Zhao *et al. EURASIP Journal on Advances in Signal Processing* (2016) 2016:106
DOI 10.1186/s13634-016-0406-3

EURASIP Journal on Advances in Signal Processing

RESEARCH Open Access

Self-organizing kernel adaptive filtering

Songlin Zhao[1], Badong Chen[2*], Zheng Cao[1], Pingping Zhu[1] and Jose C. Principe[1]

Abstract

This paper presents a model-selection strategy based on minimum description length (MDL) that keeps the kernel least-mean-square (KLMS) model tuned to the complexity of the input data. The proposed KLMS-MDL filter adapts its model order as well as its coefficients online, behaving as a self-organizing system and achieving a good compromise between system accuracy and computational complexity without a priori knowledge. Particularly, in a nonstationary scenario, the model order of the proposed algorithm changes continuously with the input data structure. Experiments show the proposed algorithm successfully builds compact kernel adaptive filters with better accuracy than KLMS with sparsity or fixed-budget algorithms.

Keywords: Kernel method, Model selection, Sparsification, Minimal description length

1 Introduction

Owing to their universal modeling capability and convex cost, kernel adaptive filters (KAFs) are attracting renewed attention. Even though these methods achieve powerful classification and regression performance, the model order (system structure) and computational complexity grows linearly with the number of processed data, for example, the model order of kernel least-mean-square (KLMS) [1] and kernel recursive-least-square (KRLS) [2] scales as $O(n)$ with respect to the number of samples, where n is the number of processed data. The computational complexity are $O(n)$ and $O(n^2)$ at each iteration, respectively. This characteristic hinders widespread use of KAFs unless the filter growth is constrained. To curb their growth to sublinear rates, not all samples are included in the dictionary and a number of different criteria for online sparsification techniques are adopted: the novelty criterion [3, 4], approximate linear dependency (ALD) criterion [2, 5], coherence [6], the surprise criterion [7], and quantization [8, 9]. Alternatively, pruning criteria discard redundant centers from the existing large dictionary [10–17].

Even though these techniques obtain a compact filter representation and even a fixed-budget model, they still have drawbacks. For example, sparsification algorithms make growth sublinear but cannot constrain network size (model order) in a predefined range in nonstationary environments. Fixed-budget algorithms [12, 14–16] still require presetting the network size a priori, which also is a major drawback in nonstationary environments. Indeed, in such environments, the complexity of the time series as seen by a filter increases during the transitions because of the mixture of modes and can switch to a very low complexity in the next mode. Online adjustment of the filter order in infinite impulse filter (IIR) or finite impulse filter (FIR) is unreasonable because all filter coefficients need to be recomputed when the filter order is increased or decreased, which will cause undesirable transients. However, this is trivial in KLMS because of one major reason: the filter grows at each iteration, while the past parameters remain fixed. Hence, the most serious disadvantage of KLMS, its continuous growth, may become a feature that allows unprecedented exploration of the design space, creating effectively a filter topology optimally tuned to the complexity of the input, both in terms of adaptive parameters and model order. The model order selection can be handled by searching an appropriate compromise between accuracy and network size [18]. We adopt the minimum description length (MDL) criterion as the criterion to adaptively decide the model structure of KAFs. The MDL principle, first proposed by Rissanen in [19], is related to the theory of algorithmic complexity [20]. Rissanen formulated model selection as data compression, where the goal is to minimize the length of the combined bit stream of the model description concatenated with the

*Correspondence: chenbd@mail.xjtu.edu.cn
[2]Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University, Xi'an, China
Full list of author information is available at the end of the article

bit stream describing the error. MDL utilizes the description length as a measure of the model complexity and selects the model with the least description length.

Besides MDL, several other criteria have been proposed to deal with the accuracy/model order compromise. The pioneering work of the Akaike information criterion (AIC) [21] was followed by the Bayesian Information Criterion (BIC, which is also known as the "Schwarz information criterion (SIC)") [22, 23], the predictive minimum description length (PDL) [24], the normalized maximum likelihood [25], and the Bayesian Ying-Yang (BYY) information criterion [26]. The utilization of the MDL in our work is based on the fact that it is robust against noise and relatively easy to estimate online, which is a requirement of our design proposal. Compared with others, MDL also has the great advantage of relatively small computational costs [27, 28].

The paper structure is as follows: in Section 2, we present a brief review of the MDL principle and the related kernel adaptive filter algorithms, KLMS and quantized KLMS (QKLMS). Section 3 proposes a novel KLMS sparsification algorithm based on an online version of MDL. Because this algorithm utilizes quantization techniques, it is called QKLMS-MDL. The comparative results of the proposed algorithms are shown in Section 4, and final conclusions and discussion are given in Section 5.

2 Foundations

2.1 Minimum description length

The MDL principle addresses a system model as a data codifier and suggests choosing the model which provides the shortest description length. The basic principle of MDL estimates both the cost of specifying the model parameters and the associated model prediction errors [27].

Let f represent the model to be estimated. The model parameters $\boldsymbol{\theta}$ are chosen from a parametric family $\boldsymbol{\Theta}$

$$\left\{f(\boldsymbol{\chi}(n))|\boldsymbol{\theta} : \boldsymbol{\theta} \in \boldsymbol{\Theta} \subset \mathbb{R}^k\right\} \tag{1}$$

based on the observation $\boldsymbol{\chi}(n) = [\boldsymbol{x}(1), \ldots \boldsymbol{x}(n)]^T$, where k is the dimension of $\boldsymbol{\theta}$. MDL is a two-stage coding scheme: the model description length $L(\hat{\boldsymbol{\theta}})$ (in bits[1]) for the estimated member $\hat{\boldsymbol{\theta}}$ (system complexity) and the error description length $L(\boldsymbol{\chi}(n)|\hat{\boldsymbol{\theta}})$ (in bits) based on $\hat{\boldsymbol{\theta}}$ (system accuracy). According to Shannon's entropy principle,

$$L(\boldsymbol{\chi}(n)|\hat{\boldsymbol{\theta}}) = -\log P(\boldsymbol{\chi}(n)|\hat{\boldsymbol{\theta}}) \tag{2}$$

where $P(\boldsymbol{\chi}(n)|\hat{\boldsymbol{\theta}})$ is the conditional distribution of $\boldsymbol{\chi}(n)$ given $\hat{\boldsymbol{\theta}}$. There are several methods to estimate $L(\hat{\boldsymbol{\theta}})$, for example, simple description [29] or mixture description lengths [30]. Since the simple description length is easy to implement (assigns the same number of bits to each model parameter) and has been successfully utilized in many practical problems [31–33], it is used in this paper. Therefore, we have

$$L(\hat{\boldsymbol{\theta}}) = \frac{k}{2}\log n \tag{3}$$

Combining the description lengths from these two stages, we establish the two-stage MDL formulation

$$L_{\text{model}}(n) = -\log P(\boldsymbol{\chi}(n)|\hat{\boldsymbol{\theta}}) + \frac{k}{2}\log n \tag{4}$$

Intuitively, this criterion penalizes large models by taking into consideration the requirement of specifying a large number of weights. The best model according to MDL is the one that minimizes the sum of the model complexity and the number of bits required to encode their errors. MDL has rich connections with other model-selection frameworks. Obviously, minimizing $-\log P(\boldsymbol{\chi}(n)|\hat{\boldsymbol{\theta}})$ is equivalent to maximizing $P(\boldsymbol{\chi}(n)|\hat{\boldsymbol{\theta}})$. Therefore, in this sense, MDL coincides with the penalized maximum likelihood (ML) [34] in the parametric estimation problem, as well as AIC. The only difference is the penalty term (AIC uses k, the model order instead of Eq. 3). Furthermore, MDL has close ties to Bayesian principles (MDL approximates BIC asymptotically). Therefore, the MDL paradigm serves as an objective platform from which we can compare Bayesian and non-Bayesian procedures alike, even though the theoretical underpinnings behind them are much different.

