IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 30, 2018, accepted November 18, 2018, date of publication November 22, 2018,
date of current version December 27, 2018.

*Digital Object Identifier 10.1109/ACCESS.2018.2882824*

# Parsimonious Kernel Recursive Least Squares Algorithm for Aero-Engine Health Diagnosis

**HAOWEN ZHOU, JINQUAN HUANG, AND FENG LU**

Jiangsu Province Key Laboratory of Aerospace Power Systems, College of Energy and Power Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China

Corresponding authors: Haowen Zhou (zhouhaowen@nuaa.edu.cn) and Jinquan Huang (jhuang@nuaa.edu.cn)

This work was supported in part by the Funding of Jiangsu Innovation Program for Graduate Education under Grant KYLX16_0400 and in part by the National Natural Science Foundation of China under Grant 61304113.

**ABSTRACT** Kernel adaptive filtering (KAF) has gained widespread popularity among the machine learning community for online applications due to its convexity, simplicity, and universal approximation ability. However, the network generated by KAF keeps growing with the accumulation of the training samples, which leads to the increasing memory requirement and computational burden. To address this issue, a pruning approach that attempts to restrict the network size to a fixed value is incorporated into a kernel recursive least squares (KRLS) algorithm, yielding a novel KAF algorithm called parsimonious KRLS (PKRLS). The basic idea of the pruning technique is to remove the center with the least importance from the existing dictionary. The importance of a center is quantified by its contribution to minimizing the cost function. The calculation of the importance measure is formulated in an efficient manner, which facilitates its implementation in online settings. Experimental results on the benchmark tasks show that PKRLS obtains a parsimonious network structure with the satisfactory prediction accuracy. Finally, a multi-sensor health diagnosis approach based on PKRLS is developed for identifying the health state of a degraded aero-engine in real time. A case study in a turbofan engine degradation data set demonstrates that PKRLS provides an effective and efficient candidate for modeling the performance deterioration of real complex systems.

**INDEX TERMS** Health diagnosis, kernel adaptive filtering, pruning method.

## I. INTRODUCTION

Kernel methods provide a unified framework to formulate nonlinear methods based on the linear counterparts and has gained numerous successful applications, including support vector machine [1], kernel principal component analysis (KPCA) [2], kernel fisher discriminant analysis [3]. The main idea behind these applications is that an implicit nonlinear mapping associated with a Mercer kernel is used to transform the data from the input space to high-dimensional reproducing kernel Hilbert spaces (RKHS) where the linear learning algorithms are implemented. However, the above-mentioned approaches are derived in a batch form and require all the training data to be available in advance. In online scenarios where the training samples arrive sequentially, these offline methods have to retrain the model from scratch once a new data point is available and are thus unsuitable for real-time applications. By contrast, a sequential learning method that updates the existing model recursively without reconsidering the historical data would be a better choice, especially when the real-time performance is emphasized.

Online kernel-based learning (OKL) is a feasible paradigm of learning the desired nonlinearity recursively. Recently, online KPCA has been proposed for feature extraction [4]. However, all the available observations are required for representing the basis functions, which leads to the high computational complexity. As a remedy, a compact dictionary which is a subset of the whole training dataset is derived and the inclusion or replacement of a dictionary member is determined at each time step [5], [6]. Due to the online adaption mechanism, they can be applied to capturing time-varying pattern features. As another subfield of OKL, kernel adaptive filtering (KAF) has become very appealing in online settings due to its universal approximation capability, convexity, and simple structure. KAF aims to reconstruct the well-established linear adaptive filters in RKHS whereby nonlinear filters with more powerful modeling capability in the original input space are obtained. The KAF family includes kernel least mean square (KLMS) [7], kernel affine projection algorithm [8], KRLS [9], and extended KRLS [10], etc.

Similar to the most kernel-based machines such as KPCA, KAF suffers from the lack of sparseness. KAF generates a

linearly growing network by allocating a kernel unit for the new data point at each iteration, which incurs an increasing demand for computational resource and memory storage. According to [11], there are mainly two strategies that can achieve a parsimonious structure. The first one is the constructive strategy which starts with a null network and gradually adds new neural nodes according to some rules. The second is the destructive strategy where a large network is first built and irrelevant nodes are pruned. Considering that the training samples arrive sequentially in online scenarios, the constructive strategy is a straightforward method for controlling the growth of the network. Concretely, only the informative samples are selected based on some sparsification criteria such as novelty criterion [12], approximation linear dependency (ALD) criterion [9], coherence criterion [13], and surprise criterion [14]. Alternatively, the destructive strategy has been adopted to eliminate the redundant centers from the existing dictionary. In [15] and [16], the $l_1$-norm regularization term is incorporated into the cost function whereby the centers associated with the negligibly small coefficients are automatically pruned. A sliding-window KRLS algorithm is proposed by utilizing only the last $N$ data points to train the model [17]. Despite its simple structure, it exhibits an excellent tracking capability in non-stationary environment. In contrast to sliding-window KRLS that omits the oldest center, the algorithm proposed in [18] eliminates the center that leads to the least approximation error. In [19], the center with the least influence on the output of the overall learning system is removed so as to keep the network size within a predefined threshold. In [20], the minimum description length principle is adopted to adapt the network size according to the variations in the input data complexity.

While the aforementioned sparsification techniques reduce the computational burden effectively, the accuracy performance deteriorates inevitably because the redundant data that are unable to satisfy a certain criterion are thrown away directly and excluded from the training process. Considering that these discarded data is useful more or less, they can be used to modify the coefficients of the network instead of updating the structure (allocating a new kernel unit). Enlightened by this idea, a quantization technique was introduced into KLMS and KRLS [21]–[24]. The quantization method divides the whole input space into small regions and each region is represented by a center. The redundant data are used to update the coefficient of the closest center in the dictionary. By doing so, the information conveyed by the data are fully perceived such that a compact network with the satisfactory accuracy performance is obtained.

Despite the utilization of the sparsification criterion, the network size may still keep growing with the accumulation of the training data. In order to keep the computational complexity moderate at each iteration, a pruning technique is integrated into KRLS algorithm. The main contributions of this study are summarized as follows.

1) A novel pruning technique that attempts to keep the network size upper bounded is proposed. The importance of each center is measured by its contribution to minimizing the cost function. In order to facilitate the online application of the importance criterion, it is calculated in an efficient manner.
2) The training data are not equally informative. Thus, flexible learning strategies are adopted according to different samples so as to allocate the computational resources appropriately. Concretely, our method combines three learning strategies and each one plays a distinct role in the training phase. The first refers to the coefficient update strategy which employs each entered sample for adjusting the coefficients of the network. The second is the structure update strategy that selects a minimal number of centers according to ALD criterion such that a compact network with the required nonlinearity is constructed. The last one is the pruning strategy and it deletes the least significant dictionary member once the dictionary size exceeds a preset threshold.
3) A multi-sensor health diagnosis method based on PKRLS is developed. To be specific, a health state classifier is generated by PKRLS. Due to its parsimonious structure, the classifier achieves fast predicting speed, which is beneficial to the online assessment of the health status of aero-engines.

The rest of this paper is organized as follows. In section II, we briefly review KRLS and ALD criterion. Section III elaborates on the derivation of PKRLS algorithm. Section IV gives experimental results on several benchmark datasets to verify the feasibility of PKRLS. Section V develops a novel multi-sensor health diagnosis method based on PKRLS. Section VI summarizes our research work.

## II. KERNEL RECURSIVE LEAST SQUARES ALGORITHM

### A. FORMULATION OF THE COST FUNCTION

Consider the task of learning a nonlinear function $f : \mathbb{U} \to \mathbb{R}$ based on a sequence of input-output pairs $\{\mathbf{u}_j, d_j\}_{j=1}^{i}$up to time step $i$, where $\mathbb{U} \subseteq \mathbb{R}^l$ is the input domain, $\mathbf{u}_j$ is the input vector, and $d_j$ is the corresponding scalar output.

A nonlinear mapping $\varphi : \mathbb{U} \to \mathbb{H}$ associated with a Mercer kernel is utilized to convert the input data into a high-dimensional feature space $\mathbb{H}$. One notable property of the kernel-induced mapping is that the inner product operation in the feature space can be calculated via a kernel function $\kappa$ as follows

$$\kappa(\mathbf{u}_1, \mathbf{u}_2) = \varphi(\mathbf{u}_1)^T \varphi(\mathbf{u}_2) \tag{1}$$

where $\kappa : \mathbb{U} \times \mathbb{U} \to \mathbb{R}$. The Gaussian kernel expressed as $\kappa(\mathbf{u}_1, \mathbf{u}_2) = \exp(-\|\mathbf{u}_1 - \mathbf{u}_2\|^2/2\sigma^2)$ where $\sigma$ denotes kernel parameter is adopted in this paper.

With the utilization of the kernel-induced mapping, a linear model in the feature space is constructed as

$$f(\mathbf{u}) = \boldsymbol{\omega}^T \varphi(\mathbf{u}) \tag{2}$$

where $\boldsymbol{\omega}$ denotes the weight vector in $\mathbb{H}$. Then the learning task boils down to selecting a weight vector $\boldsymbol{\omega}_i \in \mathbb{H}$ that minimizes the following cost function

$$\min_{\boldsymbol{\omega}_i} J_i = \sum_{j=1}^{i} |d_j - \boldsymbol{\omega}_i^T \varphi(\mathbf{u}_j)|^2 + \lambda \|\boldsymbol{\omega}_i\|^2 \tag{3}$$

where $\lambda$ is the regularization parameter.

