**Online Efficient Learning with Quantized KLMS and L₁ Regularization**

Badong Chen, Songlin Zhao, Sohan Seth, Jose C. Principe

Computational NeuroEngineering Laboratory, University of Florida, Gainesville, USA

chenbd04@mails.tsinghua.edu.cn, principe@cnel.ufl.edu

**Abstract—** In a recent work, we have proposed the quantized kernel least mean square (QKLMS) algorithm, which is quite effective in online learning sequentially a nonlinear mapping with a slowly growing radial basis function (RBF) structure. In this paper, in order to further reduce the network size, we propose a sparse QKLMS algorithm, which is derived by adding a sparsity inducing $l₁$ norm penalty of the coefficients to the squared error cost. Simulation examples show that the new algorithm works efficiently, and results in a much sparser network while preserving a desirable performance.

**Keywords—** online learning, kernel adaptive filtering, QKLMS, $l₁$ norm penalty.

I. INTRODUCTION

Kernel adaptive filters [1] belong to the family of online learning algorithms. They are developed in reproducing kernel Hilbert spaces (RKHS), by using the linear structure of this space to implement well-established linear adaptive algorithms and to obtain nonlinear filters in the input space. Among these algorithms the kernel least mean square (KLMS) is the simplest ($O(i)$ complexity), which is easy to implement, yet effective for learning complex systems [2]. When the kernel is Gaussian, the KLMS naturally creates a growing radial-basis function (RBF) network, where the weights are directly related to the errors at each sample.

The main bottleneck of the KLMS (as well as other kernel adaptive filters) is the linear growing structure with each new sample, which poses both computational as well as memory issues especially for continuous adaptation scenarios. To address this problem, several sparsification techniques have been proposed to curb the growth of the network, where the redundant input data are purely discarded. Existing sparsification criteria for data selection include the novelty criterion [3], approximate linear dependency (ALD) criterion [4], surprise criterion [5], etc. In a recent work [6], we proposed a quantization approach to constrain the network size and developed the quantized kernel least mean square (QKLMS) algorithm. The QKLMS is a first step in yielding a compact network with desirable performance, whose network size depends upon the volume of the input space and the quantization coarseness.

However, the analyses of the QKLM filter shows that many of the center coefficients are still small, which indicates that the method can be further improved. In this work, we propose a more principled approach to curb the growth of the filter coefficients by including a sparseness criterion into the QKLMS algorithm. This new algorithm is derived by adding a $l₁$ norm penalty on the center coefficients to the squared error cost function. The $l₁$ relaxation favors sparsity and has been widely used in compressive sensing literature due to its convexity. The proposed algorithm can thus produce a much sparser network, in which many coefficients are negligibly small. Pruning these negligible centers yields a much more compact network. We stress here that the $l₁$ norm penalty has been incorporated into the cost function of the adaptive algorithm, which is totally different from the conventional sparsification criteria (such as the novelty criterion). The sparsity inducing penalty terms (e.g. $l₀$ or $l₁$ norms) have already been successfully used in linear adaptive filtering algorithms, such as the least mean square (LMS) and the recursive least squares (RLS) [7-9]. These sparsity regularized adaptive algorithms are shown to be highly efficient for sparse system identification.

The rest of paper is organized as follows. In section II, we briefly introduce the QKLMS algorithm. In section III, we derive the proposed algorithm. Section IV presents the simulation results and section V gives the conclusion.

II. QKLMS

Given a sequence of training data $\{ u(i), d(i) \} , \ i = 1, 2, \cdots$, where $u(i) \in \mathbb{R}^m$ is the $m$-dimensional input vector, $d(i) \in \mathbb{R}$ is the desired output, our goal is to find an estimator of the underlying mapping $d = f(u)$ sequentially based on the last estimate and current data. The kernel methods are powerful in learning a nonlinear mapping. By kernel methods, one can transform the input data into a high dimensional feature space $\mathbb{F}$ through a mapping $\varphi$ induced by a Mercer kernel $\kappa(u, u')$, and then perform the online learning in this feature space. A commonly used kernel is the Gaussian kernel

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

where $\sigma > 0$ is the kernel size (or kernel bandwidth). The inner product between the transformed input data (feature...
vectors) satisfies \( \langle \varphi(u), \varphi(u') \rangle = \kappa(u, u') \). This simple calculation is known as the “kernel trick”.

