Fixed budget quantized kernel least-mean-square algorithm

Songlin Zhao a,*, Badong Chen b, Pingping Zhu a, José C. Príncipe a

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

Article info
Article history:
Received 26 September 2012
Received in revised form
14 February 2013
Accepted 16 February 2013
Available online 26 February 2013

Keywords:
Kernel methods
Quantized kernel least mean square
Fixed budget
Growing and pruning

Abstract
This paper presents a quantized kernel least mean square algorithm with a fixed memory budget, named QKLMS-FB. In order to deal with the growing support inherent in online kernel methods, the proposed algorithm utilizes a pruning criterion, called significance measure, based on a weighted contribution of the existing data centers. The basic idea of the proposed methodology is to discard the center with the smallest influence on the whole system, when a new sample is included in the dictionary. The significance measure can be updated recursively at each step which is suitable for online operation. Furthermore, the proposed methodology does not need any a priori knowledge about the data and its computational complexity is linear with the center number. Experiments show that the proposed algorithm successfully prunes the least “significant” centers and preserves the important ones, resulting in a compact KLMS model with little loss in accuracy.

1. Introduction

During the last few years, there has been increasing attention directed towards kernel learning methods which utilize Mercer kernels to map the data from the input space to a high-dimensional feature space. These powerful tools, such as support vector machines (SVM) [1], kernel principal component analysis (kernel PCA) [2], and kernel regression [3], show strong classification and regression performance in complicated nonlinear problems for pattern analysis and statistical signal processing. However, the price paid for good nonlinear approximation is a batch procedure with high computation cost, scaling with \( O(N^3) \), as well as up to \( O(N^2) \) for memory usage, with \( N \) being the number of processed data. This represents a huge drawback for online applications, where computing time requirement is an important factor. As a remedy, online kernel learning (OKL) [4,5] provides more computationally efficient alternatives to approximate nonlinear functions with the added advantage of tracking nonstationarities in time. As a member of OKL, kernel adaptive filters are recently making rapid progress, because they are superior to other nonlinear filters with several desirable features: (1) For universal kernels, they are universal approximators. (2) The performance surface is still quadratic, so a gradient descent learning algorithm does not suffer from local minima. Kernel affine projection algorithms (KAPA) [6] include the kernel least mean square (KLMS) [7,8] as the simplest element and kernel recursive least squares (KRLS) [9] as the most computationally demanding. Compared with other kernel adaptive filters, KLMS provides well-posedness solution with finite data [7], and is much simpler to implement in terms of both computational complexity and memory storage.

Although online kernel learning requires less computational cost per sample than batch methods, the system structure still grows with the number of processed data. To curb this growth, a variety of sparsification techniques have been proposed like the novelty criterion [10],...
the approximation linear dependency (ALD) criterion [9], sparsity constraints [11], coherence [12], the surprise criterion [13], etc. The main idea of these methods is to accept data that obeys a significance criterion into the center dictionary. Since we are dealing with online filters, the criterion chosen to discard the samples has to be online too, which creates added difficulty. More recently, quantization techniques were introduced to KLMS, yielding the quantized KLMS (QKLMS) algorithm [14]. Unlike conventional sparsification approaches, this new method does not purely discard the “redundant” data. It instead utilizes the information conveyed by these data to update the coefficients of the existing centers, and hence achieves a more compact network with better accuracy.

