Circuits, Systems, and Signal Processing (2019) 38:3133–3151
https://doi.org/10.1007/s00034-018-1006-2

Kernel Least Mean Square Based on the Nyström Method

Shi-Yuan Wang[1,2] · Wen-Yue Wang[1,2] · Lu-Juan Dang[1,2] · Yun-Xiang Jiang[3]

Received: 6 February 2018 / Revised: 11 December 2018 / Accepted: 13 December 2018 /
Published online: 18 December 2018

Abstract

The kernel least mean square (KLMS) algorithm is the simplest algorithm in kernel adaptive filters. However, the network growth of KLMS is still an issue for preventing its online applications, especially when the length of training data is large. The Nyström method is an efficient method for curbing the growth of the network size. In this paper, we apply the Nyström method to the KLMS algorithm, generating a novel algorithm named *kernel least mean square based on the Nyström method* (NysKLMS). In comparison with the KLMS algorithm, the proposed NysKLMS algorithm can reduce the computational complexity, significantly. The NysKLMS algorithm is proved to be convergent in the mean square sense when its step size satisfies some conditions. In addition, the theoretical steady-state excess mean square error of NysKLMS supported by simulations is calculated to evaluate the filtering accuracy. Simulations on system identification and nonlinear channel equalization show that the NysKLMS algorithm can approach the filtering performance of the KLMS algorithm by using much lower computational complexity, and outperform the KLMS with the novelty criterion, the KLMS with the surprise criterion, the quantized KLMS, the fixed-budget QKLMS, and the random Fourier features KLMS.

Keywords Kernel least mean square · Approximation · Nyström method · Random Fourier features

1 Introduction

Online kernel learning [13,18,31] is an efficient method for addressing the issues of classification and regression in nonlinear environments. A class of kernel adaptive filters (KAFs) is proposed using the online kernel learning method in the reproducing kernel Hilbert space (RKHS) [3,6,17,28]. The main idea of KAF is to transform the input data into a high-dimensional feature space and then perform traditional adaptive filters [30], e.g., the least mean square (LMS) algorithm, affine projection

✉ Shi-Yuan Wang
wsy@swu.edu.cn

Extended author information available on the last page of the article

algorithm (APA), and recursive least squares algorithm (RLS) in the RKHS. The reason for choosing the RKHS is that it can provide the linearity, convexity, and universal approximation capabilities [3,5,28]. To improve the calculation efficiency in the RKHS, the kernel trick based on the Mercer kernel [12,17,24,25,28] is used to calculate the inner product in the RKHS. The LMS, APA, and RLS algorithms are developed in the RKHS to generate the kernel least mean square (KLMS) algorithm [24,28], kernel affine projection algorithm (KAPA) [25,28], and kernel recursive least squares (KRLS) algorithm [12,28], respectively. The filtering performance of traditional adaptive filters in terms of the accuracy and robustness is therefore improved by KAFs. Among these aforementioned KAFs, the KLMS is simplest in both computational and space complexities.

However, the filter network of KAF [12,24,25,28] linearly grows with the length of training data, which incurs large computational burden. A sparsification method is therefore required to curb the growth of the network size by reserving the chosen input data as centers of the codebook. The commonly used sparsification methods include the novelty criterion (NC) [27], the prediction variance criterion (PVC) [10], the approximate linear dependency (ALD) criterion [12], the surprise criterion (SC) [23], and the quantized approach [8,9,15,19,22,26]. In these sparsification methods, redundant data are removed by setting different threshold parameters, thus leading to different sparsification networks and filtering performance. Among these sparsification methods, the quantized approach is the simplest and the most efficient method for constructing the filter network. In the quantized approach, only the Euclidean distance between the current input and the existing centers in the codebook is used as a criterion for constructing a sparsification network. Therefore, Chen et al. applied the quantized approach to the KLMS for obtaining the quantized KLMS (QKLMS) [8].

It is interesting to note that the network size based on the aforementioned sparsification methods is not fixed. However, in the case of limited computational capability and storage, the fixed-size network structure is necessary for practical applications. Generally, the methods based on a fixed-size network include the significance measure [33,38], the random Fourier features [4,36], and the Nyström method [11,21,35,36]. The significance measure keeps the codebook size fixed by pruning the least significant center when a new input is required to be added to the codebook. However, the calculation of significance requires extra computation. Unlike the significance measure, the random Fourier features and the Nyström method generate a vector representation of data to approximate the high-dimensional mapping. The random Fourier features approximate the kernel function using the kernel's Fourier transform with random features [14,29], and the Nyström method uses a low-rank matrix to approximate the kernel matrix. In addition, the chosen random Fourier features are independent of the input data [4,36]. And the Nyström method samples the inputs from the input data with some distribution, which is data dependent [11,21,35,36]. Therefore, due to the dependence of data, the Nyström method can provide a more efficient network structure than the random Fourier features [36]. The first two methods based on a fixed-size network have already been successfully applied to KAFs, e.g., the fixed-budget QKLMS (QKLMS-FB) [38], the simplified QKLMS-FB (SQKLMS-FB) [33], and the random Fourier features KLMS (RFFKLMS) [4].

In this paper, we propose a novel kernel least mean square algorithm based on the Nyström method (NysKLMS) by applying the approximation theory of the Nyström method to the KLMS. The Nyström method is used to approximate eigenfunctions in the RKHS with the chosen points [35], and thus, the input is transformed into a fixed-dimensional space to approximate the mapping in the RKHS. Therefore, the proposed NysKLMS algorithm provides a fixed network size in advance. To evaluate the filtering accuracy of NysKLMS, we also derive the steady-state excess mean square error (EMSE) theoretically.

2 KLMS

A continuous nonlinear input-output mapping $f\colon \mathbb{R}^D \rightarrow \mathbb{R}$ can be described as

$$d = f(\mathbf{x}), \tag{1}$$

where $\mathbf{x} \in \mathbb{X} \subseteq \mathbb{R}^D$ is the input vector and $d \in \mathbb{R}$ is the corresponding desired output. The input–output pairs $\{(\mathbf{x}(1), d(1)), (\mathbf{x}(2), d(2)), \ldots, (\mathbf{x}(n), d(n)), \ldots\}$ are used to estimate the mapping $f(\cdot)$, and the estimated output $y(n)$ of $d(n)$ is therefore obtained. An adaptive filter is constructed using a cost function based on the difference between the desired output and the estimated one.

The cost function of LMS based on the minimum mean square error (MMSE) criterion is [30]

$$J = \frac{1}{2}e(n)^2, \tag{2}$$

where the estimated error $e(n) = d(n) - y(n)$ at discrete time n is the difference between $d(n)$ and $y(n)$. The MMSE criterion is a common choice for adaptive filters due to its simplicity, and the algorithms mentioned in this paper are all equipped with (2).

Generally, the structure with the inner product $\mathbf{w}(n-1)^{\mathrm{T}}\mathbf{x}(n)$ is used in an adaptive filter, where $\mathbf{w}(n-1)$ is the weight vector and $(\cdot)^T$ is the transpose of a real vector. According to minimizing the cost function J in (2) at each iteration n, the optimal weight vector $\mathbf{w}^*$ is finally sought by the LMS. Due to the simpleness and efficiency, the gradient descent method [30] is used in the LMS to obtain the optimal solution to J in (2). Therefore, we can obtain the following update form of the weight vector in the LMS with the gradient descent method:

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \mu e(n)\mathbf{x}(n), \tag{3}$$

where μ is the step size.

