Neural Comput & Applic
DOI 10.1007/s00521-017-3096-3

ORIGINAL ARTICLE

An improved kernel-based incremental extreme learning machine with fixed budget for nonstationary time series prediction

Wei Zhang[1] · Aiqiang Xu[1] · Dianfa Ping[2] · Mingzhe Gao[1]

Received: 29 August 2016 / Accepted: 15 June 2017

Abstract In order to curb the model expansion of the kernel learning methods and adapt the nonlinear dynamics in the process of the nonstationary time series online prediction, a new online sequential learning algorithm with sparse update and adaptive regularization scheme is proposed based on kernel-based incremental extreme learning machine (KB-IELM). For online sparsification, a new method is presented to select sparse dictionary based on the instantaneous information measure. This method utilizes a pruning strategy, which can prune the least "significant" centers, and preserves the important ones by online minimizing the redundancy of dictionary. For adaptive regularization scheme, a new objective function is constructed based on basic ELM model. New model has different structural risks in different nonlinear regions. At each training step, new added sample could be assigned optimal regularization factor by optimization procedure. Performance comparisons of the proposed method with other existing online sequential learning methods are presented using artificial and real-word nonstationary time series data. The results indicate that the proposed method can achieve higher prediction accuracy, better generalization performance and stability.

Keywords Time series prediction · Extreme learning machine · Online modeling · Fixed budget · Sparsity measures · Adaptive regularization

✉ Aiqiang Xu
hjhy1989@njfu.edu.cn

Wei Zhang
linguo@njfu.edu.cn

Dianfa Ping
mingyunjiang@njfu.edu.cn

Mingzhe Gao
1601111088@pku.edu.cn

[1] Office of Research and Development, Naval Aeronautical and Astronautical University, Yantai 264001, People's Republic of China

[2] Department of Electronic and Information Engineering, Naval Aeronautical and Astronautical University, Yantai 264001, People's Republic of China

1 Introduction

Nonstationary time series prediction (TSP) plays an important role in the scientific and engineered fields, such as fault-tolerant analysis, state prediction, condition monitoring and fault diagnosis [1]. The main task of TSP is to find a proper model with appropriate structure and parameters to characterize the dynamic behavior of real systems.

As a famous nonlinear modeling method, neural networks (NNs) have been extensively used to address TSP issues over the years [2]. NNs are proven to be universal approximators under suitable conditions, thus providing the means to capture information in data that are difficult to identify using other approaches. It is, however, well known that the traditional NNs algorithms suffer from some problems such as being easily trapped into local minima, slow convergence and huge computational costs [3]. In order to overcome these issues, Huang et al. [4] proposed extreme learning machine (ELM), which is a novel learning algorithm for single-layer feedforward neural networks(SLFNs). Its salient advantage is that the input weights and hidden biases are randomly chosen instead of being exhaustively tuned. It has been reported to provide better generalization performance with much faster learning speed [5–9].

During the last few years, there has been increasing attention on kernel learning methods [10]. The fundamental idea of kernel methods is that a Mercer kernel is applied to transform the low-dimensional feature vector into a high-dimensional reproducing kernel Hilbert space (RKHS), in which many nonlinear problems will become linearly solvable. Recently, Huang et al. [11] extended ELM by using kernel functions and we denote the derived results as ELM with kernel (KELM). KELM is a special batch variant of ELM [12, 13]. The experimental and theoretical analysis has shown that KELM tends to have better scalability and can achieve similar (for regression and binary class cases) or much better (for multiclass cases) generalization performance at much faster learning speed than basic SVM and LS-SVM [11].

However, in many actual applications, online, adaptive and real-time operations are required. Such requirements pose a serious challenge, since aforementioned NNs, ELM and KELM operate in batch model. In other words, whenever a new sample arrives, these algorithms need to gather old and new data together to perform a retraining in order to incorporate the new information [14]. This will result in extra storing space consumption and make the learning time become longer and longer. Fortunately, some online sequential learning algorithms have been proposed to meet the actual demands, such as online sequential ELM (OSELM) [15], regularized online sequential ELM (ReOS-ELM) [16]. Moreover, KB-IELM was presented in Ref. [17], which is a beneficial attempt to extend KELM into online application.

There is no doubt that it will be a very significant work to research the online application of KELM. For KB-IELM, although it can handle the online tasks, some issues still need to be solved. For example, KB-IELM's model order is equal to the number of training samples, which will lead to kernel matrix expansion with learning ongoing [10, 18]. On the one hand, the algorithm will be in danger of over-fitting; on the other hand, the computational complexity and storage requirement will grow superlinearly. In a word, two key problems must be solved when KB-IELM is applied to online predict nonstationary time series, i.e., (1) how to curb the model expansion; (2) how to track or adapt the nonlinear dynamics in the time-varying and nonstationary environment.

Commonly, online sparsification strategies are employed to solve above issues. It helps to curb the growing kernel function while the training sample sequentially arrives [19]. Ref. [20] proposed KELM with forgetting mechanism (FOKELM) based on traditional sliding time window. On the basis of FOKELM, Ref. [21] proposed CF-FOKELM by use of Cholesky factorization. These methods can obtain a compact dictionary, but the dictionary largely depends on the latest k observed samples. Generally speaking, the inner structure hidden in time series data will determine the samples' significance, especially in nonstationary setting [22]. So they could not guarantee the new added sample is the most valuable for prediction [23]. Ref. [24] achieved the online sparsification by deleting those old samples which had the highest similarity with new samples. This approach cannot effectively track the system dynamics. Ref. [25] proposed ALD-KOS-ELM by use of approximate linear dependence (ALD) criterion. A major criticism that can be made of ALD criterion is that it leads to costly operations with quadratic complexity in the cardinality m of the dictionary. Ref. [26] achieved the model sparsification by using fast leave-one-out cross-validation (FLOO-CV).

The model expansion has been solved effectively by aforementioned methods, but the second problem is still not worked out fundamentally. In the process of nonlinear dynamic system modeling, empirical risk and structural risk should be considered simultaneously. Generally, KELM controls structure risk by Tikhonov regularization [16, 27]. There is no doubt that time-varying system should have different structural risks in different nonlinear regions [28]. But the current KELM-based methods employ a constant regularization factor in all time, greatly limiting their effectiveness when modeling unknown nonlinear time-varying system. So, in order to improve the capability of tracking the time-varying dynamics, regularization factor should vary over time.

The aim of this paper is to seek a new online sequential learning strategy of KB-IELM for nonstationary time series prediction, which is computationally simple, and able to simultaneously solve aforementioned two key problems. So, the fix-budget method is regarded as the basic modeling strategy, KB-IELM is regarded as basic modeling algorithm and then a unified framework is further presented by associating the proposed sparsification rule and adaptive regularization scheme.

For the online sparsification, a new method is presented to select the sparse dictionary based on the instantaneous information measure. The proposed sparsification method utilizes a pruning strategy. By online minimizing the redundancy of dictionary, it decides whether to replace one of the old dictionary members with the new kernel function. In the end, a compact dictionary with predefined size can be selected adaptively. The proposed method does not need any a priori knowledge about the data, and its computational complexity is linear with the kernel center number.

For adaptive regularization scheme, a new objective function is constructed based on basic ELM model. New model has different regularization factors in different nonlinear regions. At each training iteration, in order to assign optimal regularization factor for new added sample, LOO-CV generalization error is adopted to construct loss function, then the optimal regularization factor can be derived by minimizing loss function based on gradient descent (GD) method. Moreover, a dynamic learning rate is adopted to ensure the algorithmic convergence.

Finally, we associate the proposed sparsification rule with the adaptive regularization scheme based on KB-IELM algorithm and derive a new online sequential learning algorithm for KELM (denoted as NOS-KELM for simplicity) in this paper. Performance comparisons of proposed method with other existing algorithms are presented by using artificial and real-life time series data. The simulation results testify that NOS-KELM is an effective way to predict nonstationary time series.

The rest of this paper is organized as follows. In Sect. 2, a brief description is presented about KB-IELM and several sparsity measure criterions. The proposed algorithm is given in Sect. 3, including the online selection of key nodes, the real-time update of kernel weight coefficients and online optimization of regularization factor. In Sect. 4, the computational complexity is discussed. In Sect. 5, the proposed algorithm is evaluated by both simulation and real-world data. The conclusion is drawn in Sect. 6.

2 Preliminaries

2.1 KB-IELM

For a data set containing N different training samples, $\aleph = \{(\mathbf{x}_i, y_i)|i = 1, \ldots, N\}$, where $\mathbf{x}_i \in \mathbb{R}^p$, $y_i \in \mathbb{R}$. ELM can be expressed as a simple single-input single-output (SISO) model, i.e., $f(\mathbf{x}_i) = \mathbf{h}(\mathbf{x}_i)\boldsymbol{\beta}$, where $\mathbf{h}(\mathbf{x}_i) = [h_1(\mathbf{x}_i), \ldots, h_L(\mathbf{x}_i)]$ is a feature mapping from the p-dimensional input space to the L-dimensional hidden layer feature space; L is the number of hidden layer neurons; $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_L]^T$ is the output weights vector; and $\boldsymbol{\beta}$ can be derived by solving following objective function [3, 16],

$$\begin{aligned} &\text{Min: } L_1 = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + \gamma\frac{1}{2}\sum_{i=1}^{N}\xi_i^2 \\ &\quad \text{s.t. } y_i = \mathbf{h}(\mathbf{x}_i)\boldsymbol{\beta} + \xi_i,\ i = 1, 2, \ldots, N \end{aligned} \tag{1}$$

where ξ_i is the training error, γ is the regularization factor to relax the over-fitting problem and $\gamma \in \mathbb{R}^+$. In the end, $\boldsymbol{\beta}$ can be expressed as Eq. (2).

$$\boldsymbol{\beta} = \mathbf{H}^T\left(\gamma^{-1}\mathbf{I} + \mathbf{H}\mathbf{H}^T\right)^{-1}\mathbf{Y} \tag{2}$$

where $\mathbf{Y} = [y_1, \ldots, y_N]^T$ denotes the sample target values; $\mathbf{H} = [\mathbf{h}(\mathbf{x}_1)^T, \ldots, \mathbf{h}(\mathbf{x}_N)^T]^T$ is the mapping matrix for all input samples; and $\mathbf{I}$ is an identity matrix.

Mercer's conditions can be applied on ELM. The kernel matrix is defined as $\mathbf{G} = \mathbf{H}\mathbf{H}^T$, where $\mathbf{G}(i,j) = \mathbf{h}(\mathbf{x}_i)\cdot \mathbf{h}(\mathbf{x}_j)^T = k(\mathbf{x}_i, \mathbf{x}_j)$, and $k(\cdot,\cdot)$ is a predefined kernel function. Then, the output of SLFNs by kernel-based ELM can be given as Eq. (3).

$$\begin{aligned} f(\cdot) &= \mathbf{h}(\cdot)\mathbf{H}^T(\gamma^{-1}\mathbf{I} + \mathbf{H}\mathbf{H}^T)^{-1}\mathbf{Y} \\ &= [k(\mathbf{x}_1, \cdot), \ldots, k(\mathbf{x}_N, \cdot)](\gamma^{-1}\mathbf{I} + \mathbf{G})^{-1}\mathbf{Y} \end{aligned} \tag{3}$$

Let $\mathbf{k} = [k(\mathbf{x}_1, \cdot), \ldots, k(\mathbf{x}_N, \cdot)]$, $\boldsymbol{\theta} = (\gamma^{-1}\mathbf{I} + \mathbf{G})^{-1}\mathbf{Y}$, where $\boldsymbol{\theta}$ denotes kernel weight coefficient. Let $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_m]^T$, then $f(\cdot) = \mathbf{k}\theta = \sum_{i=1}^{N}\theta_i k(\mathbf{x}_i, \cdot)$.

In Ref. [17], KB-IELM is proposed to realize adaptive update of kernel weight coefficient when new samples arrive sequentially. Suppose that $\aleph_t = \{(\mathbf{x}_i, y_i)|i = 1, \ldots, t\}$ at time t. Let $\mathbf{A}_t = \gamma^{-1}\mathbf{I}_t + \mathbf{G}_t$, the sample $(\mathbf{x}_{t+1}, y_{t+1})$ arrives at time $t + 1$, then we have

$$\mathbf{A}_{t+1} = \begin{bmatrix} \mathbf{A}_t & \mathbf{V}_{t+1} \\ \mathbf{V}_{t+1}^T & v_{t+1} \end{bmatrix} \tag{4}$$

where $\mathbf{V}_{t+1} = [k_{1,t+1}, \ldots, k_{t,t+1}]^T$, $v_{t+1} = \gamma^{-1} + k_{t+1,t+1}$.