Although these sparsification methods curb the network size growth, they do not allow to fix in advance the exact network size $K$, i.e. establish a priori the model order as conventionally done in linear models. Fixed-budget kernel methods are crucial in practical applications, since DSP processors have limited physical memory and computational capability. In order to obtain a fixed budget model, budget maintenance strategies should be proposed, like pruning, projection, and merging. In this work, we mainly focus on pruning techniques, a series of strategies that discard the redundant units from large networks. When adopting the pruning strategies, three key problems need to be addressed: (1) Decide which center to be pruned. (2) How to deal with the nonstationarity. (3) How to decrease the influence of outliers. To solve these problems, various pruning techniques have been proposed and used not only in the field of kernel methods but also in pattern recognition and neural networks. However most of them have been developed for batch methods that are inappropriate for online learning in nonstationary environments. These pruning methods can be categorized into two groups: intuitive strategies and sensitivity analysis based methods [15]. Intuitive pruning methods are based on the direct relationship between coefficients or unit activation values. Removing the oldest pattern [16–19] and eliminating units that have the smallest magnitudes [20,21] are the two most common strategies. Sensitivity analysis based pruning methods apply a series of sensitivity measures to qualify the system’s perturbation after pruning. With orthogonal decomposition techniques, the data column that is less relevant can be removed from the radial basis function (RBF) network and a subset of regressor matrices that contain the essential time series information is preserved [22,23]. In addition, an important approach is to throw away the units resulting in a minimal influence on the system accuracy. In a classification scenario, removing the perceptron with the largest margin [24] is proposed and improved by tighter budget methods [25,26]. When it comes to the regression problem, throwing away the units resulting in minimal error increase with respect to the current training sample is widely utilized [27–29]. Due to the noise influence, only considering the pruning effect to the current training sample would result in inappropriate elimination. Therefore, in [30–32], the pruning strategies identified the centers with insignificant average contribution to the system accuracy over a set of training data. In [33], a relationship is shown to illustrate the connection between the required learning accuracy and the center contribution, through the a priori input sampling density [33]. Thus the data whose contribution is less than the learning accuracy is pruned during training. This algorithm is called Generalized Growing and Pruning RBF neural network (GGAP-RBF). Even though the pruning strategy in this GGAP-RBF algorithm is nonparametric, the prior over the input data distribution is not easy to obtain in practical cases. Going over the previously mentioned pruning techniques, we realize that, despite all the constituents (i.e. the intuitive pruning methods, the strategy considering the effect to the current sample, and the GGAP-RBF algorithm) being online, their accuracy performance is not good enough. While other methods are batch criteria which are unsuitable for KLMS.

The aforementioned pruning methods can be brought to kernel adaptive filters. Up to now, three different fixed-budget KRLS algorithms have been proposed. One uses a sliding window to discard the older training data and obtain a compact solution which only depends on the latest $K$ observed data [34,35]. The other two fixed-budget KRLS algorithms are obtained through omitting the least significant centers with the same pruning criterion mentioned in [29,36,37]. These two fixed-budget KRLS algorithms have shown good performance, not only in stationary conditions but also in nonstationary scenarios. However, because these two methods need the intermediate result of $(I+K)^{-1}$, where $K$ is the Gram matrix of the input data and $\lambda$ is the regularization term, they are not suitable for KLMS which does not need the Gram matrix. In effect, the computational cost to obtain the inverse Gram matrix is much more expensive than KLMS itself.

The aim of this paper is to seek a pruning technique for KLMS, which is online, computationally simple, and able to simultaneously solve three key problems mentioned above. The proposed method is somewhat similar to GGAP-RBF and the concept of significance of a center is adopted. However, the estimation strategy of significance is different. By utilizing the kernel density estimation technique (Parzen window) [38], the improved method approximates the input distribution online and nonparametrically. Furthermore, the exploitation of the influence factor lets the system track the nonstationary change and decay the influence of the noise outlier. So far, the computational complexity of significance is still heavy, scaling as $O(K^2)$, which motivates the use of a recursive approach. Such significance measure is estimated online and the computational complexity of the proposed method decreases to $O(K)$. Combined with the QKLMS, the proposed pruning method realizes a fixed-budget KLMS algorithm and is called the QKLMS-FB. At first, a new center will be added only if the distance between the data and the center dictionary is smaller than or equal to a predefined threshold [14]. When the network size reaches its predefined budget, the center with the smallest significance will be pruned with each new center.

The rest of this paper is organized as follows. After a brief overview of QKLMS in Section 2 and Section 3 introduces the definition of significance based on QKLMS and then the adopted estimation scheme is described in
Section 4. Two simulations are studied in Section 5 and finally, the conclusion remarks and future directions of research are listed in Section 6.

2. Quantized kernel least mean square algorithm

Consider the learning of a nonlinear function \( f : \mathbb{U} \rightarrow \mathbb{R} \) based on a known sequence \((u(1),d(1)),(u(2),d(2)), \ldots, (u(N),d(N))\), where \( \mathbb{U} \in \mathbb{R}^n \) is the input space (\( i \) is the input data dimension), \( u(i) \) is the system input at sample time \( i \), and \( d(i) \) is the corresponding desired response.

