Quantized Kernel Least Mean Square Algorithm

Badong Chen, *Member, IEEE*, Songlin Zhao, *Student Member, IEEE*,
Pingping Zhu, *Student Member, IEEE*, and José C. Príncipe, *Fellow, IEEE*

***Abstract*— In this paper, we propose a quantization approach, as an alternative of sparsification, to curb the growth of the radial basis function structure in kernel adaptive filtering. The basic idea behind this method is to quantize and hence compress the input (or feature) space. Different from sparsification, the new approach uses the "redundant" data to update the coefficient of the closest center. In particular, a quantized kernel least mean square (QKLMS) algorithm is developed, which is based on a simple online vector quantization method. The analytical study of the mean square convergence has been carried out. The energy conservation relation for QKLMS is established, and on this basis we arrive at a sufficient condition for mean square convergence, and a lower and upper bound on the theoretical value of the steady-state excess mean square error. Static function estimation and short-term chaotic time-series prediction examples are presented to demonstrate the excellent performance.**

***Index Terms*— Kernel methods, mean square convergence, quantized kernel least mean square, vector quantization.**

I. Introduction

DURING the last few years, enormous research efforts have been dedicated to the development of the kernel learning methods such as support vector machine [1]– [3], kernel regularization network [4], kernel principal component analysis [5], and so on. These methods show powerful classification and regression performance in complicated nonlinear problems when using Mercer kernels to map the original input space into a high-dimensional feature space and then performing the linear learning in feature space. However, they usually require significant memory and computational burden due to the necessity of calculating a large Gram matrix. As a remedy, some fast learning methods such as fixed-size kernel models have been studied [6]–[8]. The online kernel learning (OKL) provides more efficient alternatives that approximate the desired nonlinearity incrementally, usually with gradient descent techniques [9]. As the training data are sequentially (one by one) presented to the learning system, OKL requires much less memory and computational cost. The resource-allocating network (RAN) [10], generalized growing and pruning radial basis function (RBF) networks [11], online kernel-based classifier using adaptive projection algorithms [12], and the Forgetron [13] are typical OKL algorithms in recent literature, and more recently, the *kernel adaptive filtering* has become an emerging and promising subfield of OKL [14]. The kernel adaptive filtering algorithms are developed in reproducing kernel Hilbert spaces (RKHS) [15], [16], using linear adaptive structure in RKHS to obtain nonlinear filters in the input space, which preserve the conceptual simplicity of the classical linear adaptive filters (no local minima), while inheriting the rich expressiveness from the kernel methods (universal approximation). These algorithms include the kernel least mean square (KLMS) [17], [18], kernel affine projection algorithms (KAPAs) [19], kernel recursive least squares (KRLS) [20], extended KRLS [21], etc. When the kernel is Gaussian, they are essentially the growing RBF networks, where the weights are related to the error at each sample.

The main bottleneck of the kernel adaptive filtering algorithms is their linear growing structure with each new sample, which poses both computational as well as memory issues especially for continuous adaptation scenarios. To address this problem, a variety of sparsification techniques have been proposed to curb the growth of the networks, where only the *important* input data are accepted as the new centers. Existing sparsification criteria for data selection include the novelty criterion [10], prediction variance criterion [22], approximate linear dependency (ALD) criterion [20], and the surprise criterion [23]. The ALD and the variance criterion can be viewed as special cases of the surprise criterion [23].

The sparsification methods are quite effective in reducing the network size while preserving a desirable performance. However, there is a common drawback to these methods: the *redundant* input data are purely discarded. Actually, the redundant data are very useful and can be, for example, utilized to update the coefficients (or weights) of the existing network, although they are not so important for structure update (adding a new center). By doing so, we can expect to achieve better accuracy and a more compact network (with coefficients update, fewer centers are needed to approximate the desired nonlinearity). This idea has already been used by Platt in his RANs [10]. There are two drawbacks to Platt's method: 1) for each redundant input (pattern), the whole coefficients of the present network must be updated, which is computational expensive, and 2) as each pattern lies in a local region of the input space, the learning performance may be negatively affected by the global update (GU) of the coefficients. Thus a simple yet efficient way is to carry out a *local update* for each redundant input, i.e., only update the coefficients within the *responsive domain*. To implement such a learning scheme, we propose in this paper a novel

Manuscript received May 1, 2011; revised August 5, 2011; accepted October 9, 2011. This work was supported in part by the National Science Foundation (NSF) under Grant ECCS 0856441, Grant NSF IIS 0964197, and Grant ONR N00014-10-1-0375.

The authors are with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611 USA (e-mail: chenbd04@mails.tsinghua.edu.cn; zhaosonglin@gmail.com; ppzhu@ufl.edu; principe@cnel.ufl.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2011.2178446

quantization approach: the input space is quantized, and if the current quantized input has already been assigned a center, we do not need to add a new center (from the viewpoint of sparsification, this input is redundant), but update the coefficient of that center (only one coefficient is updated!). In this paper, for simplicity, we only apply this quantization approach to the simplest KLMS algorithm, and name the new algorithm the quantized KLMS (QKLMS).

The quantization concept has, in fact, already been used in the traditional adaptive filters [24]–[30]. The principal motivation behind these studies is to reduce the numerical complexity and dynamic range requirements of the adaptive algorithms, and hence to reduce the hardware complexity in the implementation or to meet the demand of real-time high-speed applications. For example, the power-of-two quantizer has been successfully used to simplify adaptive algorithms such as LMS by replacing multiplications with logical shifts or bit comparisons [25], [27], [29], [30].

The organization of this paper is as follows. In Section II, after briefly introducing the KLMS, we describe the QKLMS. In Section III, we carry out mean square convergence analysis for QKLMS. A sufficient condition for mean square convergence and a lower and upper bound on the steady-state excess mean square error (EMSE) are derived on the basis of the *energy conservation relation*. In Section IV, simulation examples on static function estimation and short-term chaotic time-series prediction are presented. Finally, Section V gives the concluding remarks and future lines of research.

II. QKLMS

Consider the learning of a continuous nonlinear input–output mapping $\mathbb{R}^m \mapsto \mathbb{R}$

$$d = f(\boldsymbol{u}), \quad \boldsymbol{u} \in \mathbb{U} \subseteq \mathbb{R}^m, \quad d \in \mathbb{R} \tag{1}$$

where $\boldsymbol{u}$ is the m-dimensional input vector, $\mathbb{U}$ is a compact input domain in $\mathbb{R}^m$, and d is the output signal. Here the output is assumed to be 1-D for simplicity, it can be easily generalized to the multidimensional cases. If a sequence of input–output pairs $\{\boldsymbol{u}(i), d(i), i = 1, 2, \ldots\}$ is available (we call them the training data), the problem is to find an approximation $\hat{f}$ of the mapping f based on these data. A kernel adaptive filter is a kernel-based sequential estimator of f such that f_i (the estimate at iteration i) is updated on the basis of the last estimate f_{i-1} and current example $\{\boldsymbol{u}(i), d(i)\}$. Before formally introducing the QKLMS, we briefly describe in the following the KLMS algorithm.

A. KLMS

A Mercer kernel is a continuous, symmetric, and positive definite function $\kappa : \mathbb{U} \times \mathbb{U} \to \mathbb{R}$ [15]. The commonly used Gaussian kernel is

$$\kappa(\boldsymbol{u}, \boldsymbol{u}') = \exp\left(\frac{-\|\boldsymbol{u} - \boldsymbol{u}'\|^2}{2\sigma^2}\right) \tag{2}$$

where $\sigma > 0$ is the kernel width. According to the Mercer's theorem, any Mercer kernel $\kappa(\boldsymbol{u}, \boldsymbol{u}')$ induces a mapping $\boldsymbol{\varphi}$ from the input space $\mathbb{U}$ to a high-dimensional feature space $\mathbb{F}$ (which is an inner product space) such that the following relationship (the so-called kernel trick) holds [16]:

$$\boldsymbol{\varphi}(\boldsymbol{u})^T \boldsymbol{\varphi}(\boldsymbol{u}') = \kappa(\boldsymbol{u}, \boldsymbol{u}'). \tag{3}$$[1]

The feature space $\mathbb{F}$ is essentially the same as the RKHS induced by the kernel if we identify $\boldsymbol{\varphi}(\boldsymbol{u}) = \kappa(\boldsymbol{u}, .)$. We do not distinguish these two spaces in this paper if no confusion arises.

The KLMS is actually the linear LMS algorithm in feature space $\mathbb{F}$ [17]. First, the kernel-induced mapping $\boldsymbol{\varphi}$ is employed to transform the input $\boldsymbol{u}(i)$ into $\mathbb{F}$ as $\boldsymbol{\varphi}(u(i))$. Denoting $\boldsymbol{\varphi}(i) = \boldsymbol{\varphi}(\boldsymbol{u}(i))$ and applying the LMS algorithm on the new example sequence $\{\boldsymbol{\varphi}(i), d(i)\}$ yields

$$\begin{cases} \boldsymbol{\Omega}(0) = 0 \\ e(i) = d(i) - \boldsymbol{\Omega}(i-1)^T \boldsymbol{\varphi}(i) \\ \boldsymbol{\Omega}(i) = \boldsymbol{\Omega}(i-1) + \eta e(i) \boldsymbol{\varphi}(i) \end{cases} \tag{4}$$

where $e(i)$ is the prediction error at iteration i, η is the step size, and $\boldsymbol{\Omega}(i)$ denotes the estimate of the weight vector in $\mathbb{F}$. f_i is the composition of $\boldsymbol{\Omega}(i)$ and $\boldsymbol{\varphi}$, that is $f_i = \boldsymbol{\Omega}(i)^T \boldsymbol{\varphi}(.)$. If identifying $\boldsymbol{\varphi}(\boldsymbol{u}) = \kappa(\boldsymbol{u}, .)$, we obtain the learning rule in original space for KLMS

$$\begin{cases} f_0 = 0 \\ e(i) = d(i) - f_{i-1}(\boldsymbol{u}(i)) \\ f_i = f_{i-1} + \eta e(i) \kappa(\boldsymbol{u}(i), .). \end{cases} \tag{5}$$

The KLMS produces a growing RBF network by allocating a new kernel unit for every new example with input $\boldsymbol{u}(i)$ as the center and $\eta e(i)$ as the coefficient.