Using the block matrix inverse lemma, $\mathbf{A}_{t+1}^{-1}$ can be calculated from $\mathbf{A}_t^{-1}$.

$$\mathbf{A}_{t+1}^{-1} = \begin{bmatrix} \mathbf{A}_t^{-1} + \mathbf{A}_t^{-1}\mathbf{V}_{t+1}\rho_{t+1}^{-1}\mathbf{V}_{t+1}^T\mathbf{A}_t^{-1} & -\mathbf{A}_t^{-1}\mathbf{V}_{t+1}\rho_{t+1}^{-1} \\ -\rho_{t+1}^{-1}\mathbf{V}_{t+1}^T\mathbf{A}_t^{-1} & \rho_{t+1}^{-1} \end{bmatrix} \tag{5}$$

where $\rho_{t+1} = v_{t+1} - \mathbf{V}_{t+1}^T\mathbf{A}_t^{-1}\mathbf{V}_{t+1}$.

So kernel weight coefficient can be updated by $\boldsymbol{\theta}_{t+1} = \mathbf{A}_{t+1}^{-1}\mathbf{Y}_{t+1}$, where $\mathbf{Y}_{t+1} = [y_1, \ldots, y_t, y_{t+1}]^T$.

2.2 Online sparsification and sparsity measures

The main drawback in model (3) is that the model order equals to the size of the training samples set, so when $t \to \infty$, it is unsuitable for online applications. It is significant to develop an active set strategy to control the increase in the model order as new samples become available. Suppose that there is a sparse dictionary $\mathbf{D}_t = \{k(\mathbf{c}_i^t, \cdot)\}_{i=1}^{m_t}$ with m_t key nodes at t-th training iteration, thus we have

$$\hat{f}_t(\cdot) = \sum_{i=1}^{m_t}\theta_i^t k(\mathbf{c}_i^t, \cdot) \tag{6}$$

where $\mathbf{c}_i^t$ is the kernel center of i-th kernel function, $\{\mathbf{c}_1^t, \mathbf{c}_2^t, \ldots, \mathbf{c}_{m_t}^t\} \subset \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t\}$, θ_i^t is the weight coefficient of i-th kernel function at t-th training iteration and $m_t \ll t$. So model (6) learns a kernel machine in the subspace spanned by the selected key nodes [29]. Indeed, the dictionary can be augmented whenever the novel kernel function $k(\mathbf{x}_i, \cdot)$ increases the diversity of the dictionary

[30]. Currently, there exists several sparsity measures to quantify this diversity, such as novelty criterion (NC) [31], ALD [32], coherence measure [33], cumulative coherence measure [34], significance measure [10, 19] and surprise measure (SC) [35]. These methods' main idea is to construct a sparse dictionary by only accepting those important input samples.

2.2.1 Novelty criterion

NC [31] first computes the distance of $\mathbf{x}_{t+1}$ to the current dictionary, if $dis = \min_{\mathbf{c}_i^t \in D_t} \left\| \mathbf{x}_{t+1} - \mathbf{c}_i^t \right\| < \delta_1$, then $\mathbf{x}_{t+1}$ will not be added into the dictionary. Otherwise, it further computes the prediction error, i.e., $e_{t+1} = y_{t+1} - \hat{f}_t(\mathbf{x}_{t+1})$, only if $|e_{t+1}| > \delta_2$, $\mathbf{x}_{t+1}$ will be accepted as a new center. Where δ_1 and δ_2 are two user-specified parameters.

2.2.2 Approximate linear dependence

ALD criterion is used to select the most linearly independent atoms in kernel algorithm, where only those atoms that cannot be absorbed well by the existing dictionary are selected [32]. The kernel function $k(\mathbf{x}_{t+1}, \cdot)$ is added to the dictionary if

$$\min_{\xi_1 \cdots \xi_m} \left\| k(\mathbf{x}_{t+1}, \cdot) - \sum_{i=1}^{m_t} \xi_i k(\mathbf{c}_i^t, \cdot) \right\|_H^2 \geq \delta$$

where δ is a positive threshold parameter to control the level of sparseness.

2.2.3 Coherence measure

The coherence corresponds to the largest correlation between atoms of a given dictionary [33]. The $k(\mathbf{x}_{t+1}, \cdot)$ is added to the dictionary if

$$\max_{i=1 \cdots m_t} \frac{\left| k(\mathbf{x}_{t+1}, \mathbf{c}_i^t) \right|}{\sqrt{k(\mathbf{x}_{t+1}, \mathbf{x}_{t+1}) k(\mathbf{c}_i^t, \mathbf{c}_i^t)}} \leq \delta \tag{7}$$

where δ is a given threshold, and $\delta \in (0, 1]$. When the unit-norm kernel is applied, Eq. (7) becomes

$$\max_{i=1 \cdots m_t} \left| k(\mathbf{x}_{t+1}, \mathbf{c}_i^t) \right| \leq \delta$$

2.2.4 Cumulative coherence measure

Cumulative coherence measure can be viewed as an extension of coherence criterion and provide a deeper description of a dictionary [34, 36]. The cumulative coherence of a dictionary with a Gram matrix $\mathbf{G}_t$ is defined as $\mu(\mathbf{G}_t) = \max_{i=1 \cdots m_t} \sum_{1 \leq j \leq m_t}^{j \neq i} \left| k(\mathbf{c}_i^t, \mathbf{c}_j^t) \right|$. A candidate kernel function $k(\mathbf{x}_{t+1}, \cdot)$ is included in the dictionary if

$$\max_{i=1 \cdots m_t} \sum_{1 \leq j \leq m_t}^{j \neq i} \left| k(\mathbf{c}_i^t, \mathbf{c}_j^t) \right| + \left| k(\mathbf{c}_i^t, \mathbf{x}_{t+1}) \right| \leq \delta$$

where δ is a given positive threshold.

2.2.5 Significance measure

Ref. [10] measures the significance of a center based on a weighted average contribution over all quantized input data. This method utilizes a pruning strategy, the center with the smallest influence on the whole system will be discarded when a new sample is included in the dictionary. Ref. [19] continuously examines the significance of the new training sample based on the Hessian matrix of the system loss function. This method utilizes a constructive strategy, samples with small significance are discarded and those with relative large significance are selected as the dictionary member.

2.2.6 Surprise measure

To determine useful data to be learned and remove redundant ones, a subjective information measure called surprise is introduced in Ref. [35]. Surprise is defined as the negative log likelihood of the samples given the learning system's hypothesis on the data distribution, i.e.,

$$S_{T_t}(\mathbf{x}_{t+1}, y_{t+1}) = -\ln p(\mathbf{x}_{t+1}, y_{t+1} | T_t)$$

where $p(\mathbf{x}_{t+1}, y_{t+1} | T_t)$ is the posterior probability of $(\mathbf{x}_{t+1}, y_{t+1})$ hypothesized by T_t.

3 KB-IELM with sparse updates and adaptive regularization scheme

3.1 Problem statement and formulation

In order to avoid those issues described in Sect. 1, according to Eq. (1), a new objective function is defined as Eq. (8).

$$\begin{aligned} \text{Min} : L_2 &= \frac{1}{2} \|\boldsymbol{\beta}\|^2 + \frac{1}{m} \sum_{i=1}^{m} \gamma_i \xi_i^2 \\ \text{s.t.}\, y_i &= \mathbf{h}(\mathbf{c}_i)\boldsymbol{\beta} + \xi_i, i = 1, 2, \ldots, m \end{aligned} \tag{8}$$

where $\mathbf{c}_i$ is the obtained key nodes by online selection; m is the number of key nodes; all key nodes compose sparse dictionary $\mathbf{D} = \{k(\mathbf{c}_i, \cdot)\}_{i=1}^m$; and $1/m$ is used to avoid the influence of the cumulate error. Compared with Eq. (1), there are three main improvements in Eq. (8):

1. The fixed-budget method is employed, which ensures the computational complexity is bounded.

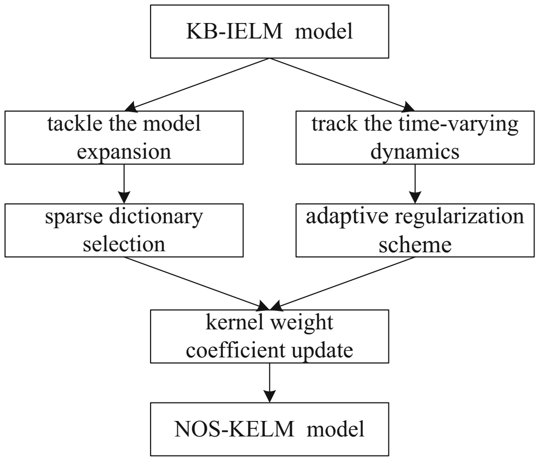
KB-IELM model
tackle the model expansion
track the time-varying dynamics
sparse dictionary selection
adaptive regularization scheme
kernel weight coefficient update
NOS-KELM model

Fig. 1 Framework of this paper

2. The online sparsification strategy is used, only those key-nodes will be accepted to update current model.
3. The adaptive regularization scheme is constructed, which makes the model has different regularization factors in different nonlinear regions.

Karush–Kuhn–Tucker(KKT) optimality conditions are employed to solve above objective function, we have $\boldsymbol{\beta}_t = \mathbf{H}^T(\boldsymbol{\Lambda}_t + \mathbf{H}\mathbf{H}^T)^{-1}\mathbf{Y}_t$. If the regularization factor vector at time t is defined as $\boldsymbol{\gamma}_t = [\gamma_1^t, \gamma_2^t, \cdots, \gamma_m^t]$, then, $\boldsymbol{\Lambda}_t = {}^m\!/_2 \cdot [\mathrm{diag}(\boldsymbol{\gamma}_t)]^{-1}$ where diag represents a diagonal matrix.

By using kernel function, we can obtain

$$\hat{f}_t(\cdot) = \mathbf{k}_t\boldsymbol{\theta}_t = \sum_{i=1}^{m} \theta_i^t \cdot k(\mathbf{c}_i^t, \cdot) \tag{9}$$

where $\mathbf{k}_t = [k(\mathbf{c}_1^t, \cdot), \ldots, k(\mathbf{c}_m^t, \cdot)]$ denotes current kernel vector; $\boldsymbol{\theta}_t = [\theta_1^t, \ldots, \theta_m^t]^T$ is the current kernel weight coefficient vector, and $\boldsymbol{\theta}_t = (\boldsymbol{\Lambda}_t + \mathbf{G}_t)^{-1}\mathbf{Y}_t$.

According to Eq. (9), we can see that the improved KB-IELM has to deal with some important problems in the process of online application, i.e., the selection of $\mathbf{D}_t$, the update of $\boldsymbol{\gamma}_t$ and $\boldsymbol{\theta}_t$. In order to solve these problems, we present a new online learning method with sparse update and adaptive regularization scheme based on KB-IELM. The framework of this paper is shown in Fig. 1.

3.2 Sparse dictionary selection based on instantaneous information measure

In this section, a novel sparsification rule is proposed based on the instantaneous information measure. This method is based on a kind of pruning strategy, which can prune the least “significant” centers and preserves the important ones, as described in Fig. 2.

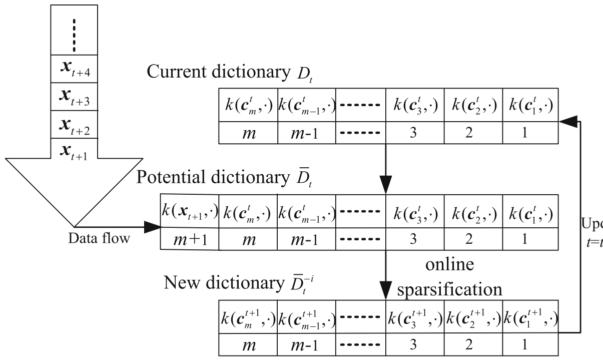

Fig. 2 Interpretation of sparsification procedure

Suppose that the obtained learning system at the training step t is $f_t = f(\mathbf{D}_t, \boldsymbol{\theta}_t, \boldsymbol{\gamma}_t)$. At the training step $t+1$, when a new training sample $(\mathbf{x}_{t+1}, y_{t+1})$ arrives, we can obtain a new kernel function $k(\mathbf{x}_{t+1}, \cdot)$. The potential dictionary is defined as $\bar{\mathbf{D}}_t = \{\mathbf{D}_t, k(\mathbf{x}_{t+1}, \cdot)\}$. In order to determine whether $k(\mathbf{x}_{t+1}, \cdot)$ can be inserted into dictionary, we firstly give two definitions based on information theory.

