Improved Sliding Window Kernel RLS Algorithm for Identification of Time-Varying Nonlinear Systems

Xinyu Guo[1], Menghua Jiang[1], Ying Gao[1], Shifeng Ou[1(✉)], Jindong Xu[2], and Zhuoran Cai[1]

[1] School of Optoelectronic Information Science and Technology, Yantai University, Yantai 264005, China
ousfeng@ytu.edu.cn

[2] School of Computer and Control Engineering, Yantai University, Yantai 264005, China

Abstract. The sliding window kernel recursive least squares (SW-KRLS) algorithm is one of the most widely used approach in dealing with nonlinear problems because of its simple structure, low computational complexity and high predictive accuracy. However, as data size increases, the computational efficiency of the SW-KRLS algorithm will be affected by the redundant data and the size of sliding window. In order to solve these problems, this paper proposes a variable sliding window sparse kernel recursive least squares (VSWS-KRLS) algorithm. It first uses the sliding window to constrain the size of the novelty criterion dictionary. After that, the algorithm combines the improved novelty criterion with the SW-KRLS to remove the less relevant data. In addition, mechanisms for window size adjustment are added to adjust the size of sliding window adaptively according to the system changes. The experimental results show that the proposed algorithm has better performance in identification of nonlinear system.

Keywords: Kernel recursive least squares · Nonlinear system · Variable sliding window · System identification

1 Introduction

Adaptive filtering algorithm is widely used in system identification, channel equalization, echo cancellation and other fields [1–4]. In 1950, Plackeet first proposed the recursive least squares [1] (RLS) algorithm, which has fast convergence and small prediction error. In 1960, Widrow and Hoff proposed the least mean square algorithm [5] (LMS), which is widely used because of its simple calculation and good convergence performance. However, the effect of linear filters in dealing with nonlinear problems is not ideal. In recent years, kernel methods [6] have been widely used to deal with nonlinear problems, such as support vector machines [7], kernel principal component analysis [8], Nuclear adaptive filter [9] and so on. The kernel recursive least squares [10] (KRLS) algorithm were proposed by Engel in 2004. This algorithm solves the nonlinear inseparable problem

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2022
Published by Springer Nature Switzerland AG 2022. All Rights Reserved
S. Liu and X. Ma (Eds.): ADHIP 2021, LNICST 417, pp. 203–220, 2022.
https://doi.org/10.1007/978-3-030-94554-1_18

by projecting the input data to the Hilbert space (Reproducing Kernel Hilbert Space, RKHS), and using the positive definite kernel function that satisfies the Mercer [11] condition to calculate the inner product. The projection is shown in Fig. 1. Because the KRLS algorithm has a strong tracking ability in solving nonlinear problems, it has been successfully applied in data mining, machine learning and other fields [12–14].

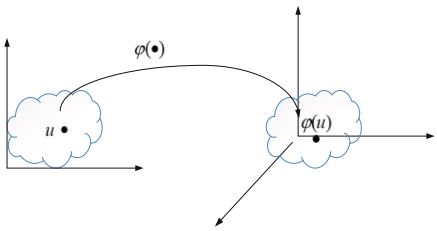

Fig. 1. Nonlinear mapping from input space to feature space.

But due to the high computational complexity of the algorithm, it is difficult to complete, so most studies mainly concentrated on the sparsification of the KRLS algorithms. In 2006, Van Vaerenbergh et al. proposed a sliding window based kernel recursive least squares algorithm [15] (SW-KRLS), which used a sliding window to limit the growing kernel matrix. This algorithm obtains lower computational complexity while retaining the advantages of KRLS. In 2010, Van Vaerenbergh et al. proposed a fixed budget recursive least squares algorithm [16] (FB-KRLS), which used the distance between the sample and the dictionary as a constraint to the sample sparsification. In 2013, Chen et al. proposed the quantized kernel recursive least squares [17] (QKRLS), which used a quantized method to reduce the input dimension. In 2018, Han et al. proposed an adaptive dynamic adjustment kernel recursive least squares method [18] (ADA-KRLS), which performs online sparsification by combining fixed budget with dynamic adaptation. In 2019, Han et al. proposed an adaptive normalized sparse quantized kernel recursive least squares [19] (ANS-QKRLS), which integrated dynamic adjustment, coherence criterion, approximate linear dependency criterion and the QKRLS algorithm for online sparsification. Zhong et al. proposed a dynamic adaptive sparse kernel recursive least squares [20] (DASKRLS) in 2020. It used approximate linear dependency (ALD) and online vector projection standards to make the data sparse, and combined with regularized maximum correlation entropy to deal with the noise impact on data. The literature [21] proposed an initial framework with forgetting factor on the basis of KRLS (FFIKRLS), which combines the ALD-KRLS algorithm with the QKRLS algorithm, and introduces a forgetting factor to sufficiently track the strongly changeable dynamic characteristics.

In summary, research based on KRLS has made good progress. However, in the face of the increasing kernel matrix size, online prediction also faces enormous challenges. In addition, KRLS has some difficulties in accommodating abrupt change. Because KRLS is based on mean square error (MSE) criterion, which is sensitive to the change. So the mutation value in the kernel matrix will affect the convergence of the algorithm. For the above problems, the variable sliding window sparse kernel recursive least squares

(VSWS-KRLS) algorithm proposes a suitable solution. In this paper, we present three major contributions:

- We reduce kernel matrix size by combining sliding window and novelty criterion with KRLS. A lot of irrelevant data in the sliding window will affect the convergence. Therefore, in order to improve calculation efficiency, we add a novelty criterion, which can raise the threshold of entering kernel matrix.
- We propose a new variable sliding window technology to enhance the capability of algorithm to track system changes. This method can adaptively adjust the window size according to the system environment.
- The algorithm has performed system identification experiments in the Wiener nonlinear systems, and has achieved good results.

The rest of this paper is arranged as follows: In Sect. 2, we introduce the KRLS algorithm. In Sect. 3, we introduce several common sparse methods. In Sect. 4, we introduce the proposed methods in this paper. In Sect. 5, we conduct simulation experiments. In Sect. 6, we present our conclusions and prospects.

