Neurocomputing 121 (2013) 160–169

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Kernel minimum error entropy algorithm

Badong Chen [a,*], Zejian Yuan [a], Nanning Zheng [a], José C. Príncipe [b]

[a] Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University, Xi'an 710049, China
[b] Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611 USA

ARTICLE INFO

Article history:
Received 1 November 2012
Received in revised form
17 April 2013
Accepted 19 April 2013
Available online 14 June 2013

Keywords:
Minimum error entropy
Kernel adaptive filters
Kernel minimum error entropy (KMEE)

ABSTRACT

As an alternative adaptation criterion, the minimum error entropy (MEE) criterion has been receiving increasing attention due to its successful use in, especially, nonlinear and non-Gaussian signal processing. In this paper, we study the application of error entropy minimization to kernel adaptive filtering, a new and promising technique that implements the conventional linear adaptive filters in reproducing kernel Hilbert space (RKHS) and obtains the nonlinear adaptive filters in original input space. The kernel minimum error entropy (KMEE) algorithm is derived, which is essentially a generalized stochastic information gradient (SIG) algorithm in RKHS. The computational complexity of KMEE is just similar to the kernel affine projection algorithm (KAPA). We also utilize the quantization approach to constrain the network size growth, and develop the quantized KMEE (QKMEE) algorithm. Further, we analyze the mean square convergence of KMEE. The energy conservation relation is derived and a sufficient condition that ensures the mean square convergence is obtained. The performance of the new algorithm is demonstrated in nonlinear system identification and short-term chaotic time series prediction.

1. Introduction

Kernel methods have become rather popular and have been successfully applied to various machine learning and signal processing tasks such as classification, regression, clustering, and feature selection. Successful examples of kernel methods include support vector machine (SVM) [1,2], kernel regularization networks [3], and kernel principal component analysis (KPCA) [4], etc. The main advantage of kernel methods is their ability to represent nonlinear functions linearly and universally in Reproducing Kernel Hilbert Space (RKHS) induced by a Mercer kernel, which in general results in a simple convex optimization problem with a unique global optimum. Since kernel methods essentially require only linear algebra, they are as simple as their linear counterparts in input space.

In recent years, more and more attention has also been paid to online kernel learning (OKL) methods [5,6], which learn the unknown nonlinear mapping incrementally and are particularly applicable to cases where data arrive sequentially. In this direction, the kernel adaptive filters (KAFs) [7] are one of the most emerging and promising areas of research. The KAFs are a family of nonlinear adaptive filters developed in RKHS by utilizing the linear structure and inner product of this space to implement the well-established linear adaptive filtering algorithms. Typical KAF algorithms include the kernel least mean square (KLMS) [8], kernel affine projection algorithms (KAPAs) [9], kernel recursive least squares (KRLS) [10], and extended kernel recursive least squares (EX-KRLS) [11], etc. With radial kernel function, the kernel adaptive filters build a linearly growing RBF network by allocating a new kernel unit for every new example with input as the center. The coefficients corresponding to each center are related to the errors at each sample.

Almost all existing KAF algorithms so far have utilized the mean square error (MSE) as the adaptation criterion. The MSE is mathematically tractable and under Gaussian assumption is an optimal criterion for linear filters. However, it is well-known that MSE may be a poor descriptor of optimality for nonlinear and non-Gaussian (e.g. heavy-tail or finite range distributions) situations, since it constraints only the second-order statistics. To address this issue, one can select some new criterion beyond second-order statistics that does not suffer from the limitation of Gaussian assumption and can improve performance in many realistic scenarios. Information theoretic quantities (such as entropy, divergence, and mutual information) as adaptation criteria attract ever-increasing attention to this end, since they can capture higher-order statistics and information content of signals rather than simply their energy [12–17]. Combining information theoretic criteria and kernel (Parzen) density estimation [18], one can obtain a nonparametric and sample based methodology to learn arbitrary

* Corresponding author. Tel.:+86 18392892686.
E-mail addresses: chenbd@mail.xjtu.edu.cn, chenbd04@mails.tsinghua.edu.cn (B. Chen).

0925-2312/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.neucom.2013.04.037

nonlinear systems, namely, the information theoretic learning (ITL) [12]. In a recent paper [19], we have applied the maximum correntropy (MC) criterion to kernel adaptive filtering, and developed the kernel maximum correntropy (KMC) algorithm, which is shown to be robust to outlier noise. The MC criterion belongs to information theoretic criteria just because correntropy is closely related to Renyi's quadratic entropy, that is, the negative logarithm of the mean of correntropy yields the Parzen estimate of Renyi's quadratic entropy [20].

In the present work, we will apply another more celebrated information theoretic criterion, the minimum error entropy (MEE) criterion [12–17], to kernel adaptive filtering. In particular, we derive the kernel minimum error entropy (KMEE) algorithm, which is essentially a generalized stochastic information gradient (SIG) algorithm [21] in kernel space. The computational complexity of KMEE is not expensive and is just similar to the KAPA algorithm. To further reduce the computational and memory requirements, one can use the quantization approach to constrain the filter network size, which leads to the quantized KMEE (QKMEE) algorithm. We also carry out the mean square convergence analysis for KMEE. The energy conservation relation is derived, and a sufficient condition for mean square convergence is obtained. The superior performance of the new algorithm has been illustrated by simulation experiments on nonlinear system identification and chaotic time series prediction.

The organization of the paper is as follows. In Section 2, we briefly review the kernel adaptive filtering. In Section 3, we derive the KMEE and QKMEE algorithms. In Section 4, we analyze the mean square convergence. In Section 5, we present simulation results to illustrate the performance. Finally, in Section 6, we give the conclusion.

2. Brief review of kernel adaptive filtering

Suppose that the input space $\mathbb{U}$ is a compact subset of the m dimensional Euclidean space $\mathbb{R}^m$, and the desired response (output) $d(i)\in\mathbb{R}$ at time instant $i\in\mathbb{N}$ is related to input vector $\boldsymbol{u}(i)\in\mathbb{U}$ via

$$d(i)=f^*(\boldsymbol{u}(i))+v(i) \tag{1}$$

where f^* denotes an unknown nonlinear mapping (between input–output) that we wish to estimate, $v(i)$ accounts for disturbance noise. The task of nonlinear adaptive filtering is then to search the mapping f^* incrementally in a hypothesis space $\mathscr{H}$ with input–output examples $\{\boldsymbol{u}(i),d(i)\}$ which arrive sequentially. In fact, the learning rule of a nonlinear adaptive filter can be, in general, expressed as

$$\begin{cases} f_0=0 \\ f_i=f_{i-1}+\Delta f_i \end{cases} \tag{2}$$

where $f_i\in\mathscr{H}$ denotes the learned mapping (hypothesis) at iteration i, and Δf_i is the adjustment term.

In kernel adaptive filtering, the hypothesis space for learning will be a Reproducing Kernel Hilbert Space (RKHS) $\mathscr{H}_k$ associated with a Mercer (reproducing) kernel $\kappa(\boldsymbol{u},\boldsymbol{u}')$, a continuous, symmetric, and positive definite function $\kappa:\mathbb{U}\times\mathbb{U}\to\mathbb{R}$ [1,2]. According to Moore–Aronszajn theorem [1,2], every Mercer kernel κ induces a unique function space, namely the RKHS $\mathscr{H}_\kappa$, whose reproducing kernel is κ, which satisfies: (1) $\forall\boldsymbol{u}\in\mathbb{U}$, the function $\kappa(\boldsymbol{u},.)\in\mathscr{H}_\kappa$, and (2) $\forall\boldsymbol{u}\in\mathbb{U}$, and for every $f\in\mathscr{H}_\kappa$, $\langle f,\kappa(\boldsymbol{u},.)\rangle_{\mathscr{H}_\kappa}=f(\boldsymbol{u})$, where $\langle .,.\rangle_{\mathscr{H}_\kappa}$ denotes the inner product in $\mathscr{H}$. If Mercer kernel κ is strictly positive definite (SPD), the induced RKHS $\mathscr{H}_\kappa$ will be universal (i.e. dense in the space of continuous functions over $\mathbb{U}$). In machine learning, the most common SPD kernel is the Gaussian kernel:

$$\kappa(\boldsymbol{u},\boldsymbol{u}')=\exp\left(-\frac{1}{2\sigma^2}||\boldsymbol{u}-\boldsymbol{u}'||^2\right)=\exp(-\zeta||\boldsymbol{u}-\boldsymbol{u}'||^2) \tag{3}$$

where $\sigma>0$ is the kernel size (or kernel width), and $\zeta=1/2\sigma^2$ is called the kernel parameter.

It is worth noting that the RKHS $\mathscr{H}_\kappa$ is isometric–isomorphic to a high dimensional feature space (a Euclidean space in which the training data are embedded). According to Mercer's theorem [1,2], any Mercer kernel $\kappa(\boldsymbol{u},\boldsymbol{u}')$ induces a mapping $\boldsymbol{\varphi}$ from the input space $\mathbb{U}$ to a high (possibly infinite) dimensional feature space $\mathbb{F}_\kappa$. In the feature space, the inner products can be calculated using the kernel trick $\boldsymbol{\varphi}(\boldsymbol{u})^T\boldsymbol{\varphi}(\boldsymbol{u}')=\kappa(\boldsymbol{u},\boldsymbol{u}')$. When identifying $\boldsymbol{\varphi}(\boldsymbol{u})=\kappa(\boldsymbol{u},.)$, the two Hilbert spaces are identical.