Definition 1: Hypothesize that the current learning system is f_t, the instantaneous posterior probability of observation sample $\mathbf{x}_{t+1}$ is $p_t(\mathbf{x}_{t+1}|f_t)$, then the information contained by $\mathbf{x}_{t+1}$, which can transfer to the current learning system, is defined as the instantaneous conditional self-information of $\mathbf{x}_{t+1}$ at time t, namely $I(\mathbf{x}_{t+1}|f_t) = -\log p_t(\mathbf{x}_{t+1}|f_t)$.

Definition 2: Hypothesize that the current learning system is f_t, the number of atoms in dictionary $\mathbf{D}_t$ is m, the instantaneous posterior probability of kernel center $\mathbf{c}_i^t (1 \le i \le m)$ is $p_t(\mathbf{c}_i^t|f_t)$, then the average self-information of $\mathbf{D}_t$ at time t is defined as the instantaneous conditional entropy of $\mathbf{D}_t$, namely

$$H(\mathbf{D}_t|f_t) = -\sum_{i=1}^{m} p_t(\mathbf{c}_i^t|f_t) \log p_t(\mathbf{c}_i^t|f_t)$$

In the actual applications, the probability distribution function (PDF) of data is hard to obtain without any priori knowledge or hypothesis. Kernel density estimator (KDE) is a reasonable method to estimate PDF. Given $\mathbf{D}_t = \{k(\mathbf{c}_1^t, \cdot), \ldots, k(\mathbf{c}_m^t, \cdot)\}$, then by use of the KDE, the instantaneous conditional PDF of kernel center can be represented as Eq. (10).

$$p_t(\mathbf{c}|\sigma, f_t) = \frac{1}{m}\sum_{i=1}^{m} k_\sigma(\mathbf{c}, \mathbf{c}_i^t) \tag{10}$$

where σ is the kernel width. According to Eq. (10), the instantaneous conditional self-information of $\mathbf{x}_{t+1}$ and the instantaneous conditional entropy of $\mathbf{D}_t$ can be, respectively, denoted as the following.

$$I(\mathbf{x}_{t+1}|\sigma,f_t) = -\log\frac{1}{m}\sum_{i=1}^{m}k_\sigma(\mathbf{x}_{t+1},\mathbf{c}_i^t)$$

$$H(\mathbf{D}_t|\sigma,f_t) = -\sum_{i=1}^{m}\left\{\left[\frac{1}{m}\sum_{j=1}^{m}k_\sigma(\mathbf{c}_i^t,\mathbf{c}_j^t)\right]\log\left[\frac{1}{m}\sum_{j=1}^{m}k_\sigma(\mathbf{c}_i^t,\mathbf{c}_j^t)\right]\right\}$$

Without loss of generality, all kernel function mentioned in this paper are unit-norm kernel, i.e., $\forall\mathbf{x}\in\mathbf{X}, k(\mathbf{x},\mathbf{x})=1$, if $k(\mathbf{x},\cdot)$ is not unit-norm, replace $k(\mathbf{x},\cdot)$ with $k(\mathbf{x},\cdot)/\sqrt{k(\mathbf{x},\mathbf{x})}$.

Let $\mathbf{e}_t=[1,\ldots,1]^T\in\mathbb{R}^{m\times1}$, then Gram matrix of dictionary $\mathbf{D}_t$ is denoted as $\mathbf{G}_t$, multiply the matrix $\mathbf{G}_t$ with the matrix $\mathbf{e}_t$, we have $\mathbf{S}_t=\mathbf{G}_t\times\mathbf{e}_t$, i.e.,

$$\mathbf{S}_t=\begin{bmatrix}\sum_{j=1}^m k_\sigma(\mathbf{c}_1^t,\mathbf{c}_j^t)\\ \sum_{j=1}^m k_\sigma(\mathbf{c}_2^t,\mathbf{c}_j^t)\\ \vdots\\ \sum_{j=1}^m k_\sigma(\mathbf{c}_m^t,\mathbf{c}_j^t)\end{bmatrix}$$

The instantaneous conditional probability of the i-th kernel center in the dictionary $\mathbf{D}_t$ under system f_t is $p_t(\mathbf{c}_i^t|\sigma,f_t)=\mathbf{S}_t(i)/m$. So the instantaneous conditional entropy can be obtained by Eq. (11).

$$H(\mathbf{D}_t|\sigma,f_t)=-\left(\frac{\mathbf{S}_t^T}{m}\right)\log\left(\frac{\mathbf{S}_t}{m}\right)\tag{11}$$

At the training step $t+1$, let $\mathbf{x}_{t+1}=\mathbf{c}_{m+1}$, then the Gram matrix of the dictionary with all potential kernel functions is denoted as $\bar{\mathbf{G}}_t$.

$$\bar{\mathbf{G}}_t=\begin{bmatrix}\mathbf{G}_t & \mathbf{K}_t\\ \mathbf{K}_t^T & 1\end{bmatrix}\tag{12}$$

where $\mathbf{K}_t=[k_\sigma(\mathbf{c}_1^t,\mathbf{c}_{m+1}),\ldots,k_\sigma(\mathbf{c}_m^t,\mathbf{c}_{m+1})]^T\in\mathbb{R}^{m\times1}$. Let $\bar{\mathbf{e}}_t=[1,\ldots,1]^T\in\mathbb{R}^{(m+1)\times1}$, compute $\bar{\mathbf{S}}_t=\bar{\mathbf{G}}_t\times\bar{\mathbf{e}}_t$, thus we can obtain Eq. (13).

$$\bar{\mathbf{S}}_t=\begin{bmatrix}\mathbf{G}_t & \mathbf{K}_t\\ \mathbf{K}_t^T & 1\end{bmatrix}\begin{bmatrix}\mathbf{e}_t\\ 1\end{bmatrix}=\begin{bmatrix}\mathbf{G}_t\times\mathbf{e}_t+\mathbf{K}_t\\ \mathbf{K}_t^T\times\mathbf{e}_t+1\end{bmatrix}=\begin{bmatrix}\mathbf{S}_t+\mathbf{K}_t\\ 1+\sum\mathbf{K}_t\end{bmatrix}\tag{13}$$

where $\sum\mathbf{K}_t$ denotes the sum of all elements in $\mathbf{K}_t$.

Let $\bar{\mathbf{E}}_t=(\bar{\mathbf{e}}_t\times\bar{\mathbf{e}}_t^T-\mathbf{I}_t)\in\mathbb{R}^{(m+1)\times(m+1)}$, where $\mathbf{I}_t$ is a $(m+1)$-order identity matrix. Obviously, $\bar{\mathbf{E}}_t$ is a matrix with its diagonal entries being 0 and all its off-diagonal entries being 1. Multiply the matrix $\bar{\mathbf{G}}_t$ with the matrix $\bar{\mathbf{E}}_t$, we can obtain

$$\bar{\mathbf{F}}_t=\bar{\mathbf{G}}_t\times(\bar{\mathbf{e}}_t\times\bar{\mathbf{e}}_t^T)-\bar{\mathbf{G}}_t=\begin{bmatrix}\mathbf{G}_t & \mathbf{k}_t\\ \mathbf{k}_t^T & 1\end{bmatrix}\underbrace{\begin{bmatrix}\mathbf{e}_t & \cdots & \mathbf{e}_t\\ 1 & \cdots & 1\end{bmatrix}}_{m+1}-\bar{\mathbf{G}}_t$$
$$=\underbrace{\begin{bmatrix}\bar{\mathbf{S}}_t & \cdots & \bar{\mathbf{S}}_t\end{bmatrix}}_{m+1}-\bar{\mathbf{G}}_t$$

Let, $\bar{\mathbf{F}}_t=\bar{\mathbf{F}}_t-\mathrm{diag}[\mathrm{diag}(\bar{\mathbf{F}}_t)]+m\mathbf{I}_t$.By substituting $\bar{\mathbf{G}}_t$ and $\bar{\mathbf{S}}_t$ into $\bar{\mathbf{F}}_t$, we can get

$$\bar{\mathbf{F}}_t=\begin{bmatrix} m & \cdots & \sum_{1\le j\le m+1}^{j\ne m}k_\sigma(\mathbf{c}_1^t,\mathbf{c}_j^t) & \sum_{1\le j\le m+1}^{j\ne m+1}k_\sigma(\mathbf{c}_1^t,\mathbf{c}_j^t)\\ \vdots & \ddots & \vdots & \vdots\\ \sum_{1\le j\le m+1}^{j\ne 1}k_\sigma(\mathbf{c}_m^t,\mathbf{c}_j^t) & \cdots & m & \sum_{1\le j\le m+1}^{j\ne m+1}k_\sigma(\mathbf{c}_m^t,\mathbf{c}_j^t)\\ \sum_{1\le j\le m+1}^{j\ne 1}k_\sigma(\mathbf{c}_{m+1}^t,\mathbf{c}_j^t) & \cdots & \sum_{1\le j\le m+1}^{j\ne m}k_\sigma(\mathbf{c}_{m+1}^t,\mathbf{c}_j^t) & m\end{bmatrix}$$

After the i-th $(1\le i\le m+1)$, kernel function in the potential dictionary $\bar{\mathbf{D}}_t$ is deleted, the new dictionary and the new learning system are, respectively, denoted as $\bar{\mathbf{D}}_t^{-i}$ and $\bar{f}_t^{-i}$. Then, the instantaneous conditional probability of the l-th $(l\ne i)$ kernel center can be denoted as the following.

$$p_t(\mathbf{c}_l^t|\sigma,\bar{f}_t^{-i})=\frac{1}{m}\sum_{1\le j\le m+1}^{j\ne i}k_\sigma(\mathbf{c}_l^t,\mathbf{c}_j^t)=\frac{1}{m}\bar{\mathbf{F}}_t(l,i)$$

According to Eq. (11), the instantaneous conditional entropy of $\bar{\mathbf{D}}_t^{-i}$ is written as Eq. (14).

$$H(\bar{\mathbf{D}}_t^{-i}|\sigma,\bar{f}_t^{-i})=-\left(\frac{\bar{\mathbf{F}}_t(:i)}{m}\right)^T\log\left(\frac{\bar{\mathbf{F}}_t(:i)}{m}\right)\tag{14}$$

The redundancy of $\bar{\mathbf{D}}_t^{-i}$ is defined as Eq. (15).

$$R_t^{-i}=1-\frac{H(\bar{\mathbf{D}}_t^{-i}|\sigma,\bar{f}_t^{-i})}{\log\left|\bar{\mathbf{D}}_t^{-i}\right|}=1-\frac{H(\bar{\mathbf{D}}_t^{-i}|\sigma,\bar{f}_t^{-i})}{\log|m|}\tag{15}$$

We aim to online minimize the redundancy of dictionary. That is because the less redundancy dictionary has, the more information dictionary contains. So the index of kernel function removed from old dictionary can be determined by Eq. (16).

$$i=\underset{1\le i\le m+1}{\arg\min}\,(R_t^{-i})\tag{16}$$

If $i=m+1$, the dictionary remains unchanged. This is because new kernel function $k(\mathbf{x}_{t+1},\cdot)$ is removed from the potential dictionary. If $i\ne m+1$, the i-th kernel function $k(\mathbf{c}_i^t,\cdot)$ of old dictionary is replaced by $k(\mathbf{x}_{t+1},\cdot)$. And $\mathbf{D}_{t+1}=\bar{\mathbf{D}}_t^{-i}$, $\mathbf{S}_{t+1}=\bar{\mathbf{S}}_t^{-i}$, where $\bar{\mathbf{S}}_t^{-i}$ can be obtained by use of Eq. (17). $\mathbf{G}_{t+1}$ can be computed based on Eq. (21), i.e., $\mathbf{G}_{t+1}=\mathbf{A}_{t+1}-\mathrm{diag}[\mathrm{diag}(\mathbf{A}_{t+1})]+\mathbf{I}_m$, where $\mathbf{I}_m$ is a m-order identity matrix. The details of dictionary selection can be summarized as Algorithm 1.

$$\begin{cases}\bar{\mathbf{S}}_t^{-i}(1:i-1)=\bar{\mathbf{F}}_t(1:i-1,i)\\ \bar{\mathbf{S}}_t^{-i}(i:m)=\bar{\mathbf{F}}_t(i+1:m+1,i)\end{cases}\tag{17}$$

Algorithm 1

```
Initialization: $t$=1, set $m$,$\sigma$ and $\mathbf{D}_t = \{k(\mathbf{c}_i^t, \cdot)\}_{i=1}^m$
1   Compute gram matrix $\mathbf{G}_t$, and compute $\mathbf{S}_t = \mathbf{G}_t \times \mathbf{e}_t$;
2   While new training sample $(\mathbf{x}_{t+1}, y_{t+1})$ is obtained, do
3   Compute $\mathbf{K}_t$,using (12), (13) ,respectively, compute $\bar{\mathbf{G}}_t$, $\bar{\mathbf{S}}_t$;
4   Compute $\bar{\mathbf{F}}_t$;
5   Using (14) compute $H(\bar{\mathbf{D}}_t^{-i}|\sigma, \bar{f}_t^{-i}), i = 1, \cdots, m+1$;
6   Using (15) compute redundancy $R_t^{-i}, i = 1, \cdots, m+1$;
7   Using (16) determine removable vector index $i$;
8   If $i$=$m$+1, then
9   $\mathbf{D}_{t+1} = \mathbf{D}_t$;$\mathbf{G}_{t+1} = \mathbf{G}_t$, $\mathbf{S}_{t+1} = \mathbf{S}_t$;
10  else
11  $\mathbf{D}_{t+1} = \bar{\mathbf{D}}_t^{-i}$, using (17) update $\mathbf{S}_{t+1}$, using (18) update $\mathbf{G}_{t+1}$;
12  End if
13  End while
14  Output $\mathbf{D}_{t+1}$ and $i$; $t$=$t$+1, return to step 2.
```

3.3 Kernel weight coefficient update

When a new sample is regarded as key node, in order to keep the dictionary size unchanged, an old sample needs to be deleted before the new sample was inserted into dictionary. According to Eq. (9), let $\mathbf{A}_t = \mathbf{\Lambda}_t + \mathbf{G}_t$, suppose that $\mathbf{A}_t^{-i}$ is a matrix composed of remaining elements after the i-th row and the i-th column elements are removed from $\mathbf{A}_t$, where i can be determined by use of Eq. (16).