When the relationship $f(\cdot)$ between the input and the output is highly nonlinear the KLMS can provide better filtering accuracy than the LMS. Actually, the KLMS [24,28] can be regarded as the LMS in the RKHS [3,6,17,28]. Assume that the weight vector at iteration n in the KLMS is denoted by $\mathbf{\Omega}(n)$. Similar to (3), the weight vector update of KLMS can be expressed as:

$$\mathbf{\Omega}(n) = \mathbf{\Omega}(n-1) + \mu e(n)\boldsymbol{\varphi}(\mathbf{x}(n)), \tag{4}$$

where $\boldsymbol{\varphi}\left(\mathbf{x}\left(n\right)\right)$ denotes the transformed input vector $\mathbf{x}\left(n\right)$. In the following, $\boldsymbol{\varphi}\left(\mathbf{x}\left(n\right)\right)$ is denoted by $\boldsymbol{\varphi}\left(n\right)$ for simplicity.

However, it is difficult to calculate $\boldsymbol{\varphi}(n)$ owing to its high dimension. Therefore, $\boldsymbol{\Omega}(n)$ cannot be computed directly using (4). Given an initial weight $\boldsymbol{\Omega}(0) = 0$, we obtain $\boldsymbol{\Omega}(n)$ by iterating

$$\boldsymbol{\Omega}(n) = \boldsymbol{\Omega}(0) + \mu \sum_{i=1}^{n} e(i)\boldsymbol{\varphi}(i) = \mu \sum_{i=1}^{n} e(i)\boldsymbol{\varphi}(i). \tag{5}$$

Then, the current estimated output can be derived as the following form of the inner product:

$$y(n) = \boldsymbol{\Omega}(n-1)^{\mathrm{T}}\boldsymbol{\varphi}(n) = \mu \sum_{i=1}^{n-1} e(i)\boldsymbol{\varphi}(i)^{\mathrm{T}}\boldsymbol{\varphi}(n). \tag{6}$$

The inner product in the RKHS can be easily computed by the kernel trick [12,24,25,28], i.e.,

$$\boldsymbol{\varphi}(n_1)^{\mathrm{T}}\boldsymbol{\varphi}\left(n_2\right) = \kappa(\mathbf{x}(n_1), \mathbf{x}(n_2)), \tag{7}$$

where $\kappa(\cdot)$ is a kernel function. The commonly used kernel function in the RKHS is the following Gaussian kernel [24,28]:

$$\kappa(\mathbf{x}(n_1), \mathbf{x}(n_2)) = \exp\left(-\frac{\|\mathbf{x}(n_1) - \mathbf{x}(n_2)\|^2}{2\sigma^2}\right), \tag{8}$$

where σ is the kernel width. It can be seen from (8) that the use of kernel function can avoid the direct calculation in the high-dimensional feature space.

Therefore, (6) can be rewritten using the kernel trick as

$$y(n) = \mu \sum_{i=1}^{n-1} e(i)\kappa(\mathbf{x}(i), \mathbf{x}(n)). \tag{9}$$

It can be seen from (9) that the KLMS produces a growing radial basis function network [12,24,25,28] by adding every new input as a center at each iteration. Therefore, the linear growth of the network increases the computational burden of KLMS and thus limits its online applications.

3 Proposed NysKLMS Algorithm

Actually, the transformed input $\boldsymbol{\varphi}(\mathbf{x})$ in the RKHS can be constructed as

$$\boldsymbol{\varphi}(\mathbf{x}) = \left[\sqrt{\lambda_1}\boldsymbol{\phi}_1(\mathbf{x}), \sqrt{\lambda_2}\boldsymbol{\phi}_2(\mathbf{x}), \ldots\right]^{\mathrm{T}}, \tag{10}$$

where λ_i and $\boldsymbol{\phi}_i, i = 1, 2, \ldots$, are the eigenvalues and the eigenfunctions of the Mercer kernel, respectively [28]. The dimension of $\boldsymbol{\varphi}(\mathbf{x})$ is infinite when the Gaussian kernel is adopted.

To avoid the calculation in an infinite dimension space and thus obtain a fixed-dimensional network, the Nyström method [11,35,36] is used to approximate the eigenfunctions with a fixed codebook. The eigenfunction problem can be described as follows:

$$\int \kappa(\mathbf{x}, \mathbf{x}')\boldsymbol{\phi}_i(\mathbf{x}')d\mathbf{x}' = \lambda_i \boldsymbol{\phi}_i(\mathbf{x}), \tag{11}$$

where $\mathbf{x}, \mathbf{x}' \in \mathbb{X}$. The codebook is composed of the randomly chosen m input vectors, i.e., $\mathbf{C} = \{\mathbf{x}(1), \mathbf{x}(2), \ldots, \mathbf{x}(m)\}$ [11]. Using the codebook to approximate the integral yields

$$\int \kappa(\mathbf{x}, \mathbf{x}')\boldsymbol{\phi}_i(\mathbf{x}')d\mathbf{x}' \approx \frac{1}{m}\sum_{k=1}^{m} \kappa(\mathbf{x}, \mathbf{x}(k))\boldsymbol{\phi}_i(\mathbf{x}(k)). \tag{12}$$

Therefore, (11) can be rewritten as

$$\frac{1}{m}\sum_{k=1}^{m} \kappa(\mathbf{x}, \mathbf{x}(k))\boldsymbol{\phi}_i(\mathbf{x}(k)) \approx \lambda_i \boldsymbol{\phi}_i(\mathbf{x}). \tag{13}$$

It is equivalent to solve the eigenfunction problem in a subspace spanned by the codebook. Eq. (13) is therefore described in a matrix form

$$\mathbf{G}\mathbf{U}_m = \mathbf{U}_m \boldsymbol{\Lambda}_m, \tag{14}$$

where $\mathbf{G}$ is the kernel matrix constructed by $\mathbf{G}_{ij} = \kappa(\mathbf{x}(i), \mathbf{x}(j))$ $(1 \leq i, j \leq m)$, $\boldsymbol{\Lambda}_m = \text{diag}(\lambda_1^{(m)}, \lambda_2^{(m)}, \ldots, \lambda_m^{(m)})$ is the diagonal matrix with the diagonal elements being the eigenvalues of $\mathbf{G}$ in a descending order, and $\mathbf{U}_m = (\mathbf{u}_1^{(m)}, \mathbf{u}_2^{(m)}, \ldots, \mathbf{u}_m^{(m)})$ is the matrix constructed by the corresponding eigenvectors of $\mathbf{G}$. Rearranging (13) and (14), we have the following approximations:

$$\lambda_i \approx \lambda_i^{(m)}/m, \tag{15}$$

$$\boldsymbol{\phi}_i(\mathbf{x}(j)) \approx \sqrt{m}\mathbf{u}_i^{(m)}(j). \tag{16}$$

Based on (15) and (16), the approximated eigenfunction [11] can be derived from (13) as

$$\hat{\boldsymbol{\phi}}_i(\mathbf{x}) = \frac{\sqrt{m}}{\lambda_i^{(m)}}\sum_{k=1}^{m} \kappa(\mathbf{x}, \mathbf{x}(k))\mathbf{u}_i^{(m)}(k). \tag{17}$$