The QKLMS algorithm uses a simple vector quantization (VQ) algorithm to quantize the feature space, and then to curb the network size of the kernel adaptive filter. 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 codebook and network size remain unchanged, but the coefficient of the closest center will be updated. Otherwise, the new center will be inserted into the codebook. The QKLMS output with \( K \) centers for a new input vector \( u(i) \) is

\[
y(i) = \eta \sum_{j=1}^K a_j \kappa_{\sigma_j}(u(i),c_j) \tag{1}
\]

where \( a_j \) is the coefficient corresponding to the \( j \)-th center. If the center \( c_q \) is removed, the system output for \( u_i \) with the remaining centers is

\[
y_i = \eta \sum_{j=1}^{k-1} a_j \kappa_{\sigma_j}(u(i),c_j) + \eta \sum_{j=k+1}^K a_j \kappa_{\sigma_j}(u(i),c_j) \tag{2}
\]

Thus, the error resulting from removing \( c_q \) is given by

\[
E_k(i) = \|y(i) - \hat{y}(i)\|_q = \eta \|a_q \kappa_{\sigma_q}(u(i),c_q)\|_q \tag{3}
\]

where \( \| \cdot \|_q \) is the \( q \)-norm of vectors, and \( q=1 \) in this paper. Such that the average error for all sequentially learned observations caused by removing \( c_q \) is

\[
E_k^2 = \frac{q}{k} (|E_k(1)| + |E_k(2)| + \cdots + |E_k(i)|) \tag{4}
\]

The computational complexity of \( E_k(i) \) is high because it involves all learned observations. Therefore, a simpler and better method needs to be adopted to approximate \( E_k(i) \). As mentioned in [33], the probability density function is an option. Suppose that the input of the observation sequence are drawn from a domain \( \mathbb{U} \) with a sampling density function \( p(u) \). According to [33], we have

\[
\lim_{i \to +\infty} E_k(i) = \eta \|a_q\|_q \int_{\mathbb{U}} \kappa_{\sigma_q}(u,c_q) p(u) \, du \tag{5}
\]

This is the statistical contribution of a center \( c_q \). In practice, the distribution of \( u \) is unknown, so here we apply the popular kernel density estimation technique (Parzen window) in order to non-parametrically estimate the density function of the input data, which yields

\[
\hat{p}(u) = \frac{1}{n} \sum_{j=1}^n \kappa_{\sigma_p}(u,u(j)) \tag{6}
\]

where \( \kappa_{\sigma_p} \) is the Parzen kernel function with kernel bandwidth \( \sigma_p \). A range of kernel functions are commonly used for density estimation: uniform, triangular, normal, Epanechnikov, and others. Eq. (5) can be further written as

\[
E_k(i) = \frac{q}{k} \|a_q\|_q \int_{\mathbb{U}} \kappa_{\sigma_q}(u,c_q) \kappa_{\sigma_p}(u,u(j)) \, du \tag{7}
\]

In Eq. (7), information from all the input data is needed and further work should be done to compute the center contribution only with the available centers, i.e. in an online mode. Inspired by QKLMS, the contribution of \( u(i) \) to the input density distribution could be approximated.
by its nearest center (the details are explained in Section 4). Moreover, in nonstationary scenarios, the influence of $c_k$ on the system performance is time-varying, which means significance representation should track the input changes. Since $\eta$ does not influence the rank of $E_k(i)$, it is dropped in the expression. In order to facilitate the implementation of these properties, Eq. (7) is modified to

$$E_k(i) = \left| a_k \right| \sum_{j=1}^{K} \int_{\mathcal{U}} \lambda_k(i)k_{\sigma_k}(-u_k(i),c_k)k_{\sigma_c}(u,C_i) \, du$$

(8)

where $\lambda_k(i)$ is the influence factor for center $c_k$ at time $i$. This factor considers how many and when data are merged into $c_k$. Intuitively, the more data merged into $c_k$, the more important its contribution and the higher should be its significance. Moreover, the center that appears more recently has relatively a higher significance than older ones. In conclusion, $E_k(i)$ in Eq. (8) is the statistical contribution of center $c_k$ to the overall output which we define as the significance of $c_k$ and represent it as $S_k(i)$.

Eq. (8) gives an abstract picture of what and how these factors affect the significance measure of a center. Some interpretations are the following:

1. The significance measure is proportional to the magnitude of the data coefficient $|a_k|$. Recall that in KMLMs, the coefficients are associated with instantaneous errors. In QKLMs the weight difference of removing center $c_k$ is $\|\Delta w_k\|^2 = \left| a_k \right|^2 k_{\sigma_k}(c_k,c_k) = \left| a_k \right|^2$ if $k_{\sigma_k}$ is a translation invariance kernel, where $C$ is a constant. When $|a_k|$ is large, removing this center produces a large effect, and it impacts negatively the system performance.

