A New Sparse Kernel RLS Algorithm for Identification of Nonlinear Systems

XINYU GUO¹, SHIFENG OU¹, MENGHUA JIANG¹, YING GAO¹, JINDONG XU² AND ZHUORAN CAI²

¹ School of Optoelectronic Information Science and Technology, Yantai University, Yantai 264005, China
² School of Computer and Control Engineering, Yantai University, Yantai 264005, China

Corresponding author: Shifeng Ou (e-mail: oufeng@ytu.edu.cn)

This work was supported by the Natural Science Foundation of Shandong Province ZR2017MF008 and National Natural Science Foundation of China 62072391.

ABSTRACT The sliding window kernel recursive least squares (SW-KRLS) algorithm has been widely used in system identification because of its simple structure, low computational complexity and high predictive accuracy. However, with the increase of input data, high computational complexity will worsen the algorithm performance, and there are some difficulties in adapting to the system with an abrupt change. In view of this, we propose a variable sliding window sparse kernel recursive least squares (VWS-KRLS) algorithm. In order to obtain a parsimonious kernel matrix with the satisfactory prediction accuracy, the basic pruning technique is applied to the traditional sliding window method. In addition, mechanism for window size adjustment is added to adjust the size of sliding window adaptively according to the system changes. Finally, Novelty criterion (NC), dictionary with sliding window, variable sliding window technique and mutation detection mechanism are combined with KRLS to form our improved KRLS algorithm. The improved algorithm can reduce the computational complexity, improve the convergence performance, and have the capability to better track the system with an abrupt change. System identification experiments are carried out in the wiener nonlinear system to prove the effectiveness of the improved algorithm.

INDEX TERMS Sparsification, variable sliding window method, system identification, kernel recursive least squares algorithm.

I. INTRODUCTION

System identification is a mathematical modeling method that uses known feature information to construct an equivalent model, which is used in many fields, such as signal processing, machine learning and control engineering [1-6]. The development of classic system identification methods has been relatively complete, including impulse response method, spectrum analysis method, least square (LS) method [7,8], maximum likelihood method [9] and deep learning [10,11]. Where, the LS method is a classic and most basic method, and it is also the most widely used method. However, the LS algorithm has certain limitations, so in order to overcome its shortcomings, scholars have proposed many adaptive algorithms. In 1950, Plackeet first proposed the recursive least squares [12] (RLS) algorithm, which has fast convergence and small prediction error. In 1960, Widrow and Hoff proposed the least mean square algorithm [13] (LMS), which is widely used because of its simple calculation and good convergence performance. However, traditional filtering algorithms have limited effectiveness in dealing with nonlinear problems.

In recent years, kernel methods [14] have been widely used to deal with nonlinear problems, such as kernel support vector machines [15,16], kernel principal component analysis [17,18], kernel adaptive filter [19]. As generalization of linear adaptive filters, the kernel recursive least squares [20] (KRLS) algorithm was proposed by Engel in 2004. This algorithm solves the nonlinear inseparable problem 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 [21] condition to calculate the inner product, this process involves nonlinear mapping from low-dimensional feature space to high-dimensional feature space, which can be shown in Figure 1.

After the KRLS algorithm was proposed, scholars proposed the kernel least mean square (KLMS) algorithm and the kernel affine projection algorithm (KAPA). According to the performance comparison results of KRLS, KLMS and KAPA
three types of models in prediction efficiency, prediction accuracy, and convergence speed, as shown in Table 1. From Table 1, it can be seen that compared with KAPA, KRLS has lower prediction efficiency, but it has more advantages in terms of speed and prediction accuracy. 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 [22,23]. However, the computational complexity of KRLS will increase due to recursive calculations, so most studies mainly concentrated on the sparsification of the KRLS algorithms. In 2006, Vaerenbergh et al. proposed a sliding window based kernel recursive least squares (SW-KRLS) algorithm [24], which uses a sliding window to limit the growing kernel matrix. This algorithm obtains lower computational complexity while retaining the advantages of KRLS. In 2010, Vaerenbergh et al. proposed a fixed budget recursive least squares (FB-KRLS) algorithm [25], which uses 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 (QKRLS) algorithm [26], which used a quantized method to reduce the input dimension. In 2018, Han et al. proposed an adaptive dynamic adjustment kernel recursive least squares (ADA-KRLS) algorithm [27], which performs online sparsification by combining fixed budget with dynamic adaptation. The parsimonious kernel recursive least squares (PKRLS) algorithm is proposed in literature [28], a pruning approach that can restrict the network size to a fixed value is applied to the KRLS algorithm, which curbs the network growth and improves learning efficiency. In 2019, Han et al. proposed an adaptive normalized sparse quantized kernel recursive least squares (ANS-QKRLS) algorithm [29], which integrates dynamic adjustment, coherence criterion, approximate linear dependency (ALD) criterion and the QKRLS algorithm for online sparsification. Zhong et al. proposed a dynamic adaptive sparse kernel recursive least squares (DASKRLS) algorithm in 2020 [30]. It uses 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 [31] 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. The literature [32] used Nyström method and k-means sampling to form novel Nyström kernel recursive least squares with k-means sampling (NysKRLS-KM).

In summary, research on the KRLS algorithm in system identification has made good progress [33,34]. However, in face of complex system environment, system identification also faces enormous challenges [35]. In addition, KRLS is based on the mean square error (MSE) criterion, which has some difficulties in accommodating abrupt change. 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 limit the size of dictionary in novelty criterion (NC) by using sliding window. As data size increases, computational complexity of calculating dictionary will raise. Therefore, in order to reduce the storage space of the dictionary, we use sliding window to fix the dictionary size. In addition, this dictionary can replace the KRLS algorithm dictionary to calculate the kernel matrix.
- We reduce kernel matrix size by combining sliding window and NC with KRLS. A lot of irrelevant data in the sliding window will affect the convergence. Therefore, in order to improve calculation efficiency, we add NC, which can raise the threshold of entering the kernel matrix.
- We propose a new variable sliding window technology to enhance the capability of the algorithm to track system changes. This method can adaptively adjust the window size according to the system environment.