With RKHS $\mathscr{H}_\kappa$ as the hypothesis space, the goal of adaptive filtering (or regression) is usually to find a function f in $\mathscr{H}_\kappa$ such that

$$\min_{f\in\mathscr{H}_k}\sum_{i=1}^{N}(d(i)-f(\boldsymbol{u}(i)))^2+\lambda|f|^2_{\mathscr{H}_k} \tag{4}$$

where $||.||_{\mathscr{H}_k}$ denotes the norm in $\mathscr{H}_k$, $\lambda\geq 0$ is the regularization factor that controls the smoothness of the solution. In feature space, the goal will be to find a high dimensional weight vector $\boldsymbol{\Omega}\in\mathbb{F}_\kappa$ that minimizes

$$\min_{\boldsymbol{\Omega}\in\mathbb{F}_\kappa}\sum_{i=1}^{N}(d(i)-\boldsymbol{\Omega}^T\boldsymbol{\varphi}(i))^2+\lambda||\boldsymbol{\Omega}||^2_{\mathbb{F}_\kappa} \tag{5}$$

where $\boldsymbol{\varphi}(i)=\boldsymbol{\varphi}(\boldsymbol{u}(i))$.

Solving (4) (or (5)) in batch manner usually requires significant memory and computational burden due to the necessity of calculating a large Gram matrix, whose dimension equals the number of input patterns. Kernel adaptive filters (KAFs), however, provide more efficient alternatives that search the solution in an online manner, without explicitly computing the Gram matrix. Up to now, there have been many KAF algorithms; in the following we only briefly describe the KLMS and KAPA algorithms. For a more comprehensive review on KAFs, we refer the readers to [7].

The KLMS is actually the least mean square (LMS) algorithm in feature space, which is just a *stochastic gradient* based algorithm without requiring explicit regularization[1] [8]. By adopting the instantaneous square error as the adaptation cost, the KLMS can be simply obtained as

$$\begin{aligned}\boldsymbol{\Omega}_i&=\boldsymbol{\Omega}_{i-1}-\eta\frac{\partial}{\partial\boldsymbol{\Omega}_{i-1}}\left(\frac{1}{2}e^2(i)\right)\\&=\boldsymbol{\Omega}_{i-1}-\eta e(i)\frac{\partial}{\partial\boldsymbol{\Omega}_{i-1}}(d(i)-\boldsymbol{\Omega}_{i-1}{}^T\boldsymbol{\varphi}(i))\\&=\boldsymbol{\Omega}_{i-1}+\eta e(i)\boldsymbol{\varphi}(i)\end{aligned} \tag{6}$$

where factor 1/2 is introduced to simplify the mathematical formulation, η denotes the step size parameter, and $e(i)$ is the prediction error at iteration i, i.e. $e(i)=d(i)-\boldsymbol{\Omega}_{i-1}{}^T\boldsymbol{\varphi}(i)$. The update Eq. (6) can alternatively be expressed in RKHS:

$$f_i=f_{i-1}+\eta e(i)\kappa(\boldsymbol{u}(i),.) \tag{7}$$

The KAPA is an affine projection algorithm (APA) [22] in feature space. Unlike KLMS, which simply uses the instantaneous values for approximating the MSE criterion $E[e^2(i)]$, KAPA employs a much better approximation. Specifically, the MSE is replaced by an approximation from the L most recent errors. Let $\boldsymbol{e}(i)=[e(i,i-L+1),e(i,i-L+2),\ldots,e(i,i)]^T$, where $e(i,j)$ denotes the prediction error with hypothesis f_{i-1} and input–output pair $(\boldsymbol{u}(j),d(j))$, that is,

[1] It has been pointed out in [8] that the gradient based search has an inherent regularization mechanism.

$e(i,j)=d(j)-f_{i-1}(\boldsymbol{u}(j))$. Then the KAPA algorithm can be derived as

$$\begin{aligned}\boldsymbol{\Omega}_i &= \boldsymbol{\Omega}_{i-1}-\eta\frac{\partial}{\partial\boldsymbol{\Omega}_{i-1}}\left(\frac{1}{2}||\boldsymbol{e}(i)||^2\right)\\ &= \boldsymbol{\Omega}_{i-1}-\eta\left[\frac{\partial}{\partial\boldsymbol{\Omega}_{i-1}}\boldsymbol{e}(i)^T\right]\boldsymbol{e}(i)\\ &= \boldsymbol{\Omega}_{i-1}-\eta\left[\frac{\partial}{\partial\boldsymbol{\Omega}_{i-1}}(\boldsymbol{d}(i)-\boldsymbol{\Phi}(i)^T\boldsymbol{\Omega}_{i-1})^T\right]\boldsymbol{e}(i)\\ &= \boldsymbol{\Omega}_{i-1}+\eta\boldsymbol{\Phi}(i)\boldsymbol{e}(i)\end{aligned} \tag{8}$$

where $\boldsymbol{d}(i)=[d(i-L+1),d(i-L+2),\ldots d(i)]^T$, $\boldsymbol{\Phi}(i)=[\boldsymbol{\varphi}(i-L+1),\boldsymbol{\varphi}(i-L+2),\ldots,\boldsymbol{\varphi}(i)]$. In RKHS the KAPA algorithm (8) becomes

$$f_i=f_{i-1}+\eta\mathbf{K}(i)\boldsymbol{e}(i) \tag{9}$$

where $\mathbf{K}(i)=[\kappa(\boldsymbol{u}(i-L+1),.),\kappa(\boldsymbol{u}(i-L+2),.),\cdots,\kappa(\boldsymbol{u}(i),.)]$. Besides the above basic form, there are also several variants of the KAPA algorithm (for details, see [7,9]).

From (7) and (9), one can see that the learned mapping by KLMS or KAPA can be expressed as

$$f_i(.)=\sum_{j=1}^{i}\boldsymbol{\alpha}_j(i)\kappa(\boldsymbol{u}(j),.) \tag{10}$$

where the coefficient vector $\boldsymbol{\alpha}(i)=[\boldsymbol{\alpha}_1(i),\boldsymbol{\alpha}_2(i),\ldots,\boldsymbol{\alpha}_i(i)]^T$ is related to the errors at each sample. With radial kernel function, the solution is exactly a linearly growing RBF network.

3. KMEE algorithm

Existing KAF algorithms are mainly based on the MSE (or least squares) criterion. MSE is not always a suitable criterion especially in nonlinear or non-Gaussian situations. Hence, it is attractive to develop new KAF algorithms based on a non-MSE (non-quadratic) criterion. Recently, we have applied the maximum correntropy (MC) criterion to kernel adaptive filtering, and developed the kernel maximum correntropy (KMC) algorithm [19]. The correntropy cost for adaptive filtering is $E[\kappa_c(e(i))]$, where κ_c is a shift invariant Mercer kernel (not to be confused with the Mercer kernel κ of RKHS H_κ). Similar to KLMS, the KMC is also a stochastic gradient based algorithm. If κ_c is a Gaussian kernel, the KMC algorithm can be expressed in the form [19]:

$$\boldsymbol{\Omega}_i=\boldsymbol{\Omega}_{i-1}+\eta\kappa_c(e(i))e(i)\boldsymbol{\varphi}(i) \tag{11}$$

which is, in fact, a KLMS algorithm with variable step size $\eta\kappa_c(e(i))$.

Although correntropy has a close relationship with Renyi's quadratic entropy, it is not a true entropy definition. Actually, the MC criterion is just a special case of Huber's M-estimation criteria [20]. In the following, we will use the error's entropy as the adaptation cost for kernel adaptive filtering. First, let's briefly describe the minimum error entropy (MEE) criterion.

3.1. MEE criterion

MEE is an important learning criterion in information theoretic learning (ITL) [12]. In ITL framework, the supervised learning problems (filtering, classification, etc.) can be formulated as minimizing the error entropy between model output and the desired response. As entropy quantifies the average uncertainty or diversity of a random variable, its minimization makes the error concentrated. With Shannon's entropy definition, the optimal weight vector in feature space, under MEE criterion, can be formulated as

$$\begin{aligned}\boldsymbol{\Omega}_{opt} &= \arg\min_{\boldsymbol{\Omega}\in\mathbb{F}_\kappa} H_S(e)\\ &= \arg\min_{\boldsymbol{\Omega}\in\mathbb{F}_\kappa}\left\{-\int p_e(\xi)\log\ p_e(\xi)d\xi\right\}\\ &= \arg\min_{\boldsymbol{\Omega}\in\mathbb{F}_\kappa} E_e[-\log\ p_e(e)]\end{aligned} \tag{12}$$

where $p_e(.)$ denotes the probability density function (PDF) of the error $e=d-\boldsymbol{\Omega}^T\boldsymbol{u}$, and E_e stands for the expectation operator with respect to random variable e.