B. QKLMS

The QKLMS algorithm can be obtained by just quantizing the feature vector $\boldsymbol{\varphi}(i)$ in the weight-update equation $\boldsymbol{\Omega}(i) = \boldsymbol{\Omega}(i-1) + \eta e(i) \boldsymbol{\varphi}(i)$ in (4), which can be expressed as

$$\begin{cases} \boldsymbol{\Omega}(0) = \mathbf{0} \\ e(i) = d(i) - \boldsymbol{\Omega}(i-1)^T \boldsymbol{\varphi}(i) \\ \boldsymbol{\Omega}(i) = \boldsymbol{\Omega}(i-1) + \eta e(i) \mathcal{Q}\left[\boldsymbol{\varphi}(i)\right] \end{cases} \tag{6}$$

where $\mathcal{Q}[.]$ denotes a quantization operator in $\mathbb{F}$. Since the dimensionality of the feature space is very high (can be even infinite), the quantization is usually performed in the original input space $\mathbb{U}$. In this situation, the learning rule for QKLMS is

$$\begin{cases} f_0 = 0 \\ e(i) = d(i) - f_{i-1}(\boldsymbol{u}(i)) \\ f_i = f_{i-1} + \eta e(i) \kappa(Q[\boldsymbol{u}(i)], .) \end{cases} \tag{7}$$

where $Q[.]$ is a quantization operator in $\mathbb{U}$. In the rest of this paper, to simplify the notation, we denote $\boldsymbol{\varphi}_q(i) = \mathcal{Q}\left[\boldsymbol{\varphi}(i)\right]$, and $\boldsymbol{u}_q(i) = Q[\boldsymbol{u}(i)]$.

Remark 1: Quantization techniques have been widely used in fields such as digitization, data compression, and speech and image coding. Here we use the quantization method to compress the input (or feature) space and hence to compact

[1] In this paper, we use the notation $\boldsymbol{\varphi}(\boldsymbol{u})^T \boldsymbol{\varphi}(\boldsymbol{u}')$ to refer to the inner product $\langle \boldsymbol{\varphi}(u), \boldsymbol{\varphi}(\boldsymbol{u}') \rangle_{\mathbb{F}}$ of the high-dimensional feature space $\mathbb{F}$.

Algorithm 1 Online VQ in $\mathbb{U}$

Input: $\{\boldsymbol{u}(i) \in \mathbb{U}\}$, $i = 1, 2, \ldots$

Initialization: Choose the quantization size $\varepsilon_{\mathbb{U}} \geq 0$, and initialize the codebook $\boldsymbol{C}(1) = \{\boldsymbol{u}(1)\}$.

Computation:

while $\{\boldsymbol{u}(i)\}$ $(i > 1)$ available **do**

1) Compute the distance between $u(i)$ and $\boldsymbol{C}(i-1)$: $dis\left(\boldsymbol{u}(i), \boldsymbol{C}(i-1)\right) = \min\limits_{1 \leq j \leq size(\boldsymbol{C}(i-1))} \left\|\boldsymbol{u}(i) - \boldsymbol{u}_j(i-1)\right\|$
2) If $dis\left(\boldsymbol{u}(i), \boldsymbol{C}(i-1)\right) \leq \varepsilon_{\mathbb{U}}$, keep the codebook unchanged: $\boldsymbol{C}(i) = \boldsymbol{C}(i-1)$, and quantize $\boldsymbol{u}(i)$ to the closest code-vector: $\boldsymbol{u}_q(i) = \boldsymbol{C}_{j^*}(i-1)$, where $j^* = \arg\min\limits_{1 \leq j \leq size(\boldsymbol{C}(i-1))} \left\|\boldsymbol{u}(i) - \boldsymbol{C}_j(i-1)\right\|$.
3) Otherwise, update the codebook: $\boldsymbol{C}(i) = \{\boldsymbol{C}(i-1), \boldsymbol{u}(i)\}$, and quantize $\boldsymbol{u}(i)$ as itself: $\boldsymbol{u}_q(i) = \boldsymbol{u}(i)$

end while

the RBF structure of the kernel adaptive filter by reducing the network size (number of centers). The network size of QKLMS, obviously, will be always smaller than the size of the quantization codebook (dictionary).

The key problem in QKLMS is the design of the vector quantizer including: 1) how to assign the code vectors to the data, and 2) how to find the closest code-vector representation. There exist a variety of vector quantization (VQ) algorithms in the literature [31]–[34]. Most of the existing VQ algorithms, however, are not suitable for online implementation because the codebook must be supplied in advance (which is usually trained on an offline data set), and the computational burden is rather heavy. In this paper, we propose a very simple online VQ method in which the codebook is trained directly from online samples and is adaptively growing. The online VQ algorithm in $\mathbb{U}$ is described in Algorithm 1.

In the above VQ algorithm, $\boldsymbol{C}_j(i-1)$ denotes the jth element (code vector) of the codebook $\boldsymbol{C}(i-1)$, and $\|.\|$ denotes the Euclidean norm in $\mathbb{U}$.

In the rest of this paper, the kernel is assumed to be Gaussian without explicit mention. For a Gaussian kernel, we have

$$\begin{aligned}\|\boldsymbol{\varphi}(i) - \boldsymbol{\varphi}(j)\|_{\mathbb{F}} &= \sqrt{(\boldsymbol{\varphi}(i) - \boldsymbol{\varphi}(j))^T (\boldsymbol{\varphi}(i) - \boldsymbol{\varphi}(j))} \\ &= \sqrt{2 - 2\kappa\left(\boldsymbol{u}(i), \boldsymbol{u}(j)\right)} \\ &= \sqrt{2 - 2\exp\left(\frac{-\|\boldsymbol{u}(i) - \boldsymbol{u}(j)\|^2}{2\sigma^2}\right)} \end{aligned} \quad (8)$$

where $\|.\|_{\mathbb{F}}$ denotes the norm in $\mathbb{F}$, i.e., $\forall \boldsymbol{\varphi} \in \mathbb{F}$, $\|\boldsymbol{\varphi}\|_{\mathbb{F}} \triangleq \sqrt{\boldsymbol{\varphi}^T \boldsymbol{\varphi}}$. Equation (8) implies that the distance in feature space $\mathbb{F}$ will be monotonically increasing with the distance in original input space $\mathbb{U}$. In this case, the VQ in $\mathbb{U}$ is actually equivalent to the VQ in $\mathbb{F}$, and we can identify $\kappa\left(\boldsymbol{u}_q(i), .\right)$ with $\boldsymbol{\varphi}_q(i)$, where $\boldsymbol{\varphi}_q(i)$ is obtained by performing similar online VQ in $\mathbb{F}$ but with quantization size

$$\varepsilon_{\mathbb{F}} = \sqrt{2 - 2\exp\left(\frac{-\varepsilon_{\mathbb{U}}^2}{2\sigma^2}\right)}. \quad (9)$$

Algorithm 2 QKLMS Algorithm

Input: $\{\boldsymbol{u}(i) \in \mathbb{U}, d(i)\}$, $i = 1, 2, \ldots$

Initialization: Choose step size $\eta > 0$, kernel width $\sigma > 0$, quantization size $\varepsilon_{\mathbb{U}} \geq 0$, and initialize the codebook (center set) and coefficient vector: $\boldsymbol{C}(1) = \{\boldsymbol{u}(1)\}$, $a(1) = [\eta d(1)]$.

Computation:

while $\{\boldsymbol{u}(i), d(i)\}$ $(i > 1)$ available **do**

1) Compute the output of the adaptive filter: $y(i) = \sum\limits_{j=1}^{size(\boldsymbol{C}(i-1))} \boldsymbol{a}_j(i-1)\kappa\left(\boldsymbol{C}_j(i-1), \boldsymbol{u}(i)\right)$
2) Compute the error: $e(i) = d(i) - y(i)$
3) Compute the distance between $\boldsymbol{u}(i)$ and $\boldsymbol{C}(i-1)$: $dis\left(\boldsymbol{u}(i), \boldsymbol{C}(i-1)\right) = \min\limits_{1 \leq j \leq size(\boldsymbol{C}(i-1))} \left\|\boldsymbol{u}(i) - \boldsymbol{C}_j(i-1)\right\|$
4) If $dis\left(\boldsymbol{u}(i), \boldsymbol{C}(i-1)\right) \leq \varepsilon_{\mathbb{U}}$, keep the codebook unchanged: $\boldsymbol{C}(i) = \boldsymbol{C}(i-1)$, and quantize $\boldsymbol{u}(i)$ to the closest center through updating the coefficient of that center: $\boldsymbol{a}_{j^*}(i) = \boldsymbol{a}_{j^*}(i-1) + \eta e(i)$, where
$$j^* = \arg\min_{1 \leq j \leq size(\boldsymbol{C}(i-1))} \left\|\boldsymbol{u}(i) - \boldsymbol{C}_j(i-1)\right\|$$
5) Otherwise, assign a new center and corresponding new coefficient: $\boldsymbol{C}(i) = \{\boldsymbol{C}(i-1), \boldsymbol{u}(i)\}$, $\boldsymbol{a}(i) = [\boldsymbol{a}(i-1), \eta e(i)]$.

end while

Remark 2: The proposed online VQ method merely relies on the distance measure (in $\mathbb{U}$ or $\mathbb{F}$) and, obviously, is not an optimal VQ method, because it is not derived by optimizing some distortion measure. A more reasonable VQ method should consider the distribution of data. However, it is very simple and suitable for our QKLMS.