At time $t+1$, when new sample $(\mathbf{x}_{t+1}, y_{t+1})$ is added into dictionary,$\mathbf{A}_{t+1}$ can be updated by Eq. (18).

$$\mathbf{A}_{t+1} = \begin{bmatrix} \mathbf{A}_t^{-i} & \mathbf{V}_{t+1} \\ \mathbf{V}_{t+1}^T & v_{t+1} \end{bmatrix} \tag{18}$$

where $v_{t+1} = m/(2\gamma_{new}^{t+1}) + 1$ and γ_{new}^{t+1} is a new regularization factor corresponding to new sample $\mathbf{x}_{t+1}$; $\mathbf{V}_{t+1} = [k_{1,t+1}, \ldots, k_{i-1,t+1}, k_{i+1,t+1}, \ldots, k_{m,t+1}]^T$. According to Eq. (5), then $\mathbf{A}_{t+1}^{-1}$ can be written as Eq. (19).

$$\mathbf{A}_{t+1}^{-1} = \begin{bmatrix} (\mathbf{A}_t^{-i})^{-1}\mathbf{V}_{t+1}\rho_{t+1}^{-1}\mathbf{V}_{t+1}^T(\mathbf{A}_t^{-i})^{-1} & -(\mathbf{A}_t^{-i})^{-1}\mathbf{V}_{t+1}\rho_{t+1}^{-1} \\ -\rho_{t+1}^{-1}\mathbf{V}_{t+1}^T(\mathbf{A}_t^{-i})^{-1} & \rho_{t+1}^{-1} \end{bmatrix} + \begin{bmatrix} (\mathbf{A}_t^{-i})^{-1} & \mathbf{O} \\ \mathbf{O} & 0 \end{bmatrix} \tag{19}$$

where $\rho_{t+1} = v_{t+1} - \mathbf{V}_{t+1}^T(\mathbf{A}_t^{-i})^{-1}\mathbf{V}_{t+1}$.

According to Eq. (19), kernel weight coefficient can be update by Eq. (20).

$$\mathbf{\theta}_{t+1} = \mathbf{A}_{t+1}^{-1}\mathbf{Y}_{t+1} \tag{20}$$

where $\mathbf{Y}_{t+1} = [y_1^t, \ldots, y_{i-1}^t, y_{i+1}^t, \ldots, y_m^t, y_{t+1}]^T$.

However, in the process of updating $\mathbf{\theta}_{t+1}$, the regularization factor γ_{new}^{t+1} is still unknown and needs to be optimized in order to adapt to the time-varying dynamics.

3.4 Adaptive regularization scheme

The objective of this section is to obtain the optimal regularization factor at each time step. Firstly, loss function is constructed based on leave-one-out cross-validation (LOO-CV) generalization error [28]. The LOO-CV method has been successfully used to a variety of applications because it has an unbiased estimation process and could avoid random disturbance.

According to Sect. 3.2, the dictionary is $\mathbf{D}_{t+1} = \{k(\mathbf{c}_i^{t+1}, \cdot)\}_{i=1}^m$ at time $t+1$, then each $(\mathbf{c}_i^{t+1}, y_i^{t+1})$ in $\mathbf{D}_{t+1}$ is chosen as the test sample and the remaining samples are used as training samples.

Both sides of Eq. (20) are multiplied by $\mathbf{A}_{t+1}$ simultaneously, we have $\mathbf{A}_{t+1}\mathbf{\theta}_{t+1} = \mathbf{Y}_{t+1}$. According to Eq. (18), we can further obtain

$$\begin{bmatrix} \mathbf{A}_t^{-i} & \mathbf{V}_{t+1} \\ \mathbf{V}_{t+1}^T & v_{t+1} \end{bmatrix} \begin{bmatrix} \bar{\theta}_{t+1} \\ \theta_m^{t+1} \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{Y}}_{t+1} \\ y_m^{t+1} \end{bmatrix} \tag{21}$$

where $\bar{\theta}_{t+1}$ denotes a vector composed of remaining elements after the m-th element θ_m^{t+1} is removed from $\mathbf{\theta}_{t+1}$; $\bar{\mathbf{Y}}_{t+1}$ represents a vector composed of remaining elements after the m-th element y_m^{t+1} is removed from $\mathbf{Y}_{t+1}$.

When LOO-CV [26, 27, 37] is applied to carry out cross-validation for $\mathbf{c}_m^{t+1}$, the kernel weight coefficient of KELM is $\bar{\theta}_{t+1} = (\mathbf{A}_t^{-i})^{-1}\bar{\mathbf{Y}}_{t+1}$, then predictive value of $\mathbf{c}_m^{t+1}$ is $\hat{y}_m^{t+1} = \mathbf{V}_{t+1}^T(\mathbf{A}_t^{-i})^{-1}\bar{\mathbf{Y}}_{t+1}$. According to Eq. (21), the following equation can be derived.

$$\begin{cases} \bar{\mathbf{Y}}_{t+1} = \mathbf{A}_t^{-i}\bar{\theta}_{t+1} + \mathbf{V}_{t+1}\theta_m^{t+1} \\ y_m^{t+1} = v_{t+1}\theta_m^{t+1} + \mathbf{V}_{t+1}^T\bar{\theta}_{t+1} \end{cases} \tag{22}$$

According to Eq. (22), then the predictive value of $\mathbf{c}_m^{t+1}$ can be rewritten as Eq. (23).

$$\hat{y}_m^{t+1} = \mathbf{V}_{t+1}^T\bar{\theta}_{t+1} + \mathbf{V}_{t+1}^T(\mathbf{A}_t^{-i})^{-1}\mathbf{V}_{t+1}\theta_m^{t+1} \tag{23}$$

Combining Eqs. (22) and (23), the LOO-CV generalization error on $(\mathbf{c}_m^{t+1}, y_m^{t+1})$ can be expressed as follows.

$$\begin{aligned} \xi_{loo}^{(-m)}(t+1) &= y_m^{t+1} - \hat{y}_m^{t+1} \\ &= (v_{t+1} - \mathbf{V}_{t+1}^T(\mathbf{A}_t^{-i})^{-1}\mathbf{V}_{t+1})\theta_m^{t+1} \end{aligned}$$

According to Eq. (19), we have $\rho_{t+1} = v_{t+1} - \mathbf{V}_{t+1}^T(\mathbf{A}_t^{-i})^{-1}\mathbf{V}_{t+1}$.

Thus, $\xi_{loo}^{(-m)}(t+1) = \rho_{t+1}\theta_m^{t+1}$, and ρ_{t+1} is the reciprocal of the m-th diagonal element of $\mathbf{A}_{t+1}^{-1}$. According to the conclusion in Ref. [26], exchanging the order of elements will not change the solution of Eq. (21). So the LOO-CV generalization error for each sample in $\mathbf{D}_{t+1}$ can be expressed as follows.

$$\xi_{loo}^{(-k)}(t+1) = \frac{(\mathbf{A}_{t+1}^{-1}\mathbf{Y}_{t+1})_k}{\mathrm{diag}(\mathbf{A}_{t+1}^{-1})_k}, \quad k = 1,\ldots,m \tag{24}$$

As a result, the generalization error vector between the estimated values and the real values can be denoted as $\mathbf{E}_{t+1} = [\xi_{loo}^{(-1)}(t+1),\ldots,\xi_{loo}^{(-m)}(t+1)]$.

So the loss function can be defined as Eq. (25).

$$J(\gamma_{new}^{t+1}) = \frac{1}{2}\langle \mathbf{E}(t+1), \mathbf{E}(t+1)\rangle = \frac{1}{2}\sum_{k=1}^{m}\left[\xi_{loo}^{(-k)}(t+1)\right]^2 \tag{25}$$

The optimal regularization factor γ_{t+1}^{*} can be obtained by minimizing loss function $J(\gamma_{new}^{t+1})$, i.e.,

$$\gamma_{t+1}^{*} = \arg\min_{\gamma_{new}^{t+1}\in\Upsilon} J(\gamma_{new}^{t+1}) \tag{26}$$

where Υ is the range of γ_{new}^{t+1}. GD algorithm is an optimization algorithm, it is also known as the steepest descent method. GD algorithm is one of the most common methods for solving unconstrained optimization problems and uses the negative gradient direction as the search direction. According to the GD method, the iteration equation can be obtained by Eq. (27).

$$\gamma_{new}^{t+1}(j+1) = \gamma_{new}^{t+1}(j) - \eta(j)\frac{\partial}{\partial\gamma_{new}^{t+1}}J(\gamma_{new}^{t+1})|_{\gamma_{new}^{t+1}=\gamma_{new}^{t+1}(j)} \tag{27}$$

where $\eta(j)$ is learning rate; the initial value of γ_{new}^{t+1} is set as $\gamma_{new}^{t+1}(1) = \gamma_m^t$.Without loss of generality, using ∇_γ to denote the function gradient with respect to γ. According to Eq. (25), we can obtain Eq. (28).

$$\nabla_\gamma J(\gamma_{new}^{t+1}) = \sum_{k=1}^{m}\left[\xi_{loo}^{(-k)}(t+1)\cdot\nabla_\gamma\xi_{loo}^{(-k)}(t+1)\right] \tag{28}$$

According to Eq. (24), we can further get Eq. (29).

$$\begin{aligned}\nabla_\gamma\xi_{loo}^{(-k)}(t+1) = {} & \frac{(\nabla_\gamma\mathbf{A}_{t+1}^{-1}\cdot\mathbf{Y}_{t+1})_k\cdot\mathrm{diag}(\mathbf{A}_{t+1}^{-1})_k}{\left[\mathrm{diag}(\mathbf{A}_{t+1}^{-1})_k\right]^2} \\ & - \frac{(\mathbf{A}_{t+1}^{-1}\mathbf{Y}_{t+1})_k\cdot\mathrm{diag}(\nabla_\gamma\mathbf{A}_{t+1}^{-1})_k}{\left[\mathrm{diag}(\mathbf{A}_{t+1}^{-1})_k\right]^2}\end{aligned} \tag{29}$$

Apparently, $\nabla_\gamma\mathbf{A}_{t+1}^{-1}$ must be calculated before the optimization problem (26) can be solved. We have

$$\rho_{t+1} = {}^{m}\!/\!{}_{2\gamma_{new}^{t+1}} + 1 - \mathbf{V}_{t+1}^{T}(\mathbf{A}_t^{-i})^{-1}\mathbf{V}_{t+1} \tag{30}$$

Let $S_{t+1} = 1 - \mathbf{V}_{t+1}^{T}(\mathbf{A}_t^{-i})^{-1}\mathbf{V}_{t+1}$,

$$\mathbf{M}_{t+1} = \begin{bmatrix}(\mathbf{A}_t^{-i})^{-1}\mathbf{V}_{t+1}\mathbf{V}_{t+1}^{T}(\mathbf{A}_t^{-i})^{-1} & -(\mathbf{A}_t^{-i})^{-1}\mathbf{V}_{t+1} \\ -\mathbf{V}_{t+1}^{T}(\mathbf{A}_t^{(-i)})^{-1} & 1\end{bmatrix}$$

$$\mathbf{N}_{t+1} = \begin{bmatrix}(\mathbf{A}_t^{-i})^{-1} & \mathbf{O} \\ \mathbf{O} & 0\end{bmatrix}$$

It is easy to see that S_{t+1},$\mathbf{M}_{t+1}$ and $\mathbf{N}_{t+1}$ have no relationship with γ_{new}^{t+1}. However, $(\mathbf{A}_t^{-i})^{-1}$ is unknown in the process of calculating S_{t+1}, $\mathbf{M}_{t+1}$ and $\mathbf{N}_{t+1}$. In order to avoid the computational burden caused by recalculating matrix $(\mathbf{A}_t^{-i})^{-1}$ and improve the operation efficiency when deleting the old sample from dictionary, we present an effective method.