### B. ITERATIVE COMPUTATION OF THE KERNEL MATRIX INVERSION

By setting the gradient of $J_i$ with respect to $\boldsymbol{\omega}_i$ to zero, the optimal solution of (3) is obtained as

$$\boldsymbol{\omega}_i = \Phi_i[\lambda \mathbf{I}_i + \Phi_i^T \Phi_i]^{-1} \mathbf{d}_i \tag{4}$$

where $\mathbf{I}_i$ is the identify matrix, $\mathbf{d}_i = [d_1, \cdots, d_i]^T$, and $\Phi_i = [\varphi(\mathbf{u}_1), \cdots, \varphi(\mathbf{u}_i)]$. The weight vector $\boldsymbol{\omega}_i$ can be also expressed in the form of a linear combination of the feature inputs as

$$\boldsymbol{\omega}_i = \Phi_i \boldsymbol{\alpha}_i \tag{5}$$

where $\boldsymbol{\alpha}_i = [\lambda \mathbf{I} + \Phi_i^T \Phi_i]^{-1} \mathbf{d}_i$. The coefficient vector $\boldsymbol{\alpha}_i$ is updated recursively by

$$\boldsymbol{\alpha}_i = \mathbf{Q}_i \mathbf{d}_i = \begin{bmatrix} \mathbf{Q}_{i-1} + r_i^{-1} \mathbf{z}_i \mathbf{z}_i^T & -\mathbf{z}_i r_i^{-1} \\ -\mathbf{z}_i^T r_i^{-1} & r_i^{-1} \end{bmatrix} \tag{6}$$

where $\mathbf{Q}_i = [\lambda \mathbf{I} + \Phi_i^T \Phi_i]^{-1}$, $\mathbf{z}_i = \mathbf{Q}_{i-1} \Phi_{i-1}^T \varphi(\mathbf{u}_i)$, and $r_i = \lambda + \kappa(\mathbf{u}_i, \mathbf{u}_i) - \mathbf{z}_i^T \Phi_{i-1}^T \varphi(\mathbf{u}_i)$.

Finally, given an input pattern $\mathbf{u}$, the corresponding output of the model approximated at the *i-th* iteration is

$$f_i(\mathbf{u}) = \boldsymbol{\omega}_i^T \varphi(\mathbf{u}) = \boldsymbol{\alpha}_i^T \Phi_i^T \varphi(\mathbf{u}) = \sum_{j=1}^{i} \alpha_i^j \kappa(\mathbf{u}_j, \mathbf{u}) \tag{7}$$

where $\alpha_i^j$ denotes *j-th* element of the coefficient vector $\boldsymbol{\alpha}_i$. Equation (7) reveals that KRLS generates a radial basis function network shown in Fig. 1. Once a new training sample comes in, KRLS allocates a new kernel unit with the new input as its center. This fact indicates the network size grows linearly with the number of training data, which incurs the increasing computation and memory requirement.

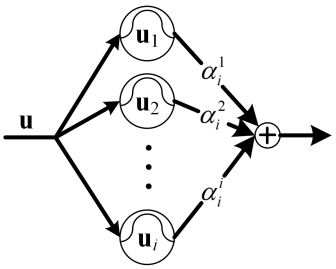



**FIGURE 1.** The network structure generated by KRLS at iteration *i*.

## III. PARSIMONIOUS KERNEL RECURSIVE LEAST SQUARES ALGORITHM

In order to develop the novel algorithm, three steps should be taken when a new training sample $\{\mathbf{u}_i, d_i\}$ arrives. During the first step, the training sample is employed to update the coefficients of the network. In the following step, ALD criterion is used to test whether this training sample is informative enough for updating the network structure. If so, $\mathbf{u}_i$ will be added into a center set $\mathcal{C}$ called the dictionary and the network size will increase by allocating a new kernel unit with $\mathbf{u}_i$ as its center. Otherwise, the network structure maintains unchanged. Finally, a pruning approach is adopted to delete the center with the least importance from the dictionary when the network size exceeds a pre-set threshold. Based on the idea mentioned above, the flowchart of the algorithm is depicted in Fig. 2.

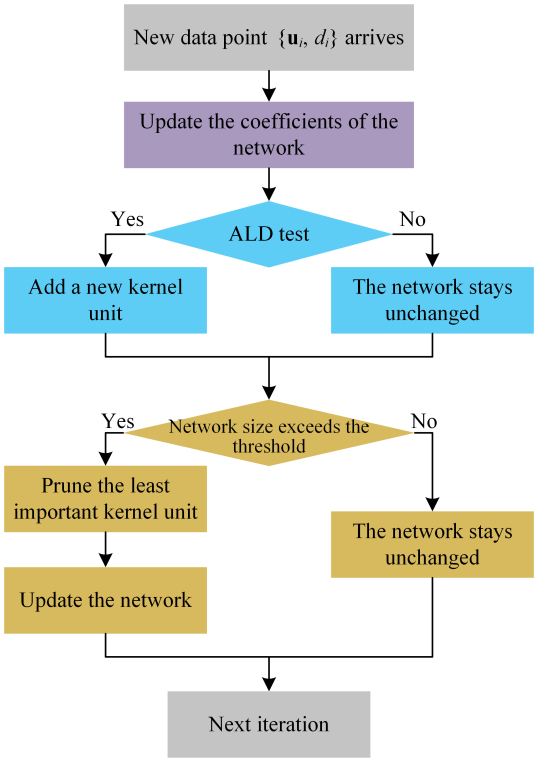



**FIGURE 2.** The flowchart of PKRLS.

### A. REFORMULATION OF THE COST FUNCTION

Assume the dictionary at iteration $i$ is $\mathcal{C}_i = \{\mathbf{c}_i^k\}_{k=1}^{m_i}$, where $\mathbf{c}_i^k$ denotes the *k-th* center and $m_i$ is the dictionary size. The approximated model with sparsification is constructed with the following form

$$f_i(\mathbf{u}) = \sum_{k=1}^{m_i} \alpha_i^k \kappa(\mathbf{c}_i^k, \mathbf{u}) \tag{8}$$

and the weight vector $\boldsymbol{\omega}_i$ is expressed as a linear combination of the feature inputs in the dictionary

$$\boldsymbol{\omega}_i = \sum_{k=1}^{m_i} \alpha_i^k \varphi(\mathbf{c}_i^k) \tag{9}$$

Then the cost function is reformulated as

$$\min J_i = \sum_{j=1}^{i} |d_j - \sum_{k=1}^{m_i} \alpha_i^k \kappa(\mathbf{c}_i^k, \mathbf{u}_j)|^2 + \lambda \|\boldsymbol{\omega}_i\|^2 \quad (10)$$

By incorporating (9) into (10), we obtain

$$\min_{\boldsymbol{\alpha}_i} J_i = \sum_{j=1}^{i} |d_j - \sum_{k=1}^{m_i} \alpha_i^k \kappa(\mathbf{c}_i^k, \mathbf{u}_j)|^2 + \lambda \boldsymbol{\alpha}_i^T \mathbf{K}_{B,i} \boldsymbol{\alpha}_i \quad (11)$$

where

$$\mathbf{K}_{B,i} = \begin{bmatrix} \kappa(\mathbf{c}_i^1, \mathbf{c}_i^1) & \cdots & \kappa(\mathbf{c}_i^1, \mathbf{c}_i^{m_i}) \\ \kappa(\mathbf{c}_i^2, \mathbf{c}_i^1) & \cdots & \kappa(\mathbf{c}_i^2, \mathbf{c}_i^{m_i}) \\ \vdots & \ddots & \vdots \\ \kappa(\mathbf{c}_i^{m_i}, \mathbf{c}_i^1) & \cdots & \kappa(\mathbf{c}_i^{m_i}, \mathbf{c}_i^{m_i}) \end{bmatrix}$$
$$\boldsymbol{\alpha}_i = [\alpha_i^1, \alpha_i^2, \cdots, \alpha_i^{m_i}]^T$$

Furthermore, (11) can be compactly written as

$$\min_{\boldsymbol{\alpha}_i} J_i = \boldsymbol{\alpha}_i^T [\mathbf{K}_{P,i} \mathbf{K}_{P,i}^T + \lambda \mathbf{K}_{B,i}] \boldsymbol{\alpha}_i - 2\mathbf{d}_i^T \mathbf{K}_{P,i}^T \boldsymbol{\alpha}_i \quad (12)$$

where

$$\mathbf{K}_{P,i} = \begin{bmatrix} \kappa(\mathbf{c}_i^1, \mathbf{u}_1) & \cdots & \kappa(\mathbf{c}_i^1, \mathbf{u}_i) \\ \kappa(\mathbf{c}_i^2, \mathbf{u}_1) & \cdots & \kappa(\mathbf{c}_i^2, \mathbf{u}_i) \\ \vdots & \cdots & \vdots \\ \kappa(\mathbf{c}_i^{m_i}, \mathbf{u}_1) & \cdots & \kappa(\mathbf{c}_i^{m_i}, \mathbf{u}_i) \end{bmatrix}$$

By setting the gradient of $J_i$ with respect to $\boldsymbol{\alpha}_i$ to zero, the optimal solution of (12) is achieved as

$$\boldsymbol{\alpha}_i = [\mathbf{K}_{P,i} \mathbf{K}_{P,i}^T + \lambda \mathbf{K}_{B,i}]^{-1} \mathbf{K}_{P,i} \mathbf{d}_i = \mathbf{Q}_i \mathbf{K}_{P,i} \mathbf{d}_i \quad (13)$$

where $\mathbf{Q}_i = [\mathbf{K}_{P,i} \mathbf{K}_{P,i}^T + \lambda \mathbf{K}_{B,i}]^{-1}$. By plugging (13) into (12), the minimal value of $J_i$ is obtained as

$$\tilde{J}_i = -\mathbf{d}_i^T \mathbf{K}_{P,i}^T [\mathbf{K}_{P,i} \mathbf{K}_{P,i}^T + \lambda \mathbf{K}_{B,i}]^{-1} \mathbf{K}_{P,i} \mathbf{d}_i \quad (14)$$