The superiority of MDL has been indicated in various applications. In neural networks, MDL was adopted to determine the number of units that mimic the underlying dynamic property of the system [27, 35]. After that, as an improvement, the MDL criterion was utilized to directly determine an optimal neural network model and successfully applied to prediction problems [36] and control systems [37]. Furthermore, the embedding dimension of an artificial neural network is decided based on constructing a global model with a least description length [38]. Starting from an overly complex model and then pruning unneeded basis function according to MDL, Leonardis and Bischof [39] proposed a radial basis function (RBF) network formulation to balance accuracy performance, training time, and network complexity. Besides neural networks, MDL has also been successfully used in vector quantization [40], clustering [31, 41, 42], graphs [43, 44], and so on. In most cases, MDL is used for supervised learning as a penalty term on the error function or as a criterion for model selection [40]. One exception is the work of Zemel [33] who applied MDL to determine a suitable data-encoding schema. Compared with MDL, AIC criterion tends to select a model that is too complex and is not appropriate for small data sets.

In this work, MDL, as described by Eq. 4, will be utilized in a very specific and unique scenario, which can be best described as a stochastic extension of MDL (MDL-SE) to preserve compatibility with online learning. Recall that we are interested in continuously estimating the best filter order (the dictionary length) online when the statistical properties of the signals change in short windows (locally stationary environments). The KAF provides a simple update of the parameter vector on a sample-by-sample basis; the error description length can be estimated from the short-term sequence of the local error, but we have to estimate appropriately the error sequence probability. Both estimations seem practical. Also recall that a KAF using a stochastic gradient descent with a small step size always reduces locally the a priori error [1]. Therefore, there is an instantaneous feedback mechanism linking the local error and the local changes in the filter weights, which also simplifies the performance testing of the algorithm. In these conditions, we cannot work anymore with statistical operators (expected value) and will use instead temporal average operators on samples in the recent past of the current sample. Moreover, estimating the probability of the error in Eq. 4 must also be interpreted as a local operation in a sliding time window over the time series, with a length relevant to reflect the local statistics of the input. So MDL-SE creates a new parameter, the window length that needs to be selected a priori, and its influence in performance needs to be appropriately quantified and compared with alternative KAF approaches.

2.2 Kernel least-mean-square algorithm

KLMS utilizes gradient-descent techniques to search for the optimal solution in reproducing kernel Hilbert space (RKHS). Specifically, the learning problem of KLMS is to find a high-dimensional weight vector $\boldsymbol{\Omega}$ in RKHS $\boldsymbol{H}$ by minimizing the empirical risk:

$$\min_{\boldsymbol{\Omega}} R_{\text{emp}}[\,\boldsymbol{\Omega} \in \boldsymbol{H}, \boldsymbol{S} \in Z^n] = \sum_{i=1}^{n} \left(d(i) - \boldsymbol{\Omega}^T \boldsymbol{\varphi}(\boldsymbol{u}(i))\right)^2 \tag{5}$$

where $\{\boldsymbol{S} = (\boldsymbol{u}(1), d(1)), \ldots, (\boldsymbol{u}(n), d(n))\} \in Z^n$ is a sequence of learning samples. $\boldsymbol{\varphi}(\cdot)$ is a nonlinear mapping to transform data from the input space to a RKHS. Assume the initial weight is $\boldsymbol{\Omega}(0) = 0$, then using the LMS algorithm in RKHS yields

$$\begin{aligned} \epsilon(n) &= d(n) - \boldsymbol{\Omega}(n-1)^T \boldsymbol{\varphi}(\boldsymbol{u}(n)) \\ \boldsymbol{\Omega}(n) &= \eta \sum_{i=1}^{n} \epsilon(i) \boldsymbol{\varphi}(\boldsymbol{u}(i)) \\ &= \sum_{i=1}^{n} \alpha(i) \boldsymbol{\varphi}(\boldsymbol{u}(i)) \end{aligned} \tag{6}$$

where $\epsilon(n)$ and $\boldsymbol{\Omega}(n)$ are, respectively, the prediction error and weight vector at time index n. η is the step size. $\alpha(i)$ is the ith element of $\boldsymbol{\alpha}(n)$ to simplify the notation in this section. $\boldsymbol{\alpha}(n) = [\,\eta\epsilon(1), \eta\epsilon(2), \ldots \eta\epsilon(n)]$ is the coefficient vector of KLMS at n.

According to the "kernel trick," the system output for the new input $\boldsymbol{u}(n+1)$ can be expressed as

$$\begin{aligned} \boldsymbol{\Omega}(n)^T \boldsymbol{\varphi}(\boldsymbol{u}(n+1)) &= \left[\sum_{i=1}^{n} \alpha(i) \boldsymbol{\varphi}(\boldsymbol{u}(i))\right]^T \boldsymbol{\varphi}(\boldsymbol{u}(n+1)) \\ &= \sum_{i=1}^{n} \alpha(i) \kappa(\boldsymbol{u}(i), \boldsymbol{u}(n+1)) \end{aligned} \tag{7}$$

where $\kappa(\cdot,\cdot)$ is a Mercer kernel. In this work, the default kernel is the Gaussian kernel, $\kappa(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-||\boldsymbol{x} - \boldsymbol{x}'||^2 / 2\sigma^2\right)$, where σ is the kernel size. The set of samples $\boldsymbol{u}(i)$, used to position the positive defined function, constitutes the dictionary of centers $\mathcal{C}(n) = \{\boldsymbol{u}(i)\}_{i=1}^{n}$, which grows linearly with the input samples. KLMS provides a well-posed solution with finite data [1]. Moreover, as an online learning algorithm, KLMS is much simpler to implement, in terms of computational complexity and memory storage, than other batch-model kernel methods because all weights (the errors) remain fixed during the weight update.

2.3 Quantized kernel least-mean-square algorithm

Quantization techniques have been introduced to KLMS to develop a sparsified kernel filter, called QKLMS. This algorithm quantizes the input space with a simple vector quantization (VQ) algorithm to curb the network size. Every time a new sample arrives, the QKLMS algorithm checks if its distance to the available centers is less than the predefined minimal distance. If there is an existing center sufficiently close to the new sample, the center dictionary and network size remain unchanged but the coefficient of the closest center will be updated. Otherwise, the new sample is included in the dictionary. For online kernel learning, most of the existing VQ algorithms are not suitable because the center dictionary is usually trained offline and the computational burden is heavy. Therefore, a very simple and greedy online VQ method is used here based on the Euclidean distance in the input space between the sample and the existing centers.[2] QKLMS has been shown to be superior to all the other methods of curbing the dictionary growth [8], and the major difference is that the information available in the input is never discarded, it is used to update the VQ coefficients with each new input sample. Because of nonstationarity, it is unclear how one can cross-validate this parameter, so the minimal distance needs to be selected a priori for each application on a representative data segment, and experience shows that

performance changes smoothly around the optimum [8]. The sufficient condition for QKLMS mean-square convergence and a lower and upper bound on the theoretical value of the steady-state excess mean square error (EMSE) are studied in [8]. The summary of the QKLMS algorithm with online VQ is presented in Algorithm 1.