The rest of this paper is organized as follows: In Section 2, a brief review of RLS and KRLS is presented. Section 3 introduces several common sparse methods. In Section 4, the VSWS-KRLS algorithm is proposed. In Section 5, experiments in system identification is conducted to evaluate the performance of the proposed algorithm. Finally, conclusions and prospects are drawn in Section 6.

II. KERNEL RECURSIVE LEAST SQUARES ALGORITHM

A. RECURSIVE LEAST SQUARES ALGORITHM

The 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 \( \omega_{i-1} \) by minimizing the cost function. The cost function is:

\[
J(\omega_i) = \min_{\omega_i} \sum_{j=1}^{i-1} \left| d_j - u_j^T \omega_i \right|^2
\]

(1)

The function is computed as:

\[
\omega_{i-1} = (U_{i-1}^T U_{i-1})^{-1} U_{i-1}^T d_i
\]

(2)

where \( U_{i-1} = [u_1, u_2, \ldots, u_{i-1}] \). \( U \) is the input vector at time \( j \), \( d_i = [d_1, d_2, \ldots, d_i]^T \), \( d_j \) is the expected response at time \( j \). By introducing the matrix inversion lemma (Woodbury identity), the RLS algorithm can be updated:

\[
r_i = 1 + u_i^T P_{i-1} u_i
\]

(3)

\[
k_i = P_{i-1} u_i / r_i
\]

(4)

\[
\omega_i = \omega_{i-1} + k_i e_i
\]

(5)

\[
P_i = P_{i-1} - k_i k_i^T r_i
\]

(6)

The RLS algorithm uses the inverse of the correlation matrix to whiten the input data, which improves the convergence performance of the filter. Figure 2 is the RLS algorithm for system identification model.

### B. KERNEL RECURSIVE LEAST SQUARES

The KRLS algorithm introduces the kernel function based on the RLS algorithm, which projects the input data to the Hilbert space (RKHS). The biggest advantage of this algorithm is that the nonlinear relationship can be solved without knowing the specific mapping form of input. First, we suppose that \( X \) represents the original space, \( X = [x_1, x_2, \ldots, x_n] \in \mathbb{R}^N \), and the mapping \( \phi \) is expressed as:

\[
\phi : X \rightarrow \mathcal{H}, \; x \rightarrow \phi(x)
\]

(7)

where, \( \mathcal{H} \) represents the Hilbert space, \( \phi(x) \) represents the projection of \( x \) in the feature space. In the kernel function, the nonlinear mapping is defined by:

\[
\kappa(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle
\]

(8)

where \( \langle , \rangle \) represents the inner product operation. The kernel function first maps the input data to a high-dimensional space, where it then performs linear operations. As the kernel function used in this paper, the Gaussian kernel function is expressed as:

\[
\kappa(x_i, x_j) = \exp(-\|x_i - x_j\|^2 / 2\sigma^2)
\]

(9)

where \( \sigma \) represents the Gaussian kernel parameter. In KRLS algorithm, given the expected sequence \( \{d_1, d_2, \ldots\} \) and the input sequence \( \{\phi_1, \phi_2, \ldots\} \), we have the cost function:

\[
J(\omega_i) = \min_{\omega_i} \sum_{j=1}^{i} \left| d_j - \phi_j^T \omega_i \right|^2
\]

(10)

where \( d_i \) and \( \phi_i \) represent the expected sequence and the input sequence. We assume that \( \Phi_i = [\phi(x_1), \ldots, \phi(x_i)] \), and a \( n \times n \) dimensional kernel matrix is defined by:

\[
K_i = \Phi_i^T \Phi_i
\]

(11)

When \( \omega_i = \Phi_i a_i \), the cost function of the KRLS algorithm at time \( i \) can be obtained:

\[
J = \min_{a_i} \left\| d_i - K_i a_i \right\|^2
\]

(12)

where \( a_i = [a_1, a_2, \ldots, a_i]^T \) is the weight vector and \( d_i = [d_1, d_2, \ldots, d_i]^T \) is the expected output vector. The estimated value of \( a_i \) is as follows:

\[
\hat{a}_i = \hat{K}_i^{-1} d_i
\]

(13)

The kernel matrix will be adapted as described below:

\[
\hat{K}_i = \hat{K}_i + \lambda I
\]

(14)

Here, \( \lambda \) is a positive number called the regularization parameter, which will be implemented to prevent overfitting.

Unlike the RLS algorithm, the KRLS algorithm focuses on updating the kernel matrix. With the widespread application of KRLS algorithm in various fields [36,37], its deficiencies have gradually emerged. 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.

### III. SPARSIFICATION

The previous part introduces and compares the RLS algorithm and the KRLS algorithm. This part we will introduce some common sparse methods.

#### A. NOVELTY CRITERION AND APPROXIMATE LINEAR DEPENDENCY

Pruning techniques can reduce redundant information and improve computational performance [38]. 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 NC [39] is a simple sparse method to check whether the latest data is useful. In 2004, Engel et al. proposed the ALD criterion to solve this problem for KRLS algorithm [20]. Besides, Richard et al. studied another similar method, called the coherence criterion [40]. The above methods can effectively optimize the KRLS algorithm from online sparsification.

This paper mainly introduces NC and ALD online sparse methods. In the NC, 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:

\[ C_i = \{c_j\}_{j=1}^{m_i} \]  

where \( c_j \) is the center at iteration \( j \), and \( m_i \) is the count of the set. When a new data pair \( \{u_{i+1}, d_{i+1}\} \) appears, algorithm will decide whether to add \( u_{i+1} \) as a new center to the dictionary.

\[ dis = \min_{c_j} |u_{i+1} - c_j| \]  

If the distance is less than the threshold \( \delta_1 \), then \( u_{i+1} \) 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 \( \delta_2 \), \( u_{i+1} \) will be added to the dictionary as a new center. In fact, in NC, increasing threshold \( \delta_1 \) will decrease the kernel matrix size, but the convergence performance of the algorithm may deteriorate. Similarly, increasing \( \delta_2 \) will decrease kernel matrix size, but the performance may deteriorate. Therefore, in this sparse method, the appropriate threshold can be selected according to different requirements.