Note that (17) can be regarded as a mapping of $\mathbf{x}$ into the ith eigenvector. Substituting (15) and (17) into (10) and using m eigenvalues and eigenvectors, we can derive an approximated transformation of input $\mathbf{x}(n)$ with a fixed-dimensional representation as follows:

$$\begin{aligned}
&\mathbf{z}(\mathbf{x}(n)) \\
&\quad= \left[\sqrt{\lambda_1}\hat{\boldsymbol{\phi}}_1(\mathbf{x}(n)), \sqrt{\lambda_2}\hat{\boldsymbol{\phi}}_2(\mathbf{x}(n)), \ldots, \sqrt{\lambda_m}\hat{\boldsymbol{\phi}}_m(\mathbf{x}(n))\right]^{\mathrm{T}} \\
&\quad= \boldsymbol{\Lambda}_m^{-\frac{1}{2}}\mathbf{U}_m^{\mathrm{T}}[\kappa(\mathbf{x}(n), \mathbf{x}(1)), \kappa(\mathbf{x}(n), \mathbf{x}(2)), \ldots, \kappa(\mathbf{x}(n), \mathbf{x}(m))]^{\mathrm{T}} \\
&\quad= \boldsymbol{\Lambda}_m^{-\frac{1}{2}}\mathbf{U}_m^{\mathrm{T}}\boldsymbol{\kappa}(n) \\
&\quad= \mathbf{P}\boldsymbol{\kappa}(n),
\end{aligned} \tag{18}$$

where $\mathbf{P} = \boldsymbol{\Lambda}_m^{-\frac{1}{2}}\mathbf{U}_m^{\mathrm{T}}$, $\boldsymbol{\kappa}(n) = [\kappa(\mathbf{x}(n), \mathbf{x}(1)), \kappa(\mathbf{x}(n), \mathbf{x}(2)), \ldots, \kappa(\mathbf{x}(n), \mathbf{x}(m))]^{\mathrm{T}}$. Denote $\mathbf{z}(n) = \mathbf{z}(\mathbf{x}(n))$ for simplicity. The subspace which $\mathbf{z}(n)$ belongs to is spanned by the codebook and has a fixed dimension. Therefore, the adaptive filter based on the transformed input $\mathbf{z}(n)$ provides a linear filter structure. The approximated degree can be evaluated by the difference between $\mathbf{z}(n)^{\mathrm{T}}\mathbf{z}(n)$ and $\kappa(\mathbf{x}(n), \mathbf{x}(n))$ as shown in [21].

Applying the gradient descent method [30] to the transformed input $\mathbf{z}(n)$ yields the weight update of NysKLMS as

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \mu e(n)\mathbf{z}(n). \tag{19}$$

In the feature space, the estimated output at the nth iteration is derived by

$$y(n) = \mathbf{w}(n-1)^{\mathrm{T}}\mathbf{z}(n). \tag{20}$$

Compared with (9), the estimated output of NysKLMS shown in (20) is obtained in a fixed-size network. We summarize the proposed NysKLMS algorithm in Algorithm 1. Considering online applications, we choose the first m input training data as the codebook, which is reasonable when the inputs are independent identically distributed (i.i.d.) [21].

Remark 1 The proposed NysKLMS algorithm inherently generates a sparsification structure due to a small codebook used, which is appropriate for online applications. In addition, The NysKLMS only uses a filter network of a fixed dimension and approaches the filtering performance of the KLMS with no sparsification.

Let N be the length of training data and m be the chosen codebook size, where $m \ll N$. Denote the magnitude of complexity by $O(\cdot)$. There exists the eigenvalue decomposition with the computational complexity $O(m^3)$ in the NysKLMS, which can be ignored reasonably since the decomposition is calculated once in the whole training process. Therefore, the computational complexity of NysKLMS is $O(m)$ which is equivalent to those of the sparsification algorithms, i.e., the QKLMS, the QKLMS-FB, and the RFFKLMS. Compared with the KLMS having the computational complexity $O(i)$ $(0 < i \leq N)$, the NysKLMS reduces the computational complexity with $m \ll N$, significantly.

Algorithm 1: Kernel Least Mean Square Algorithm Based on the Nyström Method (NysKLMS)

```
Input: ${\mathbf{x}(n), d(n)}, n = 1, 2, \ldots, N$

Initialization:
    Choose step size $\mu$, kernel width $\sigma$, and $\mathbf{w}(0) = 0$; initialize $\mathbf{C}$ with $m$ sampled inputs and compute $\mathbf{P}$.

Computation:
    while ${\mathbf{x}(n), d(n)}$ available do
        1) Compute the representation of the input: $\mathbf{z}(n) = \mathbf{P}\boldsymbol{\kappa}(n)$ using (18)
        2) Compute the error: $e(n) = d(n) - \mathbf{w}(n-1)^{\mathrm{T}}\mathbf{z}(n)$
        3) Compute the weight: $\mathbf{w}(n) = \mathbf{w}(n-1) + \mu e(n)\mathbf{z}(n)$
    end while
```

4 Mean Square Convergence Analysis

Suppose that f^* is the unknown mapping required to learn. The desired output can be modeled by

$$d(n) = f^*(\mathbf{x}(n)) + v(n), \tag{21}$$

where $v(n)$ denotes the disturbance noise.

Since $\mathbf{z}(n)$ is an approximation of $\boldsymbol{\varphi}(n)$, there exists the optimal weight vector that lies in the subspace spanned by the transformed input sequence thanks to the universal approximation property in the RKHS [3,5,7]. For the tractability of theoretical analysis, we first assume that $\mathbf{z}(n)$ approximates $\boldsymbol{\varphi}(n)$ well and the optimal weight vector $\mathbf{w}^*$ exists in the following analysis.

The following reasonable assumptions are used for tractable analysis.

A1: The noise $v(n)$ is zero mean and i.i.d. with variance $\sigma_v^2{=}E\left[v^2(n)\right]$.

A2: The noise $v(n)$ is independent of the input data $\mathbf{x}(n)$, which can show that $v(n)$ is independent of the *a priori* estimation error $e_a(n)$.

A3: At steady state, $\|\mathbf{z}(n)\|^2$ is independent of $e_a(i)$.

These assumptions are frequently used in adaptive filters [1,7,8,16,20,30,37].

4.1 Stability Analysis

First, we derive the energy conservation relation for the NysKLMS. According to the optimal weight vector $\mathbf{w}^*$, we have $f^*(\mathbf{x}(n)) = \mathbf{w}^{*\mathrm{T}}\mathbf{z}(n)$. Therefore, (21) can be rewritten as

$$d(n) = \mathbf{w}^{*\mathrm{T}}\mathbf{z}(n) + v(n). \tag{22}$$

The estimated error can be derived as

$$\begin{aligned} e(n) &= d(n) - y(n) \\ &= (\mathbf{w}^{*\mathrm{T}}\mathbf{z}(n) + v(n)) - \mathbf{w}(n-1)^{\mathrm{T}}\mathbf{z}(n) \\ &= \tilde{\mathbf{w}}(n-1)^{\mathrm{T}}\mathbf{z}(n) + v(n) \\ &= e_a(n) + v(n), \end{aligned} \tag{23}$$

where $\tilde{\mathbf{w}}(n-1) = \mathbf{w}^* - \mathbf{w}(n-1)$ denotes the weight error vector and $e_a(n) = \tilde{\mathbf{w}}(n-1)^{\mathrm{T}}\mathbf{z}(n)$ the *a priori* error.