Algorithm 1 QKLMS Algorithm

Initialization: stepsize η, quantization threshold $\xi_{\mathbb{U}} > 0$, center dictionary $\mathcal{C}(1) = \{\boldsymbol{u}(1)\}$ and coefficient vector $\boldsymbol{\alpha}(1) = [\eta d(1)]$

while $\{\boldsymbol{u}(n), d(n)\}$ is available **do**

$\epsilon(n) = d(n) - \sum_{i=1}^{M(n-1)} \boldsymbol{\alpha}_i(n-1)\kappa(\boldsymbol{u}(n), \mathcal{C}_i(n-1))$

$dis(\boldsymbol{u}(n), \mathcal{C}(n-1)) = \min_{1\le i\le M(n-1)} \|\boldsymbol{u}(n) - \mathcal{C}_i(n-1)\|$

$i^* = \arg\min_{1\le i\le M(n-1)} \|\boldsymbol{u}(n) - \mathcal{C}_i(n-1)\|$

if $dis(\boldsymbol{u}(n), \mathcal{C}(n-1)) \le \xi_{\mathbb{U}}$ **then**

$\mathcal{C}(n) = \mathcal{C}(n-1)$, $\boldsymbol{\alpha}_{i^*}(n) = \boldsymbol{\alpha}_{i^*}(n-1) + \eta\epsilon(n)$

else

$\mathcal{C}(n) = \{\mathcal{C}(n-1), \boldsymbol{u}(n)\}$, $\alpha(n)^T = [\boldsymbol{\alpha}(n-1)^T, \eta\epsilon(n)]$

end if

end while

(where $\mathcal{C}_i(n-1)$ and $M(n-1)$ are the ith element and size of $\mathcal{C}(n-1)$ respectively, $\boldsymbol{\alpha}_i(n-1)$ is the ith element of $\boldsymbol{\alpha}(n-1)$, and $\|.\|$ denotes the Euclidean norm in the input space.)

3 MDL-based quantized kernel least-mean-square algorithm

The basic idea of the proposed algorithm, QKLMS-MDL, is as follows: once a new datum is received, the cost of adding this datum as a new center or merging it to its nearest center is calculated. Here, the distance measure for the merge operation is the same as the QKLMS [8]. Then the procedure with smaller description length is adopted: the proposed algorithm compares the strategy of discarding an existing center with the one of keeping it according to the MDL criterion. This process is repeated until all existing centers are scanned, such that the network size can be adjusted adaptively for every sample particularly in nonstationary situations. We show next how to estimate the MDL-SE criterion sufficiently well in these nonstationary conditions and that KAF MDL filters outperform conventional techniques of sparsification or fixed budget presented in the literature.

3.1 Objective function

In QKLMS-MDL, the adaptive model $L_{\text{model}}(n)$ has two parts: the quantized centers $\mathcal{C}(n) = \{\boldsymbol{c}_i\}_{i=1}^{M(n)}$ and the corresponding filter coefficients $\boldsymbol{\alpha}(n)$, each of size $M(n)$, the filter order, augmented by the other free parameters, the step size η and kernel size σ. Notice we are assuming that all the centers are mapped to the same RKHS and that we are using the Gaussian kernel fully defined by one free parameter, the kernel size. Equation 4 in this particular case is expressed as

$$L_{\text{model}}(n) = -\log P(\boldsymbol{\epsilon}(n)|\mathcal{C}(n), \boldsymbol{\alpha}(n)) + \frac{L(n)}{2}\log n \quad (8)$$

where the local error sequence is $\boldsymbol{\epsilon}(n) = [\epsilon(1), \ldots, \epsilon(n)]^T$, and $L(n) = 2M(n)+2$ is the number of the model's parameters at sample n. If $M(n)$ is far greater than 2, that is $M(n) \gg 2$, we can approximate $\frac{L(n)}{2}$ by $M(n)$, yielding

$$L_{\text{model}}(n) \approx -\log P(\boldsymbol{\epsilon}(n)|\mathcal{C}(n), \boldsymbol{\alpha}(n)) + M(n)\log n \quad (9)$$

The problem is how to estimate $P(\boldsymbol{\epsilon}(n)|\mathcal{C}(n), \boldsymbol{\alpha}(n))$. Because we assume local stationary conditions, it makes more sense to utilize only a window of the latest samples than taking the full history of errors to estimate the description length. Therefore, a sliding window methodology is adopted here. Only the latest L_w errors $\boldsymbol{\epsilon}_{L_w}(n) = [\epsilon(n - L_w + 1), \ldots, \epsilon(n)]^T$ are utilized to estimate the system performance, where L_w is the free-parameter window length. Hence, the objective function for MDL is expressed as

$$L_{\text{model}}(n) = -\log P(\boldsymbol{\epsilon}_{L_w}(n)|\mathcal{C}(n), \boldsymbol{\alpha}(n)) + M(n)\log L_w \quad (10)$$

We still need to find a reasonable estimator for the error log likelihood in the window. As usual, we can choose between a nonparametric or a parametric estimator of the probability of the errors in the window. Given the stochastic approximation nature of the proposed approach, creating and updating a histogram of the errors in the window or using Parzen estimators [45] makes sense but it will be noisy because the window is small. And how to select the histogram bin size or kernel for Parzen estimators brings new difficulties, another free parameter. Alternatively, we can estimate the error log likelihood in the window simply by the log of the empirical error variance for each sample in the window as shown in Eq. 11, which worked better in our tests and has no free parameter.

$$-\log P(\boldsymbol{\epsilon}_{L_w}(n)|\mathcal{C}(n), \boldsymbol{\alpha}(n)) = \frac{L_w}{2}\log\left(\sum_{i=n-L_w+1}^{n} \epsilon(i)^2\right) \quad (11)$$

This corresponds to an implicit Gaussian model for the probability density function (pdf) of the local error; however, we are not attempting to justify theoretically that the Gaussian is the best pdf to fit local errors in this scenario; we are just estimating the log likelihood as required

by MDL-SE. Note also that the intrinsic sample by sample feedback of the QKLMS helps here, because if the estimate is not correct, this simply says that the decision of decreasing or increasing the filter order will be wrong, the local error will increase, and the filter will self-correct the order for the next sample. This will create added error power with regard to the optimal performance, and since we are going to compare QKLMS-MDL with the conventional KAF approaches, this will show up as noncompetitive performance.

3.2 Formulation of MDL-based quantized kernel least-mean-square algorithm

When a new center is received, the description length costs of adding this data into the center dictionary or merging it into the nearest center are compared. That is, we calculate $\Delta L_{\text{model\_1}}(n)$ as shown in Eq. 12.

$$\begin{aligned}\Delta L_{\text{model\_1}}(n) =& L_{\text{add}}(n) - L_{\text{merge}}(n)\\ =& \frac{L_w}{2}\log\left(\sum_{i=n-L_w+1}^{n}\hat{\epsilon}(i)^2\right)+(M(n)+1)\\ & \log L_w - \frac{L_w}{2}\log\left(\sum_{i=n-L_w+1}^{n}\bar{\epsilon}(i)^2\right) - M(n)\log L_w\\ =& \frac{L_w}{2}\log\left(\sum_{i=n-L_w+1}^{n}\hat{\epsilon}(i)^2\right) - \frac{L_w}{2}\log\left(\sum_{i=n-L_w+1}^{n}\bar{\epsilon}(i)^2\right) + \log L_w\end{aligned} \tag{12}$$

where $\hat{\epsilon}(i)$ expresses the estimated prediction error after adding a new center and $\bar{\epsilon}(i)$ is the approximated prediction error after merging. If $\Delta L_{\text{model\_1}}(n) > 0$, the cost of adding a new center is larger than the cost of merging. Therefore, this new data should be merged to its nearest center according to the MDL criteria. Otherwise, a new center is added into the center dictionary.