ALD criterion is an approximate linear correlation sparse method, the ALD criterion can be defined as:

\[ \delta_1 = \sqrt{\sum_{n=1}^{M} \alpha_n \phi(x_n) - \phi(x_i)}^2 > \nu \]  

In the ALD criterion, \( \alpha \) represents the coefficient vector, and \( \nu \) 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.

In KRLS-NC algorithms and KRLS-ALD algorithms, the weight vector \( \omega_i \) needs to be updated:

\[ \omega_i = \Phi_i (K_i + \lambda I)^{-1} d_i \]  

This process is calculated using matrix inversion lemma. Finally, the weight is substituted into the output model to obtain the output results. At iteration \( i \), given a test input \( u_i \), the output of the system is:

\[ f(u_i) = \sum_{j=1}^{l} a_j \kappa(u_j, u_i) \]  

(19)

Similar to the NC, the selection of threshold will also affect the performance of the algorithm.

B. SLIDING WINDOW METHOD

In order to limit the size of kernel matrix, Vaerenbergh et al. applied the sliding window method to the KRLS algorithm [24]. The SW-KRLS algorithm combines sliding window and traditional L2 norm regularization, which requires a fixed and unrestricted kernel matrix dimension. Meanwhile, L2 norm regularization improves the generalization ability of the model.

This algorithm shows good tracking performance in sudden changes and is mostly used in non-memory nonlinear systems. In this method, we suppose the size of window is \( M \), therefore the observation matrix is \( \Phi_i = [\phi(x_{i-M+1}), ..., \phi(x_i)] \), and a kernel matrix can be expressed as:

\[ K_i = \Phi_i \Phi_i^T + \epsilon I = \begin{bmatrix} \kappa_{1,1}(x_i) \\ \kappa_{2,1}(x_i) \\ \vdots \\ \kappa_{M,1}(x_i) \end{bmatrix} \]  

(20)

where, \( \kappa_{i,1}(x_i) = [\kappa(x_i, x_{M+1}), ..., \kappa(x_i, x_i)]^T \), \( \kappa_{i,i}(x_i, x_i) \), \( \epsilon \) 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 by (21)-(23).

\[ K = \begin{bmatrix} A & b \\ b^T & d \end{bmatrix}, \quad K^{-1} = \begin{bmatrix} E & f \\ f^T & g \end{bmatrix} \Rightarrow \begin{bmatrix} AE + bf^T = I \\ Af + bg = 0 \end{bmatrix} \]  

(21)

\[ K^{-1} = \begin{bmatrix} A^{-1} (I + b^T A^{-1} b)^{-1} g \\ -A^{-1} b g \end{bmatrix} \]  

(22)

\[ g = (d - b^T A^{-1} b)^{-1} \]  

(23)

where, \( A \) is a kernel matrix before expansion, which is nonsingular. After expansion, the kernel matrix needs to compress the expanded kernel matrix, and the oldest rows and columns are removed by (24)-(25).

\[ K = \begin{bmatrix} a & b^T \\ b & D \end{bmatrix}, \quad K^{-1} = \begin{bmatrix} e & f^T \\ f & G \end{bmatrix} \Rightarrow \begin{bmatrix} be + Df = 0 \\ bf^T + DG = I \end{bmatrix} \]  

(24)

\[ D^{-1} = G - ff^T / e \]  

(25)

The above equations can make the size of the kernel matrix fixed, and Figure 3 is the principle of the sliding window.
In summary, the SW-KRLS algorithm uses regularization to solve the overfitting problem, and combines sliding window and matrix inversion lemma to solve the limitation of computational complexity, which not only saves memory space, simplifies calculation, but also improves prediction accuracy.

IV. IMPROVED KERNEL RECURSIVE LEAST SQUARES ALGORITHM

After introducing the sparse method, in this section, we will first optimize the NC and sliding window method from online sparsification. And then, for the fixed-size “sliding window” technology, a method that can resize the kernel matrix according to the system environment is proposed. Finally, we combine the three optimization methods and propose the improved KRLS algorithm.

A. SLIDING WINDOW AND NOVELTY CRITERION

The SW-KRLS algorithm has low computational complexity and good tracking ability, but it directly adds the input data to the kernel matrix and lacks of judgment on the input data, so the data with small correlation in the sliding window may affect the convergence performance of the algorithm. In general, the algorithm remains of finding a sparse way of judging input data. Therefore, by applying the improved NC to the SW-KRLS algorithm, we propose the sliding window sparse kernel recursive least squares (SWS-KRLS) algorithm.

First and foremost, in the improved algorithm, the NC will determine whether to add input data to the sliding window at iteration $i$.

\[
dis = \min_{i, c} \left| \mathbf{u}_{i+1} \cdot \mathbf{c} \right| > \delta_1
\]  

(26)

Correspondingly, given a test input $\mathbf{u}_i$, the error of the system is:

\[
e_i = d_i - f(\mathbf{u}_i) = d_i - \sum_{j=1}^{i} a_j \kappa(\mathbf{u}_j, \mathbf{u}_i) > \delta_2
\]  

(27)

The following two situations will be judged by NC:

- If and only if the above equations are both satisfied, it means that the input has a greater influence on the algorithm, so input is allowed to enter dictionary.
- Otherwise, 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 error.

Using this sparse method, the proportion of useful input in the sliding window can be increased. Meanwhile, with the increasing of iterations, the sparse method can also reduce the computational complexity of SW-KRLS algorithm.