Subtracting $\mathbf{w}^*$ from both sides of (19) yields

$$\tilde{\mathbf{w}}(n) = \tilde{\mathbf{w}}(n-1) - \mu e(n)\mathbf{z}(n). \tag{24}$$

Let $e_p(n) = \tilde{\mathbf{w}}(n)^{\mathrm{T}}\mathbf{z}(n)$ be the *a posteriori* error. The relation between $e_p(n)$ and $e_a(n)$ is derived from (24) as

$$e_p(n) = e_a(n) - \mu e(n)\mathbf{z}(n)^{\mathrm{T}}\mathbf{z}(n). \tag{25}$$

Substituting (25) into (24) to eliminate $e(n)$, we get

$$\tilde{\mathbf{w}}(n) = \tilde{\mathbf{w}}(n-1) + (e_p(n) - e_a(n))\frac{\mathbf{z}(n)}{\mathbf{z}(n)^{\mathrm{T}}\mathbf{z}(n)}. \tag{26}$$

Squaring both sides of (26) yields

$$\begin{aligned} \tilde{\mathbf{w}}(n)^{\mathrm{T}}\tilde{\mathbf{w}}(n) &= \left[\tilde{\mathbf{w}}(n-1) + (e_p(n) - e_a(n))\frac{\mathbf{z}(n)}{\mathbf{z}(n)^{\mathrm{T}}\mathbf{z}(n)}\right]^{\mathrm{T}} \\ &\quad \times \left[\tilde{\mathbf{w}}(n-1) + (e_p(n) - e_a(n))\frac{\mathbf{z}(n)}{\mathbf{z}(n)^{\mathrm{T}}\mathbf{z}(n)}\right] \\ &= \tilde{\mathbf{w}}(n-1)^{\mathrm{T}}\tilde{\mathbf{w}}(n-1) + \frac{e_p^2(n) - e_a^2(n)}{\mathbf{z}(n)^{\mathrm{T}}\mathbf{z}(n)}. \end{aligned} \tag{27}$$

Denote the weight error power (WEP) at the nth iteration by $\|\tilde{\mathbf{w}}(n)\|^2 = \tilde{\mathbf{w}}(n)^{\mathrm{T}}\tilde{\mathbf{w}}(n)$. (27) can be rewritten as

$$\|\tilde{\mathbf{w}}(n)\|^2 + \frac{e_a^2(n)}{\|\mathbf{z}(n)\|^2} = \|\tilde{\mathbf{w}}(n-1)\|^2 + \frac{e_p^2(n)}{\|\mathbf{z}(n)\|^2}. \tag{28}$$

Taking expectations of both sides of (28), we obtain the energy conservation relation of NysKLMS by

$$E\left[\|\tilde{\mathbf{w}}(n)\|^2\right] + E\left[\frac{e_a^2(n)}{\|\mathbf{z}(n)\|^2}\right] = E\left[\|\tilde{\mathbf{w}}(n-1)\|^2\right] + E\left[\frac{e_p^2(n)}{\|\mathbf{z}(n)\|^2}\right]. \tag{29}$$

Combining (25) and (29) to eliminate $e_p(n)$ yields

$$E\left[\|\tilde{\mathbf{w}}(n)\|^2\right] = E\left[\|\tilde{\mathbf{w}}(n-1)\|^2\right] + \mu^2 E\left[e^2(n)\|\mathbf{z}(n)\|^2\right] - 2\mu E\left[e(n)e_a(n)\right]. \tag{30}$$

Substituting (23) into (30), we obtain

$$E\left[\|\tilde{\mathbf{w}}(n)\|^2\right] = E\left[\|\tilde{\mathbf{w}}(n-1)\|^2\right] + \mu^2 E\left[(e_a(n)+v(n))^2\|\mathbf{z}(n)\|^2\right] - 2\mu E\left[(e_a(n)+v(n))e_a(n)\right]. \tag{31}$$

In (31), according to Assumptions **A1** and **A2**, we have

$$2\mu E\left[(e_a(n)+v(n))e_a(n)\right] = 2\mu E\left[e_a^2(n)\right]. \tag{32}$$

Further, according to Assumptions **A2** and **A3**, we obtain

$$\mu^2 E\left[(e_a(n)+v(n))^2\|\mathbf{z}(n)\|^2\right] = \mu^2\left(E\left[e_a^2(n)\right] + E\left[v^2(n)\right]\right) E\left[\|\mathbf{z}(n)\|^2\right]. \tag{33}$$

According to $\sigma_v^2 = E[v^2(n)]$, (33) can be rewritten as

$$\mu^2 E\left[(e_a(n)+v(n))^2\|\mathbf{z}(n)\|^2\right] = \mu^2\left(E\left[e_a^2(n)\right] + \sigma_v^2\right) E\left[\|\mathbf{z}(n)\|^2\right]. \tag{34}$$

Therefore, (31) is derived as

$$E\left[\|\tilde{\mathbf{w}}(n)\|^2\right] = E\left[\|\tilde{\mathbf{w}}(n-1)\|^2\right] + \mu^2\left(E\left[e_a^2(n)\right] + \sigma_v^2\right) E\left[\|\mathbf{z}(n)\|^2\right] - 2\mu E\left[e_a^2(n)\right]. \tag{35}$$

To ensure a convergence solution, the WEPs at the nth and $(n-1)$th iterations satisfy

$$E\left[\|\tilde{\mathbf{w}}(n)\|^2\right] \le E\left[\|\tilde{\mathbf{w}}(n-1)\|^2\right]. \tag{36}$$

Thus, substituting (36) into (35) gives

$$\mu^2\left(E\left[e_a^2(n)\right] + \sigma_v^2\right) E\left[\|\mathbf{z}(n)\|^2\right] - 2\mu E\left[e_a^2(n)\right] \le 0. \tag{37}$$

To make the weight error vector monotonically decrease, the step size should satisfy

$$0 \le \mu \le \frac{2E\left[e_a^2(n)\right]}{\left(E\left[e_a^2(n)\right] + \sigma_v^2\right) E\left[\|\mathbf{z}(n)\|^2\right]}. \tag{38}$$

4.2 Steady-State Mean Square Performance

To measure the steady-state performance of NysKLMS, we denote excess mean square error (EMSE) by $S = \lim_{n\to\infty} E\left[e_a^2(n)\right]$.

Taking limits of both sides of (35) as $n \to \infty$ yields

$$\begin{aligned}\lim_{n\to\infty} E\left[\|\tilde{\mathbf{w}}(n)\|^2\right] = \lim_{n\to\infty} E\left[\|\tilde{\mathbf{w}}(n-1)\|^2\right] \\ + \mu^2\left(S+\sigma_v^2\right)\lim_{n\to\infty} E\left[\|\mathbf{z}(n)\|^2\right] - 2\mu S.\end{aligned} \tag{39}$$

Define the autocorrelation matrix of $\mathbf{z}(n)$ by $\mathbf{R}_{zz} = E\left[\mathbf{z}(n)\mathbf{z}(n)^{\mathrm{T}}\right]$. And we have the trace of $\mathbf{R}_{zz}$ as $\mathrm{Tr}(\mathbf{R}_{zz}) = E\left[\|\mathbf{z}(n)\|^2\right]$. Supposing the WEP reaches a steady-state value, we obtain

$$\lim_{n\to\infty} E\left[\|\tilde{\mathbf{w}}(n)\|^2\right] = \lim_{n\to\infty} E\left[\|\tilde{\mathbf{w}}(n-1)\|^2\right]. \tag{40}$$

Hence, we have

$$\mu^2\left(S+\sigma_v^2\right)\mathrm{Tr}\left(\mathbf{R}_{zz}\right) = 2\mu S. \tag{41}$$

According to (41), the EMSE of NysKLMS can be derived as

$$S = \frac{\mu\sigma_v^2 \mathrm{Tr}\left(\mathbf{R}_{zz}\right)}{2-\mu\mathrm{Tr}\left(\mathbf{R}_{zz}\right)}. \tag{42}$$

It can be seen from (42) that S is related to the step size, noise variance, and $\mathrm{Tr}(\mathbf{R}_{zz})$. For comparison, the EMSE of KLMS in [7] is shown as follows:

$$S_1 = \frac{\mu\sigma_v^2}{2-\mu}. \tag{43}$$

Comparing (42) with (43), we see that that the NysKLMS can achieve the same steady-state performance as the KLMS on the condition of $\mathrm{Tr}(\mathbf{R}_{zz}) \to 1$.