As shown in Algorithm 1, the beauty of QKLMS is that its coefficients are a linear combination of current prediction errors. This fact leads to an easy estimate of $\hat{\epsilon}(i)$ and $\bar{\epsilon}(i)$. Taking $\hat{\epsilon}(i)$ as an example,

$$\begin{aligned}\hat{\epsilon}(i) =& d(i) - \sum_{j=1}^{M(n)}\boldsymbol{\alpha}_j(n)\kappa(\boldsymbol{u}(i),\mathcal{C}_j(n))\\ =& d(i) - \sum_{j=1}^{M(n-1)}\alpha_j(n-1)\kappa(\boldsymbol{u}(i),\mathcal{C}_j(n-1)) - \eta\epsilon(n)\kappa(\boldsymbol{u}(i),\boldsymbol{u}(n))\\ =& \epsilon(i) - \eta\epsilon(n)\kappa(\boldsymbol{u}(i),\boldsymbol{u}(n))\end{aligned} \tag{13}$$

Assume c_{i*} is the nearest center of $u(n)$,

$$\bar{\epsilon}(i) = \epsilon(i) - \eta\epsilon(n)\kappa(\boldsymbol{u}(i),\boldsymbol{c}_{i*}) \tag{14}$$

Substituting these two equations into Eq. 12, it is straightforward to obtain $\Delta L_{\text{model\_1}}(n)$.

Similar to QKLMS, QKLMS-MDL merges a sample into its nearest center in the input space. The difference is that QKLMS-MDL quantizes the input space according to not only the input data distance but also the prediction error, which results in higher accuracy.

We have just solved the problem of how to increase the network size efficiently. But this is insufficient in a nonstationary condition where older centers should also be discarded. General methods to handle this situation utilize a nonstationary detector, like the likelihood ratio test [46], M-estimators [47], and spectra analysis, to check whether the true system or the input data shows different statistical properties. However, the computational complexity of these detectors is high and their accuracy is not good enough. The ability of MDL to estimate system complexity according to the data complexity inspired us to also apply MDL as the criterion for adaptively discarding existing centers. After checking whether a new datum should be added to the center dictionary, the proposed algorithm compares the description length costs between discarding an existing center and keeping the datum, and the strategy with the smaller description length is taken. This procedure is repeated until all existing centers are scanned.

Similar to Eq. 12, the description length difference between discarding and keeping a center is shown in Eq. 15,

$$\begin{aligned}\Delta L_{\text{model\_2}}(n) =& L_{\text{discard}}(n) - L_{\text{keep}}(n)\\ =& \frac{L_w}{2}\log\left(\sum_{i=n-L_w+1}^{n}\hat{\epsilon}(i)^2\right)+(M(n)-1)\log\\ & L_w - \frac{L_w}{2}\log\left(\sum_{i=n-L_w+1}^{n}\bar{\epsilon}(i)^2\right) - M(n)\log L_w\\ =& \frac{L_w}{2}\log\left(\sum_{i=n-L_w+1}^{n}\hat{\epsilon}(i)^2\right) - \frac{L_w}{2}\log\left(\sum_{i=n-L_w+1}^{n}\bar{\epsilon}(i)^2\right) - \log L_w\end{aligned} \tag{15}$$

where $\hat{\epsilon}(i)$ expresses the estimated prediction error after discarding an existing center, while $\bar{\epsilon}(i)$ is the approximated prediction error of keeping this center. If $\Delta L_{\text{model\_2}}(n) > 0$, the cost of discarding an existing center is larger than the cost of keeping it. Therefore, the center dictionary does not change. Otherwise, an existing center should be discarded. Assume c_k is the center to be examined,

$$\begin{aligned}\hat{\epsilon}(i) =& \epsilon(i) + \boldsymbol{\alpha}_k(n-1)\kappa(\boldsymbol{u}(i),\boldsymbol{c}_k)\\ \bar{\epsilon}(i) =& \epsilon(i)\end{aligned} \tag{16}$$

Notice that as long as an existing center is discarded, the value of $\bar{e}(i)$ for the next iteration should be changed correspondingly. Let $\boldsymbol{c}_j$ be the discarded center, so

$$\epsilon(i) = \epsilon(i) + \boldsymbol{\alpha}_k(n)\kappa(\boldsymbol{u}(i),\boldsymbol{c}_j) \tag{17}$$

Otherwise, $\epsilon(i)$ doesn't change.

Algorithm 2 gives a summary of the proposed QKLMS-MDL algorithm. Compared with traditional KLMS, the computational complexity of the proposed method is improved. At each iteration, the computational complexity to deciding whether a new center should be added or not is $O(L_w)$ and it is $O(ML_w)$ for deciding whether an existing center should be discarded or not.

Algorithm 2 QKLMS-MDL Algorithm

Initialization: window length L_w, stepsize η, center dictionary $C(1) = \{\boldsymbol{u}(1)\}$ and coefficient vector $\boldsymbol{\alpha}(1) = [\eta d(1)]$;
Computing
while $\{\boldsymbol{u}(n), d(n)\}$ is available **do**
  $\epsilon(n) = d(n) - \sum_{i=1}^{M(n-1)} \boldsymbol{\alpha}_i(n-1)\kappa(\boldsymbol{u}(n), \mathcal{C}_i(n-1))$;
  % check whether a new center is added or not
  Calculate $\Delta L_{\text{model\_1}}(n)$ according to Eq.12;
  if $\Delta L_{\text{model\_1}}(n) > 0$ **then**
    $\mathcal{C}(n) = \{\mathcal{C}(n-1), \boldsymbol{u}(n)\}, \boldsymbol{\alpha}(n)^T = [\boldsymbol{\alpha}(n-1)^T, \eta\epsilon(n)]$;
  else
    $i^* = \underset{1\le i\le M(n-1)}{\arg\min} \|\boldsymbol{u}(n) - \mathcal{C}_i(n-1)\|$;
    $\mathcal{C}(n) = \mathcal{C}(n-1), \boldsymbol{\alpha}_{i^*}(n) = \boldsymbol{\alpha}_{i^*}(n-1) + \eta\epsilon(n)$;
  end if
  % check whether the existing centers are discarded
  while $c_k \in \mathcal{C}(n)$ **do**
    Calculate $\Delta L_{\text{model\_2}}(n)$ according to Eq.15;
    if $\Delta L_{\text{model\_2}}(n) > 0$ **then**
      $\mathcal{C}(n)$ doesn't change;
    else
      $\mathcal{C}(n) = \{\mathcal{C}(n) \backslash \boldsymbol{c}_k\}$, throw away the corresponding coefficient and update prediction error $\boldsymbol{e}_{n,L_w}$ according to Eq.17;
    end if
  end while
end while

The above derivation assumes that model accuracy and simplicity are equally important, as in the stationary case. In practice, when the system model order is low, the performance accuracy plays a more important role than model simplicity. In fact, in this case, the instantaneous errors may be frequently small by chance and QKLMS-MDL will wrongly interpret that the model order is high and needs to be decreased. This is dangerous when the filter order is low, because the filer has few degrees of freedom and the penalty in accuracy will be high. Therefore, we adopt a strategy that privileges system accuracy, i.e., if the network size is smaller than a predefined minimal model order threshold N, no matter what is the value of Eq. 15, the corresponding center should be kept in the center dictionary. The disadvantage of this heuristic is that it creates an extra free parameter in the modeling besides the window size L_w. In our simulations, we kept $N = 5$ because the experiments show good results.

The other free parameter L_w adjusts the compromise between system accuracy and network size, and it is more important. Note that in Eq. 10 with the estimator of Eq. 11 L_w enters linearly in the error term and as a log in the model complexity term. Therefore, long windows put a higher constraint on the error than on the model order, which indirectly emphasizes model accuracy. When $L_w = 1$, we are just using maximum likelihood. We experimentally verified the effect of L_w in simulations because we still do not have a systematic approach to select this parameter based on the data but can advance some experimental observations: (a) the larger the L_w, the more emphasis is given to system accuracy instead of model complexity. Otherwise, a smaller L_w yields simpler (fewer parameters) models. (b) If the model order changes frequently, the value of L_w should be reduced, such that the data utilized to estimate the description length has a higher probability of being in the stationary regime. Finally, even though the procedure of discarding existing centers scales proportionally to the window size only, there is no need to check it at every sample in local stationary environments because they do not occur that fast nor all the time. We suggest that the update be done associated with the local stationary of the input data, i.e., at a rate at least twice the inverse of L_w.