In addition, although NC can eliminate less relevant data in the input, but the dictionary size in NC increases linearly with the number of training data. Therefore, we use the sliding window method to limit the size of dictionary. If the dictionary size is greater than $M$, the sliding window will delete the $i$-th data in the dictionary to keep the dictionary size unchanged. Figure 4 shows the principle of dictionary update. Supposing that the size of the fixed NC dictionary is $M$, the definition of dictionary $C$ is as follow:

\[
C_i = \{c_{i-M+1}, c_{i-M}, \ldots, c_i\}
\]  

(28)

Using this method has two advantages, on the one hand, the dictionary with sliding windows is used to calculate the kernel matrix has a better steady-state error because sliding windows delete these data can increase the proportion of the closer data, on the other hand, this dictionary can decrease memory usage.

Finally, we apply these two methods to SW-KRLS algorithm and propose the SWS-KRLS algorithm, the process of the algorithm is shown in Algorithm 1.

B. Variable Sliding Window Method

In a sense, using the 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. If the system changes, the larger the window, the worse the tracking performance of

Algorithm 1 SWS-KRLS

**Initialization:** sliding window size $M$, threshold $\delta_1$, $\delta_2$, regularization coefficient $c$, $K_0=(1+c)I$.

for $j=2,3,\ldots$ do input: $\{u_j, d_j\}$

if $dis > \delta_1$ and $c > \delta_2$, then

add the input to the dictionary $C_i=\{c_j\}_{j=1}^{m_i}$.

else

$e_0=\epsilon_{n-1}$, start the next iteration $\{u_{i+1}, d_{i+1}\}$

if dictionary size is greater than $M$

dictionary expressed as $C=\{c_{m_{i-L+1}}, c_{m_{i-L}}, \ldots, c_{m_{i-L+1}}\}$.

calculate the kernel matrix according to (18)-(22).

calculate the coefficient vector according to (14).

calculate the system error according to (24).

end for
the algorithm, on the contrary, the small the window, the worse the convergence performance of the algorithm. The trade-off relationship between the tracking performance and the convergence performance is governed by the size of sliding window. Based on these advantages and arguments, our strategy is clear: adaptive adjustment of the window size can better balance the relationship between them. Julian [41] first applied the variable sliding window method to KRLS, and the method achieved good results. In this paper, we improve this method by adding the mechanisms for window size adjustment and the change detection mechanisms.

First of all, in the mechanisms for window size adjustment, for the algorithm to operate online, it is necessary to calculate the time required to upsize and downsize the kernel matrices. Suppose the time required to upsize is \( U_m \) and the time required to downsize is \( D_m \):

\[
D_m = m^2 + m + O(1)
\]

(29)

where \( m \) is the kernel matrix size, suppose the time required to downsize is \( U_m \):

\[
U_m = 5m^2 + 2mT_x + 3m + O(1)
\]

(30)

where \( T_x \) is the calculation cost of the kernel function. Thus, the total calculation time required to upsize and downsize the kernel matrix can be calculated as:

\[
T_m = 6m^2 + 2mT_x + 4m + O(1)
\]

(31)

In the course of upsizing and downsizing, we find that if the size of the kernel matrix is smaller than \( M \), the total calculation time \( T_m \) will be 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:

\[
R_m^U = \{ \bar{m} \leq M : \sum_{i=0}^{\bar{m}} U_i \leq T_M \}
\]

(32)

We can not only increase the kernel matrix, but also reduce the size of the kernel matrix. However, enough time must be reserved to upsize, the downsizing range of the kernel matrix size is:

\[
R_m^D = \{ \bar{m} \geq 1 : \sum_{i=1}^{\bar{m}} D_i + U_m \leq T_M \}
\]

(33)

In general, when the kernel matrix size is \( m \), the resizing range \( R_m \) can be expressed as:

\[
R_m = \left( R_m^U \cup R_m^D \cup R_m^{U'} \right)
\]

(34)

where, \( R_m^U \) represent the option of discarding the regularized kernel matrix, which is an operation that takes \( O(1) \) time. The kernel matrix can be upsized and downsized within this range.

Subsequently, to adjust the window size according to system changes, the algorithms need to use change detection mechanisms. Literature [42] 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 filter coefficient \( \omega \) in the KRLS algorithm is replaced by \( \alpha \), 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 by:

\[
\Delta \epsilon_i = \epsilon_i - \epsilon_{i-1}
\]

(35)

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 Section V. Assuming that the window size of the averaging algorithm is \( L \), then the change detection function is defined by:
After getting $k_i$, we need to judge the state of the system.

\[
\begin{align*}
    k_i &> \theta: \text{change occurs} \\
    k_i &< \theta: \text{no change occurred}
\end{align*}
\]  

(37)

where $\theta$ is the threshold for detecting changes, the selection of $\theta$ represents certain degree of trade-off between the probability of false negatives and the probability of false positives. Therefore, the value of threshold is important for the performance of the system. The change threshold $\theta = 3\sigma$, where $\sigma$ is the standard deviation of the background noise. Proceeding in a fashion similar to Julian, we adopt the threshold $\theta = 3\sigma$, so that if $k_i > \theta$ the algorithm judges that the system has changed. 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 Figure 5.

In stage I, in the initial stage of the algorithm, the sliding window size gradually increases; in stage II, when no change is detected, the SW-KRLS algorithm updates with the fixed window size $M$; in stage III, when the system parameter changes are detected, the window size can be

quickly reduced. By using a smaller window size, the algorithm can track parameter changes more sensitively; in the stage IV, the window size gradually increases until it increases to the maximum allowable size $L$; finally, in stage V, the algorithm is updated with a larger window size $L$, so that the algorithm obtains a higher convergence accuracy. The process of variable sliding window kernel recursive least squares (VSW-KRLS) algorithm is shown in Algorithm 2.