2 Kernel Recursive Least Squares Algorithm

2.1 Recursive Least Squares Algorithm

The least squares [22] (LS) algorithm does not need to make assumptions about the statistical characteristics of the input signal. The RLS algorithm is a recursive extension of the LS algorithm, which can recursively update the estimated value by applying the old data. For the training data, the RLS algorithm estimates the filter coefficients $\boldsymbol{\omega}_{i-1}$ by minimizing the cost function. The cost function is:

$$J(\boldsymbol{\omega}) = \min_{\boldsymbol{\omega}} \sum_{j=1}^{i-1} \left\| d_j - \mathbf{u}_j \boldsymbol{\omega} \right\|^2 \tag{1}$$

The function is computed as:

$$\boldsymbol{\omega}_{i-1} = (\mathbf{U}_{i-1}\mathbf{U}_{i-1}^T)^{-1}\mathbf{U}_{i-1}\mathbf{d}_{i-1} \tag{2}$$

where $\mathbf{U}_{i-1} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{i-1}]_{L\times(i-1)}$, $\mathbf{u}_j$ is the input vector at time j, $\mathbf{d}_{i-1} = [d_1, d_2, \ldots, d_{i-1}]^T$, d_j is the expected response at time j. By introducing the matrix inversion lemma (Woodbury identity), the RLS algorithm can be updated:

$$r_i = 1 + \mathbf{u}_i^T \mathbf{P}_{i-1} \mathbf{u}_i \tag{3}$$

$$\mathbf{k}_i = \mathbf{P}_{i-1}\mathbf{u}_i / r_i \tag{4}$$

$$\boldsymbol{\omega}_i = \boldsymbol{\omega}_{i-1} + \mathbf{k}_i e_i \tag{5}$$

$$\mathbf{P}_i = \mathbf{P}_{i-1} - \mathbf{k}_i \mathbf{k}_i^T r_i \tag{6}$$

The RLS algorithm uses the inverse of the correlation matrix to whiten the input data, which improves the convergence performance of the filter.

2.2 Kernel Recursive Least Squares Algorithm

The KRLS algorithm projects the input data to the RKHS based on the RLS algorithm, and it can solve the nonlinear relationship without knowing the specific mapping form of input. Supposed that $\mathbf{X}$ represents the original space, $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2,\ldots,\mathbf{x}_n] \in \mathbf{R}^N$, $\mathbb{H}$ represents the Hilbert space, and the mapping φ is represented as:

$$\varphi : \mathbf{X} \rightarrow \mathbb{H}, \mathbf{x} \rightarrow \varphi(\mathbf{x}) \tag{7}$$

$\varphi(\mathbf{x})$ represents the projection of $\mathbf{x}$ in the feature space, and the nonlinear mapping is realized by the kernel function:

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = <\varphi(\mathbf{x}_i), \varphi(\mathbf{x}_j)> \tag{8}$$

where $<, >$ represents the inner product operation. As the kernel function used in this paper, the Gaussian kernel function is expressed as:

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2/2\sigma^2) \tag{9}$$

where σ represents the Gaussian kernel parameter. In KRLS algorithm, Given the expected sequence $\{d_1, d_2,\ldots\}$ and the input sequence $\{\boldsymbol{\varphi}_1, \boldsymbol{\varphi}_2,\ldots\}$, the cost function:

$$J(\boldsymbol{\omega}_i) = \min_{\boldsymbol{\omega}} \sum_{j=1}^{i} \left\| d_j - \boldsymbol{\varphi}_j^T \boldsymbol{\omega}_i \right\|^2 \tag{10}$$

where d_j and $\boldsymbol{\varphi}_j$ represent the expected sequence and the input sequence. We assume that $\boldsymbol{\Phi}_i = [\varphi(\mathbf{x}_1),\ldots,\varphi(\mathbf{x}_i)]$, and a $n \times n$-dimensional kernel matrix is defined as:

$$\mathbf{K}_i = \boldsymbol{\Phi}_i^T \boldsymbol{\Phi}_i \tag{11}$$

when $\boldsymbol{\omega}_i = \boldsymbol{\Phi}_i \boldsymbol{a}_i$, the cost function of the KRLS algorithm at time i can be obtained:

$$J = \min_{\boldsymbol{a}_i} \|\boldsymbol{d}_i - \mathbf{K}_i \boldsymbol{a}_i\|^2 \tag{12}$$

where $\boldsymbol{a}_i = [a_1,a_2,\ldots,a_i]^T$ is the weight vector and $\boldsymbol{d}_i = [d_1,d_2,\ldots,d_i]^T$ is the expected output vector. The estimated value of $\boldsymbol{\alpha}_i$ is as follows:

$$\hat{\boldsymbol{a}}_i = \hat{\mathbf{K}}_i^{-1} \boldsymbol{d}_i \tag{13}$$

The kernel matrix is expressed as:

$$\hat{\mathbf{K}}_i = \mathbf{K}_i + \lambda \mathbf{I} \tag{14}$$

where λ is the regularization factor. Unlike the RLS algorithm, the KRLS algorithm focuses on updating the kernel matrix during recursion. However, as the input data increases, the size of the kernel matrix is expanding, so the KRLS algorithm optimization problem has been transformed into the sparse problem of the kernel matrix.

3 Sparsification

The previous part makes a summary of the RLS algorithm and the KRLS, and compares the two algorithms. This part will introduce some common sparse methods.

3.1 Novelty Criterion and Approximate Linear Dependency

In the KRLS algorithm, the size of the kernel matrix will upsize linearly with the update, which will bring challenges to online prediction. Therefore, many sparse methods have been proposed, the novelty criterion [23] (NC) is a simple way to check whether the newly data is useful. In 2004, Engel et al. proposed the approximate linear correlation criterion [10] (ALD) to solve this problem. Besides, Richard et al. studied another similar method, called the coherence criterion [24]. This paper mainly introduces NC and ALD online sparse methods.