4 Simulation

In this section, we test the QKLMS-MDL in three signal-processing applications. We begin by exploring the behavior of QKLMS-MDL in a simple environment. Next, we move to a real-data time-series prediction problem , and we finalize with an application in speech processing. The tests presented are online learning tests when the weights are learned from an initial zero state and never fixed, and so, they represent performance in unseen data, similar to test set results, except that the free model parameters have been set a priori at their best values. To gauge the effect of the free parameters of the algorithms, we also present results across a set of their possible values around the optimum. Monte Carlo tests are in some conditions conducted to illustrate the effect of variability across different conditions.

4.1 System model

In this section, the underlying dynamic system is governed by

$$z(n) = a(n)\frac{z(n-1)z(n-2)z(n-3)x(n-2)(z(n-3)-1)+x(n-1)}{1+z(n-2)^2+z(n-3)^2} \quad (18)$$

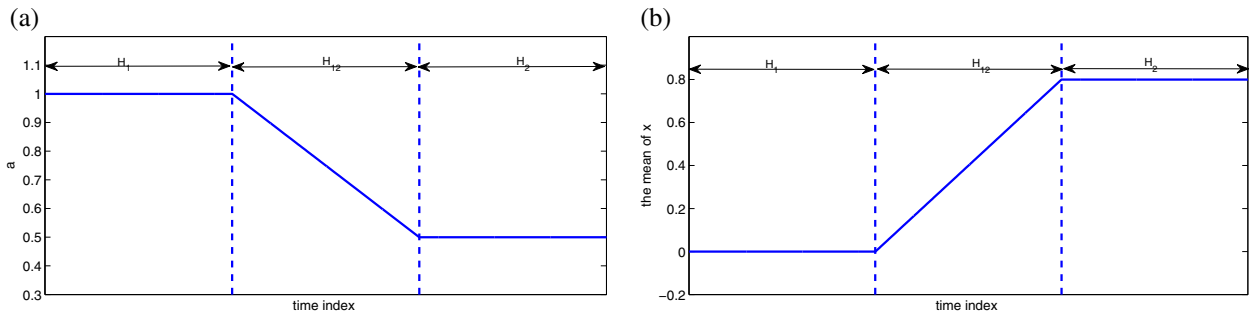

Fig. 1 Evolution curves of system parameters: **a** coefficient a; **b** mean of x

where the system input $x(n)$ is a random signal distributed as $\mathcal{N}(\mu, 1)$ and $a(n)$ is a time-varying system coefficient. The noisy system output is given by $y(n) = z(n) + v(n)$, where the noise $v(n)$ is Gaussian distributed with zero mean and standard deviation 0.1. We change the value of $a(n)$ and the mean of $x(n)$ with time to simulate a non-stationary scenario: $a = 1.0$ in the first 1000 samples (segment H_1) and decreases to 0.5 with different change rates (segment H_{12}) then is fixed to $a = 0.5$ (segment H_2). Similarly, the mean of $x(n)$ is set at 0 in segment H_1 and then linearly increase to 0.8, as shown in Fig. 1. The problem setting for system identification is as follows: the input vector of the kernel filter is

$$\boldsymbol{u}(n) = [y(n-1), y(n-2), y(n-3), x(n-1), x(n-2)]^T$$

and the corresponding design signal is $y(n)$. In this section, both the kernel size and the step size are set at 1.0. The online MSE is calculated based on the mean of the prediction error averaged in a running window of 100 samples.

At first, we compare the performance of QKLMS and QKLMS-MDL when the true system or the input data change slowly, for example, the length of transition area H_{12} is 1000. The quantization factor γ of QKLMS is set as 0.75 and the window length of QKLMS-MDL is 100 such that both QKLMS and QKLMS-MDL have relatively good compromise between system complexity and accuracy performance. The simulation results over 200 Monte Carlo runs are shown in Fig. 2. As shown in Fig. 2, the network size of QKLMS-MDL is much smaller than QKLMS while their MSE are almost the same. Furthermore, the QKLMS-MDL adjusts the network size adaptively according to the true system complexity. For example, during the transition of $a(n)$, the system parameters change which increases the signal complexity, but after iteration 2000, the network size decreases gradually signaling that the input signal complexity is reduced. Older centers are discarded, and new important centers are added to the center dictionary. On the contrary, QKLMS keeps all existing centers which results in a network size always growing larger.

In order to verify the superiority of QKLMS-MDL, in Table 1, we list the final network sizes of QKLMS and QKLMS-MDL for different transition length conditions. The parameter settings are shown in Table 2 where all parameters are selected such that both algorithms yield similar MSE. The final network size of QKLMS-MDL is noticeably smaller than QKLMS. Furthermore, in these different conditions, the final QKLMS-MDL network sizes are similar to each other. This fact also testifies

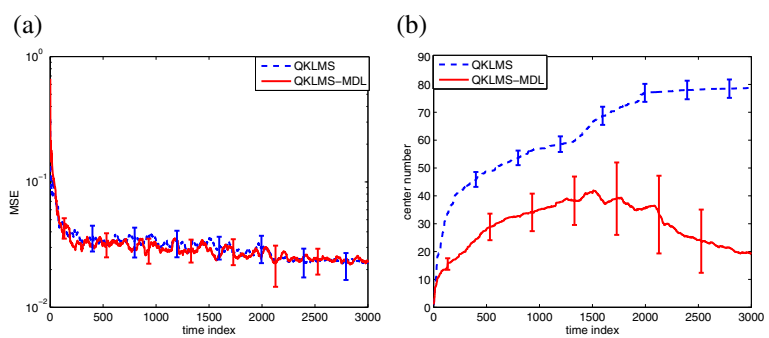

Fig. 2 Performance comparison for QKLMS and QKLMS-MDL in a system identification problem with transition area length 1000. **a** Convergence curves in terms of MSE. **b** Network size evolution curves

Table 1 The final network size comparison of QKLMS and QKLMS-MDL in a system identification problem

	Abrupt change	$\frac{1}{500}$	$\frac{1}{5000}$
QKLMS	118.4 ± 4.22	95.08 ± 3.47	124.37 ± 3.52
QKLMS-MDL	12.77 ± 7.55	14.89 ± 9.19	14.23 ± 10.85

All of these results are summarized in the form of "average $\pm$ standard deviation"

that the network size chosen by QKLMS-MDL is decided by the local complexity of current input data.

4.2 Santa Fe time-series prediction

In this experiment, the chaotic laser time series from the Santa Fe time-series competition, data set A [48] is used [49]. This time series is particularly difficult to predict, due both to its chaotic dynamics and to the fact that only three "intensity collapses" occur in the data set [2], as shown in Fig. 3. After normalizing all data to lie in the range $[0, 1]$, we consider the simple one-step prediction problem. The previous 40 points $u(i) = [x(i-40), \ldots, x(i-1)]^T$ are used as the input vector to predict the current value $x(i)$ which is the desired response.