Remark 2 The derivation of S is based on the assumption that $\mathbf{z}(n)$ approximates $\boldsymbol{\varphi}(n)$ well. We have $\mathrm{Tr}(\mathbf{R}_{zz}) = 1$ due to $\kappa(\mathbf{x}_n, \mathbf{x}_n) \equiv 1$. It can be seen from (18) that $\mathbf{z}(n)$ is related to the parameters of m and σ. Therefore, m and σ are chosen to satisfy $\mathrm{Tr}(\mathbf{R}_{zz}) \to 1$, and the EMSE of NysKLMS is verified on this condition.

5 Simulation Results

Simulations on system identification and nonlinear channel equalization are conducted to validate the performance of NysKLMS in the Gaussian and uniform noise environments. The representative algorithms, i.e., the KLMS-NC [27], the KLMS-SC [23], the QKLMS [8], the QKLMS-FB [38], the RFFKLMS [4], and the KLMS [24,28], are chosen for comparison. The KLMS-NC, the KLMS-SC, and the QKLMS are the

KLMS algorithms with sparsification. The QKLMS-FB and the RFFKLMS are the KLMS algorithms in the fixed-size network. The KLMS with no sparsification is used as the reference of performance comparison.

To evaluate the filtering performance, the mean square error (MSE) is defined as

$$\mathrm{MSE} = \frac{1}{T}\left(\sum_{n=1}^{\mathrm{T}} (d(n) - y(n))^2\right), \tag{44}$$

where T is the length of testing data. For all simulations, 50 Monte Carlo runs are performed to reduce the disturbance.

5.1 System Identification

The block diagram of system identification with finite impulse response (FIR) is shown in Fig. 1 [2]. The desired output can be expressed as

$$d(n) = \mathbf{w}^{*\mathrm{T}}\mathbf{x}(n) + v(n), \tag{45}$$

where $\mathbf{x}(n)$ is the input vector including delays of $x(n)$ and $\mathbf{w}^* = [w_1^*, w_2^*, \ldots, w_L^*]^{\mathrm{T}}$ denotes the optimal weight vector with L taps of the unknown system.

In the following simulations, the weight vector of the FIR filter is set as $\mathbf{w}^*$=[0.227, 0.460, 0.688, 0.460, 0.227]$^{\mathrm{T}}$ with $L = 5$. The input signal $x(n)$ is a Gaussian process with zero mean and unit variance.

First, to discuss the parameters selection, the relation between $\mathrm{Tr}(\mathbf{R}_{zz})$ and m and σ is plotted in Fig. 2, which is calculated by averaging 2000 training data with different m and σ. In Fig. 2, the x-axis denotes m with the range from 20 to 380 in 40 increments. The y-axis denotes σ with the range from 1 to 4 in 0.2 increments. The z-axis shows the value of $\mathrm{Tr}(\mathbf{R}_{zz})$ in a color-mapped form. From Fig. 2, we find that $\mathrm{Tr}(\mathbf{R}_{zz})$ gradually approaches 1 with the increase in m and σ. When $\mathrm{Tr}(\mathbf{R}_{zz})$ is greater than 0.995, it is approximately equal to 1. Therefore, from the region of $\mathrm{Tr}(\mathbf{R}_{zz}) > 0.995$, we choose $m = 100$ and $\sigma = 3$ for the NysKLMS to approximate the KLMS.

To verify the steady-state mean square performance of NysKLMS, the training data of length 200,000 are used to ensure its convergence, and its EMSE is obtained by the average over the last 500 iterations.

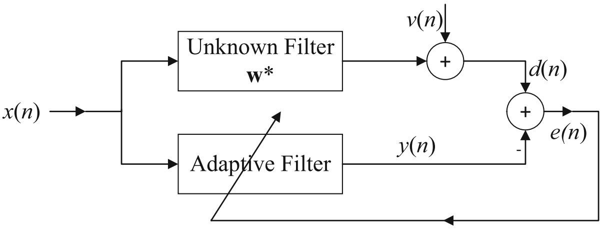


Fig. 1 System identification based on adaptive filter

Fig. 2 Tr($\mathbf{R}_{zz}$) versus m and σ

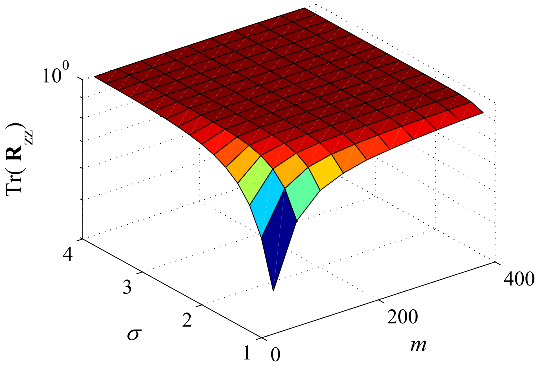
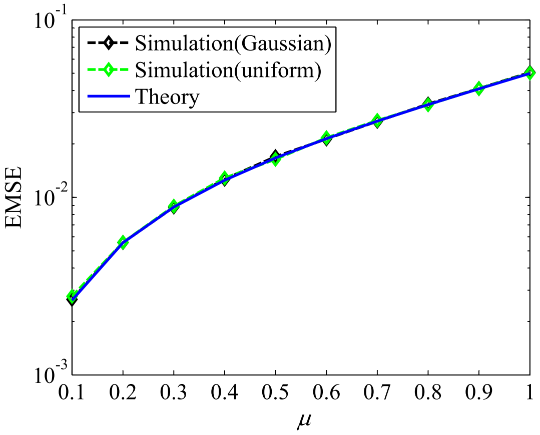


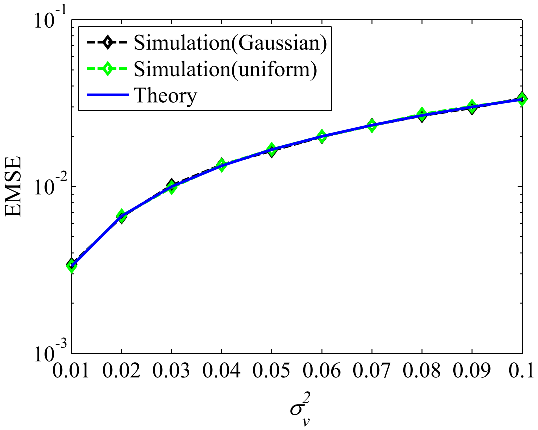
Fig. 3 Simulated and theoretical EMSEs versus μ ($\sigma_v^2 = 0.05$)



The simulated and theoretical EMSEs of NysKLMS versus step size μ and noise variance σ_v^2 in the Gaussian and uniform noises are shown in Figs. 3 and 4. From these two figures, we see that the simulated and theoretical EMSEs agree well. With the increase in μ and σ_v^2, the steady-state EMSEs of NysKLMS is decreasing. The filtering performance of NysKLMS is decided by the step size and noise variance on the condition of Tr($\mathbf{R}_{zz}$) $\to 1$.