### B. COEFFICIENT UPDATE STRATEGY

In order to update the coefficients of the network based on the new data $\{\mathbf{u}_i, d_i\}$, the inverse matrix $\mathbf{Q}$ in (13) becomes

$$\begin{aligned} \overline{\mathbf{Q}}_i &= \left[ \begin{bmatrix} \mathbf{K}_{P,i-1} & \mathbf{k}_{ib} \end{bmatrix} \begin{bmatrix} \mathbf{K}_{P,i-1}^T \\ \mathbf{k}_{ib}^T \end{bmatrix} + \lambda \mathbf{K}_{B,i-1} \right]^{-1} \\ &= \left[ \mathbf{K}_{P,i-1} \mathbf{K}_{P,i-1}^T + \lambda \mathbf{K}_{B,i-1} + \mathbf{k}_{ib} \mathbf{k}_{ib}^T \right]^{-1} \\ &= \left[ \mathbf{Q}_{i-1}^{-1} + \mathbf{k}_{ib} \mathbf{k}_{ib}^T \right]^{-1} \end{aligned} \quad (15)$$

where $\mathbf{k}_{ib} = [\kappa(\mathbf{c}_{i-1}^1, \mathbf{u}_i), \cdots, \kappa(\mathbf{c}_{i-1}^{m_{i-1}}, \mathbf{u}_i)]^T$.

By utilizing the matrix inversion lemma

$$(\mathbf{A} + \mathbf{BCD})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}(\mathbf{C}^{-1} + \mathbf{DA}^{-1}\mathbf{B})^{-1}\mathbf{DA}^{-1}$$

with the identifications

$$\mathbf{Q}_{i-1}^{-1} \to \mathbf{A}, \mathbf{k}_{ib} \to \mathbf{B}, 1 \to \mathbf{C}, \mathbf{k}_{ib}^T \to \mathbf{D}$$

we have the following recursive formula

$$\overline{\mathbf{Q}}_i = \mathbf{Q}_{i-1} - \frac{\mathbf{Q}_{i-1} \mathbf{k}_{ib} \mathbf{k}_{ib}^T \mathbf{Q}_{i-1}}{1 + \mathbf{k}_{ib}^T \mathbf{Q}_{i-1} \mathbf{k}_{ib}} \quad (16)$$

Hence, the coefficient vector is recalculated as

$$\overline{\boldsymbol{\alpha}}_i = \overline{\mathbf{Q}}_i \overline{\mathbf{K}}_{P,i} \mathbf{d}_i = \boldsymbol{\alpha}_{i-1} + \frac{\mathbf{Q}_{i-1} \mathbf{k}_{ib}}{1 + \mathbf{k}_{ib}^T \mathbf{Q}_{i-1} \mathbf{k}_{ib}} e_i \quad (17)$$

where $\overline{\mathbf{K}}_{P,i} = \begin{bmatrix} \mathbf{K}_{P,i-1} & \mathbf{k}_{ib} \end{bmatrix}$, $e_i = d_i - \mathbf{k}_{ib}^T \boldsymbol{\alpha}_{i-1}$ denotes the prediction error on the current sample using the approximation model achieved at the previous time step.

### C. STRUCTURE UPDATE STRATEGY

According to ALD criterion, the input $\mathbf{u}_i$ will be added into the dictionary, i.e., $\mathcal{C}_i = \mathcal{C}_{i-1} \cup \{\mathbf{u}_i\}$ when it satisfies the following condition

$$D_i = \kappa(\mathbf{u}_i, \mathbf{u}_i) - \mathbf{k}_{ib}^T \mathbf{K}_{B,i-1}^{-1} \mathbf{k}_{ib} > \eta \quad (18)$$

where $\eta$ is a predefined positive constant. If this condition holds, the inverse matrix $\mathbf{Q}$ needs to be augmented as

$$\begin{aligned} \mathbf{Q}_i &= \left[ \mathbf{K}_{P,i} \mathbf{K}_{P,i}^T + \lambda \mathbf{K}_{B,i} \right]^{-1} \\ &= \left[ \begin{bmatrix} \overline{\mathbf{K}}_{P,i} \\ \mathbf{k}_{ip} \end{bmatrix} \begin{bmatrix} \overline{\mathbf{K}}_{P,i}^T & \mathbf{k}_{ip}^T \end{bmatrix} + \begin{bmatrix} \lambda \mathbf{K}_{B,i-1} & \lambda \mathbf{k}_{ib} \\ \lambda \mathbf{k}_{ib}^T & \lambda \kappa(\mathbf{u}_i, \mathbf{u}_i) \end{bmatrix} \right]^{-1} \\ &= \left[ \begin{bmatrix} \overline{\mathbf{K}}_{P,i} \overline{\mathbf{K}}_{P,i}^T & \overline{\mathbf{K}}_{P,i} \mathbf{k}_{ip}^T \\ \mathbf{k}_{ip} \overline{\mathbf{K}}_{P,i}^T & \mathbf{k}_{ip} \mathbf{k}_{ip}^T \end{bmatrix} + \begin{bmatrix} \lambda \mathbf{K}_{B,i-1} & \lambda \mathbf{k}_{ib} \\ \lambda \mathbf{k}_{ib}^T & \lambda \kappa(\mathbf{u}_i, \mathbf{u}_i) \end{bmatrix} \right]^{-1} \end{aligned} \quad (19)$$

where

$$\begin{aligned} \mathbf{k}_{ip} &= [\kappa(\mathbf{u}_i, \mathbf{u}_1), \cdots, \kappa(\mathbf{u}_i, \mathbf{u}_i)] \\ \mathbf{K}_{P,i} &= \begin{bmatrix} \overline{\mathbf{K}}_{P,i} \\ \mathbf{k}_{ip} \end{bmatrix} \\ \mathbf{K}_{B,i} &= \begin{bmatrix} \mathbf{K}_{B,i-1} & \mathbf{k}_{ib} \\ \mathbf{k}_{ib}^T & \kappa(\mathbf{u}_i, \mathbf{u}_i) \end{bmatrix} \end{aligned}$$

In order to update $\mathbf{Q}_i$ iteratively, we consider the following block matrix inversion identity

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{A}^{-1} + \mathbf{A}^{-1}\mathbf{BVCA}^{-1} & -\mathbf{A}^{-1}\mathbf{BV} \\ -\mathbf{VCA}^{-1} & \mathbf{V} \end{bmatrix} \quad (20)$$

where

$$\mathbf{V} = (\mathbf{D} - \mathbf{CA}^{-1}\mathbf{B})^{-1}$$

By applying (20) to (19) with the identifications

$$\begin{aligned} \overline{\mathbf{Q}}_i^{-1} = \overline{\mathbf{K}}_{P,i} \overline{\mathbf{K}}_{P,i}^T + \lambda \mathbf{K}_{B,i-1} &\to \mathbf{A} \\ \overline{\mathbf{K}}_{P,i} \mathbf{k}_{ip}^T + \lambda \mathbf{k}_{ib} &\to \mathbf{B} \\ \mathbf{k}_{ip} \overline{\mathbf{K}}_{P,i}^T + \lambda \mathbf{k}_{ib}^T &\to \mathbf{C} \\ \mathbf{k}_{ip} \mathbf{k}_{ip}^T + \lambda \kappa(\mathbf{u}_i, \mathbf{u}_i) &\to \mathbf{D} \end{aligned}$$

the recursion to calculate $\mathbf{Q}_i$ is achieved as

$$\mathbf{Q}_i = \begin{bmatrix} \overline{\mathbf{Q}}_i & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} + r_i^{-1} \begin{bmatrix} \mathbf{z}_i \\ -1 \end{bmatrix} \begin{bmatrix} \mathbf{z}_i^{-1} & -1 \end{bmatrix} \quad (21)$$

where

$$\begin{aligned} \mathbf{z}_i &= \overline{\mathbf{Q}}_i (\overline{\mathbf{K}}_{P,i} \mathbf{k}_{ip}^T + \lambda \mathbf{k}_{ib}) \\ r_i &= \mathbf{k}_{ip} \mathbf{k}_{ip}^T + \lambda \kappa(\mathbf{u}_i, \mathbf{u}_i) - (\mathbf{k}_{ip} \overline{\mathbf{K}}_{P,i}^T + \lambda \mathbf{k}_{ib}^T) \mathbf{z}_i \end{aligned}$$

Therefore, the formula of computing $\boldsymbol{\alpha}_i$ is expressed as

$$\boldsymbol{\alpha}_i = \mathbf{Q}_i \mathbf{K}_{P,i} \mathbf{d}_i \tag{22}$$

If $D_i \leq \eta$, the network structure stays unchanged and let

$$\begin{cases} \mathcal{C}_i = \mathcal{C}_{i-1} \\ \boldsymbol{\alpha}_i = \overline{\boldsymbol{\alpha}}_i \\ \mathbf{Q}_i = \overline{\mathbf{Q}}_i \end{cases} \tag{23}$$

### D. PRUNING STRATEGY

Here, we define the importance of a center as the increase in the cost function caused by the removal of the corresponding kernel unit. Instead of computing the importance of a center directly, we will elaborate on how to calculate it in a more efficient way.