Now we give a summary of the QKLMS with online VQ:

Remark 3: The above QKLMS is somewhat similar to the sparsified KLMS with novelty criterion [14]. In fact, they have almost the same computational complexity. The key difference between the two algorithms is that the QKLMS has utilized the "redundant" data to *locally update* the coefficient of the closest center. Intuitively, the coefficient update can enhance the *utilization efficiency* of that center, and hence may yield better accuracy and a more compact network. For the case $\varepsilon_{\mathbb{U}} = 0$, the QKLMS will reduce to the true KLMS algorithm.

In QKLMS algorithm, the "redundant" data can also be used to *globally update* the whole coefficients of the current network, using the LMS algorithm, just like Platt's approach in a RAN [10]. This variant of the algorithm, which we refer to as the QKLMS-GU, however, may perform worse than the original QKLMS, although it is more computational intensive. This has been confirmed by the simulation results in Section IV. One possible reason for this is that the local update (only updates the closest center) of QKLMS avoids the negative influence caused by far away centers which are still under learning.

III. Mean Square Convergence Analysis

The convergence performance is the key aspect of an adaptive filter. We carry out in this section the mean square

convergence analysis for QKLMS. For classical linear adaptive filters, the *energy conservation relation* is shown to be a powerful tool for the mean square convergence analysis [35]– [39]. In the following, we derive the *energy conservation relation* for QKLMS, then, on this basis, present a sufficient condition for mean square convergence, and establish a lower and upper bound on the theoretical value of the steady-state EMSE.

A. Energy Conservation Relation

Consider the nonlinear system identification case in which the output data $\{d(i)\}$ are related to input vectors $\{\boldsymbol{u}(i)\}$ via

$$d(i) = f^*\left(\boldsymbol{u}(i)\right) + \upsilon(i) \tag{10}$$

where $f^*(.)$ denotes the unknown nonlinear mapping that needs to be estimated, and $\upsilon(i)$ stands for the disturbance noise. According to the universal approximation property [16], there exists a vector $\boldsymbol{\Omega}^* \in \mathbb{F}$ such that $f^* = \boldsymbol{\Omega}^{*T}\boldsymbol{\varphi}(.)$ holds. Therefore, we can rewrite (10) as

$$d(i) = \boldsymbol{\Omega}^{*T}\boldsymbol{\varphi}(i) + \upsilon(i). \tag{11}$$

And then, the prediction error $e(i)$ becomes

$$\begin{aligned} e(i) &= d(i) - \boldsymbol{\Omega}(i-1)^T\boldsymbol{\varphi}(i) \\ &= \left(\boldsymbol{\Omega}^{*T}\boldsymbol{\varphi}(i) + \upsilon(i)\right) - \boldsymbol{\Omega}(i-1)^T\boldsymbol{\varphi}(i) \\ &= \tilde{\boldsymbol{\Omega}}(i-1)^T\boldsymbol{\varphi}(i) + \upsilon(i) \\ &= e_a(i) + \upsilon(i) \end{aligned} \tag{12}$$

where $\tilde{\boldsymbol{\Omega}}(i-1) \triangleq \boldsymbol{\Omega}^* - \boldsymbol{\Omega}(i-1)$ is the weight error vector in $\mathbb{F}$, and $e_a(i) \triangleq \tilde{\boldsymbol{\Omega}}(i-1)^T\boldsymbol{\varphi}(i)$ is the *a priori* error at iteration i.

Subtracting $\boldsymbol{\Omega}^*$ from both sides of the QKLMS update equation $\boldsymbol{\Omega}(i) = \boldsymbol{\Omega}(i-1) + \eta e(i)\boldsymbol{\varphi}_q(i)$, we get

$$\tilde{\boldsymbol{\Omega}}(i) = \tilde{\boldsymbol{\Omega}}(i-1) - \eta e(i)\boldsymbol{\varphi}_q(i). \tag{13}$$

Defining the *a posteriori* error $e_p(\mathrm{i}) \triangleq \tilde{\boldsymbol{\Omega}}(i)^T\boldsymbol{\varphi}(i)$, we have

$$e_p(i) = e_a(i) + \left(\tilde{\boldsymbol{\Omega}}(i)^T - \tilde{\boldsymbol{\Omega}}(i-1)^T\right)\boldsymbol{\varphi}(i). \tag{14}$$

By incorporating (13)

$$\begin{aligned} e_p(i) &= e_a(i) - \eta e(i)\boldsymbol{\varphi}_q(i)^T\boldsymbol{\varphi}(i) \\ &= e_a(i) - \eta e(i)\kappa\left(\boldsymbol{u}_q(i), \boldsymbol{u}(i)\right) \end{aligned} \tag{15}$$

where the latter equation follows from $\boldsymbol{\varphi}_q(i) = \kappa\left(\boldsymbol{u}_q(i), .\right)$ and the *kernel trick* (3).

Combining (13) and (15) so as to eliminate the prediction error $e(i)$ yields

$$\tilde{\boldsymbol{\Omega}}(i) = \tilde{\boldsymbol{\Omega}}(i-1) + \left(e_p(i) - e_a(i)\right)\frac{\boldsymbol{\varphi}_q(i)}{\kappa\left(\boldsymbol{u}_q(i), \boldsymbol{u}(i)\right)}. \tag{16}$$

Squaring both sides of (16), we have

$$\begin{aligned} \tilde{\boldsymbol{\Omega}}(i)^T\tilde{\boldsymbol{\Omega}}(i) &= \left[\tilde{\boldsymbol{\Omega}}(i-1) + \left(e_p(i) - e_a(i)\right)\frac{\boldsymbol{\varphi}_q(i)}{\kappa\left(\boldsymbol{u}_q(i), \boldsymbol{u}(i)\right)}\right]^T \\ &\quad \times \left[\tilde{\boldsymbol{\Omega}}(i-1) + \left(e_p(i) - e_a(i)\right)\frac{\boldsymbol{\varphi}_q(i)}{\kappa\left(\boldsymbol{u}_q(i), \boldsymbol{u}(i)\right)}\right] \\ &= \tilde{\boldsymbol{\Omega}}(i-1)^T\tilde{\boldsymbol{\Omega}}(i-1) + \frac{e_p^2(i) - e_a^2(i)}{\kappa\left(\boldsymbol{u}_q(i), \boldsymbol{u}(i)\right)^2} \\ &\quad + \frac{2\left(e_p(i) - e_a(i)\right)\left\{\tilde{\boldsymbol{\Omega}}(i-1)^T\boldsymbol{\varphi}_q(i)\kappa\left(\boldsymbol{u}_q(i), \boldsymbol{u}(i)\right) - e_a(i)\right\}}{\kappa\left(\boldsymbol{u}_q(i), \boldsymbol{u}(i)\right)^2}. \end{aligned} \tag{17}$$

It follows that:

$$\begin{aligned} \left\|\tilde{\boldsymbol{\Omega}}(i)\right\|_{\mathbb{F}}^2 + \frac{e_a^2(i)}{\kappa\left(\boldsymbol{u}_q(i), \boldsymbol{u}(i)\right)^2} &= \left\|\tilde{\boldsymbol{\Omega}}(i-1)\right\|_{\mathbb{F}}^2 \\ &\quad + \frac{e_p^2(i)}{\kappa\left(\boldsymbol{u}_q(i), \boldsymbol{u}(i)\right)^2} + \beta_q \end{aligned} \tag{18}$$

where $\left\|\tilde{\boldsymbol{\Omega}}(i)\right\|_{\mathbb{F}}^2 = \tilde{\boldsymbol{\Omega}}(i)^T\tilde{\boldsymbol{\Omega}}(i)$, and β_q is expressed as

$$\beta_q = \frac{2\left(e_p(i) - e_a(i)\right)\left\{\tilde{\boldsymbol{\Omega}}(i-1)^T\boldsymbol{\varphi}_q(i)\kappa\left(\boldsymbol{u}_q(i), \boldsymbol{u}(i)\right) - e_a(i)\right\}}{\kappa\left(\boldsymbol{u}_q(i), \boldsymbol{u}(i)\right)^2}. \tag{19}$$

Remark 4: Equation (18), which we refer to in this paper as the *energy conservation relation* for QKLMS, shows how the error powers evolve during learning with QKLMS. If we omit the term β_q, this relation is very similar to the classical energy conservation relation for linear adaptive filters [35]– [39]. When the quantization size $\varepsilon_{\mathbb{U}}$ approaches zero, we have $\kappa\left(\boldsymbol{u}_q(i), \boldsymbol{u}(i)\right) \to 1$ and $\beta_q \to 0$, and in this case, (18) reduces to the *energy conservation relation* for KLMS

$$\left\|\tilde{\boldsymbol{\Omega}}(i)\right\|_{\mathbb{F}}^2 + e_a^2(i) = \left\|\tilde{\boldsymbol{\Omega}}(i-1)\right\|_{\mathbb{F}}^2 + e_p^2(i) \tag{20}$$

which in form is identical to the energy conservation relation for normalized LMS (NLMS). This is not a surprise since the KLMS (with Gaussian kernel) is essentially the NLMS in the feature space $\mathbb{F}$.