2. A center with a large coefficient but small influence factor would have a relatively small significance because few data are merged into it. Since an outlier center results in a large coefficient, the incorporation of the influence factor will decrease the significance of this outlier till it is discarded later. In this sense, our pruning criterion is less sensitive to outliers and adaptively avoids outliers from the center dictionary.

3. The influence factor of a center decreases with time if there is no new data merged into this center. This property is useful in nonstationary environments. The significance of older centers reduces gradually till it is tiny enough to be omitted. This property also guarantees that the newest center would not be dropped improperly because of less data merged into it as compared with older ones.

Remark. If the distribution of the input data is uniform and the input statistics are stationary, the significance value is exactly proportional to the magnitude of the coefficient data. Therefore, our pruning rules degenerate to eliminating centers that have the smallest magnitudes.

4. Estimation of significance

An important aspect of any online algorithm is moderate computational complexity at every iteration. Therefore, we develop a recursive method to estimate significance and compute the influence factor, thus avoiding the integration operation in Eq. (8).

4.1. Influence factor $\hat{\lambda}_k(i)$

First, let us consider the problem of approximating the input density function with limited samples. Here we exploit the quantization idea: utilizing the nearest center of an input data to approximate its contribution to the input density distribution. That is,

$$\hat{p}_u(i) = \frac{1}{T} \sum_{j=1}^{K} k_{\sigma_k}(u,\mathcal{U}(j))$$

$$= \frac{1}{T} \sum_{j=1}^{K} k_{\sigma_k}(u,Q(u(j)))$$

(9)

where $Q(\cdot)$ is a quantization operator. For simplicity, the codebook and quantization rule for density estimation are the same as the quantization procedure in QKLMS.

Therefore, the input density function is updated according to Eq. (10), where $\hat{\lambda}_k(i-1)$ is the influence factor and represents how many samples are quantized to $c_k$. The significance rank, not its exact value, is what we take into consideration as mentioned before, hence the constant $1/i$ in Eq. (10) will be ignored:

$$\hat{\lambda}_k(i) = \begin{cases} \beta \hat{\lambda}_k(i-1) - \frac{1}{T} \sum_{j=1}^{K} \hat{\lambda}_k(i-1) k_{\sigma_k}(u,c_k) + \hat{\lambda}_k(i-1) k_{\sigma_k}(u,c_{\text{merge}}) & \text{if merge to } c_{\text{merge}} \\ \beta \hat{\lambda}_k(i-1) & \text{otherwise} \end{cases}$$

(10)

Second, a general approach to forget past samples is to include a forgetting factor $\beta$. $\beta$ is a positive scalar, usually close to one:

$$0 < \beta \leq 1$$

(11)

Based on the above analysis, the expression of the influence factor is

$$\hat{\lambda}_k(i) = \begin{cases} \beta \hat{\lambda}_k(i-1) & \text{if no merge to } c_k \\ \beta \hat{\lambda}_k(i-1) + 1 & \text{Otherwise} \end{cases}$$

(12)

where $\beta$ is the forgetting factor that helps the system track the input data statistic change. The smaller the value of $\beta$ value, the more influence the current data has. If a new center is added to the center dictionary, the initial value of $\lambda$ is 1.

4.2. Integration approximation

Eq. (8) involves an integration of the Parzen window density estimation in the sampling range $\mathcal{U}$. This can be done analytically for some simple but popularly used kernels, such as the Gaussian kernel. Our algorithm involves two kernels: the Mercer kernel for mapping to
the RKHS and the Parzen density kernel. For both, the Gaussian kernel is a good choice because it is universal, differentiable and continuous. Moreover, the convolution of two Gaussian kernel functions is a Gaussian function but with different mean and variance. This property simplifies the integration computation dramatically as shown below. For all these reasons, the Gaussian kernel function is a good choice to estimate the significance measure. Taking the normalized Gaussian kernel:

$$
\int_{\mathbb{R}^D} K_{\sigma_M}(\mathbf{u}, \mathbf{c}_k) K_{\sigma_M}(\mathbf{u}, \mathbf{c}_l) d\mathbf{u} \\
\quad = \int \exp \left( -\frac{(\mathbf{u}-\mathbf{c}_k)^2}{\sigma_M^2} - \frac{(\mathbf{u}-\mathbf{c}_l)^2}{\sigma_P^2} \right) d\mathbf{x} \\
\quad = \exp \left( -\frac{(\mathbf{c}_k-\mathbf{c}_l)^2}{\sigma_M^2 + \sigma_P^2} \right) \times \int \exp \left( -\frac{\sigma_M^2 + \sigma_P^2}{\sigma_M^2 \sigma_P^2} \frac{(\mathbf{u}-\frac{\mathbf{c}_k+\mathbf{c}_l}{2})^2}{\sigma^2} \right) d\mathbf{u} \\
\quad = \exp \left( -\frac{(\mathbf{c}_k-\mathbf{c}_l)^2}{2\sigma^2} \right) \times \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left( -\frac{(\mathbf{u}-\frac{\mathbf{c}_k+\mathbf{c}_l}{2})^2}{2\sigma^2} \right) \\
\quad \times \left( \frac{\sigma_M^2 + \sigma_P^2}{\sigma_M \sigma_P} \right)^{\frac{D}{2}}
$$