At first, we verify the superiority of QKLMS-MLD versus the QKLMS, the novelty criterion (NC), and surprise criterion (SC), denoted, respectively, as KLMS-NC and KLMS-SC. In the simulation below, the free parameters related to KLMS are set as follows: the Gaussian kernel with kernel width equaling to 0.6 is selected and step size η is set to 0.9, which were cross-validated in a training set. Performance comparisons between these algorithms are presented in Figs. 4 and 5. In Fig. 4, the parameters of the four sparse algorithms are chosen such that the algorithms yield almost the same maximum network size, while in Fig. 5, the parameters of the sparse algorithms are chosen such that they produce almost the same MSE at the end of adaptation. Table 3 lists the specific parameter settings. Simulation results clearly indicate that the QKLMS-MDL exhibits much better performance, i.e., achieves either a much smaller network size or much smaller MSE than QKLMS, KLMS-NC, and KLMS-SC. Owing to coarse quantization, most of the useful information is discarded by KLMS-NC which results in its learning curve not decreasing after 190 iterations as shown in Fig. 4. Different from QKLMS, QKLMS-MDL considers the prediction error into quantization and, consequently, has better performance. More importantly, the network size of QKLMS-MDL varies. For example, as shown in Fig. 4, the network size of QKLMS-MDL peaks when the collapse occurs because of local nonstationarity and decreases between time indexes 300 and 400 because the input data complexity in this range is well predicted by a lower model. This experiment shows the proposed method has good tracking ability even for data with local nonstationarity.

Then we investigate how the window length affects the system performance. The effect of window length on final MSE and final network size are shown in Figs. 6 and 7, respectively. As shown in Fig. 6, when the window length increases, the final MSE of KLMS-MDL is closer to that of KLMS (for comparison, the final MSE of KLMS also is plotted in the figure). Figure 7 indicates that the increase of the window length results in an increase of the final network size. Fortunately, this growth trend gradually stabilizes, such that the network size is limited in a range no matter how long the window length is. In this sense, the window length is also affecting the compromise between system accuracy and network complexity as we can expect from Eqs. 12 and 15, since the window length appears in log form in one of the terms and not in the other.

Table 2 Parameter settings in a channel equalization problem to achieve almost the same final MSE in each condition

	Abrupt change	$\frac{1}{500}$	$\frac{1}{5000}$
QKLMS	$\gamma = 0.65$	$\gamma = 0.7$	$\gamma = 0.65$
QKLMS-MDL	$L_w = 100$	$L_w = 100$	$L_w = 100$

4.3 Speech signal analysis

Prediction analysis is a widely used technique for the analysis, modeling, and coding of speech signals [1]. A slowly time-varying filter could be used to model the vocal tract, while a white noise sequence (for unvoiced sounds) represents the glottal excitation. Here, we use the kernel adaptive filter to establish a nonlinear prediction model for the speech signal. Ideally, for each different phoneme, which represents a window of time series with the same basic statistics, the prediction

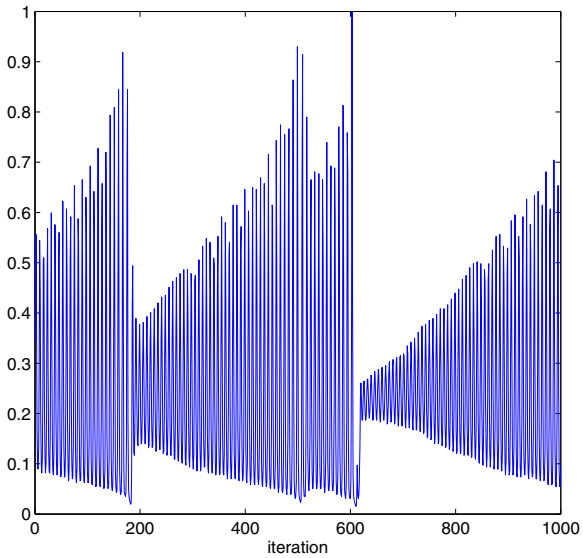

Fig. 3 Santa Fe time-series data

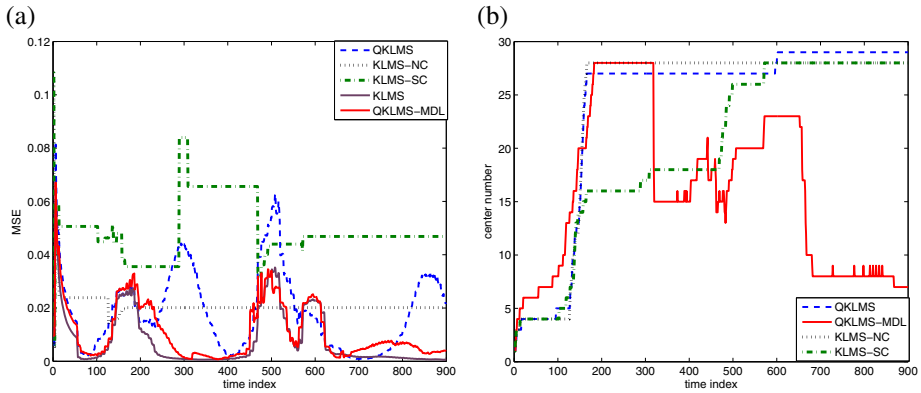

Fig. 4 Performance comparison in the Santa Fe time-series prediction (the same maximum network size). **a** Convergence curves in terms of MSE. **b** Network size evolution curves

model of speech should adjust accordingly to "track" such change. Therefore, we can conjecture to identify the phonetic changes through observing the kernel adaptive filter behavior. For example, when the filter order has local peaks, the phonetics should be changing because the filter observes two different stationary regimes, as demonstrated in Section 4.2. Therefore, such abrupt filter change can be used for phoneme segmentation, which is rather difficult to do even for the human observer.

A short whole voiced sentence from a male is used as the source signal. The original voice file can be downloaded from [50]. To increase the MDL model accuracy when the stationary window is short, this signal is upsampled to 16,000 Hz from the original 10,000 Hz. Then QKLMS is applied by the previous 11 samples $u(i) = [x(i-11),\ldots,x(i-1)]^T$. In this section, the Gaussian kernel with $\sigma = 0.3$ is selected. The step size is 0.7 according to the cross-validation test. The quantization factor of QKLMS is set as 0.67 and the window length of QKLMS-MDL is 50, such that both QKLMS and QKLMS-MDL have relatively similar prediction MSE performance.

Figure 8 shows how the network size of QKLMS-MDL varies with the input speech signal. The phonetic boundaries are obtained from SPPAS, which is a free audio annotation tool [51, 52]. As show in this plot, the network size changes close to the phonetic boundary, like [W] to [e:r] and [j] to [i]. After adapting to the speech (first couple of phonemes), the network size of QKLMS-MDL only changes dramatically during the phoneme transitions [e:r], [ei], [i], and [r], because the signal statistics change and the QKLMS-MDL initially increases the prediction model due to co-articulation and then decreases it again too much (with an undershoot) until it stabilizes for the remaining of the phoneme. On the other hand, we see that the QKLMS is always increasing size, with the largest rate of increase again during the phoneme transitions. So both respond to the changes in the signal statistics, but QKLMS-MDL preserves a low filter order, with comparable error during the full sentence.