The KLMS algorithm is essentially the LMS algorithm in feature space. Denote \( \varphi(i) = \varphi(u(i)) \), the KLMS algorithm can be expressed as [2]
\[
\begin{align*}
\Omega(0) &= 0 \\
e(i) &= d(i) - \langle \Omega(i-1), \varphi(i) \rangle \\
\Omega(i) &= \Omega(i-1) + \eta e(i) \varphi(i)
\end{align*}
\]
where \( e(i) \) is the prediction error at iteration \( i \), \( \eta \) is the step-size, and \( \Omega(i) \) denotes the estimate of the weight vector in feature space. If identifying \( \varphi(u) = \kappa(u, u) \), one obtains the learning rule in original input space:
\[
\begin{align*}
f_0 &= 0 \\
e(i) &= d(i) - f_{i-1}(u(i)) \\
f_i &= f_{i-1} + \eta e(i) \kappa(u(i), u)
\end{align*}
\]
where \( f_i \) denotes the learned mapping at iteration \( i \).

Quantizing the feature vector \( \varphi(i) \) in the weight update equation \( \Omega(i) = \Omega(i-1) + \eta e(i) \varphi(i) \), we obtain the QKLMS algorithm [6]:
\[
\begin{align*}
\Omega(0) &= 0 \\
e(i) &= d(i) - \langle \Omega(i-1), \varphi(i) \rangle \\
\Omega(i) &= \Omega(i-1) + \eta e(i) \mathcal{Q} \varphi(i)
\end{align*}
\]
where \( \mathcal{Q} [\cdot] \) denotes the quantization operator in feature space.

Since the dimensionality of the feature space is very high (can be even infinite), the quantization is usually performed in the original input space, in this case the learning rule becomes
\[
\begin{align*}
f_0 &= 0 \\
e(i) &= d(i) - f_{i-1}(u(i)) \\
f_i &= f_{i-1} + \eta e(i) \kappa(Q[u(i)], u)
\end{align*}
\]
where \( Q[\cdot] \) denotes the quantization operator in input space.

By QKLMS, the learned mapping before iteration \( i \) is
\[
f_{i,t}(x) = \sum_{j=1}^{\text{size}(C(i-1))} \alpha(i-1) \kappa(C_j(i-1), x)
\]
where \( C_j(i-1) \) denotes the \( j \)th element (code-vector) of the codebook \( C(i-1) \), \( \alpha(i-1) \) is the coefficient of the \( j \)th center \( \kappa(C_j(i-1), \cdot) \). The learned weight vector in feature space can then be written as
\[
\Omega(i-1) = \Phi(i-1) \alpha(i-1)
\]
where \( \alpha(i-1) = [\alpha(i-1), \cdots, \alpha_M(i-1)] \) denotes the coefficient vector, \( M \) is the number of the centers (\( M = \text{size}(C(i-1)) \)), and \( \Phi(i-1) = [\varphi(C_1(i-1)), \cdots, \varphi(C_M(i-1))] \).

A simple online quantization method is proposed in [6] and is described as follows:

**Online quantization**

**Initialization:** Choose the quantization size \( \epsilon \geq 0 \), and initialize the codebook \( C(1) = \{ u(1) \} \).

**Computation:**

1. Compute the distance between \( u(i) \) and \( C(1-1) \):
\[
dis(u(i), C(i-1)) = \min_{1 \leq j \leq \text{size}(C(i-1))} \| u(i) - C_j(i-1) \|
\]
2. If \( dis(u(i), C(i-1)) \leq \epsilon \), keep the codebook unchanged: \( C(i) = C(i-1) \), and quantize \( u(i) \) into the closest code-vector: \( Q[u(i)] = C_j(i-1) \), where
\[
\hat{j} = \arg \min_{1 \leq j \leq \text{size}(C(i-1))} \| u(i) - C_j(i-1) \|
\]
3. Otherwise, update codebook: \( C(i) = [C(i-1), u(i)] \), and quantize \( u(i) \) as itself: \( Q[u(i)] = u(i) \). 