Practically, if $\sigma_M = \sigma_P = \sigma$, the integration can be written as

$$
\int_{\mathbb{R}^D} K_{\sigma_M}(\mathbf{u}, \mathbf{c}_k) K_{\sigma_M}(\mathbf{u}, \mathbf{c}_l) d\mathbf{u} \\
\quad = \exp \left( -\frac{(\mathbf{c}_k-\mathbf{c}_l)^2}{2\sigma^2} \right) \times \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left( -\frac{(\mathbf{u}-\frac{\mathbf{c}_k+\mathbf{c}_l}{2})^2}{2\sigma^2} \right) d\mathbf{u} \\
\quad = \frac{\sqrt{\pi}\sigma^2}{2} \exp \left( -\frac{(\mathbf{c}_k-\mathbf{c}_l)^2}{2\sigma^2} \right) \\
$$

This assumption is reasonable. Silverman’s rule [39] is a popular and widely accepted method for kernel selection, which has been practically used over a wide range of data sets. Many studies have actually compared other selection techniques with it [40–42]. The selection of the RKHS kernel size is still an open question. However, experiments have shown that the system performance is acceptable and not very sensitive to the kernel size in a range centered at the value given by Silverman’s rule [43]. If one has good reason to believe that other kernel functions, such as polynomial kernel, result in better fitting with higher accuracy, the formulation of Eq. (13) should be modified accordingly.

### 4.3. Recursive calculation

$$
E_k(i) = \beta E_k(i-1) + \mathbf{a}_k \int_{\mathbb{R}^D} K_{\sigma_M}(\mathbf{u}, \mathbf{c}_k) K_{\sigma_M}(\mathbf{u}, \mathbf{c}_l) d\mathbf{u}
$$

The original calculation of significance in Eq. (8) is time consuming, and scales by $O(K^2)$. Therefore, a recursive method is an option to speed up its calculation. Let $E_k(0) = 0$. According to the pruning and growing demand criterion, there are three conditions:

1. A new center $\mathbf{c}_{k+1}$ is added and there is room for the center dictionary to grow. In this condition we simply append the effect of the new sample to all the current centers:

$$
E_k(i) = \left\{ \begin{array}{ll}
\beta E_k(i-1) + \mathbf{a}_k \int_{\mathbb{R}^D} K_{\sigma_M}(\mathbf{u}, \mathbf{c}_k) K_{\sigma_M}(\mathbf{u}, \mathbf{c}_l) d\mathbf{u} & \text{if } k \neq f^* \\
\beta E_k(i-1) + \mathbf{a}_k + \eta e(i) & \text{if } k = f^*
\end{array} \right.
$$

(16)

2. If $\mathbf{u}(i)$ should be quantized to $\mathbf{c}_f$, the update process is as shown in Eq. (16).

3. When a center $\mathbf{c}_i$ is pruned, the influence of this center should be eliminated, yielding

$$
E_k(i) = E_k(i) - \mathbf{a}_k \int_{\mathbb{R}^D} K_{\sigma_M}(\mathbf{u}, \mathbf{c}_k) K_{\sigma_M}(\mathbf{u}, \mathbf{c}_l) d\mathbf{u}
$$

(17)

The exploitation of recursivity reduces the computational complexity of significance from $O(K^2)$ to $O(K)$ which is acceptable for the majority of kernel methods. Furthermore, although this method has to memorize all influence factors of existing centers and hence rises the memory requirement, the cost is limited when compared with conventional KLMS algorithm.

With the implementation of recursivity, our significance criterion is an online cost function. Unlike other batch criteria, for example those in [22–32], our significance measure is updated based on the previous value at each step other than recalculating through scanning all data.

The summary of the proposed fixed-budget QKLMS algorithm is presented in Algorithm 2.