In the novelty criterion, online sparsification starts from an empty set and gradually adds samples to the central set of the dictionary according to the judgment. Assuming the current dictionary is:

$$\mathbf{C} = \{\mathbf{c}_j\}_{j=1}^{m_i} \tag{15}$$

where $\mathbf{c}_j$ is the center of time j, and m_i is the count of the set. When a new data pair $\{\mathbf{u}_{i+1}, d_{i+1}\}$ appears, algorithm will decide whether to add $\mathbf{u}_{i+1}$ as a new center to the dictionary. First we calculate the shortest distance from $\mathbf{u}_{i+1}$ to the dictionary.

$$dis = \min_{\mathbf{c}_j \in \mathbf{C}} |\mathbf{u}_{i+1} - \mathbf{c}_j| \tag{16}$$

If the distance is less than the threshold $\delta 1$, then $\mathbf{u}_{i+1}$ will not be added to the dictionary. On the contrary, the algorithm continues to compare the calculated prediction error e_{i+1} with the threshold $\delta 2$. Only if e_{i+1} is greater than the threshold $\delta 2$, $\mathbf{u}_{i+1}$ will be added to the dictionary as a new center. The approximate linear correlation criterion can be defined as:

$$\delta_i = \left\| \sum_{n=1}^{i-1} \alpha_n \varphi(\mathbf{x}_n) - \varphi(\mathbf{x}_i) \right\|^2 > \nu \tag{17}$$

In the ALD criterion, α represents the coefficient vector, and ν represents the threshold. When the input data arrives, the ALD criterion will calculate the linear dependence between the input data and the dictionary data. If the value is greater than the preset threshold, the input data will be added to the dictionary, otherwise the input will be removed. Similar to the NC criterion, through the comparison of the dictionary and the input data, the ALD criterion will perform sequential sparsification in order to reduce the size of the kernel matrix.

3.2 Sliding Window Method

In order to limit the kernel matrix size, Van Vaerenbergh et al. applied the sliding window method to the KRLS algorithm [15]. In this method, the window size M is fixed, the

observation matrix $\boldsymbol{\Phi}_i = [\boldsymbol{\varphi}(x_{i-\mathrm{M}+1}), \ldots, \boldsymbol{\varphi}(x_i)]$, a kernel matrix expressed as:

$$\mathbf{K}_i = \boldsymbol{\Phi}_i \boldsymbol{\Phi}_i^{\boldsymbol{T}} + c\mathbf{I} = \begin{bmatrix} \mathbf{K}_{i-1} & \boldsymbol{\kappa}_{i-1}(\mathbf{x}_i) \\ \boldsymbol{\kappa}_{i-1}(\mathbf{x}_i)^T & \kappa_{ii} + c \end{bmatrix} \tag{18}$$

where $\boldsymbol{\kappa}_{i-1}(\mathbf{x}_i) = [\kappa(\mathbf{x}_{i-M+1}, \mathbf{x}_i), \ldots, \kappa(\mathbf{x}_{i-1}, \mathbf{x}_i)]^T$, $\kappa_{ii} = \kappa(\mathbf{x}_i, \mathbf{x}_i)$, c is the regularization factor. In order to keep the size of kernel matrix unchanged, the kernel matrix uses the new sample data to add new rows and new columns (19)–(20).

$$\mathbf{K} = \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{b^T} & d \end{bmatrix}, \ \mathbf{K}^{-1} = \begin{bmatrix} \mathbf{E} & \mathbf{f} \\ \mathbf{f^T} & g \end{bmatrix} \quad \Rightarrow \begin{cases} \mathbf{AE} + \mathbf{bf}^T = \mathbf{I} \\ \mathbf{Af} + \mathbf{b}g = 0 \\ dg + \mathbf{b}^T\mathbf{f} = 1 \end{cases} \tag{19}$$

$$\mathbf{K}^{-1} = \begin{bmatrix} \mathbf{A}^{-1}(\boldsymbol{I} + \mathbf{bb}^T\mathbf{A}^{-1H}g & -\mathbf{A}^{-1}\mathbf{b}g \\ -(\mathbf{A}^{-1}\mathbf{b})^T g & g \end{bmatrix} \tag{20}$$

where $\mathbf{A}$ is a kernel matrix before expansion, which is non-singular, $g = (d - \mathbf{b}^{\mathrm{T}}\mathbf{A}^{-1}\mathbf{b})^{-1}$. After expansion, the kernel matrix needs to compress the expanded kernel matrix, and the oldest rows and columns are removed by (21)–(22).

$$\mathbf{K}=\begin{bmatrix} a & \mathbf{b^T} \\ \mathbf{b} & \mathbf{D} \end{bmatrix}, \ \mathbf{K}^{-1}=\begin{bmatrix} e & \mathbf{f}^T \\ \mathbf{f} & \mathbf{G} \end{bmatrix} \ \Rightarrow \begin{cases} \mathbf{b}e + \mathbf{Df} = 0 \\ \mathbf{bf}^T + \mathbf{DG} = \mathbf{I} \end{cases} \tag{21}$$

$$\mathbf{D}^{-1} = \mathbf{G} - \mathbf{ff}^T/e \tag{22}$$

The above equations can make the size of the kernel matrix fixed. The Fig. 2 is the principle of the sliding window.

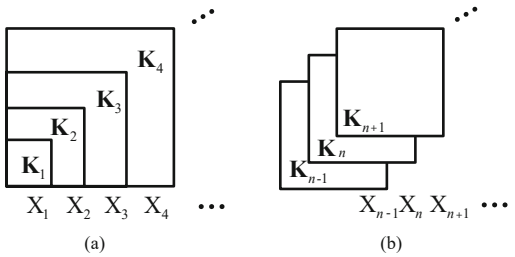

Fig. 2. (a) Kernel matrix $\mathbf{K}_i$ with growing size. (b) Kernel matrix $\mathbf{K}_i$ with a fixed size.

In summary, the advantage of the SW-KRLS method is that it can track time variation without additional computational complexity.