To start, the cost function in (12) is modified to

$$\min_{\boldsymbol{\alpha}_i} J_i^{(-j)} = \boldsymbol{\alpha}_i^T \left[\mathbf{K}_{P,i}\mathbf{K}_{P,i}^T + \lambda \mathbf{K}_{B,i} + \Lambda_i\right] \boldsymbol{\alpha}_i - 2\mathbf{d}_i^T \mathbf{K}_{P,i}^T \boldsymbol{\alpha}_i \tag{24}$$

where

$$\Lambda_i = \begin{bmatrix} 0_{11} & \cdots & 0_{1j} & \cdots & 0_{1m_i} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 0_{j1} & \cdots & \varepsilon_{jj} & \cdots & 0_{jm_i} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 0_{m_i1} & \cdots & 0_{m_ij} & \cdots & 0_{m_im_i} \end{bmatrix}$$

and $\varepsilon_{jj}$ is a positive scalar. Observe that by setting $\varepsilon_{jj} \to \infty$, $\alpha_i^j$, which is the coefficient associated with the *j-th* center is forced to move toward zero so as to minimize (24). Consequently, this center makes no contribution to the output of the network, which implies that the corresponding kernel unit is pruned from the network. Moreover, (24) can be written compactly as

$$\min_{\boldsymbol{\alpha}_i} J_i^{(-j)} = \boldsymbol{\alpha}_i^T [\mathbf{K}_{P,i}\mathbf{K}_{P,i}^T + \lambda \mathbf{K}_{B,i} + \boldsymbol{\nu}\boldsymbol{\nu}^T] \boldsymbol{\alpha}_i - 2\mathbf{d}_i^T \mathbf{K}_{P,i}^T \boldsymbol{\alpha}_i \tag{25}$$

where $\boldsymbol{\nu} = \sqrt{\varepsilon}\boldsymbol{\tau}$ with $\boldsymbol{\tau}$ being the *j-th* column of the $m_i$-order identity matrix.

Now the explicit definition of the importance of a center $\mathbf{c}_i^j$ is given by

$$\Delta J_i^{(-j)} = \tilde{J}_i^{(-j)} - \tilde{J}_i \tag{26}$$

where $\tilde{J}_i^{(-j)}$ is the minimum of (25) expressed as

$$\tilde{J}_i^{(-j)} = -\mathbf{d}_i^T \mathbf{K}_{P,i}^T [\mathbf{K}_{P,i}\mathbf{K}_{P,i}^T + \lambda \mathbf{K}_{B,i} + \boldsymbol{\nu}\boldsymbol{\nu}^T]^{-1} \mathbf{K}_{P,i}\mathbf{d}_i \tag{27}$$

Substituting (14) and (27) into (26) yields

$$\begin{aligned} \Delta J_i^{(-j)} = {} & \mathbf{d}_i^T \mathbf{K}_{P,i}^T \left[\mathbf{K}_{P,i}\mathbf{K}_{P,i}^T + \lambda \mathbf{K}_{B,i}\right]^{-1} \mathbf{K}_{P,i}\mathbf{d}_i \\ & - \mathbf{d}_i^T \mathbf{K}_{P,i}^T \left[\mathbf{K}_{P,i}\mathbf{K}_{P,i}^T + \lambda \mathbf{K}_{B,i} + \boldsymbol{\nu}\boldsymbol{\nu}^T\right]^{-1} \mathbf{K}_{P,i}\mathbf{d}_i \end{aligned} \tag{28}$$

Note that by using the matrix inversion lemma, we can easily obtain

$$\begin{aligned} & [\mathbf{K}_{P,i}\mathbf{K}_{P,i}^T + \lambda \mathbf{K}_{B,i} + \boldsymbol{\nu}\boldsymbol{\nu}^T]^{-1} \\ & \quad = [\mathbf{K}_{P,i}\mathbf{K}_{P,i}^T + \lambda \mathbf{K}_{B,i}]^{-1} \\ & \qquad - \frac{[\mathbf{K}_{P,i}\mathbf{K}_{P,i}^T + \lambda \mathbf{K}_{B,i}]^{-1} \boldsymbol{\nu}\boldsymbol{\nu}^T [\mathbf{K}_{P,i}\mathbf{K}_{P,i}^T + \lambda \mathbf{K}_{B,i}]^{-1}}{1 + \boldsymbol{\nu}^T [\mathbf{K}_{P,i}\mathbf{K}_{P,i}^T + \lambda \mathbf{K}_{B,i}]^{-1} \boldsymbol{\nu}} \end{aligned} \tag{29}$$

Then, by combining (28) and (29), we can efficiently calculate $\Delta J_i^{(-j)}$ with the equation

$$\begin{aligned} \Delta J_i^{(-j)} &= \mathbf{d}_i^T \mathbf{K}_{P,i}^T \frac{\mathbf{Q}_i \boldsymbol{\nu}\boldsymbol{\nu}^T \mathbf{Q}_i}{1 + \boldsymbol{\nu}^T \mathbf{Q}_i \boldsymbol{\nu}} \mathbf{K}_{P,i}\mathbf{d}_i \\ &= \frac{\boldsymbol{\alpha}_i^T \boldsymbol{\nu}\boldsymbol{\nu}^T \boldsymbol{\alpha}_i}{1 + \boldsymbol{\nu}^T \mathbf{Q}_i \boldsymbol{\nu}} = \frac{\varepsilon (\alpha_i^j)^2}{1 + \varepsilon \boldsymbol{\tau}^T \mathbf{Q}_i \boldsymbol{\tau}} \end{aligned} \tag{30}$$

As the value of $\varepsilon$ approaches infinity, $\Delta J_i^{(-j)}$ becomes

$$\Delta J_i^{(-j)} = \lim_{\varepsilon \to \infty} \frac{\varepsilon (\alpha_i^j)^2}{1 + \varepsilon \boldsymbol{\tau} \mathbf{Q}_i \boldsymbol{\tau}^T} = \frac{(\alpha_i^j)^2}{\boldsymbol{\tau} \mathbf{Q}_i \boldsymbol{\tau}^T} = \frac{(\alpha_i^j)^2}{q_{jj}} \tag{31}$$

where $q_{jj}$ denotes the *j-th* diagonal element of $\mathbf{Q}_i$. Considering that $\mathbf{Q}_i$ is already available, the computational load of calculating importance in (31) is acceptable. Obviously, $\Delta J_i^{(-j)}$ is positive and it is thereby suitable to quantify the importance of a center. The larger the value of $\Delta J_i^{(-j)}$ is, the more contribution the center $\mathbf{c}_i^j$ makes to the cost function minimization. Therefore, all the centers in the dictionary can be ranked in order of importance. Assume that the dictionary size $m_i$ is larger than a preset threshold $M$ and the *j-th* center $\mathbf{c}_i^j$ is determined to be the least important. In this case, $\mathbf{c}_i^j$ needs to be pruned from the existing dictionary.

For this goal, we let $\mathcal{C}_i = \mathcal{C}_i \backslash \{\mathbf{c}_j\}$ and prune the $j$-th row and column of $\mathbf{Q}_i^{-1}$. Accordingly, the inverse matrix $\mathbf{Q}_i$ and the coefficient vector $\boldsymbol{\alpha}_i$ need to be modified.

Firstly, $\mathbf{Q}_i$ is written in the following block matrix form

$$\mathbf{Q}_i = \begin{bmatrix} \mathbf{Q}_{11} & \mathbf{Q}_{12} & \mathbf{Q}_{13} \\ \mathbf{Q}_{12}^T & q_{jj} & \mathbf{Q}_{23} \\ \mathbf{Q}_{13}^T & \mathbf{Q}_{23}^T & \mathbf{Q}_{33} \end{bmatrix} \tag{32}$$

subsequently, a recursive updating scheme for $\mathbf{Q}_i$ and $\boldsymbol{\alpha}_i$ based on [17] is given by

$$\mathbf{Q}_i = \begin{bmatrix} \mathbf{Q}_{11} & \mathbf{Q}_{13} \\ \mathbf{Q}_{13}^T & \mathbf{Q}_{33} \end{bmatrix} - \frac{1}{q_{jj}} \begin{bmatrix} \mathbf{Q}_{12} \\ \mathbf{Q}_{23}^T \end{bmatrix} \begin{bmatrix} \mathbf{Q}_{12}^T & \mathbf{Q}_{23} \end{bmatrix} \tag{33}$$

and

$$\boldsymbol{\alpha}_i = \mathbf{Q}_i \mathbf{K}_{P,i} \mathbf{d}_i \tag{34}$$

where $\mathbf{K}_{P,i}$ is updated by eliminating its *j-th* row vector.

By combining the above three strategies, a parsimonious KRLS algorithm for online sequential learning is derived. Its pseudocode is shown in Algorithm 1.