Then, we compare the learning curves of NysKLMS, KLMS-NC, KLMS-SC, QKLMS, QKLMS-FB, RFFKLMS, and KLMS in the presence of Gaussian noise. The Gaussian noise is zero mean with variance 0.01. In the simulations, a segment of 5000 points of the input sequence is chosen as the training data and 100 points as the testing data. The compared results are shown in Fig. 5. To guarantee the same codebook size and initial convergence rate, the parameters of these algorithms are configured as follows. The quantization sizes are set as 2.1 and 2 in the QKLMS and the QKLMS-FB, respectively. The step size and the kernel width are set as $\mu = 0.1$ and $\sigma = 3$, respectively, for all algorithms except for the KLMS-NC and the KLMS-SC with $\mu = 0.5$ and $\sigma = 1$. The thresholds in the KLMS-NC are $\delta_1 = 1.45$ and $\delta_2 = 0.08$. The thresholds in the KLMS-SC are $T_1 = 200$ and $T_2 = 0.55$, and the regularization parameter is set as 0.01. It can be seen from Fig. 5 that the learning curve of NysKLMS

Fig. 4 Simulated and theoretical EMSEs versus σ_v^2 ($\mu = 0.5$)



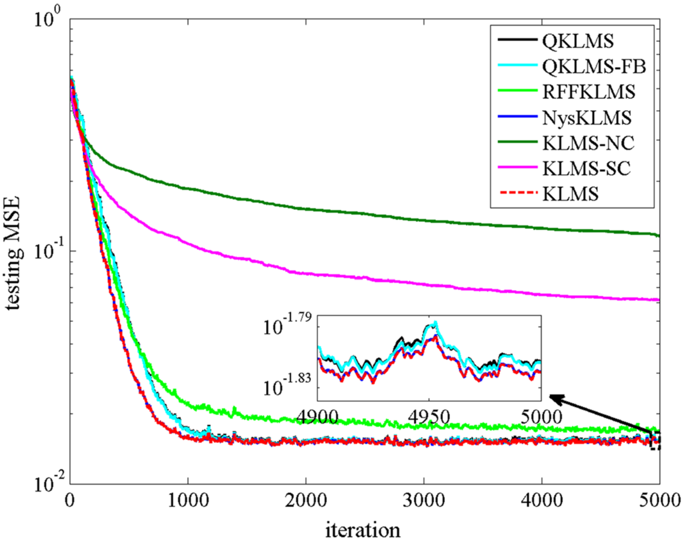


Fig. 5 Learning curves in terms of the testing MSE of different algorithms with $m = 100$

is almost the same as that of KLMS, which validates that the NysKLMS can approach the filtering performance of the KLMS with appropriate parameters. However, the NysKLMS only uses 100 centers in the codebook which is much smaller than 5000 centers in the KLMS. Hence, the NysKLMS with lower computational complexity approaches the filtering performance of the KLMS. In addition, the QKLMS and the QKLMS-FB also achieve the similar testing MSE to the KLMS and the NysKLMS and outperform the KLMS-NC and the KLMS-SC. The NysKLMS and the KLMS have faster convergence rate than the QKLMS, the QKLMS-FB, and the RFFKLMS in the beginning. The mean consumed time and testing MSEs of all algorithms in the steady state are shown in Table 1. Note that the numbers in italics in all the tables indicate the minima among the corresponding columns. It can be seen from this table

Table 1 Mean consumed time and testing MSE in the steady state with $m = 100$

Algorithm	Consumed time (s)	Testing MSE
QKLMS	1.71	0.01552
QKLMS-FB	8.60	0.01549
RFFKLMS	0.15	0.01703
NysKLMS	0.18	0.01528
KLMS-NC	*0.10*	0.11754
KLMS-SC	0.37	0.06183
KLMS	38.72	*0.01526*

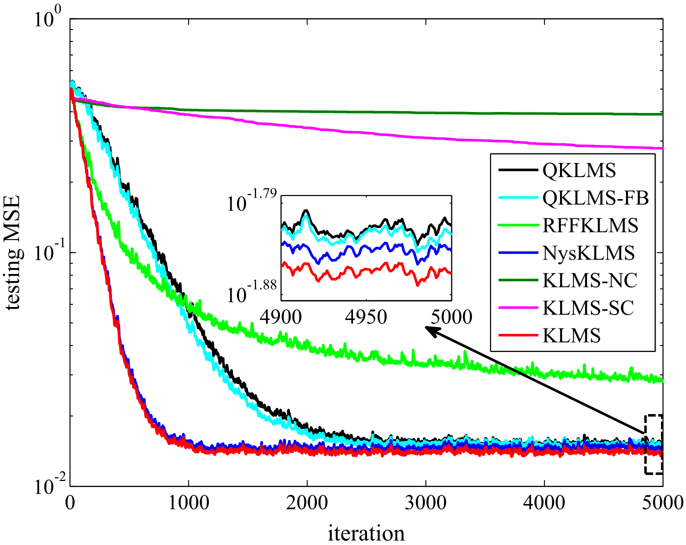


Fig. 6 Learning curves in terms of the testing MSE of different algorithms with $m = 20$

that the KLMS-NC consumes the least time, but achieves the highest testing MSE compared with other algorithms. The testing MSE of NysKLMS approaches that of KLMS. Therefore, on the condition of a fixed network size, the NysKLMS consumes less time than the QKLMS, the QKLMS-FB, the KLMS-SC, and the KLMS.

Finally, we choose a smaller codebook size $m = 20$ to observe the filtering performance of the aforementioned algorithms with the same step size and Gaussian noise. The quantization sizes are set as 5.3 and 5 in the QKLMS and the QKLMS-FB, respectively. The thresholds in the KLMS-NC are $\delta_1 = 2.4$ and $\delta_2 = 0.12$. The thresholds in the KLMS-SC are $T_1 = 200$ and $T_2 = 2.2$. We see from Fig. 6 that the NysKLMS with 20 centers approaches the filtering performance of the KLMS with 5000 centers. The computational complexity is therefore reduced by the NysKLMS, significantly. Moreover, the NysKLMS has a smaller testing MSE than the RFFKLMS with the same codebook size, which means that the NysKLMS is more efficient than the RFFKLMS in system identification. Comparing the NysKLMS with the QKLMS and the QKLMS-FB, we see that the NysKLMS generates faster convergence rate but

Table 2 Mean consumed time and testing MSE in the steady state with $m = 20$

Algorithm	Consumed time (s)	Testing MSE
QKLMS	1.14	0.0153
QKLMS-FB	3.69	0.0150
RFFKLMS	0.13	0.0286
NysKLMS	0.13	0.0145
KLMS-NC	*0.03*	0.3904
KLMS-SC	0.27	0.2792
KLMS	39.78	*0.0139*

a similar testing MSE in the steady state. The KLMS-NC and the KLMS-SC incur severe performance degradation with a smaller codebook size. The mean consumed time and testing MSEs of all algorithms are shown in Table 2. It can be seen from this table that the NysKLMS with the same consumed time as the RFFKLMS consumes less time than other algorithms except for the KLMS-NC and simultaneously approaches the filtering performance of KLMS.

5.2 Nonlinear Channel Equalization

The nonlinear channel equalization [28,34] is often used in such fields as modeling digital satellite communication channels and digital magnetic recording channels. As Fig. 7 shows, this kind of model is a combination of a linear filter and a memoryless nonlinearity. The aim of the channel equalization is to recover the input signal $s(n)$ with a low error rate according to the observed output $r(n)$ disturbed by a noise $v(n)$. $p(n)$ is defined by $p(n) = s(n) + 0.5s(n-1)$, and $r(n)$ is given by $r(n) = p(n) - 0.9p(n)^2 + v(n)$. In this simulation, the data points for regression are constructed as $\{\mathbf{x}(n), d(n)\} = \{[r(n), r(n+1), \ldots, r(n+5)]^{\mathrm{T}}, s(n-2)\}$, where a segment of 5000 points is chosen as the training data and the following 100 points as the testing data. In the following simulations, the Gaussian noise is zero mean with variance 0.1.