4 Improved Kernel Recursive Least Square Algorithm

In this section, we will introduce proposed algorithm. We begin with some relevant optimizations and introduce the improved algorithm in the end.

4.1 Sliding Window and Novelty Criterion

The SW-KRLS algorithm has low computational complexity and good tracking ability, but it lacks of judgment on the input data and directly adds the input data to the kernel matrix. Furthermore, novelty criterion can eliminate less relevant data in the input, but the size of the dictionary will increase with the number of training data. For the above problems, we use the sliding window method to limit the size of dictionary. Meanwhile, this paper applies the improved novelty criterion to the SW-KRLS algorithm, and proposes the sliding window sparse kernel recursive least squares algorithm (SWS-KRLS).

First, the NC will determine whether to add input data to the sliding window.

$$dis = \min_{\mathbf{c}_j \in \mathbf{C}} |\mathbf{u}_{i+1} - \mathbf{c}_j| > \delta 1 \tag{23}$$

$$e_i = d_i - f(\mathbf{u}_*) = d_i - \sum_{j=1}^{i} \mathbf{a}_j \kappa(\mathbf{u}_j, \mathbf{u}_*) > \delta 2 \tag{24}$$

- If the above equations are both satisfied, it means that the input has a greater influence on the algorithm, so input can be added to the sliding window and dictionary.
- If any of the above equations is not satisfied, it means that the new input has little effect on the algorithm. Therefore, the system will not add the input to the sliding window, and the algorithm error is equal to the previous time error.

Table 1. Sliding window sparse kernel recursive least squares algorithm (SWS-KRLS).

Initialization: sliding window size M, threshold δ1, δ2, regularization coefficient c, $\mathbf{K}_0=(1+c)\mathbf{I}$.
for j=2,3,… **do input:** $\{\mathbf{u}_j, d_j\}$
if dis>δ1, **and** e>δ2, **then**
add the input to the dictionary $\mathbf{C}=\{\boldsymbol{c}_j\}_{j=1}^{mi}$.
else
$e_n=e_{n-1}$, **start** the next iteration $\{\mathbf{u}_{j+1}, d_{j+1}\}$.
if dictionary size is greater than M
dictionary expressed as $\mathbf{C}=\{\boldsymbol{c}_{m_{i-L+1}}, \boldsymbol{c}_{m_{i-L}}, \ldots, \boldsymbol{c}_{m_{i-L+1}}\}$.
calculate the kernel matrix according to Eq. (18) -Eq. (22).
calculate the coefficient vector according to Eq. (14).
calculate the system error according to Eq. (24).
end for

Contrary to the standard NC, we apply sliding window method to the dictionary. When the dictionary size is greater than M, the sliding window delete the *i*-M th data in the dictionary to keep the dictionary size unchanged. Assuming that the size of the fixed novelty criterion dictionary is M, the definition of dictionary **C** is as follows:

$$\mathbf{C} = \{\mathbf{c}_{i-M+1}, \mathbf{c}_{i-M}, \ldots, \mathbf{c}_i\} \tag{25}$$

We use the dictionary **C** to calculate the kernel matrix, on the one hand, the improved algorithm has a better steady-state error because deleting these data can increase the proportion of the closer data, on the other hand, this dictionary can decrease memory usage. The process of the algorithm is shown in Table 1.

4.2 Variable Sliding Window Method

Using a sliding window to limit the size of the kernel matrix is one of the effective sparse methods. However, if a fixed size window is used, it is difficult to achieve good parameter tracking when change occurs. When the system changes, the window size will affect tracking ability. Therefore, adaptive adjustment of the window size can better track system change. Julian [25] first applied the variable sliding window method to KRLS, and the method achieved good results. This paper improves this method by adding the Mechanisms for window size adjustment and the change detection mechanisms.

Mechanisms for Window Size Adjustment
Suppose the time required to upsize is U_m and the time required to downsize is D_m:

$$D_m = m^2 + m + O(1) \tag{26}$$

$$U_m = 5m^2 + 2mT_\kappa + 3m + O(1) \tag{27}$$

where m is the kernel matrix size, T_κ is the calculation cost of the kernel function. The total calculation time can be calculated as:

$$T_m = 6m^2 + 2mT_\kappa + 4m + O(1) \tag{28}$$

If the size of the kernel matrix is smaller than M, the total calculation time T_m will less than the allowable calculation time. Thus, it is this "residual" computation time that the algorithm can adjust the size of the kernel matrix online. Suppose the size of the kernel matrix is m, where $1 < m < M$. When the size of kernel matrix is upsized, the up range of the kernel matrix sizes is:

$$R_m^U = \left\{ \overline{m} \le M : \sum_{i=m}^{\overline{m}} U_i \le T_M \right\} \tag{29}$$

When the kernel matrix is downsized, enough time must be reserved to upsize. Therefore, the donsizing range of the kernel matrix size is:

$$R_m^D = \left\{ \overline{m} \ge 1 : \sum_{i=\overline{m}}^{m} D_i + U_{\overline{m}} \le T_M \right\} \tag{30}$$

In general, when the kernel matrix scale is m, the change range can be expressed as:

$$R_m = \left\{ R_m^U \cup R_m^D \cup R_1^U \right\} \tag{31}$$

where *RU* 1 is the discarded kernel matrix data. The kernel matrix can be upsized and downsized within this range. In addition, in order to improve the parameter tracking performance of the algorithm, this paper proposes a method of adjusting window size. The implementation is shown in Fig. 3.

In stage I, when no change is detected, the SW-KRLS algorithm uses a fixed window size M to update; in stage II, when the system parameter changes are detected, the window size can quickly decrease. By using a smaller window size, the algorithm can track parameter changes more sensitively: in phases III and IV, the window size is gradually restored until it is extended to the maximum allowable size L, so as to obtain higher convergence accuracy.