B. Mean Square Convergence Analysis

Substituting $e_p(i) = e_a(i) - \eta e(i)\kappa\left(\boldsymbol{u}_q(i), \boldsymbol{u}(i)\right)$ into the energy conservation relation (18), and after some simple manipulations, yields

$$\left\|\tilde{\boldsymbol{\Omega}}(i)\right\|_{\mathbb{F}}^2 = \left\|\tilde{\boldsymbol{\Omega}}(i-1)\right\|_{\mathbb{F}}^2 + \eta^2 e^2(i) - 2\eta e(i)\tilde{\boldsymbol{\Omega}}(i-1)^T\boldsymbol{\varphi}_q(i). \tag{21}$$

Because we are interested in the mean square behavior of the QKLMS, we take expectations of both sides of (21) and write

$$\begin{aligned} E\left[\left\|\tilde{\boldsymbol{\Omega}}(i)\right\|_{\mathbb{F}}^2\right] &= E\left[\left\|\tilde{\boldsymbol{\Omega}}(i-1)\right\|_{\mathbb{F}}^2\right] + \eta^2 E\left[e^2(i)\right] \\ &\quad - 2\eta E\left[e(i)\tilde{\boldsymbol{\Omega}}(i-1)^T\boldsymbol{\varphi}_q(i)\right] \end{aligned} \tag{22}$$

where $E\left[\left\|\tilde{\boldsymbol{\Omega}}(i)\right\|_{\mathbb{F}}^2\right]$ is called the weight error power (WEP) in $\mathbb{F}$.

Now we use (22) to analyze the mean square convergence of the QKLMS. Before proceeding, we give an assumption that will be used in the rest of this paper.

A1: The noise $\upsilon(i)$ is the zero-mean, independent, identically distributed (i.i.d.), and independent of the *a priori* estimation error $e_a(i)$.

Remark 5: The i.i.d. assumption of the noise sequence is commonly used in the convergence analysis for adaptive

filtering algorithms [35]. A sufficient condition for the independence between $v(i)$ and $e_a(i)$ is the independence between $v(i)$ and the input sequence $\{\boldsymbol{u}(i)\}$.

Incorporating assumption A1 into (22) yields

$$E\left[\left\|\tilde{\boldsymbol{\Omega}}(i)\right\|_{\mathbb{F}}^2\right] = E\left[\left\|\tilde{\boldsymbol{\Omega}}(i-1)\right\|_{\mathbb{F}}^2\right] + \eta^2\left(E\left[e_a^2(i)\right] + \sigma_v^2\right) - 2\eta E\left[e_a(i)\tilde{\boldsymbol{\Omega}}(i-1)^T\boldsymbol{\varphi}_q(i)\right] \tag{23}$$

where σ_v^2 denotes the noise variance.

1) Sufficient Condition for Mean Square Convergence: From (23), we observe that

$$\begin{aligned} &E\left[\left\|\tilde{\boldsymbol{\Omega}}(i)\right\|_{\mathbb{F}}^2\right] \le E\left[\left\|\tilde{\boldsymbol{\Omega}}(i-1)\right\|_{\mathbb{F}}^2\right] \\ &\Leftrightarrow \eta^2\left(E\left[e_a^2(i)\right] + \sigma_v^2\right) - 2\eta E\left[e_a(i)\tilde{\boldsymbol{\Omega}}(i-1)^T\boldsymbol{\varphi}_q(i)\right] \le 0 \\ &\Leftrightarrow \eta \le \frac{2E\left[e_a(i)\tilde{\boldsymbol{\Omega}}(i-1)^T\boldsymbol{\varphi}_q(i)\right]}{E\left[e_a^2(i)\right] + \sigma_v^2}. \end{aligned} \tag{24}$$

Therefore, to ensure the monotonic decrease of WEP in $\mathbb{F}$, we should choose a step size η (which can be variable) such that $\forall i$

$$0 < \eta \le \frac{2E\left[e_a(i)\tilde{\boldsymbol{\Omega}}(i-1)^T\boldsymbol{\varphi}_q(i)\right]}{E\left[e_a^2(i)\right] + \sigma_v^2}. \tag{25}$$

The existence of such step size requires $\forall i$

$$E\left[e_a(i)\tilde{\boldsymbol{\Omega}}(i-1)^T\boldsymbol{\varphi}_q(i)\right] > 0. \tag{26}$$

Thus a sufficient condition for mean square convergence (monotonic decrease of WEP) will be

$$\forall i,\ \begin{cases} E\left[e_a(i)\tilde{\boldsymbol{\Omega}}(i-1)^T\boldsymbol{\varphi}_q(i)\right] > 0\ (C1) \\ 0 < \eta \le \frac{2E\left[e_a(i)\tilde{\boldsymbol{\Omega}}(i-1)^T\boldsymbol{\varphi}_q(i)\right]}{E\left[e_a^2(i)\right]+\sigma_v^2}\ (C2). \end{cases} \tag{27}$$

If $\boldsymbol{\varphi}(i)$ and $\tilde{\boldsymbol{\Omega}}(i-1)$ are statistically independent[2] we have

$$\begin{aligned} &E\left[e_a(i)\tilde{\boldsymbol{\Omega}}(i-1)^T\boldsymbol{\varphi}_q(i)\right] \\ &= E\left[\tilde{\boldsymbol{\Omega}}(i-1)^T\boldsymbol{\varphi}(i)\boldsymbol{\varphi}_q(i)^T\tilde{\boldsymbol{\Omega}}(i-1)\right] \\ &= E\left[\tilde{\boldsymbol{\Omega}}(i-1)^T E\left(\boldsymbol{\varphi}(i)\boldsymbol{\varphi}_q(i)^T\right)\tilde{\boldsymbol{\Omega}}(i-1)\right]. \end{aligned} \tag{28}$$

In this case, the condition $(C1)$ in (27) can be replaced by a more stringent condition

$$E\left(\boldsymbol{\varphi}(i)\boldsymbol{\varphi}_q(i)^T\right) > 0. \tag{29}$$

The above condition, in form, is similar to the *persistence of excitation* condition for the quantized regressor algorithm given in [26].

[2]The independence assumption between current input and weight error vector are commonly used in literature.

2) Steady-State Mean Square Performance: Let us take the limit of (23) as $i \to \infty$, and write

$$\begin{aligned} \lim_{i\to\infty} E\left[\left\|\tilde{\boldsymbol{\Omega}}(i)\right\|_{\mathbb{F}}^2\right] &= \lim_{i\to\infty} E\left[\left\|\tilde{\boldsymbol{\Omega}}(i-1)\right\|_{\mathbb{F}}^2\right] \\ &\quad +\eta^2\left(\lim_{i\to\infty} E\left[e_a^2(i)\right] + \sigma_v^2\right) \\ &\quad -2\eta \lim_{i\to\infty} E\left[e_a(i)\tilde{\boldsymbol{\Omega}}(i-1)^T\boldsymbol{\varphi}_q(i)\right]. \end{aligned} \tag{30}$$

Supposing the mean square convergence sufficient condition (27) holds and the algorithm reaches the steady state, we have

$$\lim_{i\to\infty} E\left[\left\|\tilde{\boldsymbol{\Omega}}(i)\right\|_{\mathbb{F}}^2\right] = \lim_{i\to\infty} E\left[\left\|\tilde{\boldsymbol{\Omega}}(i-1)\right\|_{\mathbb{F}}^2\right]. \tag{31}$$

And hence

$$\begin{aligned} &\eta^2\left(\lim_{i\to\infty} E\left[e_a^2(i)\right] + \sigma_v^2\right) \\ &\quad -2\eta \lim_{i\to\infty} E\left[e_a(i)\tilde{\boldsymbol{\Omega}}(i-1)^T\boldsymbol{\varphi}_q(i)\right] = 0. \end{aligned} \tag{32}$$

It follows that:

$$\begin{aligned} &\eta\left(\lim_{i\to\infty} E\left[e_a^2(i)\right] + \sigma_v^2\right) = 2\lim_{i\to\infty} E\left[e_a(i)\tilde{\boldsymbol{\Omega}}(i-1)^T\boldsymbol{\varphi}_q(i)\right] \\ &= 2\lim_{i\to\infty} E\left[e_a(i)\tilde{\boldsymbol{\Omega}}(i-1)^T\left(\boldsymbol{\varphi}(i) + \left(\boldsymbol{\varphi}_q(i) - \boldsymbol{\varphi}(i)\right)\right)\right] \\ &= 2\lim_{i\to\infty} E\left[e_a^2(i)\right] \\ &\quad +2\lim_{i\to\infty} E\left[\tilde{\boldsymbol{\Omega}}(i-1)^T\boldsymbol{\varphi}(i)\tilde{\boldsymbol{\Omega}}(i-1)^T\left(\boldsymbol{\varphi}_q(i) - \boldsymbol{\varphi}(i)\right)\right]. \end{aligned} \tag{33}$$