**Algorithm 1** PKRLS Algorithm

**Require:**

Kernel parameter $\sigma$; Regularization factor $\lambda$; A small positive constant $\eta$; A positive integer number $M$; An empty set $\mathcal{C}$;

**for** $i = 1, 2, \cdots$ **do**

A new data point $\{\mathbf{u}_i, d_i\}$ arrives;

**if** $i = 1$ **then**

$\mathcal{C}_1 = \{\mathbf{u}_1\}$; $\mathbf{Q}_1 = (\lambda + \kappa(\mathbf{u}_1, \mathbf{u}_1))^{-1}$; $\boldsymbol{\alpha}_1 = \mathbf{Q}_1 d_1$;

**else**

Compute $\overline{\mathbf{Q}}_i$ and $\overline{\boldsymbol{\alpha}}_i$ based on (16) and (17), respectively;

Compute $D_i$ based on (18);

**if** $D_i > \eta$ **then**

$\mathcal{C}_i = \mathcal{C}_{i-1} \cup \{\mathbf{u}_i\}$;

Compute $\mathbf{Q}_i$ and $\boldsymbol{\alpha}_i$ based on (21) and (22), respectively;

**else**

$\mathcal{C}_i = \mathcal{C}_{i-1}$; $\mathbf{Q}_i = \overline{\mathbf{Q}}_i$; $\boldsymbol{\alpha}_i = \overline{\boldsymbol{\alpha}}_i$;

**end if**

**if** $m_i > M$ **then**

Compute the importance $\Delta J_i$ of each center based on (31) and assume the center $\mathbf{c}_j$ is the least important;

$\mathcal{C}_i = \mathcal{C}_i \backslash \{\mathbf{c}_i^j\}$;

Update $\mathbf{Q}_i$ and $\boldsymbol{\alpha}_i$ based on (33) and (34), respectively;

**end if**

**end if**

**end for**

### E. COMPUTATIONAL COMPLEXITY ANALYSIS

The computational complexity at iteration $i$ for KRLS, KRLS with ALD criterion (KRLS-ALD), online sequential extreme learning machine (OS-ELM) [25], PKRLS are listed in Table 1, where $L$ denotes the network size of OS-ELM. In KRLS-ALD, if the sample is considered informative based on ALD, it will be used to update the network structure and coefficients; otherwise this data will be discarded directly. From Table 1, we observe the computational burden of each method is closely related to the dictionary size (or network size). Hence, the key point is how to reduce the dictionary size significantly without sacrificing the prediction accuracy.

*Remark 1:* As for KRLS-ALD, a subset of the training data are selected to train the model and the remaining data are purely discarded. Although the computational complexity is reduced effectively, the accuracy performance deteriorates inevitably. This is because each training sample contains more or less useful information and the information conveyed by the redundant data is omitted. Consequently, KRLS-ALD fails to deal with the tradeoff between the generalization performance and the real time adequately.

*Remark 2:* PKRLS attempts to get rid of this dilemma by enhancing the utilization efficiency of the training data. Specifically, all training data are employed to update the coefficients of the network and the relatively informative ones are picked up to update the network structure, i.e., add a new center into the existing dictionary. Moreover, the pruning strategy further simplifies the network by removing the center with the least importance. By doing this, fewer centers are required to construct a compact network with the desired modeling capability. In real applications, the dictionary size is much less than the number of training samples. The experimental results in the next section will support this point.

*Remark 3:* As for OS-ELM, the hidden layer is randomly generated before training. Due to the random strategy, it often needs much hidden nodes so as to achieve the desirable performance. During the training process, the hidden layer is fixed and only the output weights need to be adjusted at each iteration. Hence, the computational complexity of OS-ELM at each iteration keeps constant.

## IV. BENCHMARK DATASETS TESTING

In this section, 8 benchmark datasets including Winequality_red, BodyFat, Concrete, airfoil Self-Noise, Boston Housing, mg, abalone and mpg are utilized to show the advantage of our method. The three data sets which are Winequality_red, Concrete and Airfoil Self-Noise are obtained from UCI machine learning repository[1] and the others are downloaded from the website.[2] The details of each dataset, containing number of features (#feature), number of training data examples (trNum) and number of testing data examples (teNum), are listed in the Table 2. The input and output variables are normalized into the closed interval $[-1, 1]$ and $[0, 1]$, respectively. For comparison, we also evaluate KRLS, KRLS-ALD, KLMS and OS-ELM. KRLS, KRLS-ALD and PKRLS share the same model parameters, i.e., kernel parameter and regularization factor, which are selected using the leave-one-out cross validation strategy on KRLS. As for PKRLS and KRLS-ALD, the remaining free parameters are tuned such that they achieve nearly the same accuracy performance as KRLS.

Our experiments are carried out using Matlab 2013b on a personal computer with Intel® Core™ i5 CPU 3230M 2.6 GHz processor, 8.00 GB memory and Window 7 operation system. The root mean square error (RMSE) is employed to measure the generalization performance of all considered algorithms, which is expressed as

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{j=1}^{N} (\hat{d}_j - d_j)^2}{N}} \tag{35}$$

where $\hat{d}_j$ is the predicted value, and $N$ is the number of the testing samples.

The experimental results are listed in Table 3 where trTime and teTime denote training time and testing time, respectively. KLMS requires the least training time but obtains the worse generalization performance. This phenomenon is

[1] http://archive.ics.uci.edu/ml/datasets.html.

[2] http://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/regression.html.

**TABLE 1.** The comparison of computational complexity at iteration $i$.

| Algorithms | Calculation operations | | Computational complexity |
|---|---|---|---|
| PKRLS | Coefficient update | Update $\overline{\mathbf{Q}}_i$ | $\mathcal{O}(m_{i-1}^2)$ |
| | | Update $\overline{\boldsymbol{\alpha}}_i$ | $\mathcal{O}(m_{i-1})$ |
| | Structure update | Calculate $D_i$ | $\mathcal{O}(m_{i-1}^2)$ |
| | | Update $\mathbf{Q}_i$ | $\mathcal{O}(m_i^2)$ |
| | | Update $\boldsymbol{\alpha}_i$ | $\mathcal{O}(m_i^2)$ |
| | Pruning | Calculate $\Delta J_i$ | $\mathcal{O}(M+1)$ |
| | | Update $\mathbf{Q}_i$ | $\mathcal{O}(M^2)$ |
| | | Update $\boldsymbol{\alpha}_i$ | $\mathcal{O}(M^2)$ |
| KRLS | Update $\mathbf{Q}_i$ | | $\mathcal{O}(i^2)$ |
| | Update $\boldsymbol{\alpha}_i$ | | $\mathcal{O}(i)$ |
| KRLS-ALD | Calculate $D_i$ | | $\mathcal{O}(m_{i-1}^2)$ |
| | Update $\mathbf{Q}_i$ | | $\mathcal{O}(m_i^2)$ |
| | Update $\boldsymbol{\alpha}_i$ | | $\mathcal{O}(m_i)$ |
| OS-ELM | Update the output weights | | $\mathcal{O}(L^2)$ |

**TABLE 2.** Specifications of regression datasets.

| Datasets | #feature | trNum | teNum |
|---|---|---|---|
| Winequality_red | 11 | 1000 | 599 |
| BodyFat | 14 | 170 | 82 |
| Concrete | 5 | 700 | 330 |
| Airfoil Self-Noise | 13 | 800 | 703 |
| Boston housing | 13 | 400 | 106 |
| mg | 6 | 900 | 485 |
| abalone | 8 | 2000 | 2177 |
| mpg | 7 | 260 | 132 |

due to the fact that the computational complexity of KLMS grows linearly with the network size while that of the other algorithms is proportional to the network size at quadratic scale. PKRLS selects the relatively important samples for updating the network structure according to ALD criterion and the pruning strategy further improves the quality of the centers. Therefore, PKRLS generates the most parsimonious network structure among all the considered algorithms while achieves almost the same accuracy performance as KRLS. PKRLS also consumes the least testing time since it is just determined by the network size.

Next, we investigate the performance of PKRLS and KRLS-ALD with different network sizes. The tendency of RMSE versus the network size on Winequality_red and mg is depicted in Fig. 3 where the blue dash line produced by KRLS is treated as the benchmark for comparison. Experimental results on the other datasets are not provided because of similar results and limited space. It is easy to observe that RMSE converges to the benchmark line gradually as the network size grows. PKRLS obtains a much faster convergence rate than KRLS-ALD. In KRLS-ALD, the samples that are not informative enough for updating the network structure are omitted directly, which leads to the accuracy degradation inevitably. In the case of PKRLS, these redundant data are employed to update the coefficients of the network,

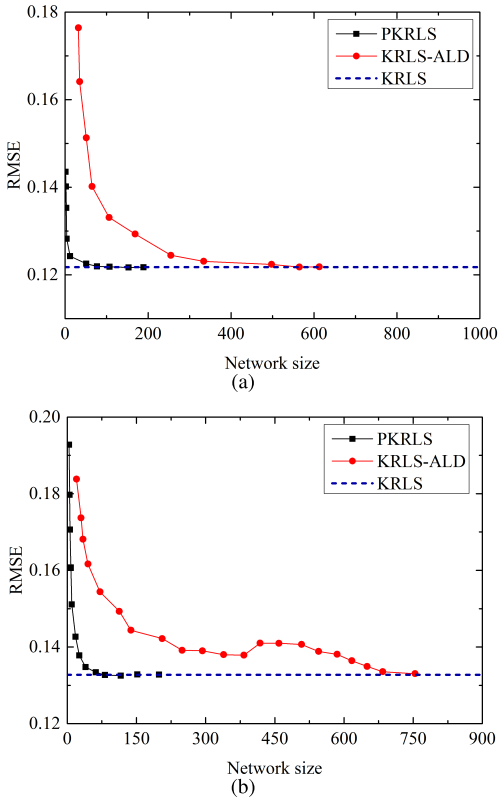



**FIGURE 3.** RMSE versus network size. (a) Winequality_red, and (b) mg.

improving the utilization efficiency of the training dataset. This explains why the accuracy performance of KRLS-ALD is more sensitive to the network size.