**Algorithm 2.** Fixed-budget QKLMS algorithm.

*Initialization:* forgetting factor $\beta$, the fixed budget size $K$, quantization threshold $a_k > 0$, the stepsize $\eta$, center dictionary $\mathbf{C}(1) = \{\mathbf{u}(1)\}$ and coefficient vector $\mathbf{a}(1) = [\eta d(1)]$

**Computing**

while $\mathbf{u}(i)$, $d(i)$ is available do

**End while**

---

Fig. 1. Lorenz prediction time series with a gradual change at iteration 2500.
The Lorenz system is a dynamical system with chaotic flow, noted for its butterfly shape. The system is nonlinear, three-dimensional and deterministic. The following set of differential equations dictates how the state of the Lorenz system evolves over time in a complex, non-repeating pattern,

\[
\begin{align*}
\dot{x} &= \sigma(y-x) \\
\dot{y} &= -xz + \gamma x - y \\
\dot{z} &= xy - Bz
\end{align*}
\]

Two set of parameters are considered: one with \(\sigma = 10, \gamma = 28, B = \frac{3}{2}\), another with \(\sigma = 16, \gamma = 45.62, B = 4\). Using these two parameter sets, we generate two time series with sampling period of 0.01 s and length \(H_1\) and \(H_2\) respectively, which we concatenate to create a nonstationary time series with a rapid transition (Fig. 1). A bias component as shown in Fig. 2 is added to the series to make the problem more challenging. A zero-mean white Gaussian noise with variance 0.01 is added to the whole sequence. The problem setting for Lorenz short term prediction is as follows: the previous five points \(\mathbf{u}(i) = [x(i-5), \ldots, x(i-1)]^T\) are used as the input vector to predict the current value \(x(i)\) which is the desired response. In the simulations below, the kernel size is kept at \(\sigma = 0.707\), and the stepsize is set to \(\eta = 0.9\). Furthermore, the training MSE is calculated based on the mean square of the prediction error in a running window of 50 samples, for visual clarity.

In order to verify that the proposed methodology keeps a relatively good learning performance in stationary condition even though centers are pruned, we first

---

Fig. 2. Bias component added to the Lorenz prediction time series.
generate the training data using the second set of parameters. The convergence curves of the training MSE (averaged over 300 Monte-Carlo simulations) for QKLMS ($\gamma = 0.65^3$) and QKLMS-FB ($\gamma = 0.7, \beta = 0.96$) are plotted in Fig. 3(a). Although the accuracy of QKLMS-FB is slightly worse than the full QKLMS (QKLMS-FB final MSE: $-16.41$ dB, QKLMS final MSE: $-17.27$ dB), the dictionary size decreases from 36.81 to 28 (reduction ratio is 23.9%). Fig. 3(b) illustrates the influence of the fixed budget size on the final MSE. As expected, performance suffers when the budget size is smaller, but it approaches smoothly that of QKLMS when the dictionary size grows.

Next, we demonstrate the performance of QKLMS-FB under nonstationary condition and compare it with QKLMS and QKLMS-GGAP (pruning strategy with significance defined in the Generalized Growing and Pruning RBF). The simulation results over 200 Monte-Carlo runs are shown in Fig. 4. The budget of QKLMS-FB and QKLMS-GGAP is set to 20, and the parameters of QKLMS are set to achieve almost the same final dictionary size. The parameters are listed in Table 1. Several a priori distributions (Gaussian, Rayleigh and Exponential) are tested with QKLMS-GGAP and yield similar performance. In the following, only results using Gaussian distribution are shown. The difference between the actual input distribution and a prior-based distribution leads to the improper elimination of centers in QKLMS-GGAP, and worsens the performance as is clearly visible in Fig. 4(a). Because QKLMS-FB discards the old centers in $H_1$ after the change at iteration 2500, it obtains better tracking ability and

\[ S. Zhao et al. / Signal Processing 93 (2013) 2759–2770 \]

\[ 2765 \]

**Table 1** Parameters setting for different algorithms in Lorenz time series prediction.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Same MSE</th>
<th>Same network size</th>
</tr>
</thead>
<tbody>
<tr>
<td>QKLMS</td>
<td>$\gamma = 0.9$</td>
<td>$\gamma = 1.35$</td>
</tr>
<tr>
<td>QKLMS-FB</td>
<td>$\gamma = 1.0$</td>
<td>$\gamma = 1.0$</td>
</tr>
<tr>
<td>QKLMS-GGAP</td>
<td>$\beta = 0.9$</td>
<td>$\beta = 0.9$</td>
</tr>
</tbody>
</table>

$^3 \gamma = \frac{\mu}{\sigma}$ is the quantization factor [14].
faster convergence speed in $H_2$. Furthermore, owing to a larger quantized factor to achieve similar final dictionary size, the performance of QKLMS is worse than QKLMS-FB.