Thus the steady-state EMSE[3] is

$$\begin{aligned} &\lim_{i\to\infty} E\left[e_a^2(i)\right] \\ &= \frac{\eta\sigma_v^2 - 2\lim_{i\to\infty} E\left[\tilde{\boldsymbol{\Omega}}(i-1)^T\boldsymbol{\varphi}(i)\tilde{\boldsymbol{\Omega}}(i-1)^T\left(\boldsymbol{\varphi}_q(i)-\boldsymbol{\varphi}(i)\right)\right]}{2-\eta}. \end{aligned} \tag{34}$$

Since

$$\begin{aligned} &\left|E\left[\tilde{\boldsymbol{\Omega}}(i-1)^T\boldsymbol{\varphi}(i)\tilde{\boldsymbol{\Omega}}(i-1)^T\left(\boldsymbol{\varphi}_q(i) - \boldsymbol{\varphi}(i)\right)\right]\right| \\ &\le E\left[\left\|\tilde{\boldsymbol{\Omega}}(i-1)\right\|_{\mathbb{F}}^2\left\|\boldsymbol{\varphi}_q(i) - \boldsymbol{\varphi}(i)\right\|_{\mathbb{F}}\right] \\ &= E\left[\left\|\tilde{\boldsymbol{\Omega}}(i-1)\right\|_{\mathbb{F}}^2\sqrt{2 - 2\exp\left(-\frac{\left\|\boldsymbol{u}_q(i) - \boldsymbol{u}(i)\right\|^2}{2\sigma^2}\right)}\right] \\ &\overset{(a)}{\le} \sqrt{2 - 2\exp\left(-\frac{\varepsilon_{\mathbb{U}}^2}{2\sigma^2}\right)}E\left[\left\|\boldsymbol{\Omega}^*\right\|_{\mathbb{F}}^2\right] \\ &= \sqrt{2 - 2\exp\left(-\frac{1}{2}\gamma^2\right)}E\left[\left\|\boldsymbol{\Omega}^*\right\|_{\mathbb{F}}^2\right] \end{aligned} \tag{35}$$

where *(a)* follows from $\left\|\boldsymbol{u}_q(i) - \boldsymbol{u}(i)\right\| \le \varepsilon_{\mathbb{U}}$ and the monotonic decrease of the WEP, γ is the *quantization factor* defined as $\gamma = \varepsilon_{\mathbb{U}}/\sigma$, then we can derive

$$\max\left\{\frac{\eta\sigma_v^2 - 2\xi_\gamma}{2-\eta}, 0\right\} \le \lim_{i\to\infty} E\left[e_a^2(i)\right] \le \frac{\eta\sigma_v^2 + 2\xi_\gamma}{2-\eta} \tag{36}$$

[3]The *a priori* error power is also referred to as the excess mean square error in the adaptive filtering community.

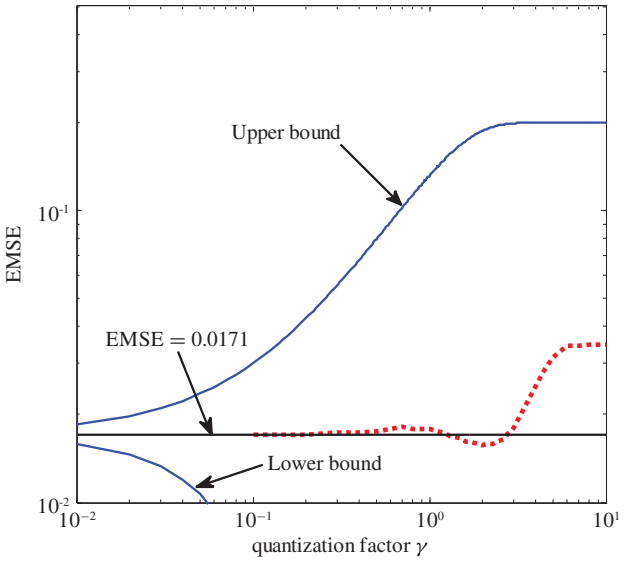

Fig. 1. Steady-state EMSE (dotted) versus the quantization factor γ in static function estimation. The derived lower and upper bounds of the steady-state EMSE are also plotted.

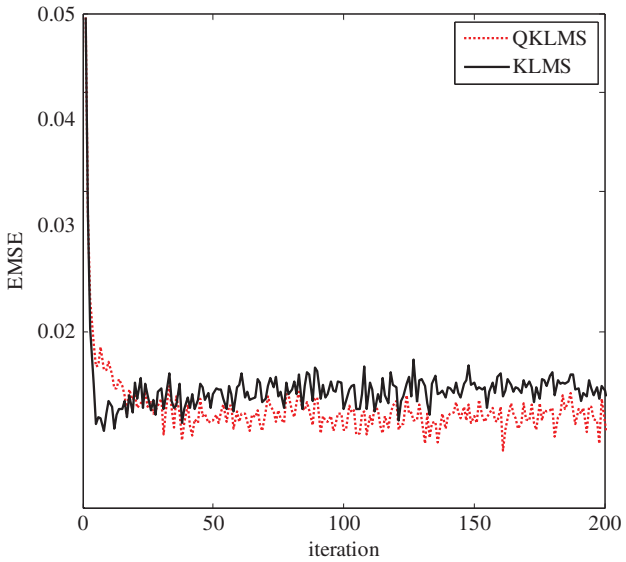

Fig. 2. Convergence curves in terms of the EMSE for QKLMS ($\gamma = 2.0$) and KLMS in static function estimation.

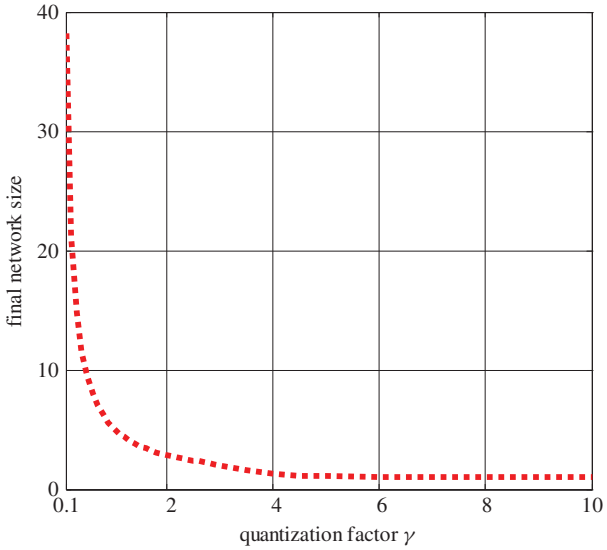

Fig. 3. Final network size versus the quantization factor γ in static function estimation.

where $\xi_\gamma = \sqrt{2 - 2\exp\left(-(1/2)\gamma^2\right)}E[\|\mathbf{\Omega}^*\|_{\mathbb{F}}^2]$.

Remark 6: We establish in (36) a lower and upper bound on the steady-state EMSE for QKLMS algorithm. For the case the quantization factor $\gamma = 0$, we have $\xi_\gamma = 0$, and

$$\lim_{i\to\infty} E\left[e_a^2(i)\right] = \frac{\eta\sigma_v^2}{2-\eta} \tag{37}$$

which is actually the steady-state EMSE for KLMS. From (36), the lower bound of the steady-state EMSE for QKLMS is smaller than the steady-state EMSE for KLMS. This is beyond our intuition, because we think that the quantization should always decrease the accuracy, and the final EMSE of the QKLMS should be always larger than that of the KLMS. In the next section, we prove by simulation that the QKLMS can really produce smaller EMSE than KLMS algorithm.

IV. Simulation Results

Now we conduct Monte Carlo simulations to demonstrate the performance of the proposed QKLMS in static function estimation and short-term chaotic time series prediction.

A. Static Function Estimation

Suppose the desired output data are generated by the following static nonlinear (mix-Gaussian) system:

$$d(i) = 0.2 \times \left[\exp\left(-\frac{(u(i)+1)^2}{2}\right) + \exp\left(-\frac{(u(i)-1)^2}{2}\right)\right] + v(i) \tag{38}$$

where the input sequence $\{u(i)\}$ is a zero-mean white Gaussian process with unit variance, and the noise $\{v(i)\}$ is another zero-mean white Gaussian process that is independent of $\{u(i)\}$. The kernel size is set as $\sigma = 1.0$.