## V. AERO-ENGINE HEALTH STATE DIAGNOSIS

The operation of an aero-engine is the interaction of various components (see Fig. 4). With the increase of service time, gas path components including fan, compressor and turbine tend to deteriorate inevitably, due to several reasons such as

**TABLE 3. Experimental results on benchmark datasets.**

| Data sets | Algorithms | Network size | trTime (s) | teTime (ms) | RMSE |
|---|---|---|---|---|---|
| Winequality_red | KLMS | 1000 | 0.0767 | 20.4561 | 0.1419 |
| | KRLS | 1000 | 5.7873 | 21.1812 | 0.1218 |
| | KRLS-ALD | 565 | 3.6335 | 11.7462 | 0.1218 |
| | PKRLS | 150 | 1.4767 | 2.8813 | 0.1218 |
| | OS-ELM | 1200 | 14.8061 | 20.1569 | 0.1228 |
| BodyFat | KLMS | 170 | 0.0098 | 1.3288 | 0.0408 |
| | KRLS | 170 | 0.0536 | 1.3446 | 0.0288 |
| | KRLS-ALD | 135 | 0.0671 | 1.2339 | 0.0292 |
| | PKRLS | 23 | 0.0371 | 0.1763 | 0.0288 |
| | OS-ELM | 250 | 0.0706 | 1.8907 | 0.0320 |
| Concrete | KLMS | 700 | 0.0464 | 10.8565 | 0.1307 |
| | KRLS | 700 | 2.3402 | 10.9805 | 0.0971 |
| | KRLS-ALD | 679 | 5.7750 | 10.5044 | 0.0969 |
| | PKRLS | 400 | 1.5711 | 5.6572 | 0.0973 |
| | OS-ELM | 1050 | 8.7113 | 12.3740 | 0.1136 |
| Airfoil Self-Noise | KLMS | 800 | 0.0498 | 25.6004 | 0.0473 |
| | KRLS | 800 | 3.6241 | 25.5257 | 0.0162 |
| | KRLS-ALD | 740 | 7.8511 | 23.3448 | 0.0165 |
| | PKRLS | 370 | 1.8507 | 11.9479 | 0.0163 |
| | OS-ELM | 950 | 10.5366 | 29.0914 | 0.0385 |
| Boston housing | KLMS | 400 | 0.0279 | 1.6884 | 0.0962 |
| | KRLS | 400 | 0.4461 | 1.6567 | 0.0675 |
| | KRLS-ALD | 391 | 1.0012 | 1.5530 | 0.0674 |
| | PKRLS | 135 | 0.1583 | 0.5378 | 0.0675 |
| | OS-ELM | 350 | 0.3696 | 7.2261 | 0.0673 |
| mg | KLMS | 900 | 0.0587 | 18.3584 | 0.1485 |
| | KRLS | 900 | 4.1613 | 18.1595 | 0.1328 |
| | KRLS-ALD | 754 | 7.4831 | 16.1444 | 0.1330 |
| | PKRLS | 120 | 2.8472 | 3.5029 | 0.1327 |
| | OS-ELM | 750 | 5.7556 | 12.4694 | 0.1347 |
| Abalone | KLMS | 2000 | 0.2444 | 158.4894 | 0.0811 |
| | KRLS | 2000 | 44.1944 | 161.1484 | 0.0728 |
| | KRLS-ALD | 1546 | 92.6601 | 142.0798 | 0.0731 |
| | PKRLS | 120 | 12.3224 | 12.0426 | 0.0728 |
| | OS-ELM | 810 | 17.8449 | 85.6687 | 0.0729 |
| mpg | KLMS | 260 | 0.0167 | 1.0553 | 0.0969 |
| | KRLS | 260 | 0.0851 | 1.0638 | 0.0867 |
| | KRLS-ALD | 255 | 0.2119 | 1.0263 | 0.0870 |
| | PKRLS | 70 | 0.0568 | 0.4299 | 0.0867 |
| | OS-ELM | 600 | 1.0465 | 1.8245 | 0.0869 |

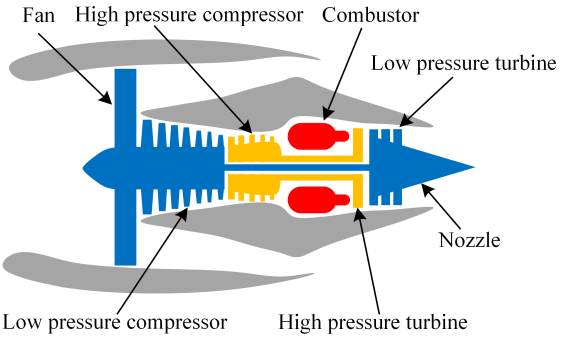



**FIGURE 4. A schematic of a turbofan engine.**

fouling, erosion of blades and seals, and blade tip clearance. As a result, aero-engines experience the gradual performance degradation throughout the whole life cycle. Health diagnosis attempts to monitor the current and past health status of a mechanical system according to the observable symptoms and plays a crucial role in mission scheduling and maintenance decision-making [26]. Health diagnosis techniques are primarily classified into model-based and data-driven methods. Model-based methods depend upon the availability of accurate mathematical models of aero-engines. The performance degradation of gas-path components can be described by the variation of health parameters, including thermodynamic efficiency and flow capacity [27], which are estimated using sequential Bayesian inference methods according to the available outputs of the system [28]–[30]. Due to advances in the sensing and communication technologies, data-driven approaches have been widely utilized to infer the underlying health status of a monitored system from the data collected from multiple sensors directly. The basic idea of data-driven methods is to transform health state diagnosis into a pattern recognition problem [31]–[33]. To be specific, a nonlinear

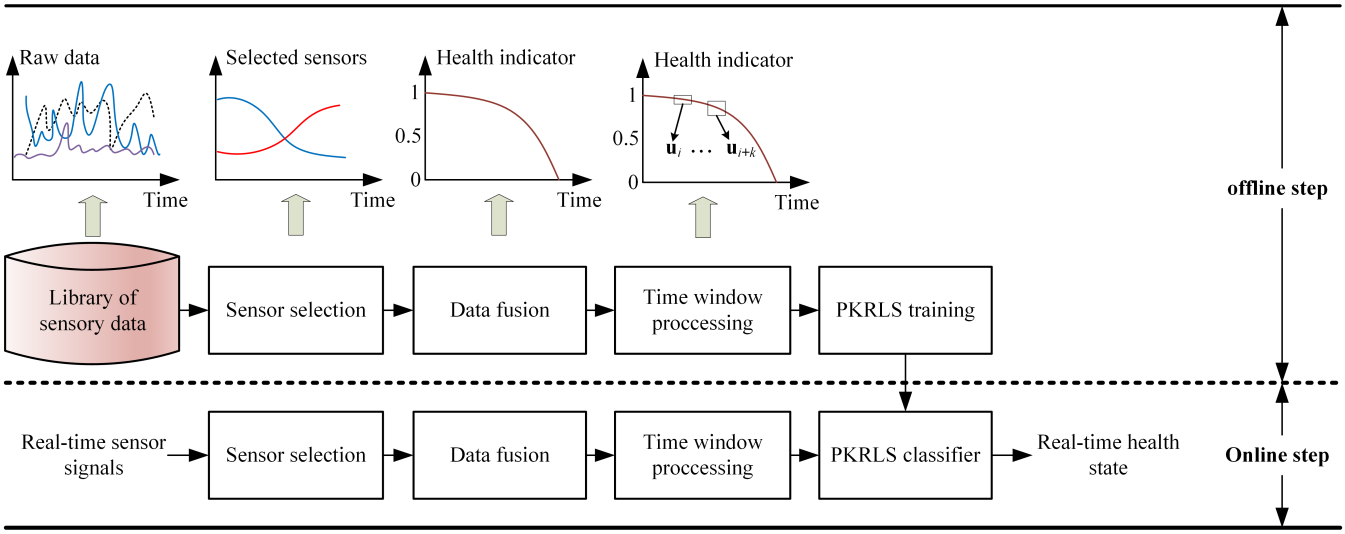



**FIGURE 5.** The schematic of the proposed multi-sensor health diagnosis approach.

relationship between the input patterns and the target class labels that represent different levels of degradation performance is learned based on a given training dataset and the trained health state classifier is employed to predict the class labels for the unknown patterns. The prediction phase is commonly carried out online so as to identify the health state of the degraded system continuously according to the real-time signals acquired from the multiple sensors. In this case, a classifier with simpler structure obtains faster predicting speed and becomes more favorable. Considering that PKRLS is efficient in modeling nonlinearity, it will be a viable choice for health diagnosis. Hence, a health diagnosis method using PKRLS is proposed. The novel method includes four steps as follows (also see Fig. 5)

1) Sensor selection, selecting the sensors closely related to the performance degradation.
2) Training dataset construction including sensory data fusion and time window processing.
3) Training a PKRLS classifier based on the training dataset.
4) Performing health diagnosis using the trained classifier.

### A. TRAINING DATASET CONSTRUCTION

#### 1) SENSORY DATA FUSION

In order to provide a better characterization of the degradation behavior of an aging engine, a simple linear regression method is used to fuse the multi-dimensional sensory data $\mathbf{y}_i = [y_i^1, \cdots, y_i^n]$ collected at the *i-th* operation cycle into a single health indicator (HI) $z_i$, which is defined in the closed interval [0, 1] [34], [35]. The data fusion model is expressed as

$$z_i = \mathbf{T}\mathbf{y}_i^T \tag{36}$$

where $\mathbf{T} \in \mathbb{R}^{1\times n}$ is the transformation vector. To obtain the linear model, the sensory data collected at the beginning and the end of an engineered system's life are first stored in two matrices $\mathbf{Y}_1 \in \mathbb{R}^{l_1\times n}$ and $\mathbf{Y}_0 \in \mathbb{R}^{l_0\times n}$, respectively. Then, the transformation vector $\mathbf{T}$ is calculated by

$$\mathbf{T} = (\mathbf{Y}^T\mathbf{Y})^{-1}\mathbf{Y}^T\mathbf{Z} \tag{37}$$

where $\mathbf{Y} = [\mathbf{Y}_1; \mathbf{Y}_0]$, $\mathbf{Z} = [\mathbf{Z}_1; \mathbf{Z}_0]$, $\mathbf{Z}_1 \in \mathbb{R}^{l_1\times 1}$ is a unity vector, and $\mathbf{Z}_0 \in \mathbb{R}^{l_0\times 1}$ is a zero vector. Once the transformation vector $\mathbf{T}$ is available, the data fusion model converts the real-time sensor signals into a noisy HI according to (36). The HI of a system is assumed to vary between 1 and 0 throughout the whole service life and its value represents different level of performance degradation. Specifically, a machine starts from a healthy state with an HI value of 1, and continues to run until a failure occurs, which lead to the value for HI reaching 0.