A well-known generalization of Shannon entropy is Renyi entropy [12,23]. The α-order Renyi entropy of e is defined by ($\alpha>0,\alpha\neq1$)

$$H_\alpha(e)=\frac{1}{1-\alpha}\log V_\alpha(e) \tag{13}$$

where $V_\alpha(e)$ is the α-order information potential (IP) [12,15]:

$$V_\alpha(e)=\int p_e^\alpha(\xi)d\xi=E_e[p_e^{\alpha-1}(e)] \tag{14}$$

By L'Hoptital's rule one can easily show that as $\alpha\to1$, Renyi entropy converges to Shannon entropy. If using Renyi entropy as the cost, the optimization problem in (12) becomes

$$\begin{aligned}\boldsymbol{\Omega}_{opt} &= \arg\min_{\boldsymbol{\Omega}\in\mathbb{F}_\kappa}\left\{\frac{1}{1-\alpha}\log\ V_\alpha(e)\right\}\\ &= \begin{cases}\arg\min_{\boldsymbol{\Omega}\in\mathbb{F}_\kappa} V_\alpha(e) & if\ \alpha<1\\ \arg\max_{\boldsymbol{\Omega}\in\mathbb{F}_\kappa} V_\alpha(e) & if\ \alpha>1\end{cases}\end{aligned} \tag{15}$$

Due to the monotonic property of the logarithm function, minimizing Renyi entropy will be equivalent to minimizing (or maximizing for $\alpha>1$) the information potential. In adaptive system training, one often uses information potential $V_\alpha(e)$ as an optimality criterion [12,15].

A more general entropy definition is the ϕ- entropy defined as [24]

$$\begin{aligned}H_\phi(e) &= \int\phi(p_e(\xi))d\xi\\ &= E_e\left\{\frac{\phi[p_e(e)]}{p_e(e)}\right\}=E_e\{\psi[p_e(e)]\}\end{aligned} \tag{16}$$

where ϕ is a continuous concave real function over $[0,\infty)$, and $\psi(x)=\phi(x)/x$. Both Shannon entropy and information potential (negative information potential for $\alpha>1$) are special cases of the ϕ1entropy (see Table 1).

3.2. KMEE

Because ϕ-entropy is a very general and flexible entropy definition, we will use the ϕ-entropy of error as the adaptation criterion to derive the KMEE algorithm. In practice, the error distribution is usually unknown, and the analytical evaluation of ϕ-entropy is not possible. So, we have to estimate the ϕ-entropy using sample data. One simple way is to estimate the error's PDF based on available samples, and substitute the estimated PDF into the *sample mean approximation* of the ϕ-entropy (approximating the expected value by its sample mean). Like in KAPA algorithm, the approximation can be based on the L most recent errors: $\boldsymbol{e}(i)=[e(i,i-L+1),e(i,i-L+2),\ldots,e(i,i)]^T$, where $e(i,j)=d(j)-f_{i-1}(\boldsymbol{u}(j))$. Thus the ϕ-entropy of the error at iteration i can be estimated as

$$\hat{H}_\phi(e(i))=\frac{1}{L}\sum_{j=i-L+1}^{i}\psi[\hat{p}_e(e(i,j))] \tag{17}$$

Table 1
Some special cases of ϕ-entropy.

Entropy type	$\phi(x)$
Shannon entropy	$-x\log x$
α-order information potential ($\alpha<1$)	$x^\alpha(\alpha<1)$
Negative α-order information potential ($\alpha>1$)	$-x^\alpha(\alpha>1)$
Havrda–Charvat entropy	$(1-\beta)^{-1}(x^\beta-x)$, $\beta>0$, $\beta\neq1$

where $\hat{p}_e$ denotes the estimated PDF of the error $e(i)$, which can be estimated based on the same L errors. By kernel density estimation approach, $\hat{p}_e$ can be expressed as [18]

$$\hat{p}_e(\xi) = \frac{1}{L}\sum_{j=i-L+1}^{i} \kappa_d(\xi - e(i,j)) \tag{18}$$

where $\kappa_d(.)$ is the kernel function for density estimation (not the Mercer kernel κ of RKHS H_κ), which in general satisfies $\kappa_d(x)\geq 0$, and $\int \kappa_d(x)\,dx = 1$. Additionally, to make the estimator smooth, the kernel function $\kappa_d(.)$ is usually continuous and differentiable (and preferably symmetric and unimodal). Plugging (18) into (17), one obtains

$$\hat{H}_\phi(e(i)) = \frac{1}{L}\sum_{l=i-L+1}^{i} \psi\left[\frac{1}{L}\sum_{j=i-L+1}^{i} \kappa_d(e(i,l) - e(i,j))\right] \tag{19}$$

The estimated ϕ-entropy in (19) is computationally inefficient since there are double summations. In an online adaptation scenario, we are more interested in approximating ϕ-entropy *stochastically* by dropping the sample mean operator (the outer summation in (19)). This leads to the instantaneous ϕ-entropy

$$\tilde{H}_\phi(e(i)) = \psi\left[\frac{1}{L}\sum_{j=i-L+1}^{i} \kappa_d(e(i,i) - e(i,j))\right] \tag{20}$$

which involves only one summation and not two. Now, based on the adaptation cost (20), one can easily derive the KMEE algorithm as follows:

$$\begin{aligned}
\boldsymbol{\Omega}_i &= \boldsymbol{\Omega}_{i-1} - \eta \frac{\partial}{\partial \boldsymbol{\Omega}_{i-1}} (\tilde{H}_\phi(e(i))) \\
&= \boldsymbol{\Omega}_{i-1} - \eta \frac{\partial}{\partial \boldsymbol{\Omega}_{i-1}} \left(\psi\left(\frac{1}{L}\sum_{j=i-L+1}^{i} \kappa_d(e(i,i) - e(i,j))\right)\right) \\
&= \boldsymbol{\Omega}_{i-1} - \frac{\eta}{L}\psi'\left(\frac{1}{L}\sum_{j=i-L+1}^{i} \kappa_d(e(i,i) - e(i,j))\right) \\
&\quad \times \sum_{j=i-L+1}^{i} \kappa'_d(e(i,i) - e(i,j))\left(\frac{\partial e(i,i)}{\partial \boldsymbol{\Omega}_{i-1}} - \frac{\partial e(i,j)}{\partial \boldsymbol{\Omega}_{i-1}}\right) \\
&= \boldsymbol{\Omega}_{i-1} + \frac{\eta}{L}\psi'\left(\frac{1}{L}\sum_{j=i-L+1}^{i} \kappa_d(e(i,i) - e(i,j))\right) \\
&\quad \times \sum_{j=i-L+1}^{i} \kappa'_d(e(i,i) - e(i,j))(\varphi(i) - \varphi(j))
\end{aligned} \tag{21}$$

where $\psi'(.)$ and $\kappa'_d(.)$ denote, respectively, the first-order derivatives of functions $\psi(.)$ and $\kappa_d(.)$.

Remark 1. One can see that the algorithm (21) is actually a generalized stochastic information gradient (SIG) algorithm [21] in feature space. By selecting a certain ϕ function, we obtain a specific KMEE algorithm. For example, if setting $\phi(x) = -x\log x$, we get the KMEE under Shannon entropy criterion:

$$\boldsymbol{\Omega}_i = \boldsymbol{\Omega}_{i-1} - \eta \frac{\sum_{j=i-L+1}^{i} \kappa'_d(e(i,i) - e(i,j))(\varphi(i) - \varphi(j))}{\sum_{j=i-L+1}^{i} \kappa_d(e(i,i) - e(i,j))} \tag{22}$$

Further, if setting $\phi(x) = -x^\alpha (\alpha > 1)$, we obtain the KMEE under α-order information potential criterion ($\alpha > 1$)

$$\begin{aligned}
\boldsymbol{\Omega}_i = \boldsymbol{\Omega}_{i-1} &- \frac{\eta(\alpha-1)}{L}\left(\frac{1}{L}\sum_{j=i-L+1}^{i} \kappa_d(e(i,i) - e(i,j))\right)^{\alpha-2} \\
&\times \sum_{j=i-L+1}^{i} \kappa'_d(e(i,i) - e(i,j))(\varphi(i) - \varphi(j))
\end{aligned} \tag{23}$$

In particular, if $\alpha = 2$, we obtain the KMEE under *quadratic information potential* (QIP) [12] criterion:

$$\boldsymbol{\Omega}_i = \boldsymbol{\Omega}_{i-1} - \frac{\eta}{L}\sum_{j=i-L+1}^{i} \kappa'_d(e(i,i) - e(i,j))(\varphi(i) - \varphi(j)) \tag{24}$$

which is the simplest KMEE, since in this case we have $\psi'(x) \equiv -1$. In practical application, to reduce the computational complexity, we often use the QIP as the adaptation cost.