### C. IMPROVED KERNEL RECURSIVE LEAST SQUARE ALGORITHM

Through the above analysis, it can be seen that the prediction accuracy of SWS-KRLS algorithm can be

![Figure 5](Image)

**FIGURE 5.** Window size adjustment in the presence of an abrupt change.

![Figure 6](Image)

**FIGURE 6.** The specific process of the VSWS-KRLS algorithm.
improved after adding the NC. Nevertheless, because the sparse method will destroy the correlation of the data, the prediction efficiency of the algorithm is severely degraded. As pointed out, the variable sliding window method enable the algorithm to perform tracking in nonstationary scenarios. In order to obtain an efficient kernel method, it is achieved by combining the two methods and we propose the sparse variable sliding window kernel recursive least square (VSWS-KRLS) algorithm.

In the initial stage of the algorithm, a decision is immediately made of whether input should be added into the dictionary. 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, there are more useful information are used to update the algorithm with low computational complexity.

In the update stage of the algorithm, if the system environment changes, the mechanisms for window size adjustment can solve the problem that SW-KRLS is not sensitive to outliers or system change. The VSWS-KRLS algorithm flow chart is shown in Figure 6.

### D. COMPUTATIONAL COMPLEXITY ANALYSIS

At present, KRLS algorithm has good application prospects because of its high prediction accuracy and fast convergence speed. However, with the increasing of iterations, the size of model will continue to increase, and the increase of computational complexity greatly limits the performance of the algorithm. So, in order to prove that the proposed VSWS-KRLS algorithm has lower computational complexity, this paper compares VSWS-KRLS algorithm with common KRLS algorithms (KRLS, KRLS-ALD, FB-KRLS, EX-KRLS, SW-KRLS). This paper uses the weight coefficient vector \( a_i \), the autocorrelation matrix \( P_i \), dictionary \( D \) and the sparse method to compare the computational complexity.

Suppose that at the iteration \( i \), the input size is \( i \), the dictionary size is \( l \) and the sliding window size is \( L \), in this paper, we use the \( O \) order method to represent the computational complexity of the algorithm. The results are shown in Table 2.

In KRLS algorithm, because sparse method is not adopted, the computational complexity of weight coefficient vector and autocorrelation matrix are both \( O(\hat{P}^2) \). In ALD-KRLS algorithm, the computational complexity of sparse method ALD, weight coefficient vector and autocorrelation matrix are both \( O(\hat{P}) \) (\( i \) is the size of the kernel matrix with sparse method, \( i \leq l \)). Similar to KRLS, the complexity of the weight coefficient vector and the autocorrelation matrix are both \( O(\hat{P}^2) \) in EX-KRLS algorithm. Both FB-KRLS algorithm and SW-KRLS algorithm limit the growth of the kernel matrix by setting a fixed size of window, so the computational complexity of weight coefficient vector and autocorrelation matrix are both \( O(L^2) \) (\( L \) is the size of the sliding window, \( L \leq i \)).

In VSWS-KRLS algorithm, two sparse methods (NC and sliding window) are added to reduce the computational complexity. And the computational complexity of the weight coefficient vector, autocorrelation matrix, sparse method and dictionary are all \( O(\hat{P}^2) \) (\( j \) is the size of the NC dictionary, and the dictionary size is controlled by a variable sliding window, so \( j \leq L \)). To summarize, compared with other algorithms, the VSWS-KRLS algorithm has the smallest computational complexity at iteration \( i \), and with the increasing of iteration, the advantage of computational complexity will become more obvious.

### V. SIMULATION

In this section, the KRLS-ALD algorithm and the KRLS-NC algorithm are evaluated to show the reason for choosing the NC as the sparse method. In addition, in order to verify the effectiveness of the proposed VSWS-KRLS algorithm, we experiment SWS-KRLS algorithm, VSWS-KRLS and VSWS-KRLS algorithms under nonlinear system identification.

### A. SIMULATION SETTINGS

This paper chooses nonlinear Wiener system as the nonlinear system, and the model is shown in Figure 7. 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, 20dB of additive white Gaussian noise (AWGN) is added. The linear channel coefficients will change at a given moment to

| Table 2. Computational complexity of different algorithms in a single iteration |

| Algorithm | Computational complexity |
|-----------|-------------------------|
| KRLS     | \( a_i \cdot O(\hat{P}) \) |
| ALD-KRLS | \( a_i \cdot O(\hat{P}) \) (\( i \leq l \)) |
| FB-KRLS  | \( a_i \cdot O(L^2) \) (\( i \leq l \)) |
| EX-KRLS  | \( a_i \cdot O(\hat{P}) \) |
| SW-KRLS  | \( a_i \cdot O(L^2) \) (\( i \leq l \)) |
| VSWS-KRLS| \( a_i \cdot O(L^2) \) (\( i \leq l \)) |
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 \( H_1(z) = 1 + 0.8362z^{-1} - 0.7732z^{-2} - 0.4484z^{-3} \), after receiving 500 data, it is changed into \( H_2(z) = 1 - 0.8045z^{-1} + 0.9962z^{-2} + 0.4678z^{-3} \). The performance of the model is shown through the MATLAB simulation.

In the experiment, the mean square error (MSE) is selected as the evaluation index of prediction accuracy. It can be calculated as:

\[
MSE = \sqrt{\frac{1}{L} \sum_{i=1}^{L} (y_i - \hat{y}_i)^2}
\]  (38)

where \( i \) represents the time, \( y_i \) represents the target output, \( \hat{y}_i \) represents the predicted output, and \( L \) represents the number of samples. 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. In the system identification experiment, the time embedding length of the input sequence is 4, the length of the training sequence is 1,500, the length of the test sequence is 100, and the parameter of the Gaussian kernel function is 1.

Meanwhile, this experiment was simulated on windows 10 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.

**B. SELECTION OF SPARSE METHODS**

In order to choose a suitable sparse method, 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 for the ALD, 80 values of \( \delta_3 \) are uniformly selected in the interval of \([0.05, 0.3]\).

![Figure 8](image8.png)

**FIGURE 8.** Comparison of different sparse methods NC and ALD of KRLS. The "*" point comes from KRLS-NC, the "+" point comes from KRLS-ALD.