#### 2) TIME WINDOW PROCESSING

The health state of the monitored machine at a certain time step is determined by the current and historical operating conditions. Hence, the temporal dependencies between data points at neighboring time steps facilitates characterizing the evolution of the health state. Hence, a fixed-size sliding window that contains consecutive data points is constructed. To be specific, at the *j-th* time step, the HIs at the current and previous time steps are concatenated into a multi-dimensional feature vector $\mathbf{u}_i = [z_i, \cdots, z_{i-l}]$, where $l$ denotes the window length and it is fed into a health state classifier as the input.

#### 3) HEALTH STATE CLASSIFIER

Assume that an aero-engine experiences $k$ distinct health states throughout the whole service life. In order to fulfill this $k$-class classification task, $k$ binary classifiers $\{f^j : \mathbb{R}^l \to \{0, 1\}\}_{j=1}^k$ are trained by PKRLS and each one aims to judge whether the input data belongs to a certain class. The estimated class label is determined by the index of the classifier with the maximum output value.

### B. A CASE STUDY IN A TURBOFAN ENGINE DEGRADATION DATASET

A turbofan engine degradation dataset produced by commercial modular aero-propulsion system simulation (C-MAPSS) is utilized to illustrate the feasibility of the proposed health diagnosis method [36]. The C-MAPSS dataset contains four sub-datasets. Each sub-dataset includes a certain number of engine units. Each unit is represented by a multivariate time series produced by collecting the sensory data over a period of flight cycles. The data for each cycle contains the unit ID, operating cycle index, 3 values that indicates the operational settings and 21 sensor measurements contaminated by high level of unknown noises. Engine unit starts with different level of initial wear caused by manufacturing and assembly variations and ends until a failure occurs. The summary of the C-MAPSS degradation dataset is provided in Table 4.

**TABLE 4. Specifications of four sub-datasets in the C-MAPSS degradation dataset.**

| Sub-datasets | Number of fault modes | Number of operational condition | Number of units |
|---|---|---|---|
| FD001 | 1 | 1 | 100 |
| FD002 | 1 | 6 | 260 |
| FD003 | 2 | 1 | 100 |
| FD004 | 2 | 6 | 249 |

In this work, the 7 sensors including T24, T30, T50, P30, Ps30, phi and BPR, are selected [34]. It should be noted that the engine units in sub-datasets #2 and #4 run under six operational conditions. To eliminate the influence of the variation in operational conditions on sensory data, six different linear fusion models related to each operational condition are trained. The sensory data that locate in different condition regimes are converted into the HIs using the corresponding fusion model. Then, the constructed HI time series are transformed into the feature sequence by time window processing where the window length $l$ is set to 7. These run-to-failure data are labeled with 4 distinct health states according to their proximity to the failure time [32]. Finally, the whole dataset is divided into training and testing datasets. The health state classifier is trained based on the training dataset and validated using the testing dataset.

For the sake of comparison, the health state classifier is also approximated by various sequential learning algorithms and the classification results are listed in Table 5. PKRLS achieves the highest correct classification rates with the smallest network among all considered algorithms, which indicates that PKRLS is efficient and effective in inferring the latent health status of the aero-engines. Moreover, PKRLS also achieves the best real-time performance during the testing phase, which facilitates its application for estimating the health state of the complex systems online. In Fig. 6, the tendency of the classification accuracy versus the network size is presented. Compared with KRLS-ALD, PKRLS requires much less kernel units when reaching the blue dash line, which indicates the efficacy and feasibility of the pruning strategy.

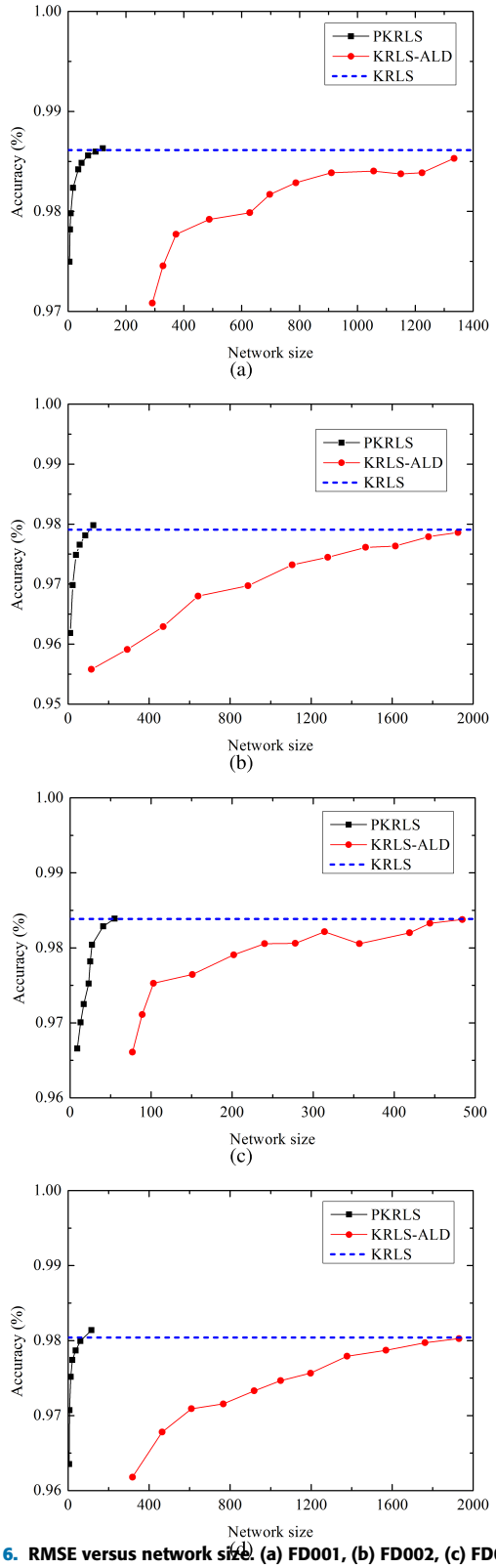



**FIGURE 6. RMSE versus network size. (a) FD001, (b) FD002, (c) FD003 and (d) FD004.**

**TABLE 5. Classification results on C-MAPSS degradation dataset.**

| Sub-datasets | Algorithms | trNum | teNum | trTime (s) | teTime (s) | Network size | Classification accuracy (%) |
|---|---|---|---|---|---|---|---|
| FD001 | KLMS | 2000 | 18031 | 0.2439 | 1.3437 | 2000 | 97.42 |
| | KRLS | 2000 | 18031 | 51.5505 | 1.3402 | 2000 | 98.61 |
| | KRLS-ALD | 2000 | 18031 | 64.4243 | 1.0270 | 1334 | 98.53 |
| | PKRLS | 2000 | 18031 | 20.4008 | 0.0868 | 120 | 98.63 |
| | OS-ELM | 2000 | 18031 | 22.9250 | 0.4948 | 1000 | 96.59 |
| FD002 | KLMS | 2000 | 50199 | 0.2491 | 3.8628 | 2000 | 97.16 |
| | KRLS | 2000 | 50199 | 50.3093 | 3.7930 | 2000 | 97.91 |
| | KRLS-ALD | 2000 | 50199 | 163.2806 | 3.6436 | 1925 | 97.86 |
| | PKRLS | 2000 | 50199 | 19.5690 | 0.2515 | 125 | 97.98 |
| | OS-ELM | 2000 | 50199 | 32.3684 | 1.5214 | 1200 | 94.48 |
| FD003 | KLMS | 2000 | 22120 | 0.2329 | 1.6322 | 2000 | 97.83 |
| | KRLS | 2000 | 22120 | 51.6326 | 1.6833 | 2000 | 98.39 |
| | KRLS-ALD | 2000 | 22120 | 12.1745 | 0.4224 | 484 | 98.38 |
| | PKRLS | 2000 | 22120 | 6.0239 | 0.0488 | 55 | 98.39 |
| | OS-ELM | 2000 | 22120 | 24.1880 | 0.6067 | 1000 | 96.68 |
| FD004 | KLMS | 2000 | 57755 | 0.2544 | 4.6388 | 2000 | 97.26 |
| | KRLS | 2000 | 57755 | 52.5977 | 4.6116 | 2000 | 98.04 |
| | KRLS-ALD | 2000 | 57755 | 167.7720 | 4.5088 | 1930 | 98.03 |
| | PKRLS | 2000 | 57755 | 19.1572 | 0.2869 | 115 | 98.14 |
| | OS-ELM | 2000 | 57755 | 34.6446 | 1.8705 | 1200 | 95.10 |

## VI. CONCLUSION

In online scenarios where the computational resources and memory storage are commonly limited, it is necessary to keep the network size upper bounded. Furthermore, a compact network can accelerate the learning speed at each iteration. Therefore, we propose a novel PKRLS algorithm by devising the effective pruning procedure. This pruning method can rank the centers in order of importance and remove the least important one from the existing dictionary, thereby curbing the network growth. The importance criterion is defined as the contribution of the center to the cost function. The computational burden of calculating the importance criterion is alleviated significantly by a speedup scheme. In addition, the data that are not informative enough for updating the network structure are utilized to adapt the coefficients of the network, which improves the utilization efficiency of the whole training dataset. Since PKRLS can adapt the learning strategies flexibly according to different training samples, it makes a good compromise between the computational complexity and the approximation accuracy.