The compared results with the same codebook size $m = 200$ are shown in Fig. 8. The parameters of algorithms are configured for the same initial convergence rate and codebook size. The step size is set as $\mu = 0.5$ for all algorithms. The kernel width is $\sigma = 0.7$ for the KLMS-NC and the KLMS-SC and $\sigma = 2$ for others. The quantization sizes are set as 1.2 and 1 in the QKLMS and the QKLMS-FB, respectively. The thresholds are $\delta_1 = 1.1$ and $\delta_2 = 0.08$ for the KLMS-NC and $T_1 = 1$ and $T_2 = 0.3$ for the KLMS-SC. It can be seen from Fig. 8 that the NysKLMS with efficient approximation representation of the input, can provide better filtering performance than the QKLMS, the QKLMS-FB, the KLMS-NC, the KLMS-SC, and the RFFKLMS. And

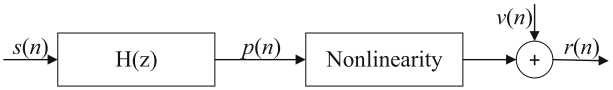


Fig. 7 Structure of a nonlinear channel

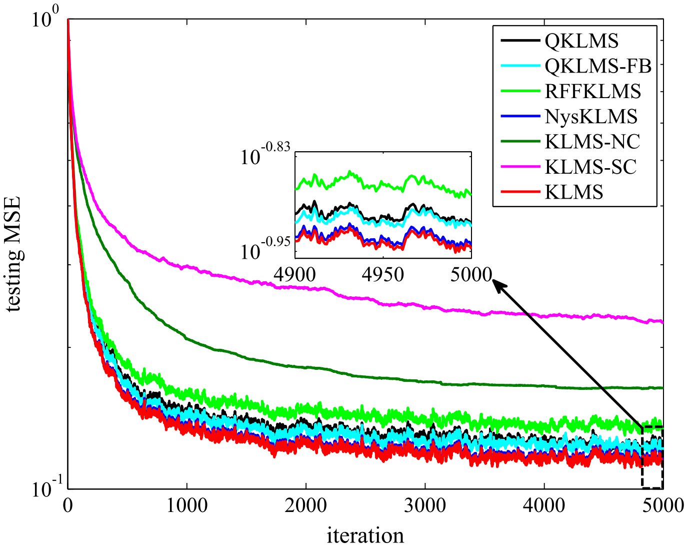


Fig. 8 Learning curves in terms of the testing MSE of different algorithms with the same codebook size

Table 3 Mean consumed time and testing MSE in the steady state with $m = 200$

Algorithm	Consumed time (s)	Testing MSE
QKLMS	4.84	0.1251
QKLMS-FB	20.66	0.1235
RFFKLMS	*0.20*	0.1363
NysKLMS	0.27	0.1173
KLMS-NC	0.34	0.1640
KLMS-SC	0.69	0.2274
KLMS	86.28	*0.1159*

the NysKLMS using 200 centers achieves similar filtering performance to the KLMS that has no sparsification. The mean consumed time and testing MSEs of all algorithms are shown in Table 3. It can be seen from this table that the NysKLMS with less consumed time obtains lower testing MSE than other sparsification algorithms. The filtering performance of NysKLMS is almost the same as that of KLMS.

In Fig. 9, the parameters of the QKLMS, the QKLMS-FB, the KLMS-NC, the KLMS-SC, the RFFKLMS, and the NysKLMS are chosen such that almost the same testing MSEs are obtained. Besides the same step sizes and kernel widths as those in Fig. 8, the quantization size is set as 0.8 in the QKLMS and the QKLMS-FB. The thresholds are $\delta_1 = 0.5$ and $\delta_2 = 0.37$ for the KLMS-NC and $T_1 = 200$ and $T_2 = -0.2$ for the KLMS-SC. The corresponding mean codebook size and the consumed time of these algorithms are shown in Table 4. We see from this table that the NysKLMS with the same filtering performance as others consumes the least time and codebook size among all the compared algorithms. Therefore, from the aspect of sparsification

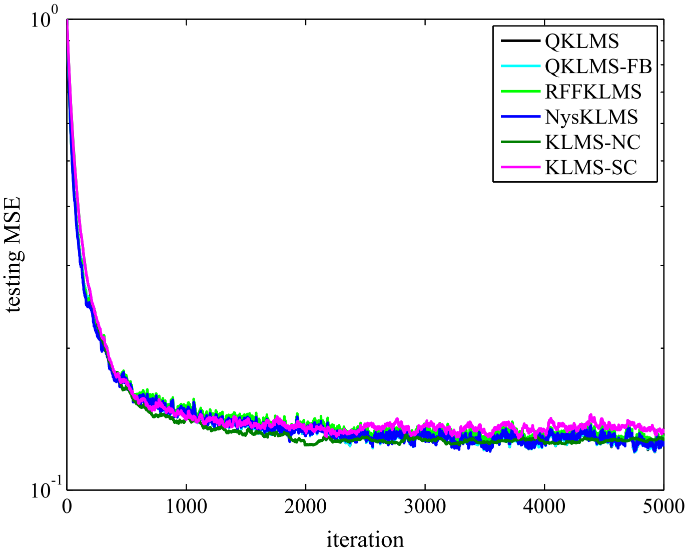


Fig. 9 Learning curves in terms of the same testing MSE of different algorithms

Table 4 Mean codebook size and consumed time with the same testing MSE

Algorithm	Codebook size	Consumed time (s)
QKLMS	365	7.25
QKLMS-FB	350	29.52
RFFKLMS	500	0.29
NysKLMS	*130*	*0.22*
KLMS-NC	536	1.52
KLMS-SC	921	4.07

of KLMS, the NysKLMS is more efficient than the QKLMS, the QKLMS-FB, the KLMS-NC, the KLMC-SC, and the RFFKLMS.

6 Conclusion

In this paper, combining the kernel least mean square (KLMS) algorithm with the Nyström method, we propose a novel sparsification algorithm in a fixed-size network, named Nyström KLMS (NysKLMS). The Nyström method is used to construct a new vector representation from the input space to a relatively high-dimensional feature space. The representation of a fixed dimension can approximate the transformed input of KLMS, efficiently. The proposed NysKLMS algorithm avoids a growing network and thus reduces computational complexity with a small codebook size, significantly. The sufficient condition for the mean square convergence and the theoretical value of the steady-state EMSE in the NysKLMS are also derived for theoretical analysis. Analytical results of the NysKLMS are supported by simulations. From the aspect of the construction of sparsification network, the NystKLMS provides an efficient method for approximating the KLMS.

Acknowledgements This work was supported by National Natural Science Foundation of China (Grant No. 61671389), China Postdoctoral Science Foundation Funded Project (Grant No. 2017M620779), Chongqing Postdoctoral Science Foundation Special Funded Project (Grant No. Xm2017107), and the Fundamental Research Funds for the Central Universities (Grant No. XDJK2016C096).