For each threshold, we use KRLS-NC and KRLS-ALD to train, and record the kernel matrix size and MSE values. Figure 8 shows the MSE values corresponding to two different sparse methods in different kernel matrix size.

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.

**C. SLIDING WINDOW SPARSE KERNEL RECURSIVE LEAST SQUARES ALGORITHM**

To further test the performance of the SWS-KRLS algorithm, this section shows the influence of algorithm performance with different NC thresholds. We set the size of the sliding window is 150, the threshold \( \delta_1=[0.2, 0.5, 0.7] \), and threshold \( \delta_2=0.05 \). The simulation result is shown in Figure 9. It is showed that the steady-state error is improved by adding improved NC to the SW-KRLS algorithm. For thresholds, the greater the value of \( \delta_1 \) is, the better the steady-state error of the system. But the sparse methods will destroy the complex relationship between the data, and it will reduce the convergence speed of the algorithm.

![Figure 9](image9.png)

**FIGURE 9.** MSE learning curves of SW-KRLS and SWS-KRLS with different NC thresholds.

For each threshold, we use KRLS-NC and KRLS-ALD to train, and record the kernel matrix size and MSE values. Figure 8 shows the MSE values corresponding to two different sparse methods in different kernel matrix size.

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.

**C. IMPROVED KERNEL RECURSIVE LEAST SQUARES ALGORITHM**

This paper introduces a change detection mechanism in the variable sliding window method. Different from the literature \([41]\), 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 Figure 10.
The error difference unprocessed is unstable at each iteration step, so if we use it to observe the system, the judgment is inaccurate. By comparison, the error difference with synchronous superposition average algorithm can better reflect the situation of the system, because it can select adjacent time to decrease the influence of instability. In order to evaluate the impact of different window size on the performance of the algorithm, we choose the window size M of the SW-KRLS algorithm to be 50, 100, and 150. Meanwhile, the variable sliding window method is implemented in the algorithm of M = 150. The simulation results are shown in Figure 11 and Figure 12. Clearly, when the SW-KRLS
algorithm undergoes a system change in the system identification, the fixed window size affects the performance of the algorithm, its steady-state MSE decreases and tracking ability worsens when the size of sliding window grows. However, the trade-off between the efficiency and accuracy in this algorithm can be adjusted by adjusting the size of sliding window. The experimental results demonstrate that the VSW-KRLS algorithm has a faster convergence rate as well as a smaller steady-state misalignment than the SW-KRLS algorithm with fixed size.

As shown in Figure 13, the VSWS-KRLS algorithm is compared with the NORMA algorithm, KRLS-T algorithm, SW-KRLS algorithm and SWS-KRLS algorithm. Gaussian kernel with \( \sigma = 1 \) is used in all algorithms, and the regularization is set to 0.01. We apply the KRLS-T algorithm with forgetting factor \( \lambda = 0.999 \). The simulation results show that the VSWS-KRLS algorithm has not only a fast convergence rate but also a small steady-state error compared with other algorithms. Meanwhile, the VSWS-KRLS algorithm has the same convergence accuracy as the SWS-KRLS algorithm. With the application of the variable sliding window method, the convergence rate of the VSWS-KRLS algorithm has been greatly improved.

The dynamic change of the dictionary size is plotted in Figure 14. It can be observed that in the KRLS algorithm, the network sizes increase quickly with the increasing of iterations. By using the sliding window method, the size of the dictionary is limited, the dictionary size will stop growing when it grows to the preset size. And in fact, we can stop adding new centers after convergence by noticing that the MSE does not change after convergence. In the VSWS-KRLS algorithm, the variable sliding window method and novel criterion guarantee the dictionary size changes more slowly, and at the same time they can also reduce the usage rate of the dictionary. Moreover, Figure 15 shows a more gradual linear channel transition from \( H_1 \) to \( H_2 \), when the system is linearly varying, the VSWS-KRLS algorithm achieve better performance than other algorithms.

Table 3 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 \( \theta \) represent the threshold and change detection threshold, \( \mu \) represents the learning rate, \( \lambda \) represents the forgetting factor and MSE represents the mean square error value. \( t \) represents the system running time of a single iteration. As can be seen, in the SWS-KRLS algorithm, the increase of the NC threshold will not only improve the convergence accuracy of the algorithm, but also reduce the computational complexity of the algorithm. Compared with SW-KRLS algorithm, the convergence performance, steady state performance and calculation performance of the VSWS-KRLS algorithm has been significantly improved.

| Algorithm   | \( M \) | \( \delta_1 \) | \( \delta_2 \) | \( \theta \) | \( \mu \) | \( \lambda \) | MSE  | \( t \)  |
|-------------|--------|------------|------------|--------|--------|--------|-------|-------|
| NORMA       | \( \infty \) | 0.1        |           |        |        | 0.999  | 11.08dB | 1.19  |
| KRLS-T      | 150    | 0.999      |           |        |        |        | 16.56dB | 1.35  |
| SW-KRLS     | 150    | 0.2        | 0.05      |        |        |        | -12.26dB | 0.42  |
|             | 150    | 0.5        | 0.05      |        |        |        | -13.63dB | 0.59  |
|             | 200    | 0.7        | 0.05      |        |        |        | -14.95dB | 0.73  |
| SWS-KRLS    | 150    | 3\( \sigma \) |            |        |        |        | -14.21dB | 0.60  |
|             | 150    | 0.5        | 0.05      |        |        |        | -15.35dB | 0.45  |
|             | 150    | 0.7        | 0.05      |        |        |        | -15.89dB | 0.37  |
| VSWS-KRLS   | 150    | 0.7        | 0.05      | 3\( \sigma \) | 0.05  |        | -15.61dB | 0.60  |
| VSWS-KRLS   | 150    | 0.7        | 0.05      | 3\( \sigma \) |        |        | -15.86dB | 0.36  |
Finally, Figure 16 shows the actual value of the model and the predicted value using the SW-KRLS and VSWS-KRLS algorithms. In case of different amounts of data, the proposed algorithm can better estimate the model value. And it can be seen in Table 3 and Figure 16 that compared with the SW-KRLS algorithm, the VSWS-KRLS algorithm can better estimate the model value. In general, the VSWS-KRLS algorithm can play a great role in the field of system identification and make greater contributions to data processing.