Due to the wide utilization of multiple sensors for condition monitoring, massive data collected from mechanical systems become available. Nevertheless, how to fully capture the degradation pattern based on the large amount of sensory data in an efficient manner is an intricate problem. In view of its efficient learning procedure, PKRLS is used for health diagnosis. A case study in a turbofan engine degradation dataset shows the health state classifier based on PKRLS can recognize the health state with the satisfactory accuracy. It is noteworthy that the classifier is represented by the parsimonious network structure, which facilitates the application of the novel diagnosis method for estimating the health state of a system in real time.

## REFERENCES


[1] V. N. Vapnik, "An overview of statistical learning theory," *IEEE Trans. Neural Netw.*, vol. 10, no. 5, pp. 988–999, Sep. 1999.

[2] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Comput.*, vol. 10, no. 5, pp. 1299–1319, Jul. 1998.

[3] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K.-R. Müller, "Fisher discriminant analysis with kernels," in *Proc. IEEE Signal Process. Soc. Workshop Neural Netw. Signal Process.*, Aug. 1999, pp. 41–48.

[4] T. Tanaka, Y. Washizawa, and A. Kuh, "Adaptive kernel principal components tracking," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2012, pp. 1905–1908.

[5] Y. Washizawa, "Adaptive subset kernel principal component analysis for time-varying patterns," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 12, pp. 1961–1973, Dec. 2012.

[6] J. B. O. S. Filho and P. S. R. Diniz, "A fixed-point online kernel principal component extraction algorithm," *IEEE Trans. Signal Process.*, vol. 65, no. 23, pp. 6244–6259, Dec. 2017.

[7] W. Liu, P. P. Pokharel, and J. C. Príncipe, "The kernel least-mean-square algorithm," *IEEE Trans. Signal Process.*, vol. 56, no. 2, pp. 543–554, Feb. 2008.

[8] W. Liu and J. C. Príncipe, "Kernel affine projection algorithms," *EURASIP J. Adv. Signal Process.*, vol. 2008, no. 1, p. 784292, Mar. 2008, doi: 10.1155/2008/784292.

[9] 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.

[10] W. Liu, I. Park, Y. Wang, and J. C. Príncipe, "Extended kernel recursive least squares algorithm," *IEEE Trans. Signal Process.*, vol. 57, no. 10, pp. 3801–3814, Oct. 2009.

[11] N. Wang, M. J. Er, and M. Han, "Parsimonious extreme learning machine using recursive orthogonal least squares," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 10, pp. 1828–1841, Oct. 2014.

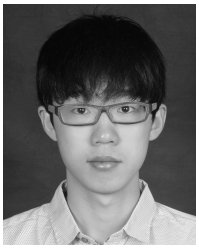
[12] J. Platt, "A resource-allocating network for function interpolation," *Neural Comput.*, vol. 3, no. 2, pp. 213–225, Jun. 1991, doi: 10.1162/neco.1991.3.2.213.

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

[14] W. Liu, I. Park, and J. C. Príncipe, "An information theoretic approach of designing sparse kernel adaptive filters," *IEEE Trans. Neural Netw.*, vol. 20, no. 12, pp. 1950–1961, Dec. 2009.

[15] B. Chen, S. Zhao, S. Seth, and J. C. Príncipe, "Online efficient learning with quantized klms and l1 regularization," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jun. 2012, pp. 1–6.

[16] B. Chen, N. Zheng, and J. C. Principe, "Sparse kernel recursive least squares using $L_1$ regularization and a fixed-point sub-iteration," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2014, pp. 5257–5261.
[17] S. van Vaerenbergh, J. Via, and I. Santamaria, "A sliding-window kernel RLS algorithm and its application to nonlinear channel identification," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, vol. 5, May 2006, pp. V-789–V-792.
[18] S. van Vaerenbergh, I. Santamaria, W. Liu, and J. C. Príncipe, "Fixed-budget kernel recursive least-squares," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2010, pp. 1882–1885.
[19] S. Zhao, B. Chen, P. Zhu, and J. C. Príncipe, "Fixed budget quantized kernel least-mean-square algorithm," *Signal Process.*, vol. 93, no. 9, pp. 2759–2770, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0165168413000662
[20] S. Zhao, B. Chen, Z. Cao, P. Zhu, and J. C. Príncipe, "Self-organizing kernel adaptive filtering," *EURASIP J. Adv. Signal Process.*, vol. 2016, no. 1, p. 106, Oct. 2016, doi: 10.1186/s13634-016-0406-3.
[21] B. Chen, S. Zhao, P. Zhu, and J. C. Principe, "Quantized kernel least mean square algorithm," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 1, pp. 22–32, Jan. 2012.
[22] B. Xi, L. Sun, B. Chen, J. Wang, N. Zheng, and J. C. Príncipe, "Density-dependent quantized kernel least mean square," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2016, pp. 3564–3569.
[23] 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.
[24] Y. Zheng, S. Wang, J. Feng, and K. T. Chi, "A modified quantized kernel least mean square algorithm for prediction of chaotic time series," *Digit. Signal Process.*, vol. 48, pp. 130–136, Jan. 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1051200415002833
[25] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan, "A fast and accurate online sequential learning algorithm for feedforward networks," *IEEE Trans. Neural Netw.*, vol. 17, no. 6, pp. 1411–1423, Nov. 2006.
[26] A. Heng, S. Zhang, A. C. C. Tan, and J. Mathew, "Rotating machinery prognostics: State of the art, challenges and opportunities," *Mech. Syst. Signal Process.*, vol. 23, no. 3, pp. 724–739, 2009.
[27] Y. G. Li and P. Nilkitsaranont, "Gas turbine performance prognostic for condition-based maintenance," *Appl. Energy*, vol. 86, no. 10, pp. 2152–2161, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0306261909000506
[28] H. Hanachi, J. Liu, A. Banerjee, and Y. Chen, "Sequential state estimation of nonlinear/non-Gaussian systems with stochastic input for turbine degradation estimation," *Mech. Syst. Signal Process.*, vols. 72–73, pp. 32–45, May 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0888327015004823
[29] F. Lu, J. Huang, and Y. Lv, "Gas path health monitoring for a turbofan engine based on a nonlinear filtering approach," *Energies*, vol. 6, no. 1, pp. 492–513, 2013.
[30] F. Lu, C. Jiang, J. Huang, and X. Qiu, "Aero engine gas path performance tracking based on multi-sensor asynchronous integration filtering approach," *IEEE Access*, vol. 6, pp. 28305–28317, May 2018.
[31] A. Widodo and B.-S. Yang, "Support vector machine in machine condition monitoring and fault diagnosis," *Mech. Syst. Signal Process.*, vol. 21, no. 6, pp. 2560–2574, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0888327007000027
[32] P. Tamilselvan and P. F. Wang, "Failure diagnosis using deep belief learning based health state classification," *Rel. Eng., Syst. Saf.*, vol. 115, pp. 124–135, Jul. 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0951832013000574
[33] C. Ma, X. Gu, and Y. Wang, "Fault diagnosis of power electronic system based on fault gradation and neural network group," *Neurocomputing*, vol. 72, no. 13, pp. 2909–2914, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231209001015
[34] T. Wang, J. Yu, D. Siegel, and J. Lee, "A similarity-based prognostics approach for remaining useful life estimation of engineered systems," in *Proc. IEEE Int. Conf. Prognostics Health Manage.*, Oct. 2008, pp. 1–6.
[35] J. Sun, H. Zuo, W. Wang, and M. G. Pecht, "Application of a state space modeling technique to system prognostics based on a health index for condition-based maintenance," *Mech. Syst. Signal Process.*, vol. 28, pp. 585–596, Apr. 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0888327011003979
[36] A. Saxena, K. Goebel, D. Simon, and N. Eklund, "Damage propagation modeling for aircraft engine run-to-failure simulation," in *Proc. IEEE Int. Conf. Prognostics Health Manage.*, Oct. 2008, pp. 1–9.



**HAOWEN ZHOU** received the B.S. degree in aircraft propulsion engineering from the Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2014, where he is currently pursuing the Ph.D. degree in aerospace propulsion theory and engineering.

His current research interests include gas turbine engine health prognostics, machine learning, and pattern recognition.

**JINQUAN HUANG** received the M.S. and Ph.D. degrees in aerospace propulsion system theory and engineering from the Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 1987 and 1998, respectively.

He has been a Professor with the College of Energy and Power Engineering, Nanjing University of Aeronautics and Astronautics, since 1999. His current research interests include gas turbine engine modeling, control, and health management.

**FENG LU** received the Ph.D. degree in system control and simulation from the Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2009. From 2012 to 2015, he was with the Aviation Motor Control System Institute as a Post-Doctoral Fellow, where he was involved in aircraft engine control system modeling and health monitoring. From 2016 to 2017, he was a Visiting Professor with the Department of Mechanical and Industrial Engineering, University of Toronto. Since 2009, he has been a Lecturer with the College of Energy and Power Engineering, Nanjing University of Aeronautics and Astronautics, where he has been an Associate Professor since 2013.

His current research interests include gas turbine engine modeling, control, and health prognostics.