The weight update equation in (21) can be written in a compact form:

$$\boldsymbol{\Omega}_i = \boldsymbol{\Omega}_{i-1} + \eta \boldsymbol{\Phi}(i)\boldsymbol{h}_\phi(\boldsymbol{e}(i)) \tag{25}$$

where $\boldsymbol{h}_\phi(\boldsymbol{e}(i))$ is a vector-valued function of $\boldsymbol{e}(i)$, expressed as

$$\begin{aligned}
\boldsymbol{h}_\phi(\boldsymbol{e}(i)) = \frac{1}{L}\psi'\left(\frac{1}{L}\sum_{j=i-L+1}^{i} \kappa_d(e(i,i) - e(i,j))\right) \\
\times \begin{pmatrix} -\kappa'_d(e(i,i) - e(i,i-L+1)) \\ \vdots \\ -\kappa'_d(e(i,i) - e(i,i-1)) \\ \sum_{j=i-L+1}^{i-1} \kappa'_d(e(i,i) - e(i,j)) \end{pmatrix}
\end{aligned} \tag{26}$$

Comparing (25) and (8), we observe that the KMEE algorithm is very similar to the KAPA algorithm, except now the error vector $\boldsymbol{e}(i)$ is nonlinearly transformed by function $\boldsymbol{h}_\phi(.)$. In RKHS, the KMEE becomes

$$f_i = f_{i-1} + \eta \mathbf{K}(i)\boldsymbol{h}_\phi(\boldsymbol{e}(i)) \tag{27}$$

The learned mapping by KMEE, obviously, has the same structure as those learned by KLMS and KAPA, and can be represented as a linear combination of kernels centered in each data points, as expressed in (10). From (26) and (27), the updating on the coefficients of (10) can be readily derived as

$$\boldsymbol{a}_j(i) = \begin{cases} \frac{\eta}{L}\psi'\left(\frac{1}{L}\sum_{l=i-L+1}^{i} \kappa_d(e(i,i) - e(i,l))\right)\sum_{l=i-L+1}^{i-1} \kappa'_d(e(i,i) - e(i,l)), & j = i \\ \boldsymbol{a}_j(i-1) - \frac{\eta}{L}\psi'\left(\frac{1}{L}\sum_{l=i-L+1}^{i} \kappa_d(e(i,i) - e(i,l))\right)\kappa'_e(e(i,i) - e(i,j)), & i-L < j < i \\ \boldsymbol{a}_j(i-1), & 1 \leq j \leq i-L \end{cases} \tag{28}$$

The pseudocode for KMEE is summarized in Table 2.

Remark 2. As one can observe, the most time-consuming part of the computation in both KMEE and KAPA is to calculate the L errors. In [7], it has been shown that the computation complexity of this part is $O(i + L^2)$. The overall computation complexity of KMEE is still $O(i + L^2)$, since the complexity of the coefficients update Eq. (28) is only $O(L)$.

3.3. Quantized KMEE

The main bottleneck of all KAF algorithms is their growing structure with each sample, which results in increasing computational costs and memory requirements especially for continuous

Table 2
The kernel minimum error entropy (KMEE) algorithm.

Input: $\{\boldsymbol{u}(i) \in \mathbb{U}, d(i)\}, i = 1, 2, \ldots$
Initialization
a) Assigning the ϕ function and the kernel functions κ and κ_d;
b) Choose the step-size η, and the error samples length L;
c) Initialize the coefficient vector $\boldsymbol{a}(1) = [\eta d(1)]$.

Computation
while $\{\boldsymbol{u}(i), d(i)\}(i > 1)$ available **do**
(1) Allocate a new unit: $\boldsymbol{a}(i) = [\boldsymbol{a}(i-1)^T, 0]^T$
(2) For $\max\{1, i-L+1\} \leq j \leq i$, compute the errors:
$e(i,j) = d(j) - \sum_{l=1}^{i-1} \boldsymbol{a}_l(i-1)\kappa(\boldsymbol{u}(l), \boldsymbol{u}(j))$
(3) Update the coefficient vector $\boldsymbol{a}(i)$ using (28)

end while

adaptation scenarios. In order to curb the network growth and to obtain a compact representation, a variety of sparsification techniques have been proposed, where only the *important* input data are accepted as the new centers. Typical sparsification criteria include the novelty criterion [25], coherence criterion [26], approximate linear dependency (ALD) criterion [10], surprise criterion [27], and so on. Recently, we have proposed a novel quantization approach to constrain the network size (number of centers) and developed a quantized kernel least mean square (QKLMS) algorithm [28], which is shown to be very effective in yielding a compact network with desirable performance.

Recall that the QKLMS can be simply expressed as [28]

$$f_i = f_{i-1} + \eta e(i)\kappa(Q[\boldsymbol{u}(i)],.) \tag{29}$$

where $Q[.]$ denotes a vector quantization (VQ) operator in input space $\mathbb{U}$. Assume that the quantization codebook $\boldsymbol{C}$ contains M code vectors, i.e. $\boldsymbol{C}=\{\boldsymbol{c}(m)\in\mathbb{U}\}_{m=1}^{M}$. Then the vector quantizer $Q(\boldsymbol{u})$ is a function which maps the input $\boldsymbol{u}$ in $\mathbb{U}$ into one of the M code vectors in $\boldsymbol{C}$. The vector quantizer $Q(\boldsymbol{u})$ is specified by the values of the code vectors and by a partition of the input space $\mathbb{U}$ into M disjoint and exhaustive regions $S_1, S_2, \ldots, S_M$, where $S_m = Q^{-1}(\boldsymbol{c}(m))\subset\mathbb{U}$. We have $Q[\boldsymbol{u}(i)]=\boldsymbol{c}(m)$, if $\boldsymbol{u}(m)\in S_m$. By quantization approach, the network size of the learned mapping, obviously, can never be larger than the codebook size M. A simple online VQ method was also proposed in [28], where the codebook was trained directly and sequentially from the sample data (see Table 3 for the description in pseudocode).

One can immediately apply the same idea to KMEE, and obtain the quantized KMEE (QKMEE)

$$f_i = f_{i-1} + \eta Q[\mathbf{K}(i)]\boldsymbol{h}_\phi(\boldsymbol{e}(i)) \tag{30}$$

where $Q[\mathbf{K}(i)]$ is defined as

$$Q[\mathbf{K}(i)] = [\kappa(Q[\boldsymbol{u}(i-L+1)],.),\kappa(Q[\boldsymbol{u}(i-L+2)],.),\ldots,\kappa(Q[\boldsymbol{u}(i)],.)] \tag{31}$$

If adopting the online VQ method described in Table 3, the learned mapping will be

$$f_i(.) = \sum_{j=1}^{|\boldsymbol{C}(i)|}\boldsymbol{\alpha}_j(i)\kappa(\boldsymbol{C}_j(i),.) \tag{32}$$

where the coefficients are updated as follows (we assume $\boldsymbol{\alpha}_j(i-1)=0$ if $j>|\boldsymbol{C}(i-1)|$)

$$\boldsymbol{\alpha}_j(i) = \boldsymbol{\alpha}_j(i-1) + \sum_{l\in\mathbf{I}_j(i)}\beta_l(i), \qquad 1\le j\le|\boldsymbol{C}(i)| \tag{33}$$

Table 3
Online VQ method.

Input: $\{\boldsymbol{u}(i)\in\mathbb{U}\}, i=1,2,\ldots$
Initialization:
Select the quantization size $\varepsilon>0$, and initialize the codebook $\boldsymbol{C}(1)=\{\boldsymbol{u}(1)\}$
Computation:
while $\{\boldsymbol{u}(i)\}(i>1)$ available **do**
(1) Compute the distance between $\boldsymbol{u}(i)$ and $\boldsymbol{C}(i-1)$:
$dis(\boldsymbol{u}(i),\boldsymbol{C}(i-1)) = ||\boldsymbol{u}(i)-\boldsymbol{C}_{j^*}(i-1)||$, where $j^* = \underset{1\le j\le|\boldsymbol{C}(i-1)|}{\operatorname{argmin}}||\boldsymbol{u}(i)-\boldsymbol{C}_j(i-1)||$.
(2) If $dis(\boldsymbol{u}(i),\boldsymbol{C}(i-1))\le\varepsilon$, keep the codebook unchanged: $\boldsymbol{C}(i)=\boldsymbol{C}(i-1)$, and quantize $\boldsymbol{u}(i)$ to the closest code vector: $Q[\boldsymbol{u}(i)]=\boldsymbol{C}_{j^*}(i-1)$.
(3) Otherwise, update the codebook: $\boldsymbol{C}(i)=\{\boldsymbol{C}(i-1),\boldsymbol{u}(i)\}$, and quantize $\boldsymbol{u}(i)$ as itself: $Q[\boldsymbol{u}(i)]=\boldsymbol{u}(i)$.
end while
(where $||.||$ denotes the Euclidean norm, and $\boldsymbol{C}_j(i-1)$ denotes the jth element of the codebook $\boldsymbol{C}(i-1)$)

where $\mathbf{I}_j(i)=\{l\,|\,i-L+1\le l\le i, Q[\boldsymbol{u}(l)]=\boldsymbol{C}_j(i)\}$, and

$$\beta_l(i)=\begin{cases}\frac{\eta}{L}\psi'\left(\frac{1}{L}\sum_{j=i-L+1}^{i}\kappa_d(e(i,i)-e(i,j))\right)\sum_{j=i-L+1}^{i-1}\kappa'_d(e(i,i)-e(i,j)), & l=i\\ -\frac{\eta}{L}\psi'\left(\frac{1}{L}\sum_{j=i-L+1}^{i}\kappa_d(e(i,i)-e(i,j))\right)\kappa'_e(e(i,i)-e(i,l)), & i-L<l<i\end{cases} \tag{34}$$

4. Mean square convergence analysis

In this section, we study the mean square convergence of the KMEE algorithm (25). The unknown system is assumed to be a nonlinear regression model as in (1). First, we derive an important *energy conservation relation* in feature space.

