15. Lee C-C, Chiang Y-C, Shih C-Y, Tsai C-L (2009) Noisy time series prediction using M-estimator based robust radial basis function neural networks with growing and pruning techniques. Expert Syst Appl 36(3):4717–4724
16. Liang NY, Huang GB, Saratchandran P, Sundararajan N (2006) A fast and accurate online sequential learning algorithm for feedforward networks. IEEE Trans Neural Netw 17(6):1411–1423
17. Huang G-B, Zhu Q-Y, Siew C-K (2006) Extreme learning machine: theory and applications. Neurocomputing 70(1):489–501
18. Yin J, Zou Z, Xu F, Wang N (2014) Online ship roll motion prediction based on grey sequential extreme learning machine. Neurocomputing 129:168–174
19. Wang H, Qian G, Feng XQ (2013) Predicting consumer sentiments using online sequential extreme learning machine and intuitionistic fuzzy sets. Neural Comput Appl 22(3–4):479–489
20. Wang X, Han M (2014) Online sequential extreme learning machine with kernels for nonstationary time series prediction. Neurocomputing 145:90–97
21. Wang J, Mao W, Wang L, Tian M (2015) Online sequential extreme learning machine with new weight-setting strategy for nonstationary time series prediction. In: Proceedings of ELM-2014, vol 1. Springer, pp 263–272
22. Pan F, Zhao H (2013) Online sequential extreme learning machine based multilayer perception with output self feedback for time series prediction. J Shanghai Jiaotong Univ (Sci) 18:366–375
23. Liao B, Zhang Z, Chan S-C (2010) A new robust Kalman filter-based subspace tracking algorithm in an impulsive noise environment. IEEE Trans Circuits Syst II Express Briefs 57(9):740–744
24. Deng W, Zheng Q, Chen L (2009) Regularized extreme learning machine. In: Proceedings of IEEE symposium on computational intelligence and data mining, pp 389–395
25. Li G, Niu P (2013) An enhanced extreme learning machine based on ridge regression for regression. Neural Comput Appl 22(3–4): 803–810
26. Huang G-B, Zhou H, Ding X, Zhang R (2012) Extreme learning machine for regression and multiclass classification. IEEE Trans Syst Man Cybern Part B Cybern 42(2):513–529
27. Huynh HT, Won Y (2011) Regularized online sequential learning algorithm for single-hidden layer feedforward neural networks. Pattern Recogn Lett 32(14):1930–1935
28. Huang G-B, Chen L, Siew C-K (2006) Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Trans Neural Netw 17(4):879–892
29. Huang G-B, Chen L (2007) Convex incremental extreme learning machine. Neurocomputing 70(16):3056–3062
30. Huang G-B, Chen L (2008) Enhanced random search based incremental extreme learning machine. Neurocomputing 71(16): 3460–3468
31. Zhang R, Lan Y, Huang G-B, Xu Z-B (2012) Universal approximation of extreme learning machine with adaptive growth of hidden nodes. IEEE Trans Neural Netw Learn Syst 23(2):365–371
32. Chng E, Chen S, Mulgrew B (1996) Gradient radial basis function networks for nonlinear and nonstationary time series prediction. IEEE Trans Neural Netw 7(1):190–194
33. Chan S-C, Zou Y-X (2004) A recursive least M-estimate algorithm for robust adaptive filtering in impulsive noise: fast algorithm and convergence performance analysis. IEEE Trans Signal Process 52(4):975–991
34. Golub GH, Van Loan CF (2013) Matrix computations, vol 4. JHU Press, Baltimore
35. Zou Y, Chan S, Ng T (2000) A recursive least M-estimate (RLM) adaptive filter for robust filtering in impulse noise. IEEE Signal Process Lett 7(11):324–326
36. Rousseeuw PJ, Leroy AM (2005) Robust regression and outlier detection, vol 589. Wiley, New York
37. Horata P, Chiewchanwattana S, Sunat K (2015) Enhancement of online sequential extreme learning machine based on the householder block exact inverse QRD recursive least squares. Neurocomputing 149:239–252
38. 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. Cybernet Syst 44(4):351–377
39. Chandra R, Zhang M (2012) Cooperative coevolution of Elman recurrent neural networks for chaotic time series prediction. Neurocomputing 86:116–123
40. 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
41. Zhang X, Wang H-L (2011) Selective forgetting extreme learning machine and its application to time series prediction. Acta Phys Sin 60(8):68–74
42. Uçar A, Yavşan E (2016) Behavior learning of a memristor-based chaotic circuit by extreme learning machines. Turk J Elec Eng & Comp Sci 24(1):121–140