For convenience of description, $\mathbf{A}_t$ can be rewritten as following form, as shown in Fig. 3, where i is the removable index searched by Eq. (16).

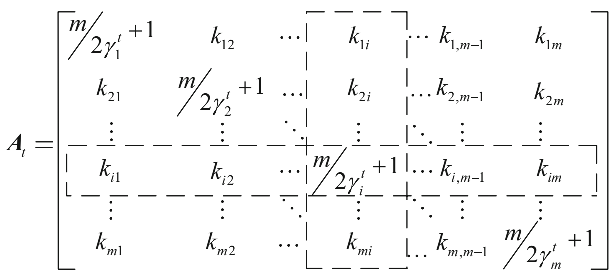

Fig. 3 Detailed composition of matrix $\mathbf{A}_t$

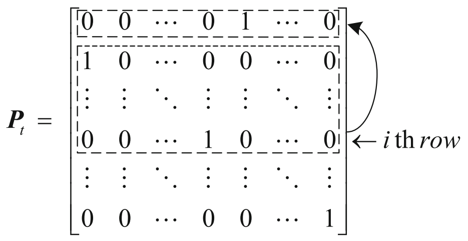

Fig. 4 Elementary matrix $\mathbf{P}_t$

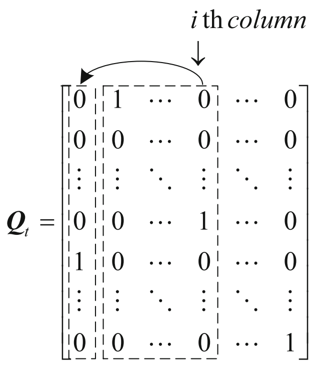

Fig. 5 Elementary matrix $\mathbf{Q}_t$

Moving the i-th row and i-th column of $\mathbf{A}_t$ to the first row and first column, respectively, this transformation process can be mathematically formulated as:

$$\tilde{\mathbf{A}}_t = \mathbf{P}_t\mathbf{A}_t\mathbf{Q}_t \tag{31}$$

where $\mathbf{P}_t$ and $\mathbf{Q}_t$ are two m-order elementary matrix, and their structures are shown in Figs. 4 and 5.

It is easy to see that $\mathbf{P}_t\mathbf{P}_t^T = \mathbf{I}_m$ and $\mathbf{Q}_t\mathbf{Q}_t^T = \mathbf{I}_m$, where $\mathbf{I}_m$ is a m-order identity matrix. So $\mathbf{P}_t$ and $\mathbf{Q}_t$ are orthogonal matrixes. Based on the properties of orthogonal matrix, we have $\mathbf{P}_t^{-1} = \mathbf{P}_t^T$, $\mathbf{Q}_t^{-1} = \mathbf{Q}_t^T$. Furthermore, $\mathbf{P}_t = \mathbf{Q}_t^T$, so the following conclusion can be obtained: $\mathbf{Q}_t^{-1} = \mathbf{P}_t$, $\mathbf{P}_t^{-1} = \mathbf{Q}_t$.

According to Eq. (31), we can get

$$\tilde{\mathbf{A}}_t^{-1} = (\mathbf{P}_t\mathbf{A}_t\mathbf{Q}_t)^{-1} = \mathbf{P}_t\mathbf{A}_t^{-1}\mathbf{Q}_t \tag{32}$$

$\tilde{\mathbf{A}}_t^{-1}$ can be rewritten as the following form.

$$\tilde{\mathbf{A}}_t^{-1} = \begin{bmatrix} (\tilde{\mathbf{A}}_t^{-1})^{(1,1)} & (\tilde{\mathbf{A}}_t^{-1})^{(1,2:\text{end})} \\ (\tilde{\mathbf{A}}_t^{-1})^{(2:\text{end},1)} & (\tilde{\mathbf{A}}_t^{-1})^{(2:\text{end},2:\text{end})} \end{bmatrix}$$

We can also obtain the block matrix form of $\tilde{\mathbf{A}}_t$.

$$\tilde{\mathbf{A}}_t = \begin{bmatrix} \bar{v}_t & \bar{\mathbf{V}}_t \\ \bar{\mathbf{V}}_t^T & \mathbf{A}_t^{-i} \end{bmatrix}$$

where $\bar{\mathbf{V}}_t = [k_{i,1}, \ldots, k_{i,i-1}, k_{i,i+1}, \cdots, k_{i,m}]$, $\bar{v}_t = m/(2\gamma_i^t) + 1$. According to the conclusion in Ref [20], we can obtain

$$(\mathbf{A}_t^{-i})^{-1} = (\tilde{\mathbf{A}}_t^{-1})^{(2:\text{end},2:\text{end})} - \frac{(\tilde{\mathbf{A}}_t^{-1})^{(2:\text{end},1)}(\tilde{\mathbf{A}}_t^{-1})^{(1,2:\text{end})}}{(\tilde{\mathbf{A}}_t^{-1})^{(1,1)}} \tag{33}$$

According to Eq. (33), we can obtain S_{t+1}, $\mathbf{M}_{t+1}$ and $\mathbf{N}_{t+1}$. By substituting them into (19), (19) can be rewritten as Eq. (34).

$$\mathbf{A}_{t+1}^{-1} = \frac{2\gamma_{new}^{t+1}}{m + 2\gamma_{new}^{t+1}S_{t+1}}\mathbf{M}_{t+1} + \mathbf{N}_{t+1} \tag{34}$$

According to Eq. (34), $\nabla_\gamma \mathbf{A}_{t+1}^{-1}$ can be obtained.

$$\begin{aligned} \nabla_\gamma \mathbf{A}_{t+1}^{-1} &= \frac{\partial}{\partial \gamma_{new}^{t+1}}\left(\frac{2\gamma_{new}^{t+1}}{m + 2\gamma_{new}^{t+1}S_{t+1}}\mathbf{M}_{t+1} + \mathbf{N}_{t+1}\right) \\ &= \frac{2m}{(m + 2\gamma_{new}^{t+1}S_{t+1})^2}\mathbf{M}_{t+1} \end{aligned} \tag{35}$$

By substituting Eq. (29) into (28), (28) can be rewritten as Eq. (36).

$$\nabla_\gamma J(\gamma_{new}^{t+1}) = \sum_{k=1}^{m}\left\{\frac{\Delta_{1k} - \Delta_{2k}}{[\text{diag}(A_{t+1}^{-1})_k]^3}\right\} \tag{36}$$

where $\Delta_{1k} = (\mathbf{A}_{t+1}^{-1}\mathbf{Y}_{t+1})_k \cdot (\nabla_\gamma \mathbf{A}_{t+1}^{-1} \cdot \mathbf{Y}_{t+1})_k \cdot \text{diag}(\mathbf{A}_{t+1}^{-1})_k$ and $\Delta_{2k} = (\mathbf{A}_{t+1}^{-1}\mathbf{Y}_{t+1})_k^2 \cdot \text{diag}(\nabla_\gamma \mathbf{A}_{t+1}^{-1})_k$.

According to Eq. (30), we can know that ρ_{t+1}^{-1} and $\nabla_\gamma \rho_{t+1}^{-1}$ are two functions of γ_{new}^{t+1}, and

$$\begin{cases} \rho_{t+1}^{-1} = 2\gamma_{new}^{t+1}\big/(m + 2\gamma_{new}^{t+1}S_{t+1}) \\ \nabla_\gamma \rho_{t+1}^{-1} = 2m\big/(m + 2\gamma_{new}^{t+1}S_{t+1})^2 \end{cases} \tag{37}$$

Combining Eqs. (34), (35) and (37), we can further obtain Eq. (38).

$$\begin{cases} \text{diag}(\mathbf{A}_{t+1}^{-1}) = \rho_{t+1}^{-1}\text{diag}(\mathbf{M}_{t+1}) + \text{diag}(\mathbf{N}_{t+1}) \\ \text{diag}(\nabla_\gamma \mathbf{A}_{t+1}^{-1}) = \nabla_\gamma \rho_{t+1}^{-1}\text{diag}(\mathbf{M}_{t+1}) \\ \mathbf{A}_{t+1}^{-1}\mathbf{Y}_{t+1} = \rho_{t+1}^{-1}\mathbf{M}_{t+1}\mathbf{Y}_{t+1} + \mathbf{N}_{t+1}\mathbf{Y}_{t+1} \\ \nabla_\gamma \mathbf{A}_{t+1}^{-1}\mathbf{Y}_{t+1} = \nabla_\gamma \rho_{t+1}^{-1}\mathbf{M}_{t+1}\mathbf{Y}_{t+1} \end{cases} \tag{38}$$

By substituting (38) into (36), $\nabla_\gamma J(\gamma_{new}^{t+1})$ can be obtained. At each iterative step in Eq. (27), the change of γ_{new}^{t+1} will not impact on $\text{diag}(\mathbf{M}_{t+1})$, $\text{diag}(\mathbf{N}_{t+1})$, $\mathbf{M}_{t+1}\mathbf{Y}_{t+1}$,$\mathbf{N}_{t+1}\mathbf{Y}_{t+1}$ and S_{t+1}, and only has influence on ρ_{t+1}^{-1} and $\nabla_\gamma \rho_{t+1}^{-1}$. So, at each training iteration, we only need to update ρ_{t+1}^{-1} and $\nabla_\gamma \rho_{t+1}^{-1}$.

For the iteration equation shown in Eq. (27), a dynamic learning rate is adopted to ensure the convergence of the algorithm.

$$\eta(j) = \begin{cases} \eta(1 - \delta/\varepsilon_{t+1}(j)) & \text{if } \varepsilon_{t+1}(j) > \delta \\ 0 & \text{if } \varepsilon_{t+1}(j) \le \delta \end{cases} \tag{39}$$

where η is a constant, and $0<\eta\le 1$;$\varepsilon_{t+1}(j)$ is the mean value of generalization errors in the j-th iteration at time $t+1$, i.e., $\varepsilon_{t+1}(j) = \sum_{k=1}^{m}|\xi_{loo}^{(-k)}(t+1)|\big/m$; δ represents algorithm termination threshold.

By substituting the results of Eqs. (36) and (39) into (27), the optimization problem can be solved. In the end, the regularization factor vector can be updated by Eq. (40).

$$\begin{cases} \gamma_k^{t+1} = \gamma_k^t\,, \ k<i \\ \gamma_k^{t+1} = \gamma_{k+1}^t\,, \ i\le k<m \\ \gamma_k^{t+1} = \gamma_{new}^{t+1}\,, \ k = m \end{cases} \tag{40}$$

4 Complexity analysis

A brief computational framework of NOS-KELM is described in Fig. 6.

The details can be summarized as Algorithm 2. Where the initial dictionary $\mathbf{D}_0$ is composed of the first m samples $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$, kernel width σ and initial regularization factor γ_0 are determined by grid search method. Initial regularization factor vector is defined as $\boldsymbol{\gamma}_0 = [\gamma_1^0, \ldots, \gamma_m^0]$, then $\mathbf{\Lambda}_0 = {}^{m}\!/_{2} \cdot [\text{diag}(\boldsymbol{\gamma}_0)]^{-1}$. Moreover, $\mathbf{G}_0 = [k(\mathbf{x}_i, \mathbf{x}_j)]_{m\times m}$, $\mathbf{Y}_0 = [y_1, y_2, \ldots, y_m]^T$.