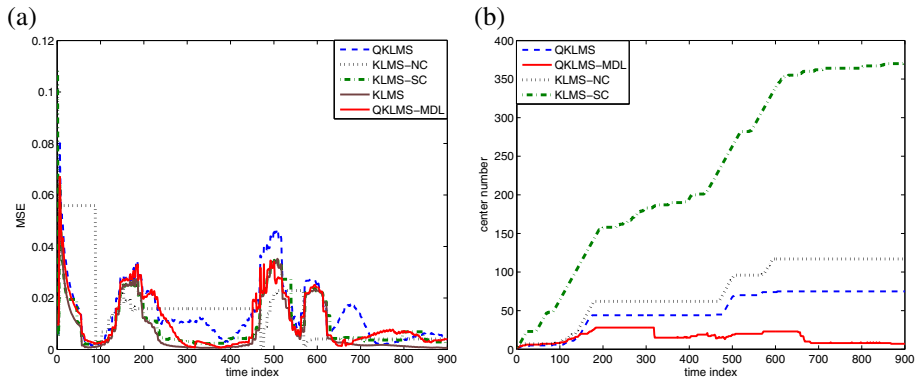

Fig. 5 Performance comparison in the Santa Fe time-series prediction (the same MSE at the final stage). **a** Convergence curves in terms of MSE. **b** Network size evolution curves

Table 3 Parameter settings for different algorithms in a time-series prediction

	Same network size	Same final MSE
QKLMS	$\gamma = 1.97$	$\gamma = 1.54$
QKLMS-MDL	$L_w = 150$	$L_w = 150$
KLMS-NC	$\sigma_1 = 1.38$	$\sigma_1 = 0.85$
	$\sigma_2 = 0.001$	$\sigma_2 = 0.001$
KLMS-SC	$\lambda = 0.005$	$\lambda = 0.005$
	$T_1 = 90$, $T_2 = -0.085$	$T_1 = 300$, $T_2 = -1.6$

σ_1 the distance threshold, σ_2 the error threshold, λ the regularization parameter, T_1 the upper threshold of the surprise measure, T_2 the lower threshold of the surprise measure

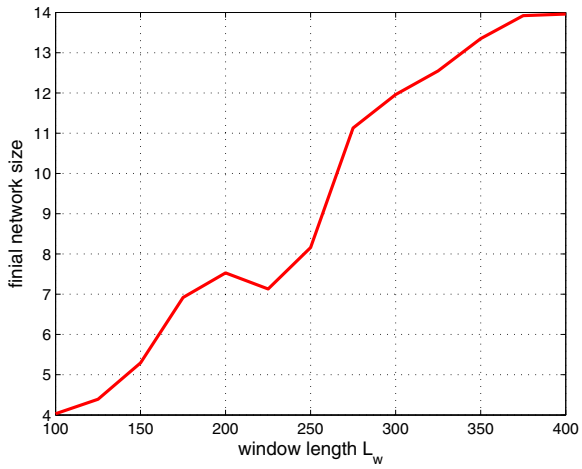

Fig. 7 Effect of the window length on the final network size in the Santa Fe time-series data

5 Conclusions

This paper proposes for the first time a truly self-organizing adaptive filter where all the filter weight and order are adapted online from the input data. How to choose an efficient kernel model to make a trade-off between computational complexity and system accuracy in nonstationary environments is the ultimate test for an online adaptive algorithm. Based on the popular and powerful MDL principle, a sparsification algorithm for kernel adaptive filters is proposed. Experiments show that the QKLMS-MDL successfully adjusts the network size according to the input data complexity while keeping the accuracy in an acceptable range. This property is very useful in nonstationary conditions while other existing sparsification methods keep increasing the network size. Fewer free parameters and an online model makes QKLMS-MDL practical in real-time applications.

We believe that many applications will benefit from QKLMS-MDL. For example, speech signal processing is an extremely interesting area. Nonstationary behavior is the nature of speech. Furthermore, owing to a sufficient number of samples to quantify short-term stationarity, front-end signal processing (acoustic level) based on QKLMS-MDL seems to be a good methodology to improve the quantification of speech recognition because of its fast response. However, this is still a very preliminary paper, and many more results and better arguments for the solutions are required. Although self-organization solves some problems, it also brings new ones in the processing. In fact, as described in Section 3, the QKLMS-MDL algorithm "forgets" in nonstationary conditions the previous learning results and needs to relearn the input-output mapping when the system switches to a new state. First, the filter weights would have to be read and clustered to identify the sequence of phonemes. Second, the MDL strategy keeps the system model at the simplest structure, but when a previous state appears

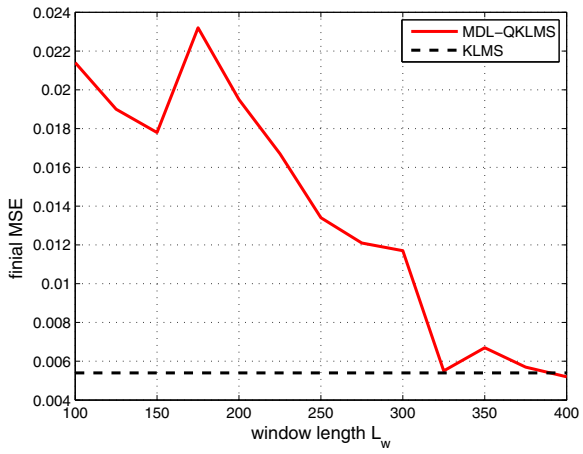

Fig. 6 Effect of the window length on the final MSE in the Santa Fe time-series data

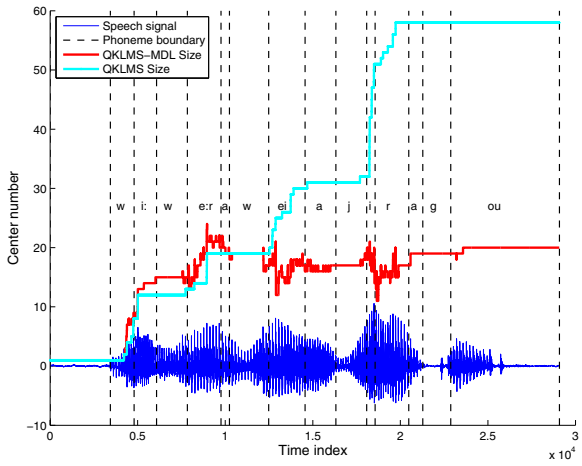

Fig. 8 Adaptive filter performance in a speech signal

again, QKLMS-MDL must relearn from scratch because the former centers are removed from the center dictionary. This relearning should be avoided, and increasingly accurate modeling techniques should be developed. Future work will have to address this aspect of the technique.

Endnotes

[1]The unit of description length also could be *nat*, which is calculated by ln rather than $\log_2$ as we do here. For simplification, we use log to present $\log_2$ in the following part of this paper.

[2]A translation invariant kernel establishes a continuous mapping from the input space to the feature space. As such, the distance in feature space is monotonically increasing with the distance in the original input space. Therefore, for a translation invariant kernel, the VQ in the original input space also induces a VQ in the feature space.

Acknowledgments
This work was supported by NSF Grant ECCS0856441, NSF of China Grant 61372152, and the 973 Program 2015CB351703 in China.

Authors' contributions
SZ conceived of the idea and drafted the manuscript. JP, BC, and PZ have contributed in the supervision and guidance of the manuscript. CZ participated in the simulation and experiment. All authors did the research and data collection. All authors read and approved the final manuscript.

Competing interests
The authors declare that they have no competing interests.

Author details
[1]Department of Electrical and Computer Engineering, University of Florida, Gainesville, USA. [2]Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University, Xi'an, China.