First, we investigate how the quantization factor affects the performance. Assume the noise variance is 0.04 and let the step size be 0.6. We plot in Fig. 1 the steady-state EMSE, as well as the lower and upper bounds of (36), versus the quantization factor γ. Here the steady-state EMSE is calculated as an average of 2000 Monte Carlo simulations. For each Monte Carlo simulation, 500 iterations are run and the last 100 iterations are used to compute the EMSE. From Fig. 1 we observe: 1) the steady-state EMSE indeed lies between the derived lower and upper bounds; 2) when the quantization factor approaches zero, the steady-state EMSE of the QKLMS approaches that of the KLMS, which can be theoretically calculated as 0.0171; 3) with quantization factor increasing, the steady-state EMSE will not always increase; and 4) the steady-state EMSE of the QKLMS can even be smaller than that of the KLMS. In this example, the steady-state EMSE attains its smallest value when $\gamma \approx 2.0$. One possible reason for this is that, when quantization factor (also the quantization size) equals 2.0, it is more likely to yield a codebook that

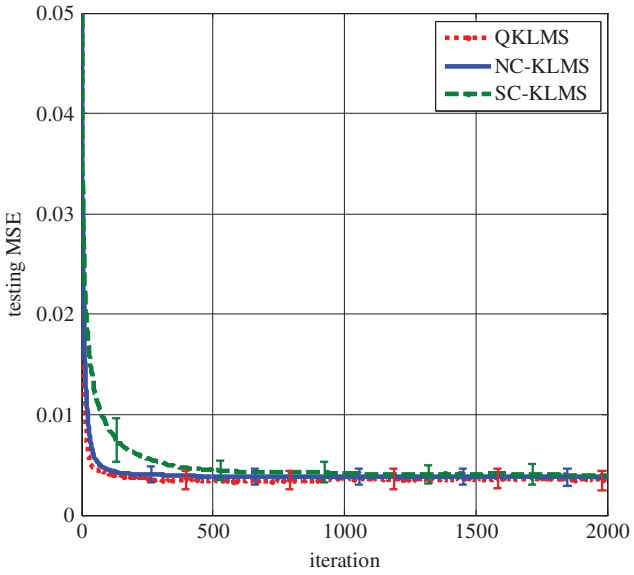

Fig. 4. Convergence curves in terms of the testing MSE for QKLMS, NC-KLMS, and SC-KLMS in static function estimation. The parameters of the algorithms are chosen such that they produce almost the same testing MSE at final stage of adaptation.

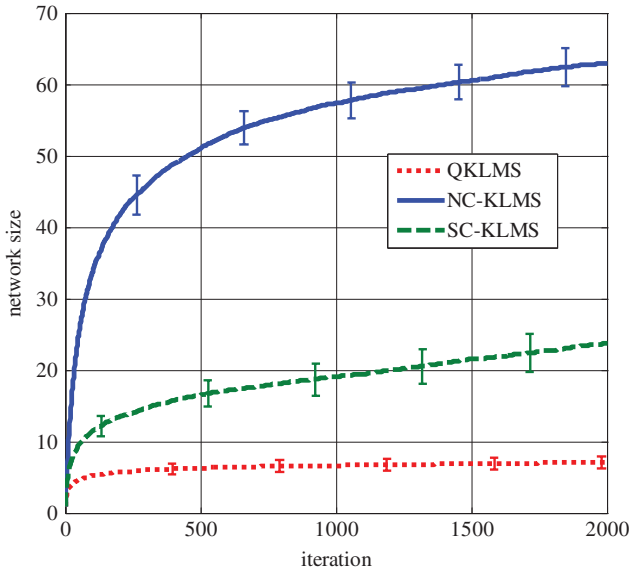

Fig. 5. Network size evolution curves for QKLMS, NC-KLMS, and SC-KLMS in static function estimation. The parameters of the algorithms are chosen such that they produce almost the same testing MSE at final stage of adaptation (see Fig. 4).

is *close* to the ideal codebook $\{-1, 1\}$ which, corresponds to the centers of the desired mix-Gaussian function (note that the distance between the desired centers is just 2.0). Fig. 2 illustrates the ensemble convergence curves for QKLMS ($\gamma = 2.0$) and KLMS. It is evident that the QKLMS can outperform the original KLMS in terms of the EMSE. The network size of QKLMS can be very small. Fig. 3 shows the final network size averaged over 2000 Monte Carlo simulations versus the quantization factor. As expected, when quantization factor γ increases, the network size will decrease dramatically. For very large quantization factor, the network size reduces to 1 (using only one Gaussian function to estimate the desired mapping).

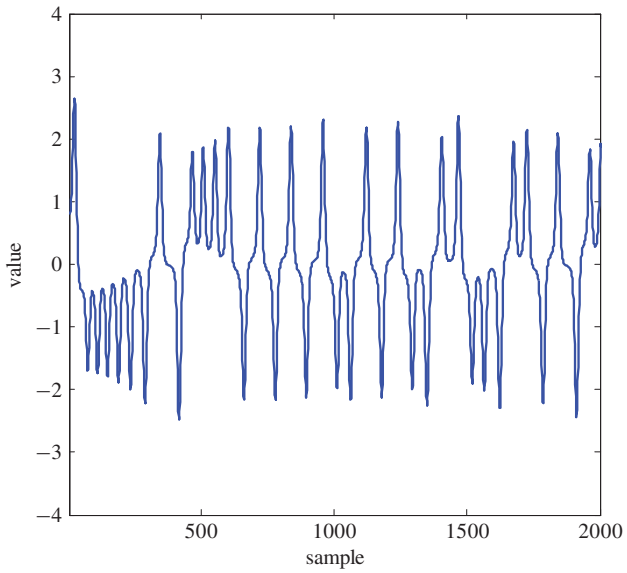

Fig. 6. Segment of the processed Lorenz time series.

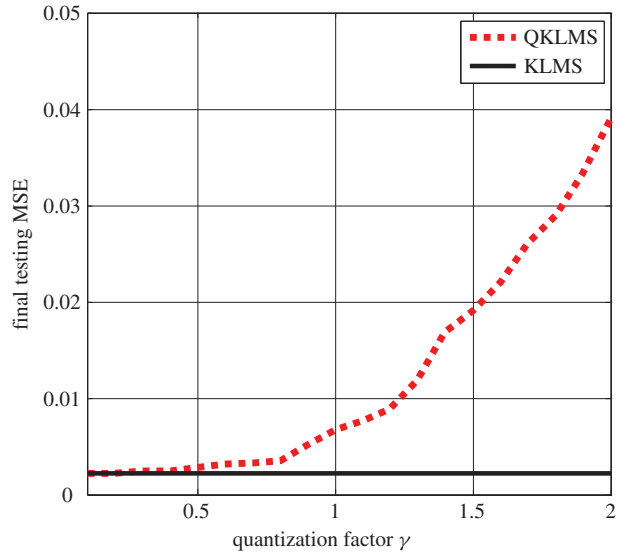

Fig. 7. Effect of the quantization factor γ on final testing MSE in Lorenz time series prediction.

Second, we compare the performance of the QKLMS with that of the sparsified KLMS. The novelty criterion and surprise criterion (which includes the ALD and variance criterion as special cases) are two typical sparsification criteria. In the following, we use NC-KLMS and SC-KLMS to denote, respectively, the novelty and surprise criterion based sparsified KLMS. Assume the noise variance is 0.01. The convergence curves in terms of the testing MSE (along with its standard deviation) averaged over 100 Monte Carlo simulations for QKLMS, NC-KLMS and SC-KLMS are plotted in Fig. 4, and the corresponding network size evolution curves are depicted in Fig. 5. In the simulation, the testing MSE is calculated on the basis of 200 noise-free test samples (the filter is fixed in the testing phase). The step sizes of all the algorithms are set as 0.1, and the other parameters of the three algorithms are selected such that they produce almost the same testing

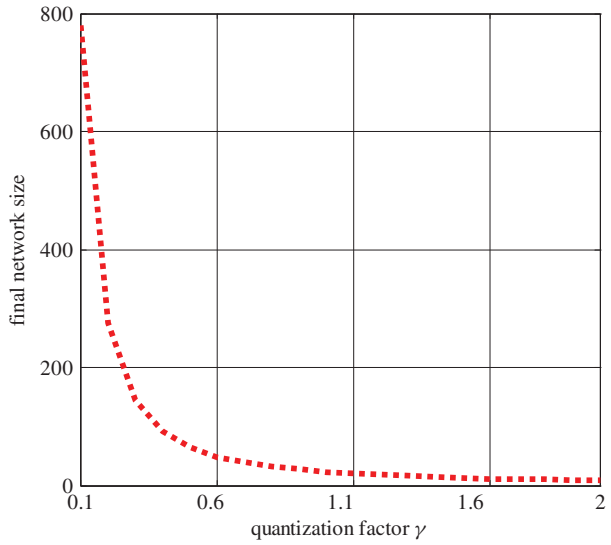

Fig. 8. Effect of the quantization factor γ on final network size in Lorenz time-series prediction.

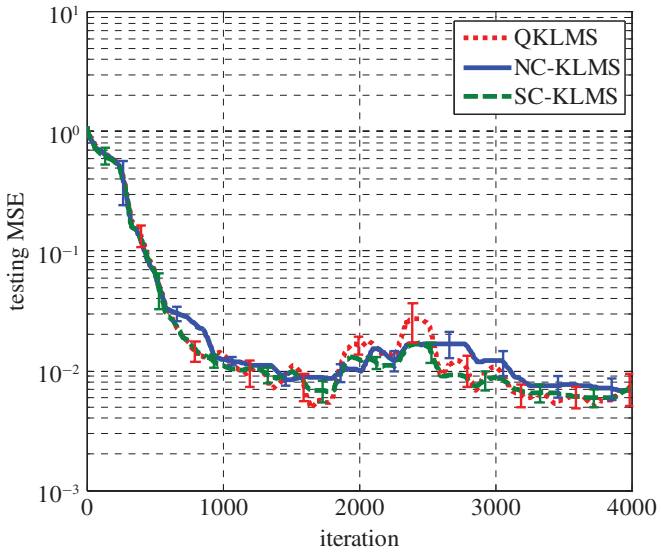

Fig. 9. Convergence curves in terms of the testing MSE for QKLMS, NC-KLMS, and SC-KLMS in Lorenz time series prediction. The parameters of the algorithms are chosen such that they produce almost the same testing MS.