In Fig. 5, the performance of QKLMS-FB, QKLMS-GGAP, QKLMS and KLMS are compared. This time the parameters are selected to yield almost the same final MSE\(^4\) (except QKLMS-GGAP, since it does not work well in nonstationary conditions). In order to achieve the same final MSE as QKLMS-FB, QKLMS requires a smaller quantization factor, which speeds up the dictionary growth (see Fig. 5). Fig. 6 shows the influence of the fixed budget size on the final MSE in nonstationary data. In this example, our simulation results confirm that QKLMS-FB behaves much better than the others, achieving either higher accuracy or smaller dictionary size.

5.2. Nonlinear channel equalization

Consider the nonlinear channel consisting of a linear filter and a memoryless nonlinearity, as shown in Fig. 7. This nonlinear channel model is named as Wiener model and has been used to model digital communications system and digital magnetic recording channels. A series of binary signals, $s_i \in \{+1, -1\}$, are the channel input. At the receiver, the signal is corrupted by additive noise $n(i)$. The channel equalization problem can be formulated as a regression problem, by setting the pairwise input–output as $[(r(i), r(i+1), ..., r(i+l)), s(i-D))$, where $l$ is the time-embedding length and $D$ is the equalization time lag. $l=5$ and $D=2$ are set in this simulation. The nonlinear channel model is defined by $x(i) = s(i) + 0.5s(i-1), r(i) = a(x(i) - 0.9x(i)^2) + n(i)$, where $n(i)$ is a $-20$ dB white Gaussian noise. $a=3.0$ in the first 1000 samples (area $H_1$) and

---

\(^4\)The final MSE is the steady MSE after input statistical property changes.
Table 2
The final MSE (dB) comparison of different algorithms in channel equalization. The parameters of the algorithms are chosen as to produce almost the same final network size.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Abrupt change</th>
<th>$\gamma$</th>
<th>$\gamma$</th>
<th>$\gamma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>QKLMS</td>
<td>101</td>
<td>166.5</td>
<td>188.3</td>
<td>190.8</td>
</tr>
<tr>
<td>QKLMS-FB</td>
<td>64</td>
<td>84</td>
<td>84</td>
<td>84</td>
</tr>
</tbody>
</table>

Table 3
The final network size comparison of different algorithms in channel equalization. The parameters of the algorithms are chosen as to produce almost the same final MSE in each condition.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Abrupt change</th>
<th>$\gamma$</th>
<th>$\gamma$</th>
<th>$\gamma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>QKLMS</td>
<td>101</td>
<td>166.5</td>
<td>188.3</td>
<td>190.8</td>
</tr>
<tr>
<td>QKLMS-FB</td>
<td>64</td>
<td>84</td>
<td>84</td>
<td>84</td>
</tr>
</tbody>
</table>

Table 4
Parameters setting in channel equalization to achieve almost the same final network size in each condition.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Abrupt change</th>
<th>$\gamma$</th>
<th>$\gamma$</th>
<th>$\gamma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>QKLMS</td>
<td>2.7</td>
<td>3.2</td>
<td>3.4</td>
<td>3.4</td>
</tr>
<tr>
<td>QKLMS-FB</td>
<td>2.0</td>
<td>2.0</td>
<td>2.0</td>
<td>2.0</td>
</tr>
<tr>
<td>QKLMS-GGAP</td>
<td>0.98</td>
<td>0.95</td>
<td>0.92</td>
<td>0.9</td>
</tr>
</tbody>
</table>

Table 5
Parameters setting in channel equalization to achieve almost the same MSE in each condition.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Abrupt change</th>
<th>$\gamma$</th>
<th>$\gamma$</th>
<th>$\gamma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>QKLMS</td>
<td>1.5</td>
<td>1.7</td>
<td>1.6</td>
<td>1.6</td>
</tr>
<tr>
<td>QKLMS-FB</td>
<td>2.0</td>
<td>2.0</td>
<td>2.0</td>
<td>2.0</td>
</tr>
<tr>
<td>QKLMS-GGAP</td>
<td>0.98</td>
<td>0.95</td>
<td>0.92</td>
<td>0.9</td>
</tr>
</tbody>
</table>

Fig. 9. Performance comparison for QKLMS, QKLMS-FB and QKLMS-GGAP for the channel equalization problem under the condition of abrupt change. (a) Convergence curves in terms of the training MSE. (b) Network size evolution curves.
decreases to 2.0 with different change rates (area $H_{12}$, in this paper change rate is defined as $1/\text{length}(H_{12})$) then fixed to $a = 2.0$ (area $H_2$). Fig. 8 represents one realization of $r(i)$ with the change rate of being 1/500. In this example, the kernel size is set to $\sigma = 0.707$, and the stepsize to $\eta = 1.0$.