Received: 20 June 2016 Accepted: 28 September 2016
Published online: 12 October 2016

References

1. W Liu, PP Pokharel, JC Príncipe, The kernel least mean square algorithm. IEEE Trans. Sig. Process. **56**(2), 543–554 (2008)
2. Y Engel, S Mannor, R Meir, The kernel recursive least-squares algorithm. IEEE Trans. Sig. Process. **52**(8), 2275–2285 (2004)
3. J Platt, A resource-allocating network for function interpolation. Neural Comput. **3**(4), 213–225 (1991)
4. P Bouboulis, S Theodoridis, Extension of Wirtinger's calculus to reproducing kernel Hilbert spaces and the complex kernel LMS. J. IEEE Trans. Sig. Process. **59**(3), 964–978 (2011)
5. K Slavakis, S Theodoridis, Sliding window generalized kernel affine projection algorithm using projection mappings. EURASIP J. Adv. Sig. Process. **1**, 1–16 (2008)
6. C Richard, JCM Bermudez, P Honeine, Online prediction of time series data with kernels. IEEE Trans. Sig. Process. **57**(3), 1058–1066 (2009)
7. W Liu, Il Park, JC Príncipe, An information theoretic approach of designing sparse kernel adaptive filters. IEEE Trans. Neural Netw. **20**(12), 1950–1961 (2009)
8. B Chen, S Zhao, P Zhu, JC Príncipe, Quantized kernel least mean square algorithm. IEEE Trans. Neural Netw. Learn. Syst. **23**(1), 22–32 (2012)
9. B Chen, S Zhao, P Zhu, JC Príncipe, Quantized kernel recursive least squares algorithm. IEEE Trans. Neural Netw. Learn. Syst. **24**(9), 1484–1491 (2013)
10. SV Vaerenbergh, J Via, I Santamana, A sliding-window kernel RLS algorithm and its application to nonlinear channel identification. IEEE Int. Conf. Acoust. Speech Sig. Process, 789–792 (2006)
11. SV Vaerenbergh, J Via, I Santamana, Nonlinear system identification using a new sliding-window kernel RLS algorithm. J. Commun. **2**(3), 1–8 (2007)
12. SV Vaerenbergh, I Santamana, W Liu, JC Príncipe, Fixed-budget kernel recursive least-squares. IEEE Int. Conf. Acoust. Speech Sig. Process, 1882–1885 (2010)
13. M Lázaro-Gredilla, SV Vaerenbergh, I Santamana, A Bayesian approach to tracking with kernel recursive least-squares. IEEE Int. Work. Mach. Learn. Sig. Process. (MLSP), 1–6 (2011)
14. S Zhao, B Chen, P Zhu, JC Príncipe, Fixed budget quantized kernel least-mean-square algorithm. Sig. Process. **93**(9), 2759–2770 (2013)
15. D Rzepka, in *2012 IEEE 17th Conference on Emerging Technologies & Factory Automation (ETFA)*. Fixed-budget kernel least mean squares, (Krakow, 2012), pp. 1–4
16. K Nishikawa, Y Ogawa, F Albu, in *Signal and Information Processing Association Annual Summit and Conference (APSIPA)*. Fixed order implementation of kernel RLS-DCD adaptive filters, (Asia-Pacific, 2013), pp. 1–6
17. K Slavakis, P Bouboulis, S Theodoridis, Online learning in reproducing kernel Hilbert spaces. Sig. Process. Theory Mach. Learn. **1**, 883–987 (2013)
18. W Gao, J Chen, C Richard, J Huang, Online dictionary learning for kernel LMS. IEEE Trans. Sig. Process. **62**(11), 2765–2777 (2014)
19. J Rissanen, Modeling by shortest data description. Sig. Process. **14**(5), 465–471 (1978)
20. M Li, PMB Vitányi, *An introduction to Kolmogorov complexity and its applications*. (Publisher, Springer-Verlag, New York Inc, 2008)
21. H Akaike, A new look at the statistical model identification. IEEE Trans. Autom. Control. **19**(2), 716–723 (1974)
22. G Schwarz, Estimating the dimension of a model. Ann. Stat. **6**(2), 461–464 (1978)
23. A Barron, J Rissanen, B Yu, The minimum description length principle in coding and modeling. IEEE Trans. Inf. Theory. **44**(6), 2743–2760 (1998)
24. J Rissanen, Universal coding, information, prediction, and estimation. IEEE Trans. Inf. Theory. **30**(4), 629–636 (1984)
25. J Rissanen, MDL denoising. IEEE Trans. Inf. Theory. **46**(7), 2537–2543 (2000)
26. L Xu, Bayesian Ying Yang learning (II): a new mechanism for model selection and regularization. Intell. Technol. Inf. Anal, 661–706 (2004)
27. Z Yi, M Small, Minimum description length criterion for modeling of chaotic attractors with multilayer perceptron networks. IEEE Trans. Circ. Syst. I: Regular Pap. **53**(3), 722–732 (2006)
28. T Nakamura, K Judd, AI Mees, M Small, A comparative study of information criteria for model selection. Int. J. Bifurcation Chaos Appl. Sci. Eng. **16**(8) (2153)
29. T Cover, J Thomas, *Elements of information theory*. (Wiley, 1991)
30. M Hansen, B Yu, Minimum description length model selection criteria for generalized linear models. Stat. Sci. A Festschrift for Terry Speed. **40**, 145–163 (2003)
31. K Shinoda, T Watanabe, MDL-based context-dependent subword modeling for speech recognition. Acoust. Sci. Technol. **21**(2), 79–86 (2000)
32. AA Ramos, The minimum description length principle and model selection in spectropolarimetry. Astrophys. J. **646**(2), 1445–1451 (2006)
33. RS Zemel, *A minimum description length framework for unsupervised learning, Dissertation*. (University of Toronto, 1993)
34. AWF Edwards, *Likelihood*, (Cambridge Univ Pr, 1984)
35. M Small, CK Tse, Minimum description length neural networks for time series prediction. Astrophys. J. **66**(6), 066701 (2002)
36. A Ning, H Lau, Y Zhao, TT Wong, Fulfillment of retailer demand by using the MDL-optimal neural network prediction and decision policy. IEEE Trans. Ind. Inform. **5**(4), 495–506 (2009)
37. JS Wang, YL Hsu, An MDL-based Hammerstein recurrent neural network for control applications. Neurocomputing. **74**(1), 315–327 (2010)
38. YI Molkov, DN Mukhin, EM Loskutov, AM Feigin, GA Fidelin, Using the minimum description length principle for global reconstruction of dynamic systems from noisy time series. Phys. Rev. E. **80**(4), 046207 (2009)
39. A Leonardis, H Bischof, An efficient MDL-based construction of RBF networks. Neural Netw. **11**(5), 963–973 (1998)

40. H Bischof, A Leonardis, A Selb, MDL principle for robust vector quantisation. Pattern Anal. Appl. **2**(1), 59–72 (1999)
41. T Rakthanmanon, EJ Keogh, S Lonardi, S Evans, MDL-based time series clustering. Knowl. Inf. Syst., 1–29 (2012)
42. H Bischof, A Leonardis, in *15th International Conference on Pattern Recognition*. Fuzzy c-means in an MDL-framework, (Barcelona, 2000), pp. 740–743
43. I Jonye, LB Holder, DJ Cook, MDL-based context-free graph grammar induction and applications. Int. J. Artif. Intell. Tools. **13**(1), 65–80 (2004)
44. S Papadimitriou, J Sun, C Faloutsos, P Yu, Hierarchical, parameter-free community discovery. Mach. Learn. Knowl. Discov. Databases., 170–187 (2008)
45. E Parzen, On estimation of a probability density function and mode. Ann. Math. Stat. **33**(3), 1065–1076 (1962)
46. AM Mood, FA Graybill, DC Boes, *Introduction to the Theory of Statistics*. (McGraw-Hill, USA, 1974)
47. SA Geer, *Applications of empirical process theory*. (The Press Syndicate of the University of Cambridge, Cambridge, 2000)
48. The Santa Fe time series competition data. http://www-psych.stanford.edu/~andreas/Time-Series/SantaFe.html, Accessed June 2016
49. AS Weigend, NA Gershenfeld, *Time series prediction: forecasting the future and understanding the past*. (Westview Press, 1994)
50. Sound files obtained from system simulations. http://www.cnel.ufl.edu/~pravin/Page_7.htm, Accessed June 2016
51. B Bigi, in *The eighth international conference on Language Resources and Evaluation*. SPPAS: a tool for the phonetic segmentations of Speech, (Istanbul, 2012), pp. 1748–1755
52. SPPAS: automatic annotation of speech. http://www.sppas.org, Accessed June 2016