MSE at the final stage of adaption. Specifically, for QKLMS, we set the quantization factor $\gamma = 0.8$, for NC-KLMS, we set the distance threshold $\delta_1 = 0.07$, and the error threshold $\delta_2 = 0.005$, for SC-KLMS, we set the regularization parameter $\lambda = 0.1$, the upper threshold of the surprise measure $T_1 = 100$, and the lower threshold $T_2 = -0.7$. As one can see clearly from Fig. 5, the QKLMS performs best, as it yields the smallest network size.

B. Short-Term Chaotic Time-Series Prediction

Consider the Lorenz chaotic system whose states are governed by the differential equations [21]

$$\begin{cases} \frac{dx}{dt} = -\beta x + yz \\ \frac{dy}{dt} = \delta(z - y) \\ \frac{dz}{dt} = -xy + \rho y - z \end{cases} \tag{39}$$

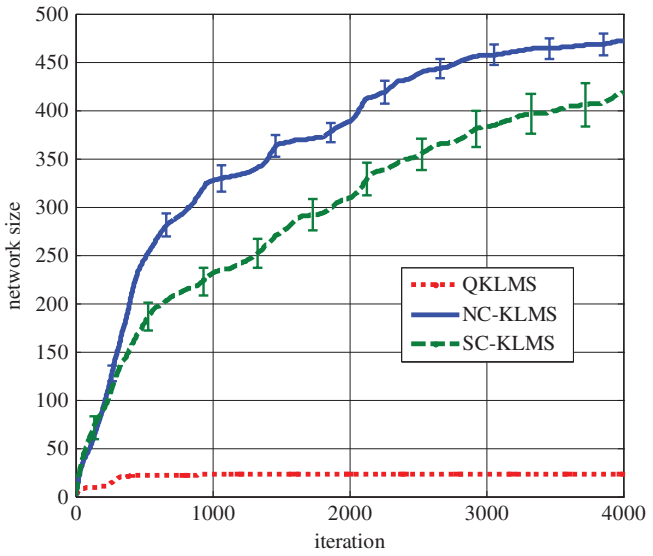

Fig. 10. Network size evolution curves for QKLMS, NC-KLMS, and SC-KLMS in Lorenz time-series prediction. The parameters of the algorithms are chosen such that they produce almost the same testing MSE (see Fig. 9).

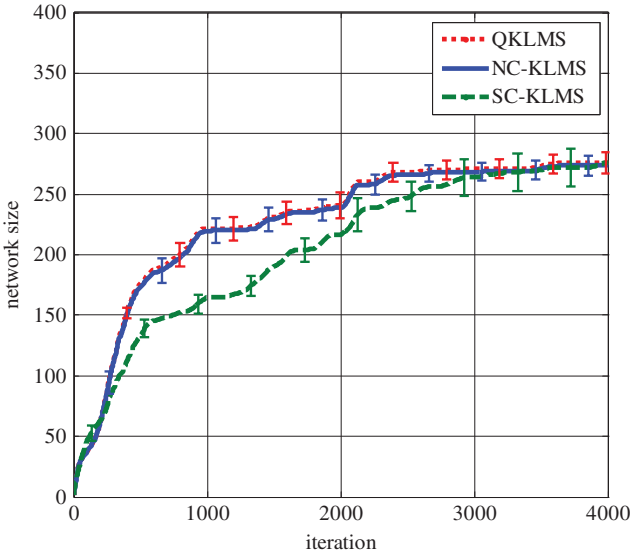

Fig. 11. Network size evolution curves for QKLMS, NC-KLMS, and SC-KLMS in Lorenz time-series prediction. The parameters of the algorithms are chosen such that they produce almost the same final network size.

where the parameters are set as $\beta = 8/3$, $\delta = 10$, and $\rho = 28$. The sample data are obtained using first-order approximation with step size 0.01. The first component (namely x) is used in the following for the short-term prediction task. The signal is preprocessed to be zero mean and unit variance before the modeling. A segment of the processed Lorenz time series is shown in Fig. 6.

The problem setting for short-term prediction is as follows: the previous five points $\boldsymbol{u}(i) = [x(i-5), x(i-4), \ldots, x(i-1)]^T$ are used as the input vector to predict the current value $x(i)$, which is the desired response here. In the simulations below, the kernel size is set as $\sigma = 0.707$, and unless mentioned otherwise, the step sizes involved are set as $\eta = 0.1$. The effect of the quantization factor on final

TABLE I
PARAMETER SETTINGS FOR DIFFERENT ALGORITHMS

	To produce the same testing MSE	To produce the same final network size
QKLMS	$\gamma = 1.0$	$\gamma = 0.2$
NC-KLMS	$\delta_1 = 0.1,\ \delta_2 = 0.001$	$\delta_1 = 0.142,\ \delta_2 = 0.001$
SC-KLMS	$\lambda = 0.005,$ $T_1 = 300,\ T_2 = -0.5$	$\lambda = 0.01,$ $T_1 = 300,\ T_2 = 0.0$

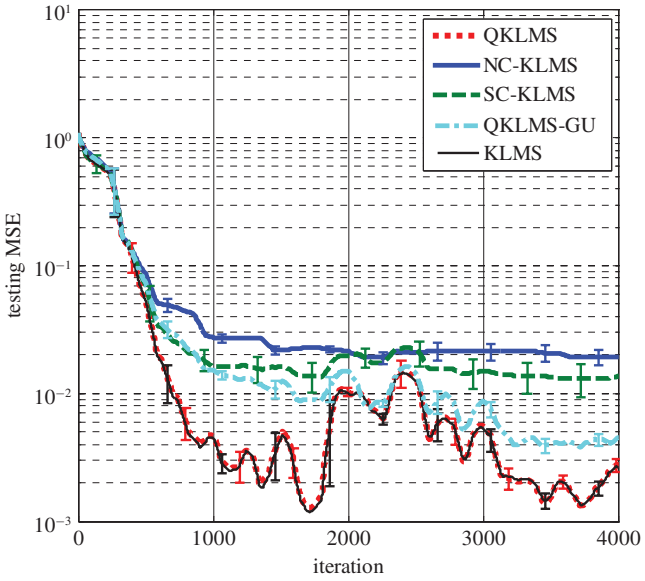

Fig. 12. Convergence curves in terms of the testing MSE for QKLMS, NC-KLMS, SC-KLMS, QKLMS-GU, and the original KLMS in Lorenz time-series prediction. The parameters of the algorithms (except KLMS) are chosen such that they produce almost the same final network size (see Fig. 11).

testing MSE is shown in Fig. 7, whilst the effect on the final network size is shown in Fig. 8. For each quantization factor, 100 independent simulations are run with different segments of the signal. The length of each segment (or equivalently, the number of iterations in each simulation) is 4000. The final network size is calculated as an average over 100 simulations. The final testing MSE is calculated as an average over the last 1000 iterations in the ensemble learning curves over 100 simulations. For each iteration cycle, the testing MSE is calculated on the basis of 200 test data (the filter is fixed in the testing phase). It can be seen from Fig. 7 that, when the quantization factor is small (say, less than 0.5), the final testing MSE of the QKLMS will be very close to that of the KLMS (for comparison purpose, we also plot in the figure the final testing MSE of the KLMS). As shown in Fig. 8, with quantization factor increasing, the final network size will decrease dramatically. Even for a small quantization factor (say, 0.1), the final network size will be much smaller than 4000 (which is the final network size of the KLMS).

The performance comparisons between QKLMS, NC-KLMS, and SC-KLMS in short-term Lorenz time series prediction, are presented in Figs. 9–12, where all the simulation results are averaged over 100 simulations run with different segments of the signal. In Figs. 9 and 10, the parameters of the three algorithms are chosen such that they produce almost the same testing MSE (see Fig. 9)[4], while in Figs. 11 and 12, the parameters are selected such that the algorithms yield almost the same final network size (see Fig. 11). Table I lists the specific parameter settings. Simulation results clearly indicate that the QKLMS exhibits much better performance, i.e., achieves either much smaller network size or much smaller testing MSE than NC-KLMS and SC-KLMS. For further comparison purpose, we also plot in Fig. 12 the convergence curves of the QKLMS-GU and the original KLMS. It is evident that the QKLMS performs better than QKLMS-GU and achieves almost the same testing MSE as the KLMS. In the simulation, the QKLMS-GU and QKLMS are set the same quantization factor, and hence their network sizes are always identical. The step size for GU (LMS) in QKLMS-GU is experimentally set to 0.002.

V. CONCLUSION

Quantization techniques have been widely used in data compression where a larger dataset is represented by a smaller set of code vectors. This paper used the idea of quantization to compress the input (or feature) space of the kernel adaptive filters so as to curb the growth of the RBF structure. Unlike conventional sparsification methods, the new approach utilizes the "redundant" data to update the coefficient of the closest center, and hence may yield better accuracy and a more compact network. Based on a simple online VQ method, we developed a QKLMS algorithm. The mean square convergence analysis for this algorithm has been studied. A sufficient condition for mean square convergence, as well as a lower and upper bound on the theoretical value of the steady-state EMSE, was derived based on energy conservation relation. Simulation results have confirmed the theoretical analysis and shown the excellent performance of QKLMS.