4.1. Energy conservation relation

Let's define the *a priori* error vector $\boldsymbol{e}_a(i)$ and *a posteriori* error vector $\boldsymbol{e}_p(i)$ as follows:

$$\begin{cases}\boldsymbol{e}_a(i)=\boldsymbol{\Phi}(i)^T\tilde{\boldsymbol{\Omega}}_{i-1}\\ \boldsymbol{e}_p(i)=\boldsymbol{\Phi}(i)^T\tilde{\boldsymbol{\Omega}}_i\end{cases} \tag{35}$$

where $\tilde{\boldsymbol{\Omega}}_{i-1}$ and $\tilde{\boldsymbol{\Omega}}_i$ are, respectively, the *weight error vectors* in feature space at iteration $i-1$ and i, that is, $\tilde{\boldsymbol{\Omega}}_{i-1}=\boldsymbol{\Omega}^*-\boldsymbol{\Omega}_{i-1}$, $\tilde{\boldsymbol{\Omega}}_i=\boldsymbol{\Omega}^*-\boldsymbol{\Omega}_i$, where $\boldsymbol{\Omega}^*$ stands for the weight vector in feature space corresponding to the desired mapping f^* (i.e. $\boldsymbol{\Omega}^{*T}\boldsymbol{\varphi}(.)=f^*(.)$). Note that the *a priori* error vector $\boldsymbol{e}_a(i)$ and the error vector $\boldsymbol{e}(i)$ are related via

$$\begin{aligned}\boldsymbol{e}(i)&=\boldsymbol{d}(i)-\boldsymbol{\Phi}(i)^T\boldsymbol{\Omega}_{i-1}\\&=(\boldsymbol{\Phi}(i)^T\boldsymbol{\Omega}^*+\boldsymbol{v}(i))-\boldsymbol{\Phi}(i)^T\boldsymbol{\Omega}_{i-1}\\&=\boldsymbol{e}_a(i)+\boldsymbol{v}(i)\end{aligned} \tag{36}$$

where $\boldsymbol{v}(i)=[v(i-L+1),\ldots,v(i)]^T$ is the noise vector. In addition, $\boldsymbol{e}_a(i)$ and $\boldsymbol{e}_p(i)$ have the following relationship:

$$\boldsymbol{e}_p(i)=\boldsymbol{e}_a(i)+\boldsymbol{\Phi}(i)^T(\tilde{\boldsymbol{\Omega}}_i-\tilde{\boldsymbol{\Omega}}_{i-1})=\boldsymbol{e}_a(i)-\boldsymbol{\Phi}(i)^T(\boldsymbol{\Omega}_i-\boldsymbol{\Omega}_{i-1}) \tag{37}$$

Combining (37) and (25), we have

$$\boldsymbol{e}_p(i)=\boldsymbol{e}_a(i)-\eta\boldsymbol{\Phi}(i)^T\boldsymbol{\Phi}(i)\boldsymbol{h}_\phi(\boldsymbol{e}(i))=\boldsymbol{e}_a(i)-\eta\boldsymbol{G}(i)\boldsymbol{h}_\phi(\boldsymbol{e}(i)) \tag{38}$$

where $\boldsymbol{G}(i)=\boldsymbol{\Phi}(i)^T\boldsymbol{\Phi}(i)$ is an $L\times L$ Gram matrix with elements $\boldsymbol{G}_{jk}(i)=\kappa(\boldsymbol{u}(i-L+j),\boldsymbol{u}(i-L+k))$. Then we can derive

$$\begin{aligned}&\boldsymbol{e}_p(i)=\boldsymbol{e}_a(i)-\eta\boldsymbol{G}(i)\boldsymbol{h}_\phi(\boldsymbol{e}(i))\\&\Rightarrow\boldsymbol{G}(i)^{-1}(\boldsymbol{e}_p(i)-\boldsymbol{e}_a(i))=-\eta\boldsymbol{h}_\phi(\boldsymbol{e}(i))\\&\Rightarrow\boldsymbol{\Phi}(i)\boldsymbol{G}(i)^{-1}(\boldsymbol{e}_p(i)-\boldsymbol{e}_a(i))=-\eta\boldsymbol{\Phi}(i)\boldsymbol{h}_\phi(\boldsymbol{e}(i))\\&\Rightarrow\boldsymbol{\Phi}(i)\boldsymbol{G}(i)^{-1}(\boldsymbol{e}_p(i)-\boldsymbol{e}_a(i))=\tilde{\boldsymbol{\Omega}}_i-\tilde{\boldsymbol{\Omega}}_{i-1}\end{aligned} \tag{39}$$

It follows that

$$\tilde{\boldsymbol{\Omega}}_i=\tilde{\boldsymbol{\Omega}}_{i-1}+\boldsymbol{\Phi}(i)\boldsymbol{G}(i)^{-1}(\boldsymbol{e}_p(i)-\boldsymbol{e}_a(i)) \tag{40}$$

Squaring both sides of (40), we have

$$\begin{aligned}||\tilde{\boldsymbol{\Omega}}_i||^2_{\mathbb{F}_\kappa}&=(\tilde{\boldsymbol{\Omega}}_{i-1}+\boldsymbol{\Phi}(i)\boldsymbol{G}(i)^{-1}(\boldsymbol{e}_p(i)-\boldsymbol{e}_a(i)))^T\\&\quad\times(\tilde{\boldsymbol{\Omega}}_{i-1}+\boldsymbol{\Phi}(i)\boldsymbol{G}(i)^{-1}(\boldsymbol{e}_p(i)-\boldsymbol{e}_a(i)))\\&=||\tilde{\boldsymbol{\Omega}}_{i-1}||^2_{\mathbb{F}_\kappa}+(\boldsymbol{e}_p(i)-\boldsymbol{e}_a(i))^T\boldsymbol{G}(i)^{-1}(\boldsymbol{e}_p(i)-\boldsymbol{e}_a(i))\\&\quad+2\tilde{\boldsymbol{\Omega}}_{i-1}^T\boldsymbol{\Phi}(i)\boldsymbol{G}(i)^{-1}(\boldsymbol{e}_p(i)-\boldsymbol{e}_a(i))\\&=||\tilde{\boldsymbol{\Omega}}_{i-1}||^2_{\mathbb{F}_\kappa}+||\boldsymbol{e}_p(i)||^2_{\boldsymbol{G}(i)^{-1}}-||\boldsymbol{e}_a(i)||^2_{\boldsymbol{G}(i)^{-1}}\end{aligned} \tag{41}$$

where $||\tilde{\boldsymbol{\Omega}}_i||^2_{\mathbb{F}_\kappa}=\tilde{\boldsymbol{\Omega}}_i^T\tilde{\boldsymbol{\Omega}}_i$, $||\boldsymbol{e}_a(i)||^2_{\boldsymbol{G}(i)^{-1}}=\boldsymbol{e}_a(i)^T\boldsymbol{G}(i)^{-1}\boldsymbol{e}_a(i)$, and $||\boldsymbol{e}_p(i)||^2_{\boldsymbol{G}(i)^{-1}}=\boldsymbol{e}_p(i)^T\boldsymbol{G}(i)^{-1}\boldsymbol{e}_p(i)$. Taking the expectations, we obtain

$$E[\|\tilde{\boldsymbol{\Omega}}_i\|^2_{\mathbb{F}_\kappa}]+E[||\boldsymbol{e}_a(i)||^2_{\boldsymbol{G}(i)^{-1}}]=E[||\tilde{\boldsymbol{\Omega}}_{i-1}||^2_{\mathbb{F}_\kappa}]+E[||\boldsymbol{e}_p(i)||^2_{\boldsymbol{G}(i)^{-1}}] \tag{42}$$

where $E[||\tilde{\mathbf{\Omega}}_i||^2_{\mathbb{F}_\kappa}]$ is called the weight error power (WEP) in feature space at iteration i.

Remark 3. Eq. (42) is referred in this paper as the energy conservation relation for KMEE, which shows an interesting identity relationship between different error powers. Similar relations were found for classical linear adaptive filters [22] and KLMS algorithms [29].

4.2. Sufficient condition for mean square convergence

Based on the energy conservation relation (42), one can easily obtain a sufficient condition that ensures the mean square convergence of KMEE. Substituting $\boldsymbol{e}_p(i)=\boldsymbol{e}_a(i)-\eta \boldsymbol{G}(i)\boldsymbol{h}_\phi(\boldsymbol{e}(i))$ into (42), we have

$$E[||\tilde{\mathbf{\Omega}}_i||^2_{\mathbb{F}_\kappa}]=E[||\tilde{\mathbf{\Omega}}_{i-1}||^2_{\mathbb{F}_\kappa}]-2\eta E[\boldsymbol{e}_a(i)^T\boldsymbol{h}_\phi(\boldsymbol{e}(i))] +\eta^2 E[\boldsymbol{h}_\phi(\boldsymbol{e}(i))^T\boldsymbol{G}(i)\boldsymbol{h}_\phi(\boldsymbol{e}(i))] \tag{43}$$