5.2.1. Comparison in different change rate conditions

We first compare the performances of QKLMS, QKLMS-FB, QKLMS-GGAP under different change rate conditions: abrupt change, $1/500$, $1/2500$ and $1/5000$ respectively. In Tables 2 and 3, the parameters are selected such that all of the algorithms yield almost the same final network size or final MSE for each condition. Fig. 9 visually shows the result under abrupt change condition. The specific parameters setting are listed in Tables 4 and 5.

These results indicate again that QKLMS-FB is superior in all conditions with better fitting performance or fewer centers. In Table 2, QKLMS-FB’s performance is stable, which implies that it has good tracking ability under different change rate conditions. Table 3 shows that the network size of QKLMS increases with the transition time, because there are more important centers to preserve the filter accuracy. On the other hand, QKLMS-FB does not struggle with this problem because it discards the older centers in a time learning–varying environment.

5.2.2. Outlier influence

We now verify the performance of QKLMS-FB in a non-Gaussian noise situation and use a mix-Gaussian density to simulate the influence of outliers. The noise distribution is

$$p_{\text{noise}}(i) = 0.95\mathcal{N}(0,0.03) + 0.05\mathcal{N}(0.4,0.03)$$  \hspace{1cm} (19)

Table 6 shows parameters setting and final MSE for different algorithms with almost the same final network size (QKLMS: 85.2, QKLMS-FB and QKLMS-GGAP: 84). This result indicates that QKLMS-FB still obtains good performance under the influence of outlier noise.

5.2.3. How the three free parameters, $\gamma$, $\beta$ and $K$, affect the performance of QKLMS-FB

Setting the rate change to $1/500$ for example and fixing the network size budget at 84, we plot the final training MSE versus the quantization factor $\gamma$ and forgetting factor $\beta$ (Fig. 11). From the plot we observe that these parameters affect the system performance heavily. We still do not have a systematic approach to select these parameters based on the data, but can advance some observations: (a) under a fixed-budget limitation, small and large $\gamma$ both result in low accuracy. A too small quantization factor does not yield good performance. Thus the limited

![Fig. 10. The change of the number of centers allocated during $H_1$ for channel equalization problem.](image)
dictionary cannot provide enough detail of the signal structure to provide good identification. On the other hand, a large quantization factor decreases the steady-state EMSE of the quantization procedure, which also decreases the performance of QKLMS-FB. (b) Selecting the forgetting factor depends on how the data statistics change. If the statistical property changes frequently, a smaller forgetting factor is preferred. Otherwise, a large forgetting factor brings good performance ($\beta = 1$ in stationary condition). A small forgetting factor forces the current data to have more influence on the significance measure. However, a too small forgetting factor makes the system sensitive to outliers. In conclusion, setting the forgetting factor is a tradeoff between outlier rejection and tracking ability.

6. Conclusion

Since practical computing devices have only limited physical memory and computational capacity, fixed-budget kernel methods are crucial in real world applications. In this paper, we presented a new efficient pruning and growing strategy in designing a fixed-budget QKLMS system. To constrain its network size, we proposed a discarding criterion, significance, inspired by GGAP-RBF, but our method is more practical because it does not need a priori knowledge about the input distribution. Moreover, online and recursive calculations result in a reasonable and efficient updating strategy to carry on the computation in real time. As we show theoretically and experimentally, the proposed significance measure successfully quantifies the importance of a center with respect to the overall learning system, and keeps the system structure under a predefined size particularly in nonstationary condition.

Although the QKLMS-FB results in a good performance, determining the appropriate network size remains an important issue. So far, the systematic relationship between the budget size, free parameters such as quantization factor, and system accuracy is unknown. Such relationship would help us choose appropriate parameters to fix the budget (model order) to meet a predetermined accuracy. In our opinion, the rate distortion theory is a powerful tool to analyze this problem because it provides the theoretical foundation for lossy data compression. As such, the model selection problem would be translated in a data compression problem. This will be the topic of a future study.

References