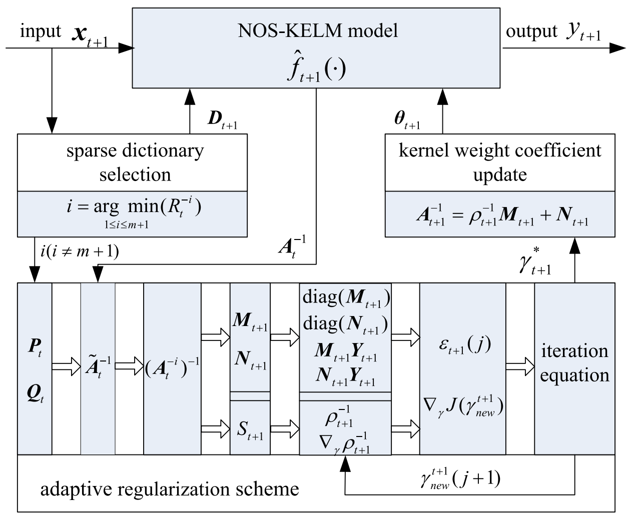

Fig. 6 Brief computational framework for NOS-KELM

Algorithm 2

Initialization: Set σ, γ_0, m, η and δ;

```
Let $t=0$, compute $\mathbf{A}_t^{-1}=(\boldsymbol{\Lambda}_t+\mathbf{G}_t)^{-1}$,$\boldsymbol{\theta}_t=\mathbf{A}_t^{-1}\mathbf{Y}_t$
1   $t=t+1$, new sample $(\mathbf{x}_{t+1},y_{t+1})$ is obtained;
2   Using Algorithm 1 to determine removable index $i$;
3   If $i=m+1$
4       $\boldsymbol{\gamma}_{t+1}=\boldsymbol{\gamma}_t$;$\mathbf{A}_{t+1}^{-1}=\mathbf{A}_t^{-1}$ ; $\boldsymbol{\theta}_{t+1}=\boldsymbol{\theta}_t$;
5   else
6       Compute $\boldsymbol{P}_t$ and $\boldsymbol{Q}_t$ according to Fig. 4 and Fig. 5;
7       Using (31), (32) to compute $\tilde{\mathbf{A}}_t$ and $\tilde{\mathbf{A}}_t^{-1}$;
8       Using (33) to compute $(\mathbf{A}_t^{-i})^{-1}$;
9       Compute $\mathbf{V}_{t+1}$ and $\mathbf{Y}_{t+1}$;and compute $\mathbf{M}_{t+1}$,$\mathbf{N}_{t+1}$ and $S_{t+1}$;
10      Compute diag($\mathbf{M}_{t+1}$),diag($\mathbf{N}_{t+1}$),$\mathbf{M}_{t+1}\mathbf{Y}_{t+1}$ and $\mathbf{N}_{t+1}\mathbf{Y}_{t+1}$;
11      Let $j=1$, $\gamma_{new}^{t+1}(j)=\gamma_m^t$
12      While ($j<100$)do
13          Using (37) to compute $\rho_{t+1}^{-1}$,$\nabla_\gamma\rho_{t+1}^{-1}$; using (38) compute to
              diag($\mathbf{A}_{t+1}^{-1}$),diag($\nabla_\gamma\mathbf{A}_{t+1}^{-1}$),$\mathbf{A}_{t+1}^{-1}\mathbf{Y}_{t+1}$,$\nabla_\gamma\mathbf{A}_{t+1}^{-1}\mathbf{Y}_{t+1}$ ;
14          Using (36) to compute $\nabla_\gamma J(\gamma_{new}^{t+1})$, and compute $\varepsilon_{t+1}(j)$ ;
15          If $\varepsilon_{t+1}(j)>\delta$
16              $\eta(j)=\eta[1-\delta/\varepsilon_{t+1}(j)]$;
                Using(27) to compute $\gamma_{new}^{t+1}(j+1)$; $j=j+1$;
17          else
18              $\gamma_{new}^{t+1}(j+1)=\gamma_{new}^{t+1}(j)$; break;
19          End if
20      End while
21      $\gamma_{t+1}^{*}=\gamma_{new}^{t+1}(j+1)$;
22      Using (40) to update $\boldsymbol{\gamma}_{t+1}$ ;using(34) to update $\mathbf{A}_{t+1}^{-1}$; using (20)
          update $\boldsymbol{\theta}_{t+1}$;
23  End if
24  Output $\boldsymbol{\theta}_{t+1}$; return to step 1.
```

We will make a brief analysis to algorithmic complexity at each training step from following three aspects. (1) Sparse dictionary selection: the time complexity of updating $\bar{\mathbf{G}}_t$ and $\bar{\mathbf{S}}_t$ is $O(m)$;the time complexity of computing $\bar{\mathbf{F}}_t$ and $H(\bar{\mathbf{D}}_t^{-i}|\sigma,\bar{f}_t^{-i})$ is $O(m+1)$; the time complexity of computing R_t^{-i} is $O(1)$; the time complexity of finding out removable index i is $O(m+1)$, so the time complexity of this stage is $\boldsymbol{O}(m+1)$. (2) Adaptive regularization scheme: the time complexity of computing $\boldsymbol{P}_t$ and $\boldsymbol{Q}_t$ are $\boldsymbol{O}(i)$; the time complexity of computing $\tilde{\mathbf{A}}_t^{-1}$, $(\mathbf{A}_t^{-i})^{-1}$ is, respectively, $O(m^2)$, $O((m-1)^2)$;the time complexity of computing S_{t+1}, $\mathbf{M}_{t+1}$ and $\mathbf{N}_{t+1}$ is, respectively, $O((m-1)^2)$, $\boldsymbol{O}(m^2)$ and $O(m)$; the time complexity of computing $\mathbf{M}_{t+1}\mathbf{Y}_{t+1}$ and $\mathbf{N}_{t+1}\mathbf{Y}_{t+1}$ are $\boldsymbol{O}(m^2)$; the time complexity of computing ρ_{t+1}^{-1} and $\nabla_\gamma\rho_{t+1}^{-1}$ are $\boldsymbol{O}$ (1); the time complexity of computing $\nabla_\gamma J(\gamma_{new}^{t+1})$ is $\boldsymbol{O}(m)$. Assuming that the optimization process needs l iterations, then the time complexity of the whole optimization procedure is $\boldsymbol{O}(lm)$. So the time complexity of this stage are $\boldsymbol{O}(m^2)$ or $\boldsymbol{O}(lm)$ (when $l>m$). (3) Kernel weight coefficient update: the time complexity of updating $\mathbf{A}_{t+1}^{-1}$ and $\boldsymbol{\gamma}_{t+1}$ is $\boldsymbol{O}(m)$, and the time complexity of updating $\boldsymbol{\theta}_{t+1}$ is $\boldsymbol{O}(m^2)$. So the time complexity of this stage is $\boldsymbol{O}(m^2)$.

5 Experimental analysis

In this section, we will give three examples to demonstrate effectiveness of the proposed method. In order to further show its performance, the proposed method is compared with three existing KELM methods:(1) KB-IELM [17] (no online sparsification procedure);(2) FOKELM [20] (based on traditional sliding time window);(3) ALD-KOS-ELM [25] (based on ALD criterion).

All kernel methods employ Gaussian kernel as kernel function, i.e., $k(\mathbf{x}_i,\mathbf{x}_j)=\exp(-||\mathbf{x}_i-\mathbf{x}_j||^2/\sigma)$, where σ is kernel width. All simulation studies are conducted in MATLAB R2010b environment running on a Windows XP PC with Intel Core i3-2120 2.2 GHz CPU and 2 GB RAM.

Moreover, the root-mean-square error (RMSE) is regarded as performance measure index of model accuracy, the maximal absolute prediction error (MAPE) and mean relative prediction error (MRPE) are regarded as performance measure indices of model stability. They are, respectively, defined as

$$\mathrm{RMSE}=\sqrt{\frac{1}{n}\sum_{i=1}^{n}|\hat{y}(i)-y(i)|^2}$$

$$\mathrm{MAPE}=\max_{i=1,\cdots,n}|\hat{y}(i)-y(i)|$$

$$\mathrm{MRPE} = \frac{1}{n}\sum_{i=1}^{n}\frac{|\hat{y}(i) - y(i)|}{y(i)}$$

5.1 Nonstationary Mackey–Glass chaotic time series

This example is an artificial nonstationary time series, and it is generated by the Mackey–Glass chaotic time series mixed with a sinusoid. The Mackey–Glass chaotic time series are generated by the following time-delay differential equation.

$$\frac{dx(t)}{dt} = \frac{ax(t-\tau)}{1 + x(t-\tau)^{10}} - bx(t)$$

where $x(t)$ is the value of time series at time t. Initial conditions are set as: $a = 0.2$, $b = 0.1$, $\tau = 17$, $x(0) = 1.2$ and $x(t) = 0$ for $t < 0$. We apply the fourth-order Runge–Kutta method with time step size $\Delta = 0.1$ to get the numerical solution of differential equation. Then, a sinusoid $0.3\sin(2\pi t/3000)$ is added to the series to create the nonstationary chaotic time series. Sampling interval is set as $T_s = 10\Delta$. In this example, the first 800 points for training and the last 400 points for testing, and all points are shown in Fig. 7. The time embedding dimension is set as 10, i.e., the input is $\mathbf{u}(t) = (x(t - T_s), \ldots, x(t - 10T_s))$.

In this example, the selected parameters are depicted in Table 1. All methods are applied to learn training samples one-by-one. Let Z denote predictive step, when Z is equal to 200 and 400, respectively, the prediction results of different methods are shown in Table 2, where the bold values are the optimal values corresponding to every evaluation index.

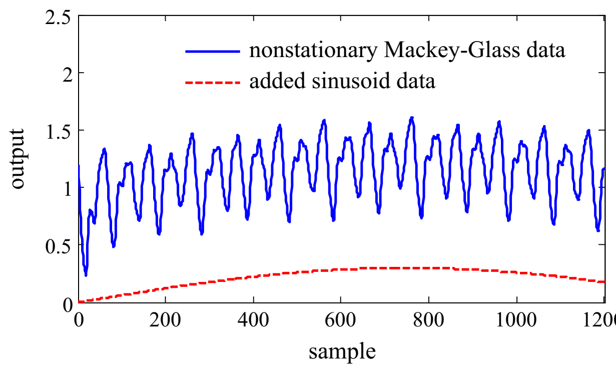

Fig. 7 Nonstationary Mackey–Glass chaotic time series

Table 1 Selected parameters for example 1

Methods	γ	σ	Else
KB-IELM [17]	10	10	–
FOKELM [20]	2e + 4	10	$m = 50$
ALD-KOS-ELM [25]	1e + 3	10	$\delta = $ 1e−5
NOS-KELM	2e + 4	10	$m = 50$, $\delta = 0.01$, $\eta = 0.8$

From Table 2, we can see that compared with KB-IELM, FOKELM and ALD-KOS-ELM, when predictive step is equal to 200, the RMSE is, respectively, reduced by 39.2, 24.4 and 14.7% when predictive step is equal to 400, the RMSE is, respectively, reduced by 42.2, 70.8 and 27.8%. So our proposed method has the higher modeling accuracy.

When the predictive step is equal to 400, the prediction results of proposed method are shown in Fig. 8. It is clear that the prediction curve can express the trend of the actual curve effectively, and the prediction errors are in a relatively low level. Besides, Fig. 9 shows the distribution of model regularization factors when the training process is finished. Obviously, the obtained model has different regularization factors in different nonlinear regions.

Figure 10 shows the learning curves of different methods, where Y-axis denotes mean square error (MSE), X-axis denotes training sample. We can see that the learning curve of our proposed method is more smooth and converges to a more accurate stage. So the proposed method has better performance than other ones.

5.2 Lorenz chaotic time series

Lorenz chaotic time series is shown as the following equation:

$$\begin{cases} dx/dt = \sigma(y - x) \\ dy/dt = rx - y - xz \\ dz/dt = -bz + xy \end{cases}$$

The initial values are set as:$\sigma = 10$, $r = 28$, $b = 8/3$, $x(0) = 1$, $y(0) = 2$ and $z(0) = 9$. The fourth-order Runge–Kutta method is used to generate a sample set with the first 800 points for training and last 400 points for testing. At the same time, a Gaussian white noise is added into time series, and SNR = 5 dB. All samples are shown in Fig. 11. In this example, $x(t)$, $y(t)$ and $z(t)$ series are used together to predict $x(t + d)$. The delayed times and embedding dimensions are set as $\tau_1 = \tau_2 = \tau_3 = 1$, $d_1 = d_2 = d_3 = 6$, respectively. And the resulting reconstructed vector is used as the input while $x(t + d)$ is served as the target output.

In this example, the selected parameters are depicted in Table 3. All methods are applied to learn training samples one-by-one.