VI. CONCLUSIONS AND FUTURE WORK

This paper proposes an improved sparse kernel RLS algorithm for nonlinear system identification. NC, dictionary with sliding window, variable sliding window size technique and mutation detection mechanism are combined with the KRLS algorithm to form our improved KRLS algorithm. Compared with the other algorithm, the VSWS-KRLS algorithm had better prediction performance and low computational complexity in identification of nonlinear systems.

Although the VSWS-KRLS algorithm can realize online prediction and limit the dimension of kernel matrix, the accuracy of online prediction will be affected by NC condition threshold. If the threshold is too small, the amount of data will be large and the efficiency will be low. However, if the threshold is too large, some important data may be deleted by mistake, resulting in the decline of accuracy, therefore, it needs to be improved in the future.

In future work, we will focus on selection of the other sparse methods in the future, which has certain effects on performance and deserves further studied. Furthermore, we will study the influence of adding forgetting factor and quantification to the performance of the algorithm.

REFERENCES

[1] R. Ah Yap, “System identification,” IEEE Trans. Automat. Contr., vol. 18, pp. 85–86, Feb. 1973.
[2] Y. Lin, X. Zhu, Z. Zheng, Z. Dou, and R. Zhou, “The individual identification method of wireless device based on dimensionality reduction and machine learning,” J. Supercomput., vol. 5, pp. 1–18, Dec. 2017.
[3] N. Simidjievski, L. Todorovski, J. Kocijan and S. Džeroski, “Equation discovery for nonlinear system identification,” IEEE Access, vol. 8, pp. 29930–29943, Feb. 2020.
[4] S. Liu, S. Wang, X. Liu, C. T. Lin, and Z. Lv, “Fuzzy detection aided realtime and robust visual tracking under complex environments,” IEEE Trans. Fuzzy Syst., vol. 29, no. 1, pp. 90–102, Jan. 2021.
[5] Y. Hu, J. Wu, and C. Zeng, “Robust adaptive identification of linear time-varying systems under relaxed excitation conditions,” IEEE Access, vol. 8, pp. 8268–8274, Jan. 2020.
[6] Y. Lin, Y. Tu, Z. Dou, L. Chen, and S. Mao, “Countour stella image and deep learning for signal recognition in the physical layer,” IEEE Trans. Cognit. Commun. Netw., vol. 7, no. 1, pp. 34–46, Mar. 2021.
[7] A. H. Sayed, Fundamentals of Adaptive Filtering. Hoboken, NJ, USA: Wiley, 2003.
[8] S. Li, Y. Zhou, R. Li, and X. Zhao, “Online lithium battery fault diagnosis based on least square support vector machine optimized by Ant Lion algorithm,” Int. J. Performability Eng. vol. 16, no. 10, pp. 1637–1645, Oct. 2020.
[9] A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum likelihood from incomplete data via the EM algorithm,” J. Roy. Stat. Soc.: B (Methodol.), vol. 39, 1, pp. 1–22, 1977.
[10] Y. Tu, Y. Lin, J. Wang, and J. U. Kim, “Semi-supervised learning with generative adversarial networks on digital signal modulation classification,” Comput. Mater. Continua, vol. 55, no. 2, pp. 243–254, Jan. 2018.
[11] H. Zhou, J. Bai, Y. Wang, L. Jiao, S. Zheng, W. Shen, J. Xu, and X. Yang, “Few-shot electromagnetic signal classification: A data union augmentation method,” Chin. J. Aeronaut., Sep. 15, 2021. [Online]. Available: https://doi.org/10.1016/j.cja.2021.07.014.
[12] S. S. Haykin, Adaptive Filter Theory. Chennai, India: Pearson, 2005.
[13] B. Widrow and M. E. Hoff, “Adaptive switching circuits,” IRE WESCON Conv. Rec., vol. 4, no. 1, pp. 96–104, Jun. 1960.
[14] J. Kivinen, A. J. Smola, and R. C. Williamson, “Online learning with kernels,” IEEE Trans. Signal Process., vol. 52, no. 8, pp. 2165–2176, Aug. 2004.
[15] S. S. Keerthi and C. J. Lin, “Asymptotic behaviors of support vector machines with Gaussian kernel,” Neural Comput., vol. 15, no. 7, pp. 1667–1689, Aug. 2003.
[16] Y. Yong, “Wireless sensor node location based on Igwo-LSSVM,” Int. J. Performability Eng., vol. 16, no. 9, pp. 1434–1442, Sep. 2020.
[17] R. Rosipal and L. J. Trejo, “Kernel partial least squares regression in reproducing kernel Hilbert space,” J. Mach. Learn. Res., vol. 2, pp. 97–123, Mar. 2001.

[18] D. Li, W. E. Wong, W. Yang, Y. Yao, and M. Chau. “Detection and mitigation of label-flipping attacks in federated learning systems with KPCA and K-Means,” in Proc. 8th Int. Conf. Dependable Syst. Their Appl. (DSAt), Yinchuan, China, pp. 551–559, Aug. 2021.

[19] W. Liu, J. Principe, and S. Haykin, Kernel Adaptive Filtering, Hoboken, NJ, USA: Wiley, 2010.

[20] Y. Engel, S. Mannor, and R. Meir, “The kernel recursive least-squares algorithm,” IEEE Trans. Signal Process., vol. 52, no. 8, pp. 2275–2285, Aug. 2004.

[21] N. Aronszajn, “Theory of reproducing kernels,” Trans. Amer. Math. Soc., vol. 68, no. 3, pp. 337–404, May 1950.