There are many problems that need to be studied in the future. The first and probably the most important one is how to design a more efficient VQ method. The simple online VQ method of this paper considers only the distance measure in input (or feature) space. It might be more reasonable to incorporate other information, such as the prediction errors and the distance in the desired signal space. The idea of coding theory or information bottleneck can also be applied in developing a new online VQ method. How to determine the quantization size (coarseness of quantization) was another important problem. In general, the choice of the quantization size depends upon a tradeoff between accuracy and complexity (network size). For a nonstationary learning system, the quantization size should be adaptive. In future study, we should also develop the quantized version of other kernel adaptive algorithms, such as the quantized KAPA, quantized KRLS, and so on.

ACKNOWLEDGMENT

The authors would like to thank the reviewers for pointing out some related and significant references they had overlooked.

[4]The largest fluctuations in all the convergence curves are due to the switch between two attractors in Lorenz system.

REFERENCES

[1] B. Scholkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond*. Cambridge, MA: MIT Press, 2002.

[2] V. Vapnik, *The Nature of Statistical Learning Theory*. New York: Springer-Verlag, 1995.

[3] J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle, *Least Squares Support Vector Machine*. Singapore: World Scientific, 2002.

[4] F. Girosi, M. Jones, and T. Poggio, "Regularization theory and neural networks architectures," *Neural Comput.*, vol. 7, no. 2, pp. 219–269, Mar. 1995.

[5] B. Scholkopf, A. J. Smola, and K. Muller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Comput.*, vol. 10, no. 5, pp. 1299–1319, Jul. 1998.

[6] M. Espinoza, J. A. K. Suykens, and B. De Moor, "Fixed-size least squares support vector machines: A large scale application in electrical load forecasting," *Comput. Manage. Sci., Special Issue Support Vector Mach.*, vol. 3, no. 2, pp. 113–129, Apr. 2006.

[7] L. Hoegaerts, J. A. K. Suykens, J. Vandewalle, and B. De Moor, "Subset based least squares subspace regression in RKHS," *Neurocomputing*, vol. 63, pp. 293–323, Jan. 2005.

[8] K. De Brabanter, J. De Brabanter, J. A. K. Suykens, and B. De Moor, "Optimized fixed-size kernel models for large data sets," *Comput. Stat. Data Anal.*, vol. 54, no. 6, pp. 1484–1504, 2010.

[9] J. Kivinen, A. J. Smola, and R. C. Williamson, "Online learning with kernels," *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2165–2176, Aug. 2004.

[10] J. Platt, "A resource-allocating network for function interpolation," *Neural Comput.*, vol. 3, no. 2, pp. 213–225, Jun. 1991.

[11] G. Huang, P. Saratchandran, and N. Sundararajan, "A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation," *IEEE Trans. Neural Netw.*, vol. 16, no. 1, pp. 57–67, Jan. 2005.

[12] K. Slavakis, S. Theodoridis, and I. Yamada, "Online kernel-based classification using adaptive projection algorithms," *IEEE Trans. Signal Process.*, vol. 56, no. 7, pp. 2781–2796, Jul. 2008.

[13] O. Dekel, S. Shalev-Shwartz, and Y. Singer, "The Forgetron: A kernel-based perceptron on a budget," *SIAM J. Comput.*, vol. 37, no. 5, pp. 1342–1372, Jan. 2008.

[14] W. Liu, J. Príncipe, and S. Haykin, *Kernel Adaptive Filtering: A Comprehensive Introduction*. New York: Wiley, 2010.

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

[16] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Min. Knowl. Disc.*, vol. 2, no. 2, pp. 121–167, 1998.

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

[18] P. Bouboulis and S. Theodoridis, "Extension of Wirtinger's calculus to reproducing kernel Hilbert spaces and the complex kernel LMS," *IEEE Trans. Signal Process.*, vol. 59, no. 3, pp. 964–978, Mar. 2011.

[19] W. Liu and J. Príncipe, "Kernel affine projection algorithms," *EURASIP J. Adv. Signal Process.*, vol. 2008, no. 1, pp. 1–13, 2008.

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

[21] 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.

[22] L. Csato and M. Opper, "Sparse on-line Gaussian processes," *Neural Comput.*, vol. 14, no. 3, pp. 641–668, Mar. 2002.

[23] 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.

[24] D. Duttweiler, "Adaptive filter performance with nonlinearities in the correlation multiplier," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 30, no. 4, pp. 578–586, Aug. 1982.

[25] P. Xue and B. Liu, "Adaptive equalizer using finite-bit power-of-two quantizer," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 34, no. 6, pp. 1603–1611, Dec. 1986.

[26] W. Sethares and C. R. Johnson, "A comparison of two quantized state adaptive algorithms," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 37, no. 1, pp. 138–143, Jan. 1989.

[27] E. Eweda, "Convergence analysis and design of an adaptive filter with finite-bit power-of-two quantized error," *IEEE Trans. Circuits Syst. II: Analog Digital Signal Process.*, vol. 39, no. 2, pp. 113–115, Feb. 1992.

[28] J. C. M. Bermudez and N. J. Bershad, "Transient and tracking performance analysis of the quantized LMS algorithm for time-varying system identification," *IEEE Trans. Signal Process.*, vol. 44, no. 8, pp. 1990–1997, Aug. 1996.

[29] M. A. Aldajani, "Logarithmic quantization in the least mean squares algorithm," *Digital Signal Process.*, vol. 18, no. 3, pp. 321–333, May 2008.

[30] M. U. Otaru, A. Zerguine, and L. Cheded, "Channel equalization using simplified least mean-fourth algorithm," *Digital Signal Process.*, vol. 21, no. 3, pp. 447–465, May 2011.

[31] Y. Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Commun.*, vol. 28, no. 1, pp. 84–95, Jan. 1980.

[32] P. A. Chou, T. Lookabaugh, and R. M. Gray, "Entropy-constrained vector quantization," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 37, no. 1, pp. 31–42, Jan. 1989.

[33] T. Lehn-Schiøler, A. Hegde, D. Erdogmus, and J. C. Príncipe, "Vector quantization using information theoretic concepts," *Natural Comput.*, vol. 4, no. 1, pp. 39–51, 2005.

[34] S. Craciun, D. Cheney, K. Gugel, J. C. Sanchez, and J. C. Príncipe, "Wireless transmission of neural signals using entropy and mutual information compression," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 19, no. 1, pp. 35–44, Feb. 2011.

[35] A. H. Sayed, *Fundamentals of Adaptive Filtering*. New York: Wiley, 2003.

[36] N. R. Yousef and A. H. Sayed, "A unified approach to the steady-state and tracking analyses of adaptive filters," *IEEE Trans. Signal Process.*, vol. 49, no. 2, pp. 314–324, Feb. 2001.

[37] T. Y. Al-Naffouri and A. H. Sayed, "Adaptive filters with error nonlinearities: Mean-square analysis and optimum design," *EURASIP J. Appl. Signal Process.*, vol. 2001, no. 4, pp. 192–205, 2001.

[38] T. Y. Al-Naffouri and A. H. Sayed, "Transient analysis of data-normalized adaptive filters," *IEEE Trans. Signal Process.*, vol. 51, no. 3, pp. 639–652, May 2003.

[39] T. Y. Al-Naffouri and A. H. Sayed, "Transient analysis of adaptive filters with error nonlinearities," *IEEE Trans. Signal Process.*, vol. 51, no. 3, pp. 653–663, Mar. 2003.

Badong Chen (M'10) received the Ph.D. degree in computer science and technology from Tsinghua University, Beijing, China, in 2008.

He is currently a Post-Doctoral Associate with the Computational Neuro-Engineering Laboratory, University of Florida, Gainesville. His current research interests include statistical signal processing, information theoretic learning, and kernel adaptive filters, as well as their applications in neuroscience, brain-machine interface, bioinformatics, sensors, and social networks.

Songlin Zhao (S'09) was born in Jiangsu, China. She received the B.S. and M.S. degrees in electrical engineering from Xi'an Jiao Tong University, Shannxi, China, in 2006 and 2009, respectively. She is currently pursuing the Ph.D. degree with the Computational Neuro-Engineering Laboratory, University of Florida, Gainesville.

Her current research interests include signal processing, machine learning, and pattern recognition.

Pingping Zhu (S'10) received the B.S. and M.S. degrees in electronics and information engineering from the Institute for Pattern Recognition & Artificial Intelligence, Huazhong University of Science and Technology, Wuhan, China, in 2006 and 2008, respectively. He is currently pursuing the Ph.D. degree in electrical and computer engineering with the University of Florida, Gainesville.

He has been with the Computational NeuroEngineering Laboratory, University of Florida, since 2009, under the supervision of Dr. J. C. Príncipe. His current research interests include kernel methods, Kalman filter, signal processing, and machine learning.

José C. Príncipe (M'83–SM'90–F'00) is currently a Distinguished Professor of electrical and biomedical engineering with the University of Florida, Gainesville. He is the BellSouth Professor and the Founder and Director of the Computational NeuroEngineering Laboratory, University of Florida. His current research interests include biomedical signal processing, in particular electroencephalogram signals, and the modeling and applications of adaptive systems.

He is a past Editor-in-Chief of the IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING, past President of the International Neural Network Society, former Secretary of the Technical Committee on Neural Networks of the IEEE Signal Processing Society, and a former member of the Scientific Board of the Food and Drug Administration. He was awarded the IEEE Computational Intelligence Society Neural Networks Pioneer Award in 2011.