It follows that

$$\begin{aligned} &E[||\tilde{\mathbf{\Omega}}_i||^2_{\mathbb{F}_\kappa}]\le E[||\tilde{\mathbf{\Omega}}_{i-1}||^2_{\mathbb{F}_\kappa}] \\ &\Leftrightarrow -2\eta E[\boldsymbol{e}_a(i)^T\boldsymbol{h}_\phi(\boldsymbol{e}(i))]+\eta^2 E[\boldsymbol{h}_\phi(\boldsymbol{e}(i))^T\boldsymbol{G}(i)\boldsymbol{h}_\phi(\boldsymbol{e}(i))]\le 0 \\ &\Leftrightarrow \eta \le \frac{2E[\boldsymbol{e}_a(i)^T\boldsymbol{h}_\phi(\boldsymbol{e}(i))]}{E[\boldsymbol{h}_\phi(\boldsymbol{e}(i))^T\boldsymbol{G}(i)\boldsymbol{h}_\phi(\boldsymbol{e}(i))]} \end{aligned} \tag{44}$$

Thus, if $\forall i$, the step-size η satisfies the inequality

$$0<\eta\le\frac{2E[\boldsymbol{e}_a(i)^T\boldsymbol{h}_\phi(\boldsymbol{e}(i))]}{E[\boldsymbol{h}_\phi(\boldsymbol{e}(i))^T\boldsymbol{G}(i)\boldsymbol{h}_\phi(\boldsymbol{e}(i))]} \tag{45}$$

the weight error power in feature space will monotonically decrease (and hence converge). The above inequality also implies $E[\boldsymbol{e}_a(i)^T\boldsymbol{h}_\phi(\boldsymbol{e}(i))]>0$, $\forall i$. Therefore, a sufficient condition for the mean square convergence of KMEE will be $\forall i$

$$\begin{cases} E[\boldsymbol{e}_a(i)^T\boldsymbol{h}_\phi(\boldsymbol{e}(i))]>0 \\ 0<\eta\le\frac{2E[\boldsymbol{e}_a(i)^T\boldsymbol{h}_\phi(\boldsymbol{e}(i))]}{E[\boldsymbol{h}_\phi(\boldsymbol{e}(i))^T\boldsymbol{G}(i)\boldsymbol{h}_\phi(\boldsymbol{e}(i))]} \end{cases} \tag{46}$$

Remark 4. The above sufficient condition for the mean square convergence is only of theoretical importance, and it is difficult to check it in practice. In traditional linear adaptive filters [22], the mean-square convergence behavior can be rigorously analyzed in a satisfactory way if given the noise distribution and assuming that the *a priori* error is Gaussian distributed and independent of the input. In kernel adaptive filters, however, the *a priori* error cannot be assumed to be Gaussian, since the central limit theorem (CLT) does not hold in general with a nonlinear model.

5. Simulation results

We present simulation results to demonstrate the performance of KMEE. For comparison purpose, we also show the performance of several other KAF algorithms, such as KLMS, KMC and KAPA. In the simulations below, all the kernels (kernel of RKHS, kernel of correntropy, and kernel for density estimation) are chosen to be Gaussian kernels. Except mentioned otherwise, the QIP is selected as the adaptation cost in KMEE (or QKMEE) due to its simplicity.

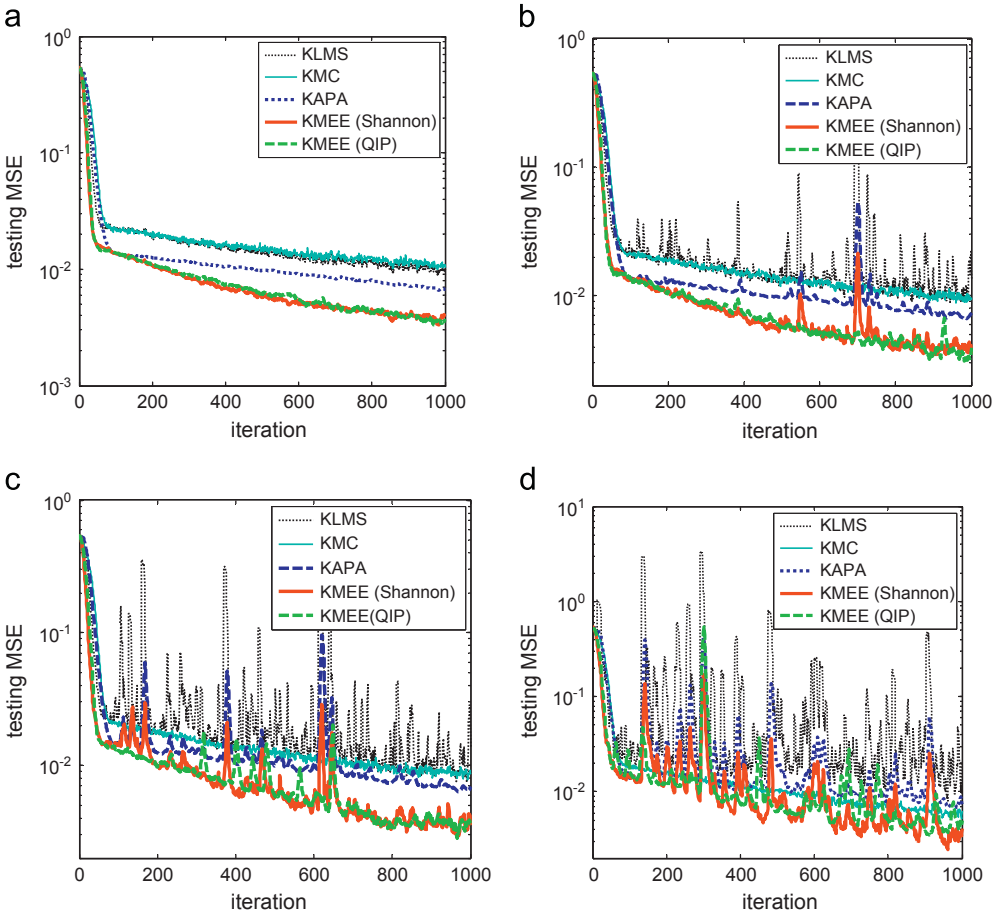


Fig. 1. Ensemble learning curves for different α values in nonlinear system identification: (a)$\alpha=2.0$, (b) $\alpha=1.9$, (c) $\alpha=1.8$, and (d) $\alpha=1.5$.

Table 4
Testing MSE at final iteration for different α values in nonlinear system identification.

Algorithms	Testing MSE			
	$\alpha = 2.0$	$\alpha = 1.9$	$\alpha = 1.8$	$\alpha = 1.5$
KLMS	0.0095 ± 0.0079	0.0134 ± 0.0647	0.0136 ± 0.1218	0.0203 ± 0.3829
KMC	0.0103 ± 0.0082	0.0096 ± 0.0089	0.0088 ± 0.0086	0.0063 ± 0.0064
KAPA	0.0067 ± 0.0015	0.0069 ± 0.0055	0.0073 ± 0.0078	0.0072 ± 0.0205
KMEE (Shannon)	$\mathbf{0.0040 \pm 0.0027}$	$\mathbf{0.0035 \pm 0.0028}$	$\mathbf{0.0035 \pm 0.0051}$	$\mathbf{0.0041 \pm 0.0180}$
KMEE (QIP)	$\mathbf{0.0035 \pm 0.0020}$	$\mathbf{0.0034 \pm 0.0022}$	$\mathbf{0.0036 \pm 0.0046}$	$\mathbf{0.0048 \pm 0.0138}$

Table 5
Performance comparison of KMEE with different step sizes and KRLS with different regularization parameters in nonlinear system identification ($\alpha = 1.8$).

Algorithms	Testing MSE		Average running time (s)
KMEE	$\eta = 0.5$	0.0079 ± 0.0022	0.0664
	$\eta = 1.0$	0.0053 ± 0.0032	
	$\eta = 2.0$	0.0036 ± 0.0046	
	$\eta = 3.0$	0.0047 ± 0.0049	
	$\eta = 4.0$	0.0076 ± 0.0091	
KRLS	$\lambda = 0.0$	N/A	2.2753
	$\lambda = 0.1$	0.0011 ± 0.0019	
	$\lambda = 1.0$	0.0034 ± 0.0008	
	$\lambda = 10$	0.0138 ± 0.0006	