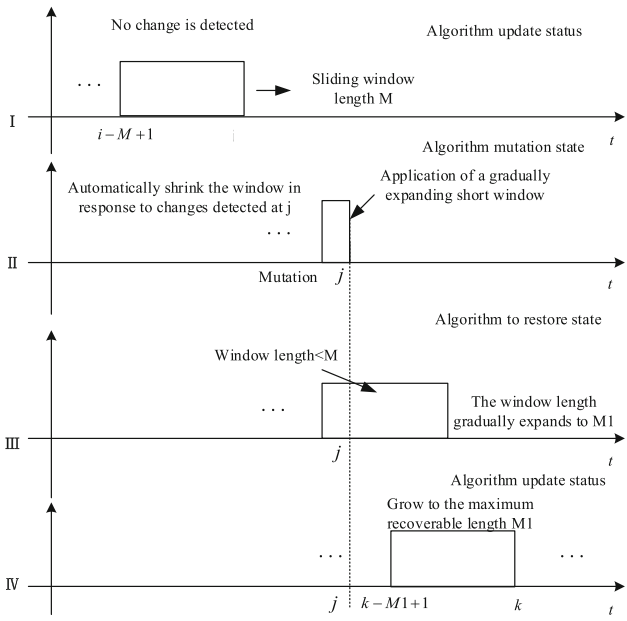

Fig. 3. Method of adjusting window size.

Change Detection Mechanisms

To adjust the window size according to system changes, the algorithms need to use change detection mechanisms, Literature [26] introduced several different methods of detecting parameter changes in adaptive filtering algorithms, which can be divided into parameter detection methods and error detection methods. However, because the parameter ω in the KRLS algorithm is replaced by α, the parameter detection method cannot be used. Therefore, this paper uses error detection to detect system change. The mean square error of adjacent time is defined as Δe:

$$\Delta e_i = e_i - e_{i-1} \tag{32}$$

In order to reduce the false alarm rate, we use the synchronous superposition averaging algorithm to update the system error difference, and this method will be verified in

Sect. 6. Suppose that the window size of the averaging algorithm is L, then the change detection function is defined as:

$$k_i = \frac{1}{L} \sum_{j=i-L+1}^{i} \Delta e_j \tag{33}$$

As defined in [26], the change threshold $\theta = 3\sigma$, where σ is the standard deviation of the background noise. When $k_i > \theta$, the algorithm judges that the system has changed. The process of variable sliding window kernel recursive least squares algorithm is shown in Table 2.

Table 2. Variable sliding window kernel recursive least squares algorithm (VSW-KRLS).

Initialization: sliding window size M, threshold θ, regularization coefficient c, $\mathbf{K}_0$=(1+c)**I**.
for j=2,3,…**do input:** {$\mathbf{u}_j$, d_j}
calculate the error k_i according to Eq. (32) and Eq. (33).
if k_i <θ **then**
calculate the kernel matrix according to Eq. (18) –Eq. (22).
calculate the coefficient vector according to Eq. (14).
else
calculate the maximum changeable window size M1 .
calculate the kernel matrix according to Eq. (18)-Eq. (20).
when the size of the kernel matrix is restored to M1
calculate the kernel matrix according to Eq. (18)-Eq. (22).
end for

4.3 Improved Kernel Recursive Least Square Method

Through above analysis, novelty criterion, variable sliding window method combined with sliding window kernel recursive least squares algorithm to form our improved KRLS algorithm, which is called the sparse variable sliding window kernel recursive least square algorithm (VSWS-KRLS).

The proposed algorithm needs to judge whether to add the input to the dictionary according to the NC. Only if the distance and error are both greater than the preset threshold, the input data will be added to the dictionary and participate in the calculation of the kernel matrix. Therefore, this method can use more useful information to update the algorithm with low computational complexity.

In addition, we introduced a variable sliding window method to improve the ability to track time-varying characteristics. Because it can solve the problem that KRLS is not sensitive to outliers or time-varying characteristic environmental changes. The algorithm flow chart is shown in Fig. 4.

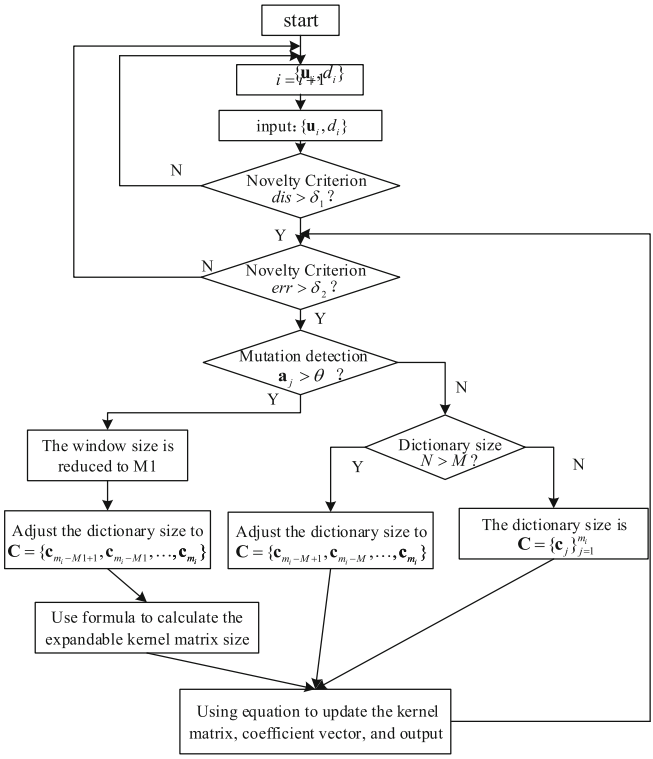

Fig. 4. The specific process of the VSWS-KRLS algorithm.

Table 3. Computational complexity of different algorithms in a single iteration.