References

1. T.Y. Al-Naffouri, A.H. Sayed, Transient analysis of adaptive filters with error nonlinearities. IEEE Trans. Signal Process. **51**(3), 653–663 (2003)
2. A.Z.U.M. Al-Saggaf, M. Moinuddin, M. Arif, The q-least mean squares algorithm. Signal Process. **111**, 50–60 (2015)
3. N. Aronszajn, Theory of reproducing kernels. Trans. Am. Math. Soc. **68**(3), 337–404 (1950)
4. P. Bouboulis, S. Pougkakiotis, S. Theodoridis, Efficient KLMS and KRLS algorithms: a random Fourier feature perspective, in *IEEE Statistical Signal Processing Workshop (SSP)*, Palma de Mallorca, Spain, 26–29 June 2016, pp. 1–5
5. C. Burges, A tutorial on support vector machines for pattern recognition. Data Min. Knowl. Disc. **2**(2), 121–167 (1998)
6. B. Chen, L. Li, W. Liu, J.C. Príncipe, *Nonlinear adaptive filtering in kernel spaces* (Springer, Berlin, 2014), pp. 715–734
7. B. Chen, S. Zhao, P. Zhu, J.C. Príncipe, Mean square convergence analysis for kernel least mean square algorithm. Signal Process. **92**(11), 2624–2632 (2012)
8. B. Chen, S. Zhao, P. Zhu, J.C. Príncipe, Quantized kernel least mean square algorithm. IEEE Trans. Neural Netw. Learn. Syst. **23**, 22–32 (2012)
9. S. Craciun, D. Cheney, K. Gugel, J.C. Sanchez, J.C. Príncipe, Wireless transmission of neural signals using entropy and mutual information compression. IEEE Trans. Neural Syst. Rehabil. Eng. **19**(1), 35–44 (2011)
10. L. Csato, M. Opper, Sparse on-line Gaussian processes. Neural Comput. **14**(3), 641–668 (2002)
11. P. Drineas, M.W. Mahoney, On the nystrom method for approximating a Gram matrix for improved kernel-based learning. J. Mach. Learn. Res. **6**, 2153–2175 (2005)
12. Y. Engel, S. Mannor, R. Meir, The kernel recursive least-squares algorithm. IEEE Trans. Signal Process. **52**(8), 2275–2285 (2004)
13. H. Fan, Q. Song, A linear recurrent kernel online learning algorithm with sparse updates. Neural Netw. **50**(2), 142–153 (2014)
14. Z. Hu, M. Lin, C. Zhang, Dependent online kernel learning with constant number of random fourier features. IEEE Trans. Neural Netw. Learn. Syst. **26**(10), 2464–2476 (2015)
15. A. Khalili, A. Rastegarnia, M.K. Islam, T.Y. Rezaii, Codebook design for vector quantization based on a kernel fuzzy learning algorithm. Circuits Syst. Signal Process. **30**(5), 999–1010 (2011)
16. A. Khalili, A. Rastegarnia, M.K. Islam, T.Y. Rezaii, Steady-state tracking analysis of adaptive filter with maximum correntropy criterion. Circuits Syst. Signal Process. **36**(4), 1725–1734 (2017)
17. S. Khan, I. Naseem, R. Togneri, M. Bennamoun, A novel adaptive kernel for the RBF neural networks. Circuits Syst. Signal Process. **36**(4), 1639–1653 (2017)
18. J. Kivinen, A. Smola, R. Williamson, Online learning with kernels. IEEE Trans. Signal Process. **52**(8), 2165–2176 (2004)
19. T. Lehn-Schiøler, A. Hegde, D. Erdogmus, J.C. Príncipe, Vector quantization using information theoretic concepts. Natural Comput. **4**(1), 39–51 (2005)
20. S. Li, L. Song, T. Qiu, Steady-state and tracking analysis of fractional lower-order constant modulus algorithm. Circuits Syst. Signal Process. **30**(6), 1275–1288 (2011)
21. M. Lin, L. Zhang, R. Jin, S. Weng, C. Zhang, Online kernel learning with nearly constant support vectors. Neurocomputing **179**, 26–36 (2016)
22. Y.Y. Linde, A. Buzo, R.M. Gray, An algorithm for vector quantizer design. IEEE Trans. Commun. **28**(1), 84–95 (1980)

23. W. Liu, I. Park, J.C. Príncipe, An information theoretic approach of designing sparse kernel adaptive filters. IEEE Trans. Neural Netw. **20**(12), 1950–1961 (2009)
24. W. Liu, P. Pokharel, J.C. Príncipe, The kernel least-mean-square algorithm. IEEE Trans. Signal Process. **56**(2), 543–554 (2008)
25. W. Liu, J.C. Príncipe, Kernel affine projection algorithms. EURASIP J. Adv. Signal Process. **2008**(1), 1–13 (2008)
26. S. Nan, L. Sun, B. Chen, Z. Lin, K.A. Toh, Density-dependent quantized least squares support vector machine for large data sets. IEEE Trans. Neural Netw. Learn. Syst. **28**(1), 94–106 (2015)
27. J. Platt, A resource-allocating network for function interpolation. Neural Comput. **3**(2), 213–225 (1991)
28. J.C. Príncipe, W. Liu, S.S. Haykin, *Kernel adaptive filtering: a comprehensive introduction* (Wiley, New York, 2010)
29. A. Rahimi, B. Recht, Random features for large scale kernel machines, in *Advances in Neural Information Processing Systems*, Vancouver, Canada, 3–6 December 2007, pp. 1177–1184
30. A.H. Sayed, *Fundamentals of adaptive filtering* (Wiley, New York, 2003)
31. K. Slavakis, S. Theodoridis, I. Yamada, Online kernel-based classification using adaptive projection algorithms. IEEE Trans. Signal Process. **56**(7), 2781–2796 (2008)
32. S. Smale, D. Zhou, Geometry on probability spaces. Constr. Approx. **30**(3), 311–323 (2009)
33. S. Wang, Y. Zheng, S. Duan, L. Wang, Simplified quantised kernel least mean square algorithm with fixed budget. Electron Lett. **52**, 1453–1455 (2016)
34. S. Wang, Y. Zheng, S. Duan, L. Wang, H. Tan, Quantized kernel maximum correntropy and its mean square convergence analysis. Digital Signal Process. **63**(3), 164–176 (2017)
35. C. Williams, M. Seeger, Using the nystrom method to speed up kernel machines, in *Advances in Neural Information Processing Systems*, Vancouver, Canada, 3–8 December 2001, pp. 682–688
36. T. Yang, Y. Li, M. Mahdavi, R. Jin, Z. Zhou, Nystrom method vs random Fourier features: a theoretical and empirical comparison, in *Advances in Neural Information Processing Systems*, Lake Tahoe, USA, 3–6 December 2012, pp. 485–493
37. N.R. Yousef, A.H. Sayed, A unified approach to the steady-state and tracking analyses of adaptive filters. IEEE Trans. Signal Process. **49**(2), 314–324 (2001)
38. S. Zhao, B. Chen, P. Zhu, J.C. Príncipe, Fixed budget quantized kernel least-mean-square algorithm. Signal Process. **93**(9), 2759–2770 (2013)

Affiliations

Shi-Yuan Wang[1,2] · Wen-Yue Wang[1,2] · Lu-Juan Dang[1,2] · Yun-Xiang Jiang[3]

Wen-Yue Wang
wangwenyue@email.swu.edu.cn

Lu-Juan Dang
ljdang@email.swu.edu.cn

Yun-Xiang Jiang
jiang.yunxiang@sz.tsinghua.edu.cn

[1] College of Electronic and Information Engineering, Southwest University, Chongqing 400715, China

[2] Chongqing Key Laboratory of Nonlinear Circuits and Intelligent Information Processing, Chongqing 400715, China

[3] Graduate School at Shenzhen of Tsinghua University, Shenzhen 518055, China