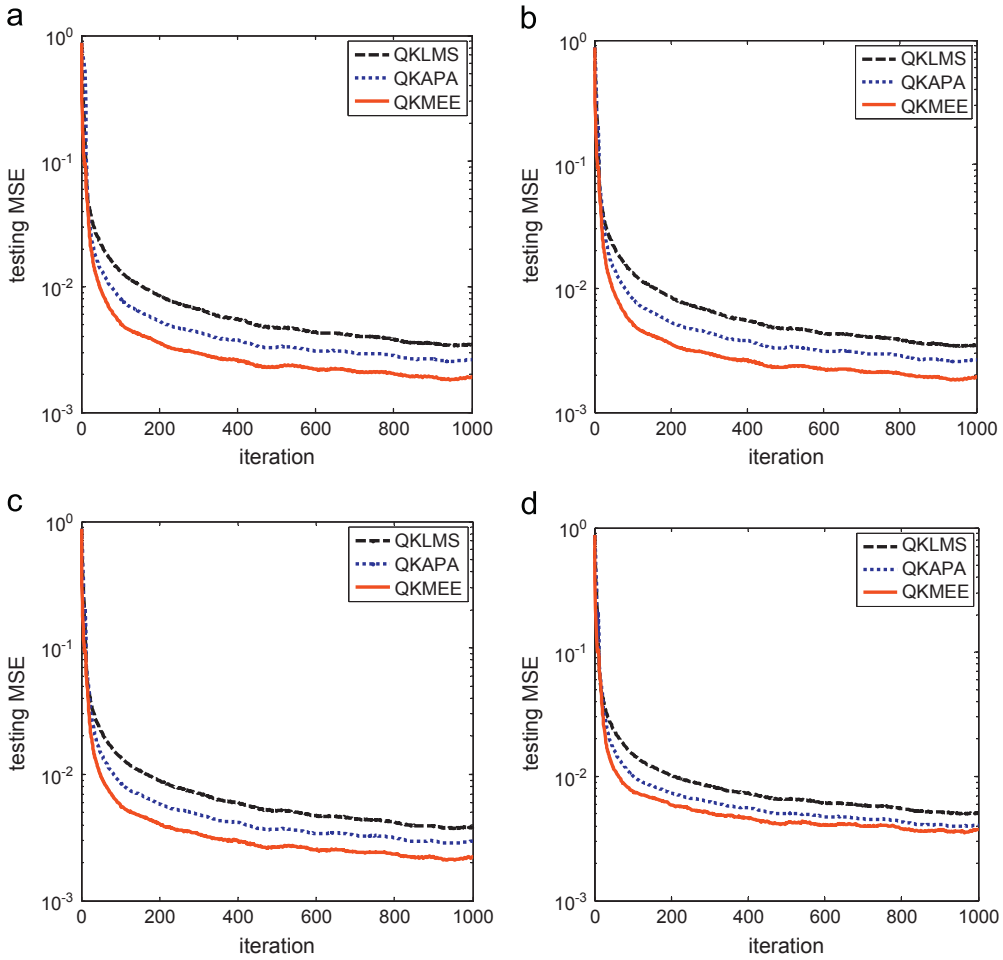


Fig. 2. Ensemble learning curves for different ε values in MG time series prediction: (a) $\varepsilon = 0.0$, (b) $\varepsilon = 0.1$, (c) $\varepsilon = 0.3$ and (d) $\varepsilon = 0.5$.

5.1. Nonlinear system identification

The first example is on the nonlinear system identification, where the nonlinear system is of the form [30]

$$y(i) = (0.8 - 0.5\exp(-y(i-1)^2))y(i-1) - (0.3 + 0.9\exp(-y(i-1)^2))y(i-2) + 0.1\sin(3.1415926y(i-1)) + v(i) \quad (47)$$

The noise $v(i)$ is assumed to be of symmetric α-stable ($S\alpha S$) distribution with characteristic function

$$\psi(\omega) = \exp(-\gamma|\omega|^{\alpha}) \quad (48)$$

where $\gamma = 0.005$, $0 < \alpha \leq 2.0$. For $\alpha = 2.0$, the distribution reduces to a zero-mean Gaussian distribution with variance 0.01; while for $\alpha < 2.0$, the distribution corresponds to an impulsive noise with infinite variance.

First, we compare the performance of five algorithms: KLMS, KMC, KAPA, KMEE (Shannon), and KMEE (QIP). Fig. 1 illustrates the ensemble learning curves (over 200 Monte Carlo runs) for different α values, and Table 4 lists the testing MSE at final iteration. In the simulation, 1000 samples are used for training and another 100 clean samples are used for testing. The filters are fixed during the testing phase. In addition, since the MEE criteria (Shannon, QIP) are shift-invariant, the testing MSEs for KMEE (Shannon, QIP) are calculated by adding a bias value to the testing errors. This bias value was adjusted so as to yield zero-mean error over the training set. The kernel parameter for RKHS is set at 0.2, the kernel size for correntropy is 0.4, and the kernel width for density estimation is 1.0. The error lengths for KMEE and KAPA are set at $L = 10$. The step-sizes for KLMS, KMC, KAPA, KMEE (Shannon), and KMEE (QIP), are, respectively, set at 0.8, 1.0, 0.05, 1.0, and 2.0. These parameters are experimentally selected to achieve a desirable tradeoff between convergence speed and final accuracy. From Fig. 1 and Table 4, we can see that the KMEE algorithms outperform all other algorithms except for the case of small α, the KMC algorithm may achieve a relatively smaller deviation in testing MSE. Notice that the MC criterion is especially robust to the impulsive noises, however, in case of non-impulsive noises (i.e. Gaussian noise), its performance may be not so good. Simulation results also show that the performances of KMEE (Shannon) and KMEE (QIP) are very close.

Next, we compare the performance of KMEE with different step sizes and KRLS with different regularization factors. The results of the testing MSE at final iteration and the average execution time for each simulation run (1000 iterations) are summarized in Table 5. The running time is measured on a PC with a 2.2 GHz processor and 3 GB memory. As one can see, by using a suitable regularization parameter, the KRLS may achieve a better (but not significant) performance. However, it requires significantly much more time to finish a simulation, which is to be expected, as its computational complexity is $O(i^2)$ at iteration i. In Table 5, when the regularization parameter is zero, KRLS performs rather poorly (often diverges to very large values) on the test data. From Table 5 one can also see how the step size affects the learning performance of the KMEE: when the step size is too small or too large, the final testing MSE will become large due to the slow convergence speed or large misadjustment. In general, the role of the step size in KMEE remains in principle the same as the step size in other kernel adaptive filtering algorithms, which controls the compromise between convergence speed and misadjustment.

5.2. Chaotic time series prediction

The second example is on Mackey–Glass (MG) chaotic time series prediction. The time series is generated from the differential equation with the same parameter settings as in [27]. The goal is to predict the current value using the previous seven points.

In this example, we show the performance of QKMEE and compare it with that of QKLMS and QKAPA (quantized KAPA). The kernel parameter for RKHS is set at 0.5, and the kernel width for density estimation is 1.0. The step-sizes for QKLMS, QKAPA, and QKMEE are set at 0.2, 0.05, and 2.0, respectively. Fig. 2 shows the ensemble learning curves for different quantization sizes (ε values), and Table 6 gives the results of testing MSE at final iteration and average execution time for each simulation run. All learning curves are averaged over 200 independent simulations, and each simulation is run with different segments of the signal. As expected, the QKMEE achieves the best performance with smallest testing MSE for all ε values. The time consumption of

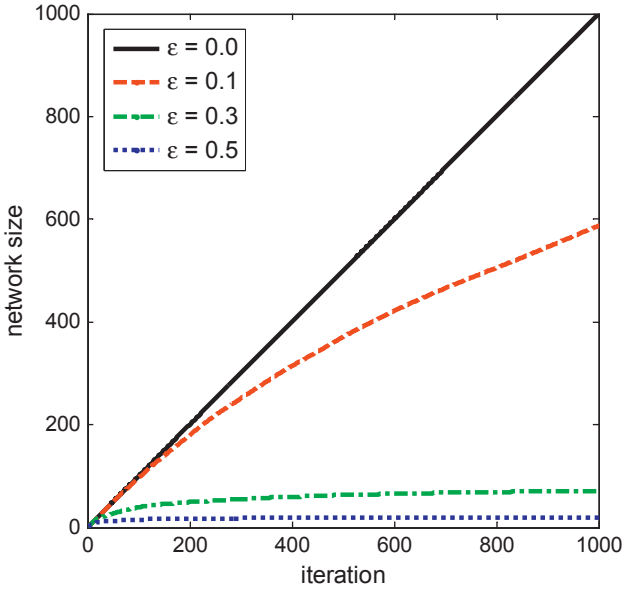


Fig. 3. Network growth curves for different ε values in MG time series prediction.

Table 6
Performance comparison of QKLMS, QKAPA and QKMEE with different quantization sizes in MG time series prediction.

Algorithms	Testing MSE		Average running time (s)
QKLMS	$\varepsilon = 0.0$	0.0035 ± 0.0008	0.0162
	$\varepsilon = 0.1$	0.0035 ± 0.0008	0.0105
	$\varepsilon = 0.3$	0.0039 ± 0.0009	0.0043
	$\varepsilon = 0.5$	0.0051 ± 0.0014	0.0032
QKAPA	$\varepsilon = 0.0$	0.0026 ± 0.0007	0.0583
	$\varepsilon = 0.1$	0.0027 ± 0.0007	0.0411
	$\varepsilon = 0.3$	0.0029 ± 0.0008	0.0185
	$\varepsilon = 0.5$	0.0041 ± 0.0014	0.0152
QKMEE	$\varepsilon = 0.0$	0.0019 ± 0.0005	0.0662
	$\varepsilon = 0.1$	0.0019 ± 0.0005	0.0490
	$\varepsilon = 0.3$	0.0022 ± 0.0007	0.0260
	$\varepsilon = 0.5$	0.0038 ± 0.0013	0.0212

Table 7
Network sizes at final iteration for different ε values in MG time series prediction.

Quantization size	Network size
$\varepsilon = 0.0$	1000 ± 0
$\varepsilon = 0.1$	587 ± 21
$\varepsilon = 0.3$	72 ± 4
$\varepsilon = 0.5$	20 ± 2

QKMEE is, obviously, higher than that of QKLMS, and is moderately higher than, but still comparable to, that of QKAPA, which confirms the fact that the computational complexity of KMEE is similar to the KAPA algorithm.

From the simulation results one can also observe that, when quantization size is small (say smaller than 0.3), there is little loss in the learning performance (i.e. testing MSE) of all algorithms. Even with small quantization size, however, the network size will decrease significantly. This is confirmed by Fig. 3, which shows the network growth curves for different ε values. The network sizes at final iteration for different quantization sizes are presented in Table 7.