Algorithm	Computational complexity	
ALD-KRLS	$\boldsymbol{a}_i$	$O(l^2)$ $(l \leq i)$
	Sparse method (ALD)	$O(l^2)$ $(l \leq i)$
	$\mathbf{P}_i$	$O(l^2)$ $(l \leq i)$
FB-KRLS	$\boldsymbol{a}_i$	$O(L^2)$ $(L < i)$
	$\mathbf{P}_i$	$O(L^2)$ $(L < i)$
EX-KRLS	$\boldsymbol{a}_i$	$O(i^2)$
	$\mathbf{P}_i$	$O(i^2)$
SW-KRLS	$\boldsymbol{a}_i$	$O(L^2)$
	$\mathbf{P}_i$	$O(L^2)$
VSWS-KRLS	$\boldsymbol{a}_i$	$O(j^2)$ $(j \leq L)$
	Sparse method (NC)	$O(j^2)$ $(j \leq L)$
	Dictionary $\mathbf{D}_i$	$O(j^2)$ $(j \leq L)$

4.4 Computational Complexity Analysis

In order to prove that the proposed VSWS-KRLS algorithm has lower computational complexity, this paper compares this algorithm with common KRLS algorithms (KRLS-ALD, FB-KRLS, EX-KRLS, SW-KRLS). This paper uses the weight coefficient vector $\boldsymbol{\alpha}_i$, the autocorrelation matrix $\mathbf{P}_i$ and the sparse method to compare the computational complexity. The results are shown in Table 3.

Suppose that at the i-th moment, the input size is i, the dictionary size is l, and the size of sliding window is L. For ALD-KRLS, the computational complexity of sparse method ALD, weight coefficient vector and autocorrelation matrix are both $O(l^2)$. In EXKRLS, the complexity of the weight coefficient vector and the autocorrelation matrix are $O(i^2)$. Both FB-KRLS and SW-KRLS limit the growth of the kernel matrix by setting a fixed size of window, so the computational complexity of their weight coefficient vector and autocorrelation matrix are both $O(L^2)$. In VSWS-KRLS, two sparse methods (NC and sliding window) reduce the computational complexity. And the computational complexity of the weight coefficient vector, sparse method and dictionary are all $O(j^2)$ (j is the size of the NC dictionary, and the dictionary size is controlled by a variable sliding window, so $j \leq$ L). Therefore, compared with other algorithms, the SWS-KRLS algorithm has the smallest computational complexity in a single iteration, and as the iteration progresses, the advantage of computational complexity will become more obvious.

5 Simulation

In this section, in order to prove the effectiveness of the algorithm, we apply the algorithm to a nonlinear time-varying system for system identification. The linear channel coefficients will change at a given moment to compare the tracking ability of the algorithm. This paper uses four channels for simulation: In the first part of the simulation, the linear channel is $\mathrm{H}_1(\mathrm{z}) = 1 + 0.8362\mathrm{z}^{-1} - 0.7732\mathrm{z}^{-2} - 0.4484\mathrm{z}^{-3}$, after receiving 500 data, it is changed into $\mathrm{H}_2(\mathrm{z}) = 1 - 0.8045\mathrm{z}^{-1} + 0.9962\mathrm{z}^{-2} + 0.4678\mathrm{z}^{-3}$. This paper nonlinear Wiener system as the nonlinear system, and the model is shown in Fig. 5.

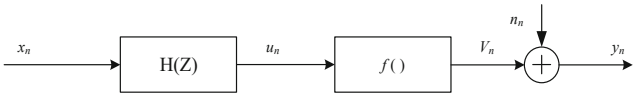

Fig. 5. Nonlinear Wiener system.

A binary signal x_n is sent through this channel, then a nonlinear function $v = \tanh(x)$ is applied to it, and v_n is the channel output. Finally, 20 dB of additive white Gaussian noise (AWGN) is added. The performance of the model is shown through the MATLAB simulation. In addition, the ALD-KRLS algorithm and the NC-KRLS algorithm are evaluated to show the reason for choosing the novelty criterion as the sparse method. In the experiment, the mean square error (MSE) is selected as the evaluation index of prediction accuracy. If the value of MSE is smaller, it means that the prediction

performance is better. At the same time, this paper also uses training time as an indicator of computational complexity.

This experiment was simulated on windows10 operating system, Intel Core i3-9100 CPU 3.60 GHz, RAM 8.00 GB, and the codes are operated in MATLAB R2016a software, and the experiment results are obtained through 100 Monte Carlo experiments.

5.1 Selection of Sparse Methods

In order to choose a suitable sparse method, we compare different sparse methods, we uniformly select 20 values of $\delta 1$ in the interval of [0.1, 0.3] and 20 values of $\delta 2$ in the interval of [0.05, 0.1] for the NC. And we select 80 values of $\delta 3$ are uniformly selected in the interval of [0.05, 0.3] for the ALD. For each threshold, we use KRLS-NC and KRLS-ALD to train, and record the kernel matrix size and MSE values. Figure 6 shows the MSE values corresponding to two different sparse methods in different kernel matrix size.

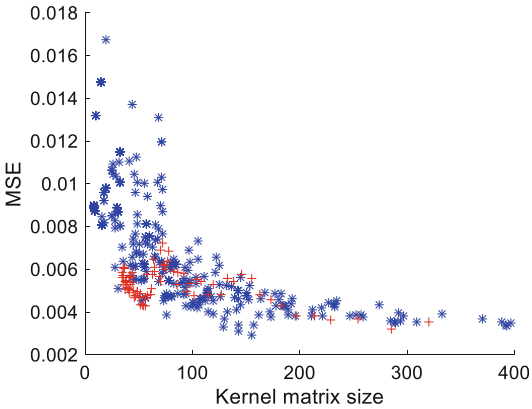

Fig. 6. Comparison of different sparse methods NC and ALD of KRLS. The "*" point comes from KRLS-NC, the " + " point comes from KRLS-ALD.

The results show that when the kernel matrix size is less than 70, the performance of ALD criterion is better than NC, and NC can actually perform better than ALD in other places. Because the sliding window size in the SW-KRLS algorithm is concentrated between 100–200, so this paper applies the NC to the SW-KRLS algorithm to solve the sparse problem.