When Z is equal to 200 and 400, respectively, the prediction results of different methods are shown in Table 4, where the bold values are the optimal values corresponding to every evaluation index. It is clear that proposed method obtains the smallest prediction errors and has the better modeling performance. Compared with KB-IELM, FOKELM and ALD-KOS-ELM, when predictive step is equal to 200, the RMSE is reduced by 59.0.3, 48.4 and 21.7%, respectively;when predictive step is equal to 400, the RMSE is reduced by 47.8, 15.3 and 16.2%,

Table 2 Prediction results of different methods for example 1

Methods	Training		Testing			
	Tr-time/s	RMSE	Te-time/s	RMSE	MAPE	MRPE
$Z = 200$						
[17]	38.438	0.0174	0.0293	0.0153	0.0420	0.0108
[20]	**0.2100**	0.0268	0.0009	0.0123	0.0317	0.0083
[25]	0.2327	0.0112	0.0009	0.0109	**0.0249**	0.0081
NOS-KELM	0.9793	**0.0104**	**0.0009**	**0.0093**	0.0263	**0.0063**
$Z = 400$						
[17]	16.799	0.0190	0.0436	0.0166	0.0438	0.0112
[20]	0.1840	0.0526	**0.0012**	0.0329	0.0842	0.0241
[25]	**0.1637**	0.0351	0.0013	0.0133	0.0384	0.0088
NOS-KELM	0.8215	**0.0112**	0.0031	**0.0096**	**0.0258**	**0.0066**

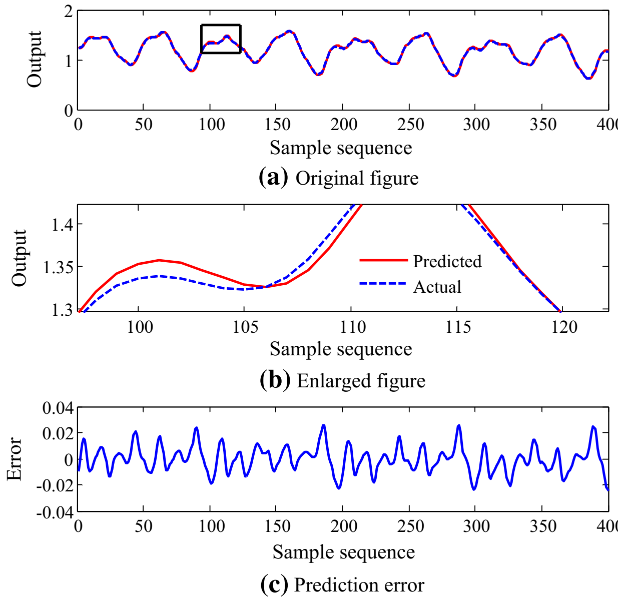

Fig. 8 Comparison of real values and prediction values obtained by NOS-KELM in example 1

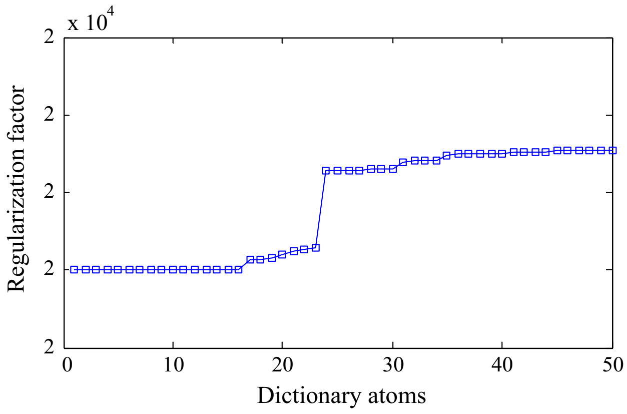

Fig. 9 Distribution of NOS-KELM regularization factors in example 1 when training is ended

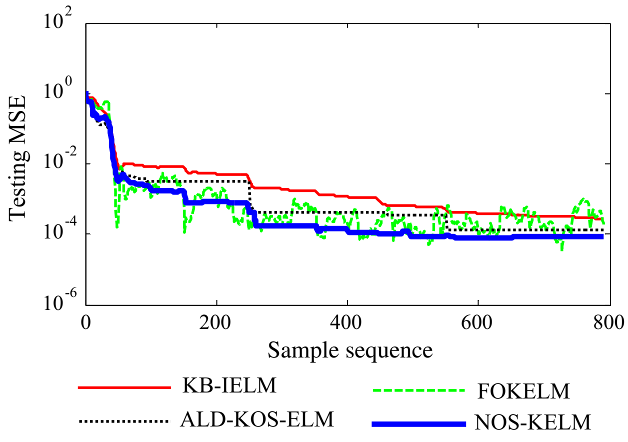

Fig. 10 Learning curves of different methods for example 1

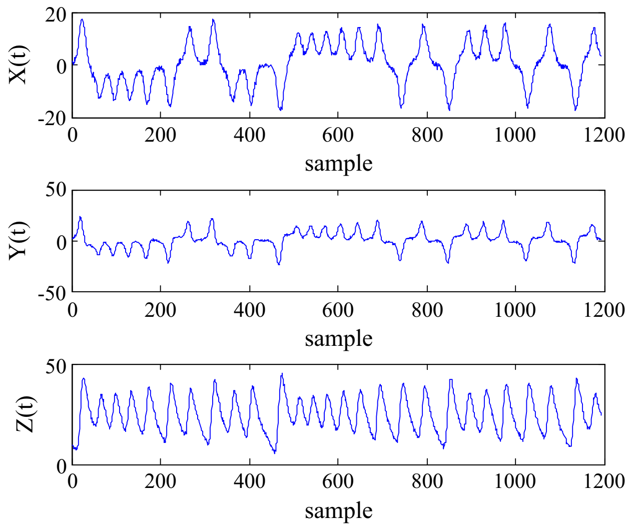

Fig. 11 Lorenz chaotic time series with Gaussian white noise

respectively. Moreover, Table 4 also shows the proposed method has the smallest MAPE and MRPE, which indicates the proposed method has better stability.

Table 3 Selected parameters for example 2

Methods	γ	σ	Else
KB-IELM [17]	7e + 4	1e + 6	–
FOKELM [20]	4e + 4	1e + 6	$m = 80$
ALD-KOS-ELM [25]	2e + 4	1e + 6	$\delta = 1\text{e}{-6}$
NOS-KELM	5e + 4	1e + 6	$m = 80, \delta = 0.2, \eta = 0.8$

As shown in Table 4, KB-IELM spends much more learning time than other methods because of no online sparsification procedure. The training time of proposed method is slightly longer than FOKELM and ALD-KOS-ELM, but it is also in a relatively low level.

When the predictive step is equal to 400, the prediction results of proposed method are shown in Fig. 12. It is clear that the prediction curve fits the actual curve well, and the prediction errors are in a relatively low level. Besides that, Fig. 13 shows the distribution of model regularization factors when the training process is ended. Obviously, compared with other methods, the obtained model has different regularization factors in different nonlinear regions.

Figure 14 shows the learning curves of different methods. Compared with Fig. 10, the same conclusion can be obtained.

5.3 Yearly sunspot numbers time series

In this section, the proposed method is applied to a real-life example. Yearly sunspot numbers time series is nonlinear, nonstationary and non-Gaussian, and has long been a benchmark in chaotic time series research. The yearly sunspot numbers we used are from 1700 to 2010 about 311 samples (see Fig. 15). In this example, the input is set as $\mathbf{u}(t) = (x(t-1), \ldots, x(t-n_s))$, the time embedding length is $n_s = 5$ in phase reconstruction stage.

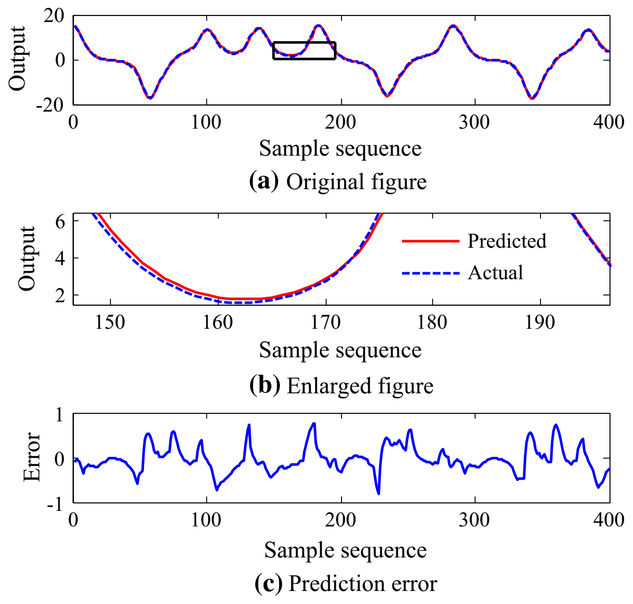

Fig. 12 Comparison of real values and prediction values obtained by NOS-KELM in example 2

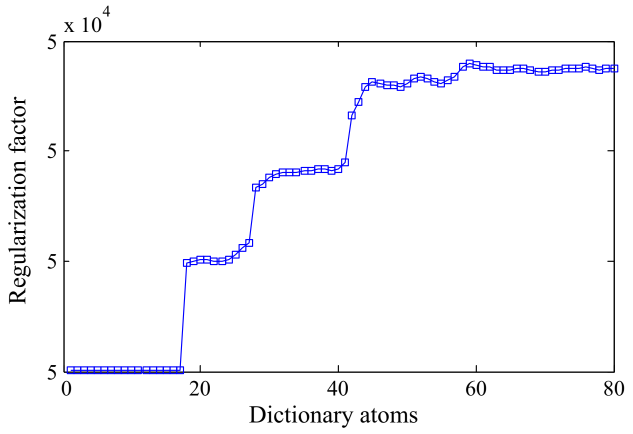

Fig. 13 Distribution of NOS-KELM regularization factors in example 2 when training is ended

Table 4 Prediction results of different methods for example 2

Methods	Training		Testing			
	Tr-time/s	RMSE	Te-time/s	RMSE	MAPE	MRPE
$Z = 200$						
[17]	40.789	0.6517	0.0395	0.6035	1.5822	0.4889
[20]	0.3603	0.5168	0.0011	0.4795	1.5484	0.2034
[25]	**0.1556**	0.3622	**0.0007**	0.3157	1.1968	0.1704
NOS-KELM	0.7155	**0.3499**	0.0008	**0.2472**	**0.7831**	**0.1278**
$Z = 400$						
[17]	17.279	0.5652	0.0485	0.5469	1.5114	**0.2929**
[20]	0.2766	0.4124	0.0018	0.3366	1.1129	0.5131
[25]	**0.1153**	0.3744	**0.0011**	0.3403	1.1962	0.7162
NOS-KELM	0.5755	**0.3646**	0.0013	**0.2851**	**0.9441**	0.6454

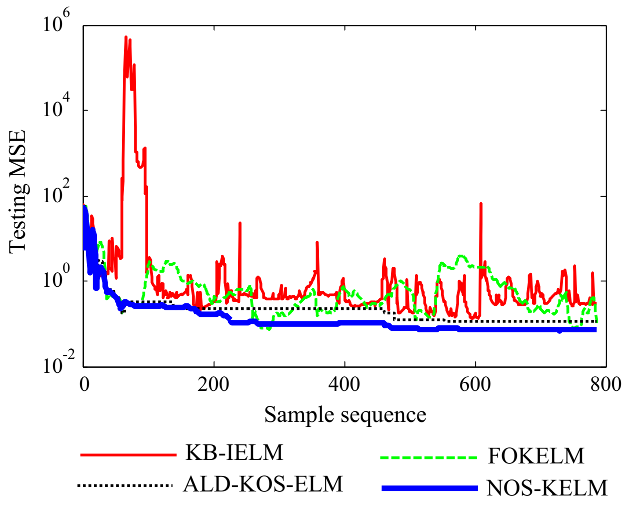

Fig. 14 Learning curves of different methods for example 2

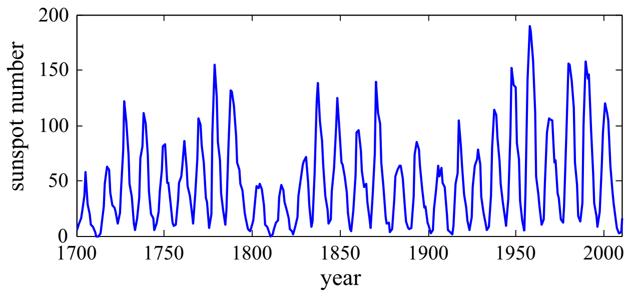

Fig. 15 Yearly sunspot numbers time series

Table 5 Selected parameters in example 3

Methods	γ	σ	Else
KB-IELM [17]	1e + 4	1e + 6	–
FOKELM [20]	2e + 4	1e + 6	$m = 50$
ALD-KOS-ELM [25]	2e + 4	1e + 6	$\delta = 3e{-}7$
NOS-KELM	2e + 4	1e + 6	$m = 50$, $\delta = 18$, $\eta = 0.8$