[22] M. Han, S. Zhang, M. Xu, T. Qiu, and N. Wang, “Multivariate chaotic time series online prediction based on improved kernel recursive least squares algorithm,” IEEE Trans. Cybern., vol. 49, no. 4, pp. 1160–1172, Apr. 2019.

[23] H. Zhou, J. Huang, F. Lu, J. Thiyagalingam, and T. Kirubarajan, “Echo state kernel recursive least square algorithm for machine condition prediction,” Mech. Syst. Signal Process., vol. 111, pp. 68–86, Oct. 2018.

[24] S. Van Vaerenbergh, J. V. A., and I. Santamaría, “A sliding-window kernel RLS algorithm and its application to nonlinear channel identification,” in Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP), vol. 5, pp. 789–792, May 2006.

[25] S. Van Vaerenbergh, I. Santamaría, W. Liu, and J. C. Príncipe, “Fixed budget kernel recursive least-squares,” in Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP), pp. 1882–1885, Mar. 2010.

[26] B. Chen, S. Zhao, P. Zhu, and J. C. Príncipe, “Quantized kernel recursive least squares algorithm,” IEEE Trans. Neural Netw. Learn. Syst., vol. 24, no. 9, pp. 1484–1491, Sep. 2013.

[27] M. Han, J. Ma, and S. Kanae, “Time series online prediction based on adaptive dynamic adjustment kernel recursive least squares algorithm,” in Proc. IEEE 9th Int. Conf. Intell. Control Inf. Process. (ICICIP), Wanzhou, China, pp. 66–72, Nov. 2018.

[28] H. Zhou, J. Huang, and F. Lu, “Parsimonious kernel recursive least squares algorithm for aero-engine health diagnosis,” IEEE Access, vol. 6, pp. 74687–74698, Nov. 2018.

[29] M. Han, S. Zhang, M. Xu, T. Qiu, and N. Wang, “Multivariate chaotic time series online prediction based on improved kernel recursive least squares algorithm,” IEEE Trans. Cybern., vol. 49, no. 4, pp. 1160–1172, Apr. 2019.

[30] K. Zhong, J. Ma, and M. Han, “Online prediction of noisy time series: dynamic adaptive sparse kernel recursive least squares from sparse and adaptive tracking perspective,” Eng. Appl. Artif. Intell., vol. 91, pp. 103547, May 2020.

[31] J. Guo, H. Chen, and S. Chen, “Improved kernel recursive least squares algorithm based online prediction for nonstationary time series,” IEEE Signal Process. Lett., vol. 27, pp. 1365–1369, July 2020.

[32] T. Zhang, S. Wang, X. Huang, and L. Jia, “Kernel recursive least squares algorithm based on the Nyström method with k-means sampling,” IEEE Signal Process. Lett., vol. 27, pp. 361–365, Feb. 2020.

[33] B. Chen, N. Zheng, and J.C. Príncipe, “Sparse kernel recursive least squares using L1 regularization and a fixed-point sub-iteration,” in Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP), pp. 5257–5261, July 2014.

[34] Y. Tu, Y. Lin, H. Zha, J. Zhang, Y. Wang, G. Gui, and S. Mao, “Large-scale real-world radio signal recognition with deep learning,” Chin. J. Aeronaut., Oct. 13, 2021. [Online]. Available: https://doi.org/10.1016/j.cja.2021.07.014.

[35] J. D. A. Santos, G. A Barreto, “An outlier-robust kernel RLS algorithm for nonlinear system identification,” Nonlinear Dyn., vol. 90, pp. 1707–1726, Aug. 2017.

[36] S. Liu, S. Wang, X. Liu, A. Gandomi, M. Daneshmand, K. Muhammad, and V. H. De Albuquerque, “Human memory update strategy: A multi-layer template update mechanism for remote visual monitoring,” IEEE Trans. Multimedia, vol. 23, pp. 2188–2198, Mar. 2021.

[37] M. Xu, Z. Lu, Y. Qiao, and Y. Min, “Modelling of wind power forecasting errors based on kernel recursive least-squares method,” J. Modern Power Syst. Clean Energy, vol. 5, no. 5, pp. 735–745, Sep. 2017.

[38] Y. Lin, Y. Tu, and Z. Dou, “An improved neural network pruning technology for automatic modulation classification in edge devices,” IEEE Trans. Veh. Technol., vol. 69, no. 5, pp. 5703–5706, May 2020.

[39] J. Platt, “A resource-allocation networking for function interpolation,” Neural Comput., vol. 3, no. 2, pp. 213–225, June 1991.

[40] C. Richard, J. C. M. Bermudez, and P. Honcine, “Online prediction of time series data with kernels,” IEEE Trans. Signal Process., vol. 57, no. 3, pp. 1058–1067, Mar. 2009.

[41] B. J. Julian, “Modifications to the sliding-window kernel RLS algorithm for time-varying nonlinear systems: online resizing of the kernel matrix,” in Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP), Taipei, Taiwan, pp. 3389–3392, Apr. 2009.

[42] F. Gustafsson and F. Gustafsson, Adaptive Filtering and Change Detection, Hoboken, NJ, USA: Wiley, 2000.
YING GAO received the Ph.D. degrees from Jilin University, Jilin, China, in 2008. She is currently the associate professor and master tutor with School of Optoelectronic Information Science and Technology, Yantai University. Her main research interests include speech signal processing, adaptive filtering algorithm and blind signal processing.

JINDONG XU received the Ph.D. degrees from Beijing Normal University. He is currently the associate professor and master tutor with School of Computer and Control Engineering, Yantai University. His main research interest is image processing.

ZHUORAN CAI received the Ph.D. degree in information and communication engineering from the Harbin Institute of Technology, in 2013. He is currently an associate professor with the School of Optoelectronic Information Science and Technology, Yantai University, Yantai, China. His research interests include deep learning methods, cognitive radio, and compressed sensing.