5.2 Sliding Window Sparse Kernel Recursive Least Squares Algorithm

To further test the performance of SWS-KRLS algorithm, this section shows the influence of algorithm performance with different NC thresholds. The size of the sliding window is 150, and the threshold $\delta 1 = [0.2, 0.5, 0.7]$. The simulation result is shown in Fig. 7. It is observed that the steady-state error is improved by adding novel criterion to the

SW-KRLS algorithm. For thresholds, the greater the value of δ1 is, the better the steady-state error of the system. But the sparse methods will destroy the complex relationship between the data, it will reduce the convergence speed of the algorithm.

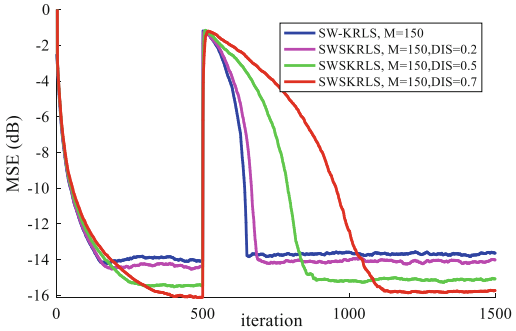

Fig. 7. MSE learning curves of SW-KRLS and SWS-KRLS with different NC thresholds.

5.3 Improved Algorithm

This paper introduces a change detection mechanism in the variable sliding window method. Different from the literature [25], the error difference after simultaneous superposition and averaging can better detect change. In a single experiment, the algorithm mean square error difference and mean square error value after using synchronous superposition averaging algorithm is shown in Fig. 8. In one iteration, the error difference is unstable, so the judgment is also inaccurate. The synchronous superposition average algorithm selects adjacent time to calculate the average value to reduce the influence of instability.

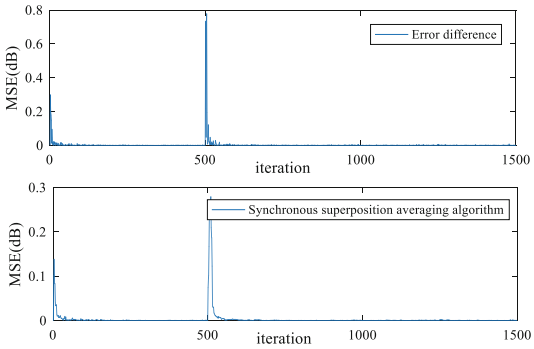

Fig. 8. Algorithm mean square error difference and mean square error value after using synchronous superposition averaging algorithm.

In order to evaluate the impact of different window lengths on the performance of the algorithm, we choose the window length M of the SW-KRLS algorithm to be 50, 100,

and 150. Meanwhile, the variable sliding window method is applied in the algorithm of M = 150. The simulation results are shown in Figs. 9 and 10. Clearly, when the SW-KRLS algorithm undergoes a sudden change, the window size affects the performance of the algorithm. However, the VSW-KRLS algorithm can improve the tracking ability of the algorithm by changing the window size.

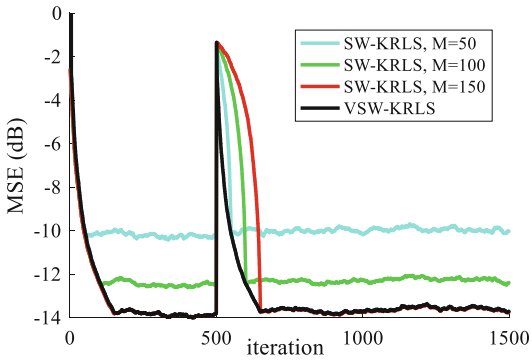

Fig. 9. MSE learning curves of VSW-KRLS and SW-KRLS with different window lengths.

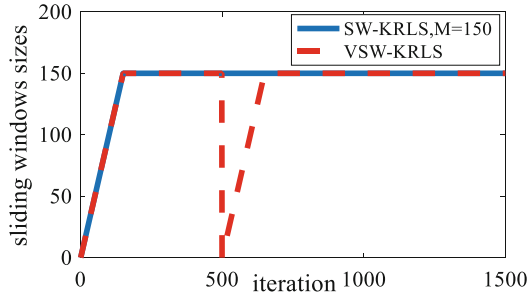

Fig. 10. Changes in the sliding window length of the VSW-KRLS and SW-KRLS.

As shown in Fig. 11, the VSWS-KRLS algorithm is compared with the SW-KRLS algorithm and SWS-KRLS algorithm. The simulation results confirm that the VSWS-KRLS algorithm has not only a fast convergence rate but also a small steady-state misalignment in comparison to SW-KRLS algorithms. Table 4 shows the simulation results of different algorithms in a nonlinear system, where, M represents the size of the sliding window, $\delta 1$, $\delta 2$ and θ represent the NC threshold and change detection threshold, and MSE represents the mean square error value. t represents the system running time of a single iteration. As can be seen, compared with other algorithms, the calculation performance of the VSWS-KRLS algorithm has been greatly improved.

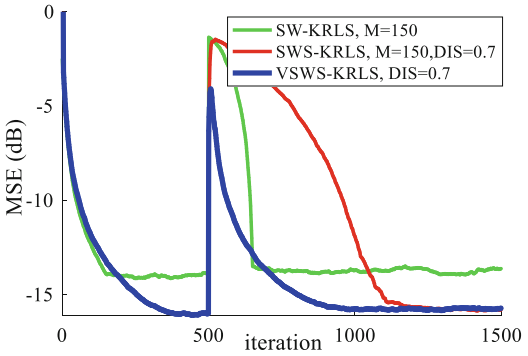

Fig. 11. MSE learning curves of SW-KRLS SWS-KRLS and VSWS-KRLS.

Table 4. Simulation results of different algorithms in nonlinear time-varying systems.