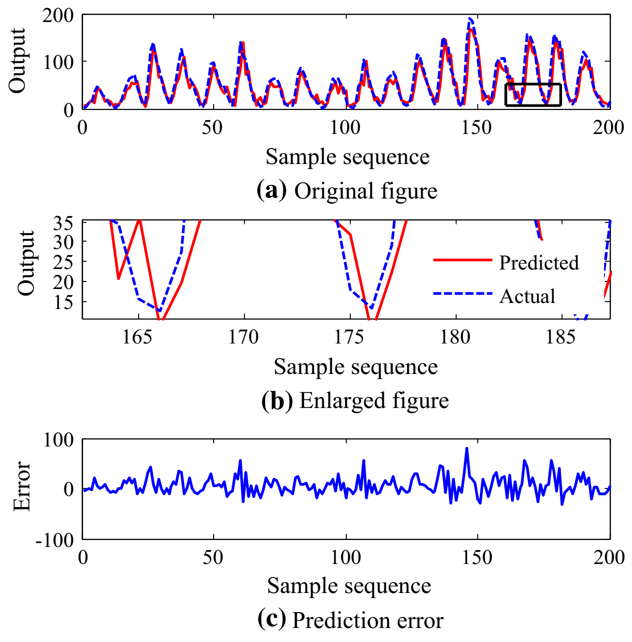

Fig. 16 Comparison of real values and prediction values obtained by NOS-KELM in example 3

In this example, the selected parameters are depicted in Table 5. Table 6 shows that the prediction results of different methods for yearly sunspot number time series when predictive step is equal to 100 and 200, respectively, where the bold values are the optimal values corresponding to every evaluation index. It is clear that our proposed method gets the smallest prediction errors and has the better modeling performance. Compared with KB-IELM, FOKELM and ALD-KOS-ELM, when predictive step is equal to 100, the RMSE is, respectively, reduced by 3.1, 15.2 and 8.5%;when predictive step is equal to 200, the RMSE is, respectively, reduced by 3.3, 4.2 and 5.6%.

When the predictive step is equal to 200, the prediction results of proposed method are shown in Fig. 16. It is clear that the prediction curve can express the trend of the actual

Table 6 Prediction results of different methods for example 3

Methods	Training		Testing			
	Tr-time/s	RMSE	Te-time/s	RMSE	MAPE	MRPE
$Z = 100$						
[17]	0.1190	13.5935	0.0012	18.5265	66.7773	0.5508
[20]	**0.0633**	14.5808	0.0006	21.1775	88.1223	0.4867
[25]	0.2447	12.9152	0.0012	19.6364	65.4295	0.4827
NOS-KELM	0.3410	**12.4182**	**0.0012**	**17.9586**	**59.7291**	**0.4774**
$Z = 200$						
[17]	0.0650	14.0217	0.0026	16.3969	66.0040	0.6895
[20]	**0.0280**	14.0356	0.0026	16.5558	65.1638	0.7141
[25]	0.0314	13.3749	0.0032	15.9481	**59.1508**	**0.5219**
NOS-KELM	0.1395	**13.2193**	**0.0015**	**15.8586**	61.8346	0.5238

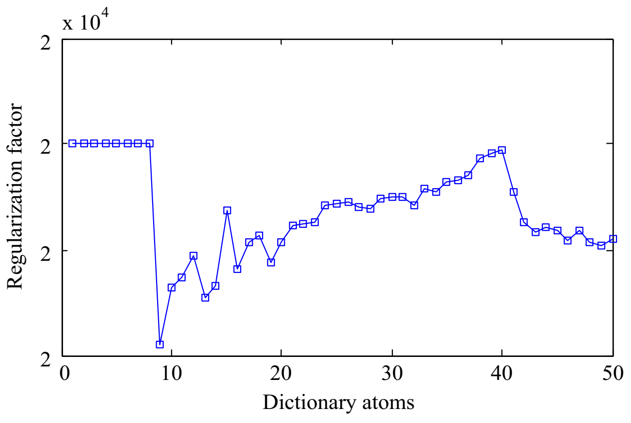

Fig. 17 Distribution of NOS-KELM regularization factors in example 3 when training is ended

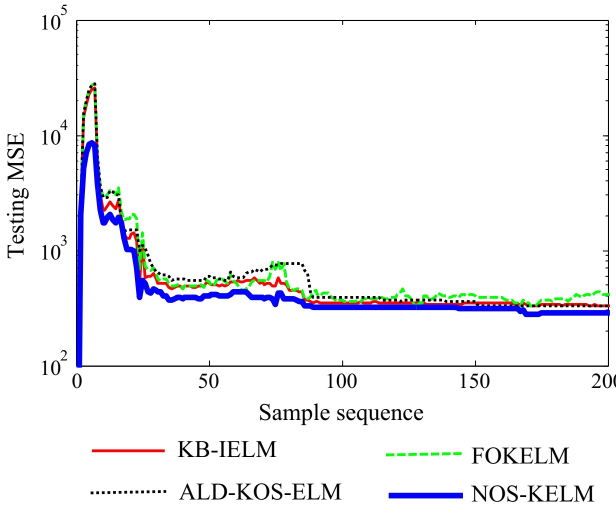

Fig. 18 Learning curves of different methods in example 3

curve effectively, and the prediction errors are in a relatively low level. Besides that, Fig. 17 shows the distribution of model regularization factors when training is ended.

According to Fig. 18, we can see that the proposed method has better stability than other ones. Obviously, its learning curve converges to a more accurate stage.

6 Conclusion

In order to curb the model expansion and adapt the nonlinear dynamics in nonstationary environment, a new online sequential learning algorithm for KELM, namely NOS-KELM, is presented. The experimental results show that NOS-KELM utilizing sparsification rule and adaptive regularization scheme can achieve higher modeling accuracy, higher convergence rate and better stability than other KELM-based online sequential learning methods.

The proposed method has the following advantages: (1) A novel sparsification rule is proposed, which can prune the least "significant" samples and preserves the important ones; (2) An adaptive regularization scheme is constructed, which ensures the new model has different structural risks in different nonlinear regions by adjusting regularization factors; (3) An unified learning framework is established, which realizes the simultaneous update of the kernel weight coefficients and regularization factors when some samples are added or removed.

In the future, we aim to investigate how the kernel types and kernel parameters would affect the prediction performance of NOS-KELM.

Acknowledgements The research was supported by National Science Foundation of China under Grant Nos. 61571454.

Compliance with ethical standards

Conflict of interest The authors declare that there is no conflict of interests regarding the publication of this paper.

References

1. Mourad E, Amiya N (2012) Comparison-based system-level fault diagnosis: a neural network approach. IEEE Trans Parallel Distrib Syst 23(6):1047–1059
2. Tian Z, Qian C, Gu B, Yang L, Liu F (2015) Electric vehicle air conditioning system performance prediction based on artificial neural network. Appl Therm Eng 89:101–104
3. Cambria E, Huang GB (2013) Extreme learning machine [trends and controversies]. IEEE Intell Syst 28(6):30–59
4. Huang GB, Zhu QY, Siew CK (2006) Extreme learning machine: theory and application. Neurocomputing 70(1–3):489–501
5. Yin G, Zhang YT, Li ZN, Ren GQ, Fan HB (2014) Online fault diagnosis method based on incremental support vector data description and extreme learning machine with incremental output structure. Neurocomputing 128:224–231
6. Rong HJ, Huang GB, Sundararajan N, Saratchandran P (2009) Online sequential fuzzy extreme learning machine for function approximation and classification problems. IEEE Transactions on systems, man and cybernetics—part B: cybernetics 39(4):1067–1072
7. Mirza B, Lin ZP, Liu N (2015) Ensemble of subset online sequential extreme learning machine for class imbalance and concept drift. Neurocomputing 149:316–329
8. Xia SX, Meng FR, Liu B, Zhou Y (2015) A kernel clustering-based possibilistic fuzzy extreme learning machine for class imbalance learning. Cogn Comput 7:74–85
9. Li XD, Mao WJ, Wei Jiang (2016) Multiple-kernel-learning-based extreme learning machine for classification design. Neural Comput Appl 27:175–184
10. Zhao SL, Chen BD, Zhu PP, Principe JC (2013) Fixed budget quantized kernel least-mean-square algorithm. Sig Process 93:2759–2770
11. Huang GB, Zhou H, Ding X, Zhang R (2011) Extreme learning machine for regression and multiclass classification. IEEE Trans Syst Man Cybern—Part B: Cybern 42(2):513–529
12. Wang XY, Han M (2014) Online sequential extreme learning machine with kernels for nonstationary time series prediction. Neurocomputing 145:90–97

13. Deng WY, Ong YS, Tan PS, Zheng QH (2016) Online sequential reduced kernel extreme learning machine. Neurocomputing 174:72–84
14. Wong SY, Yap KS, Yap HJ, Tan SC (2015) A truly online learning algorithm using hybird fuzzy ARTMAP and online extreme learning machine for pattern classification. Neural Process Lett 42:585–602
15. 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
16. Huynh HT, Won Y (2011) Regularized online sequential learning algorithm for single-hidden layer feedforward neural networks. Pattern Recogn Lett 32:1930–1935
17. Guo L, Hao JH, Liu M (2014) An incremental extreme learning machine for online sequential learning problems. Neurocomputing 128:50–58
18. Fan HJ, Song Q, Yang XL, Xu Z (2015) Kernel online learning algorithm with state feedbacks. Knowl Based Syst 89:173–180
19. Fan HJ, Song Q (2013) A sparse kernel algorithm for online time series data prediction. Expert Syst Appl 40:2174–2181
20. Zhou XR, Liu ZJ, Zhu CX (2014) Online regularized and kernelized extreme learning machines with forgetting mechanism. Math Probl Eng. doi:10.1155/2014/938548
21. Zhou XR, Wang CS (2016) Cholesky factorization based online regularized and kernelized extreme learning machines with forgetting mechanism. Neurocomputing 174:1147–1155
22. Gu Y, Liu JF, Chen YQ, Jiang XL, Yu HC (2014) TOSELM: timeliness online sequential extreme learning machine. Neurocomputing 128:119–127
23. Lim J, Lee S, Pang HS (2013) Low complexity adaptive forgetting factor for online sequential extreme learning machine (OS-ELM) for application to nonstationary system estimation. Neural Comput Appl 22:569–576
24. He X, Wang HL, Lu JH, Jiang W (2015) Online fault diagnosis of analog circuit based on limited-samples sequence extreme learning machine. Control Decis 30(3):455–460
25. Scardapance S, Comminiello D, Scarpiniti M, Uncini A (2015) Online sequential extreme learning machine with kernel. IEEE Trans Neural Netw Learn. Syst 26(9):2214–2220
26. Zhang YT, Ma C, Li ZN, Fan HB (2014) Online modeling of kernel extreme learning machine based on fast leave-one-out cross-validation. J Shanghai Jiaotong Univ 48(5):641–646
27. Shao ZF, Meng JE (2016) An online sequential learning algorithm for regularized extreme learning machine. Neurocomputing 173:778–788
28. Lu XJ, Zhou C, Huang MH, Lv WB (2016) Regularized online sequential extreme learning machine with adaptive regulation factor for time-varying nonlinear system. Neurocomputing 174:617–626
29. Lin M, Zhang LJ, Jin R, Weng SF, Zhang CS (2016) Online kernel learning with nearly constant support vectors. Neurocomputing 179:26–36
30. Honeine P (2015) Analyzing sparse dictionaries for online learning with kernels. IEEE Trans Signal Process 63(23):6343–6353
31. Platt J (1991) A resource-allocating network for function interpolation. Neural Comput 3(2):213–225
32. Engel Y, Mannor S, Meir R (2004) The kernel recursive least-squares algorithm. IEEE Trans Signal Process 52(8):2275–2285
33. Richard C, Bermudez JCM, Honeine P (2009) Online prediction of time series data with kernels. IEEE Trans Signal Process 57(3):1058–1067
34. Fan HJ, Song Q, Xu Z (2014) Online learning with kernel regularized least mean square algorithms. Expert Syst Appl 41:4349–4359
35. Liu WF, Park I, Principe JC (2009) An information theoretic approach of designing sparse kernel adaptive filters. IEEE Trans Neural Netw 20(12):1950–1961
36. Fan HJ, Song Q, Shrestha SB (2016) Kernel online learning with adaptive kernel width. Neurocomputing 175:233–242
37. Zhao YP, Wang KK (2014) Fast cross validation for regularized extreme learning machine. J Syst Eng Electron 25(5):895–900