6. Conclusion

Existing approaches to kernel adaptive filtering in general aim to optimize the second-order statistics (least squares, mean square error, etc.). However, in most situations, a more appropriate approach would be to capture higher order statistics or information content of signals rather than simply their energy. Recent studies indicate that the error entropy as an adaptation cost can improve the performance in many realistic scenarios, since it provides a natural framework where information content of available data can be assessed. In this work, we apply the minimum error entropy (MEE) criterion to kernel adaptive filtering, and derive the kernel minimum error entropy (KMEE) algorithm and its quantized version. The developed algorithm is in essence a generalized stochastic information gradient (SIG) algorithm in kernel space, whose computational complexity is just similar to the KAPA algorithm. The mean square convergence is also analyzed. Based on the energy conservation relation, we obtain a sufficient condition that ensures the mean square convergence of KMEE. Simulation results on synthetic data demonstrate the superior performance of the proposed algorithm. The application to real data is interesting, and is left for future research.

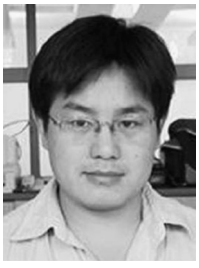
Acknowledgments

This work was supported by NSF Grant ECCS 0856441, National Natural Science Foundation of China (No. 60904054), and supported by 973 Program (No. 2012CB316400).

References

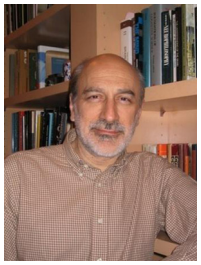
[1] V. Vapnik, The Nature of Statistical Learning Theory, Springer, New York, 1995.
[2] B. Scholkopf, A.J. Smola, Learning with Kernels, Support Vector Machines, Regularization, Optimization and Beyond, MIT Press, Cambridge, MA, USA, 2002.
[3] F. Girosi, M. Jones, T. Poggio, Regularization theory and neural networks architectures, Neural Comput. 7 (1995) 219–269.
[4] B. Scholkopf, A.J. Smola, K. Muller, Nonlinear component analysis as a kernel eigenvalue problem, Neural Comput. 10 (1998) 1299–1319.
[5] J. Kivinen, A.J. Smola, R.C. Williamson, Online learning with kernels, IEEE Trans. Signal Process. 52 (8) (Aug. 2004) 2165–2176.
[6] F. Orabona, J. Keshet, B. Caputo, Bounded kernel-based online learning, J. Mach. Learn. Res. 10 (2009) 2643–2666.
[7] W. Liu, J. Principe, S. Haykin, Kernel Adaptive Filtering: A Comprehensive Introduction, Wiley, 2010.
[8] W. Liu, P. Pokharel, J. Principe, The kernel least mean square algorithm, IEEE Trans. Signal Process. 56 (2008) 543–554.
[9] W. Liu, J. Principe, Kernel affine projection algorithm, EURASIP J. Adv. Signal Process. (2008), http://dx.doi.org/10.1155/2008/784292, Article ID 784292, 12 pages.
[10] Y. Engel, S. Mannor, R. Meir, The kernel recursive least-squares algorithm, IEEE Trans. Signal Process. 52 (2004) 2275–2285.
[11] W. Liu, Il Park, Y. Wang, J.C. Principe, Extended extended kernel recursive least squares algorithm, IEEE Trans. Signal Process. 57 (2009) 3801–3814.
[12] J.C. Principe, Information Theoretic Learning: Renyi's Entropy and Kernel Perspectives, Springer, New York, 2010.
[13] D. Erdogmus, J.C. Principe, An error-entropy minimization algorithm for supervised training of nonlinear adaptive systems, IEEE Trans. Signal Process. 50 (7) (2002) 1780–1786.
[14] D. Erdogmus, J.C. Principe, From linear adaptive filtering to nonlinear information processing—the design and analysis of information processing systems, IEEE Signal Process. Mag. 23 (6) (2006) 14–33.
[15] D. Erdogmus, J.C. Principe, Generalized information potential criterion for adaptive system training, IEEE Trans. on Neural Networks 13 (2002) 1035–1044.
[16] I Santamaria, D Erdogmus, J.C. Principe, Entropy minimization for supervised digital communications channel equalization, IEEE Trans. Signal Process. 50 (5) (2002) 1184–1192.
[17] B. Chen, J. Hu, L. Pu, Z. Sun, Stochastic gradient algorithm under (h, ϕ)-entropy criterion, Circ. Syst. Signal Process. 26 (2007) 941–960.
[18] B.W. Silverman, Density Estimation for Statistic and Data Analysis, Chapman & Hall, NY, 1986.
[19] S. Zhao, B. Chen, J. C. Principe, Kernel adaptive filtering with maximum correntropy criterion, in: Proceedings of the International Joint Conference on Neural Networks (IJCNN), 2012–2017, 2011.
[20] W. Liu, P.P. Pokharel, J.C. Principe, , Correntropy: properties and applications in non-Gaussian signal processing, IEEE Trans. Signal Process. 55 (11) (2007) 5286–5298.
[21] D. Erdogmus, E.H. Kenneth, J.C. Principe, Online entropy manipulation: stochastic information gradient, IEEE Signal Process. Lett. 10 (2003) 242–245.
[22] A. Sayed, Fundamentals of Adaptive Filtering, Wiley, New York, 2003.
[23] A. Renyi (Ed.), Selected Papers of Alfred Renyi, 2, Akademia Kiado, Budapest, 1976.
[24] J. Burbea, C. Rao, Entropy differential metric, distance and divergence measures in probability spaces: a unified approach, J. Multivar. Anal. 12 (1982) 575–596.
[25] J. Platt, A resource-allocating network for function interpolation, Neural Comput. 3 (1991) 213–225.
[26] C. Richard, J.C.M. Bermudez, P. Honeine, Online prediction of time series data with kernels, IEEE Trans. Signal Process. 57 (2009) 1058–1067.
[27] W. Liu, Il Park, J.C. Principe, An information theoretic approach of designing sparse kernel adaptive filters, IEEE Trans. Neural Networks 20 (2009) 1950–1961.
[28] B. Chen, S. Zhao, P. Zhu, J.C. Principe, Quantized kernel least mean square algorithm, IEEE Trans. Neural Networks Learn. Syst. 23 (1) (2012) 22–32.
[29] B. Chen, S. Zhao, P. Zhu, J.C. Principe, Mean square convergence analysis of the kernel least mean square algorithm, Signal Process. vol. 92 (2012) 2624–2632.
[30] S. Chen, S.A. Billings, P.M. Grant, Recursive hybrid algorithm for non-linear system identification using radial basis function networks, Int. J. Control 55 (1992) 1051–1070.

Badong Chen received his PhD degree in Computer Science and Technology from Tsinghua University, Beijing, China, in 2008. He was a postdoctoral researcher at Tsinghua University from 2008 to 2010, and a Postdoctoral Associate at the University of Florida Computational Neuro-Engineering Laboratory (CNEL) from 2010 to 2012. He is currently a professor at the Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University, Xi'an, China. His research interests are in signal processing and machine learning, and their applications in cognition and neuroscience.



Zejian Yuan received the MS degree in electronic engineering from Xi'an University of Technology in 1999, and the PhD degree in pattern recognition and intelligent system from Xi'an Jiaotong University, China, in 2003. He was a visiting scholar in the Advanced Robotics Lab of Chinese University of Hong Kong during 2008–2009. He is currently an associate professor in the Department of Automatic Engineering, Xi'an Jiaotong University, and a member of the Chinese Association of Robotics. His research interests include image processing, pattern recognition, as well as machine learning methods in computer vision.

Nanning Zheng graduated from the Department of Electrical Engineering, Xi'an Jiaotong University, Xi'an, China, in 1975, and received the MS degree in information and control engineering from Xi'an Jiaotong University in 1981 and the PhD degree in electrical engineering from Keio University, Yokohama, Japan, in 1985. He is currently a professor and director of the Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University. His research interests include computer vision, pattern recognition and image processing, and hardware implementation of intelligent systems. He became a member of the Chinese Academy of Engineering in 1999, and he is the Chinese Representative on the Governing Board of the International Association for Pattern Recognition. He also serves as an executive deputy editor of the Chinese Science Bulletin.



José C. Príncipe is currently the Distinguished Professor of Electrical and Biomedical Engineering at the University of Florida, Gainesville, where he teaches advanced signal processing and artificial neural networks (ANNs) modeling. He is BellSouth Professor and Founder and Director of the University of Florida Computational Neuro-Engineering Laboratory (CNEL). He is involved in biomedical signal processing, in particular, the electroencephalogram (EEG) and the modeling and applications of adaptive systems. He is the past Editor-in-Chief of the IEEE Transactions on Biomedical Engineering, past President of the International Neural Network Society, and former Secretary of the Technical Committee on Neural Networks of the IEEE Signal Processing Society. He is an IEEE Fellow and an AIMBE Fellow and a recipient of the IEEE Engineering in Medicine and Biology Society Career Service Award. He is also a former member of the Scientific Board of the Food and Drug Administration, and a member of the Advisory Board of the McKnight Brain Institute at the University of Florida.