Algorithm	M	$\delta 1$	$\delta 2$	θ	MSE	t
SW-KRLS	100				−12.26 dB	0.42
	150				−13.63 dB	0.59
	200				−14.95 dB	0.73
SWS-KRLS	150	0.2	0.05		−14.21 dB	0.60
	150	0.5	0.05		−15.35 dB	0.45
	150	0.7	0.05		−15.89 dB	0.37
VSW-KRLS	150			3σ	−13.61 dB	0.60
VSWS-KRLS	150	0.7	0.05	3σ	−15.86 dB	0.36

6 Conclusion and Prospect

This paper proposed VSWS-SKRLS algorithm for identification of nonlinear systems. The model used sliding window and NC to make the dictionary and kernel matrix sparse, it could reduce the computational complexity and memory occupancy of algorithm. Moreover, online sparsification could improve prediction accuracy by reducing impact of irrelevant data, but it also influenced the tracking ability of model. To further improve the tracking ability of the model, adaptive mechanisms for sliding window size were employed. Finally, the experimental results showed that the VSWS-SKRLS algorithm had better prediction performance and low computational complexity in identification of nonlinear time-varying systems. Furthermore, our work will focus on selection the other sparse methods in the future, which has certain effects on performance and deserves further studied.

References

1. Haykin, S.: Adaptive Filter Theory, 4th edn. Prentice-Hall, Englewood Cliffs (2001)

2. Wu, C., Wang, X., Guo, Y.: Robust uncertainty control of the simplified Kalman filter for acoustic echo cancelation. Circuits Syst. Signal Process. **35**(12), 4584–4595 (2016)
3. Liu, Y., Mikhael, W.B.: A fast-converging adaptive fir technique for channel equalization. In: 2012 IEEE 55th International Midwest Symposium on Circuits and Systems (MWSCAS), pp. 828–831. IEEE, Boise (2012)
4. Mariappan, S., Rao, G.S.: Enhancing GPS receiver tracking loop performance in multipath environment using an adaptive filter algorithm. Indian J. Sci. Technol. **7**(S7), 156–164 (2014)
5. Widrow, B., Hoff, M.E.: Adaptive switching circuits. In: IRE WESCON Convention Record, vol. 4, pp. 96–104. IRE, New York (1960)
6. Kuh, A.: Adaptive least square kernel algorithms and applications. In: 2002 International Joint Conference on Neural Networks, vol. 3, pp. 2104–2107. IEEE, Honolulu (2002)
7. Williamson, R.C., Kivinen, J., Smola, J.: A online learning with kernels. Signal Process. **100**, 1–12 (2004)
8. Bai, Y., Xiao, J., Long, Y.: Kernel partial least-squares regression. In: International Joint Conference on Neural Networks, IJCNN 2006, Part of the IEEE World Congress on Computational Intelligence, WCCI 2006, pp. 1231–1238. IEEE, Vancouver (2006)
9. Liu, W., Príncipe, J.C., Haykin, S.: Kernel Adaptive Filtering: A Comprehensive Introduction. Wiley, Hoboken (2010)
10. Engel, Y., Mannor, S., Meir, R.: The kernel recursive least-squares algorithm. IEEE Trans. Signal Process. **52**(8), 2275–2285 (2004)
11. Aronszajn, N.: Theory of reproducing kernels. Trans. Am. Math. Soc. **68**(3), 337–404 (1950)
12. Wang, X., Han, M.: Multivariate time series prediction based on multiple kernel extreme learning machine. In: IEEE International Joint Conference on Neural Networks (IJCNN), pp. 198–201. IEEE, Beijing (2014)
13. Han, M., Zhang, S., Xu, M.: Multivariate chaotic time series online prediction based on improved kernel recursive least squares algorithm. IEEE Trans. Cybern. **99**, 1–13 (2018)
14. Zhou, H., Huang, J., Lu, F.: Echo state kernel recursive least squares algorithm for machine condition prediction. Mech. Syst. Signal Process. **111**, 68–86 (2018)
15. Van Vaerenbergh, S., Via, J., Santamaria, I.: A sliding-window kernel RLS algorithm and its application to nonlinear channel identification. In: 2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings (ICASSP 2006), vol. 5, pp. 789–792. IEEE, Toulouse (2006)
16. Van Vaerenbergh, S., Santamaria, I., Liu, W.: Fixed-budget kernel recursive least-squares. In: 2010 IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP 2010), pp.1882–1885. IEEE, Dallas (2010)
17. Chen, B., Zhao, S., Zhu, P.: Quantized kernel recursive least squares algorithm. IEEE Trans. Neural Netw. Learn. Syst. **29**(4), 1484–1491 (2013)
18. Han, M., Ma, J., Kanae, S.: Time series online prediction based on adaptive dynamic adjustment kernel recursive least squares algorithm. In: The 2018 Ninth International Conference on Intelligent Control and Information Processing (ICICIP), pp. 66–72. IEEE, Wanzhou (2018)
19. Han, M., Zhang, S., Xu, M.: Multivariate chaotic time series online prediction based on improved kernel recursive least squares algorithm. IEEE Trans. Cybern. **49**(4), 1160–1172 (2019)
20. Zhong, K., Ma, J., Han, M.: Online prediction of noisy time series: dynamic adaptive sparse kernel recursive least squares from sparse and adaptive tracking perspective. Eng. Appl. Artif. Intell. **91**, 103547 (2020)
21. Guo, J., Chen, H., Chen, S.: Improved Kernel recursive least squares algorithm based online prediction for nonstationary time series. IEEE Signal Process. Lett. **27**, 1365–1369 (2020)
22. Sayed, A.H.: Fundamentals of adaptive filtering. IEEE Control. Syst. **25**(4), 77–79 (2003)
23. Platt, J.: A resource - allocating network for function interpolation. Neural Comput. **3**(2), 213–225 (1991)

24. Richard, C., Bermudez, J.C.M., Honeine, P.: Online prediction of time series data with kernels. IEEE Trans. Signal Process. **57**(3), 1058–1067 (2009)
25. Julian, B.J.: Modifications to the sliding-window kernel RLS algorithm for time-varying nonlinear systems: online resizing of the kernel matrix. In: 2009 IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 3389–3392. IEEE, Taipei (2009)
26. Gustafson, F.: Adaptive Filtering and Change Detection. Wiley, New York (2000)