**End while**

III. QKLMS WITH L1 NORM REGULARIZATION

In QKLMS the cost function is the squared error \( \| e(i) \|^2 \). In order to yield sparser center coefficients, we define a new cost by combining the squared error and the \( l_1 \) norm penalty of the coefficient vector:
\[
L = \frac{1}{2} \| e(i) \|^2 + \gamma \| \alpha(i-1) \|_1
\]
where \( \gamma > 0 \) is a weight factor to balance the \( l_1 \) penalty and the estimation error.

First we express the \( l_1 \) penalty as
\[
\| \alpha(i-1) \|_1 = \sum_{j=1}^{M} | \alpha_j(i-1) |
\]
\[
= \alpha(i-1) \begin{bmatrix} \text{sign}(\alpha(i-1)) \end{bmatrix}^T
= \alpha(i-1) \begin{bmatrix} K(i-1) K(i-1)^{-1} \end{bmatrix} \begin{bmatrix} \text{sign}(\alpha(i-1)) \end{bmatrix}
= \begin{bmatrix} \Phi(i-1) \alpha(i-1) \end{bmatrix}^T \begin{bmatrix} \Phi(i-1) K(i-1)^{-1} \text{sign}(\alpha(i-1)) \end{bmatrix}
= \begin{bmatrix} \Omega(i-1) \Phi(i-1) K(i-1)^{-1} \text{sign}(\alpha(i-1)) \end{bmatrix}
\]

\( \text{sign}(\alpha(i-1)) \)
where $K(i-1) = \Phi(i-1)\Phi(i-1)^T$ is an $M \times M$ Gram matrix, $\text{sign}(\alpha(i-1))$ is the component-wise sign vector of $\alpha(i-1)$. One can then derive the gradient-based weight update equation in feature space:

$$\Omega(i) = \Omega(i-1) - \frac{\partial L}{\partial \Omega(i-1)}$$

$$= \Omega(i-1) + \eta\epsilon(i)\frac{\partial \epsilon(i)}{\partial \Omega(i-1)} - \rho\frac{\partial \|\alpha(i-1)\|}{\partial \Omega(i-1)}$$

$$= \Omega(i-1) + \eta\epsilon(i) - \rho\Phi(i-1)K(i-1)^{-1}\text{sign}(\alpha(i-1))$$

where $\rho = \eta\gamma$. The mapping update equation in original input space will be

$$f_i = f_{i-1} + \eta\epsilon(i)\kappa(u(i),\cdot) - \rho\psi(i-1)K(i-1)^{-1}\text{sign}(\alpha(i-1))$$

Quantizing the current input $u(i)$ we obtain the QKLMS with $l_i$ norm regularization (namely the sparse QKLMS):

$$f_i = f_{i-1} + \eta\epsilon(i)\kappa(Q[u(i)],\cdot) - \rho\psi(i-1)K(i-1)^{-1}\text{sign}(\alpha(i-1))$$

When we adopt the previous online quantization method, we have the following coefficients update rule:

a) If $\text{dis}(u(i),C(i-1)) > \varepsilon$

$$\alpha(i) = \frac{\alpha(i-1) - \rho K(i-1)^{-1}\text{sign}(\alpha(i-1))}{\eta\epsilon(i)}$$

b) If $\text{dis}(u(i),C(i-1)) \leq \varepsilon$

$$\alpha(i) = \alpha(i-1) + \eta\epsilon(i)\xi_{j^*} - \rho\Phi(i-1)^{-1}\text{sign}(\alpha(i-1))$$

where $j^* = \arg\min_{j=1,\ldots,M} |u(i) - C_j(i-1)|$. $\xi_j$ is an $M$-dimensional vector whose $j$th element is 1 and all other elements are 0.

In the algorithm (12), the inverse matrix $K(i-1)^{-1}$ can be calculated recursively [4]. In practice, in order to avoid singularity, we can add a regularization parameter $\lambda$, and calculate $(K(i-1) + \lambda I)^{-1}$ instead of $K(i-1)^{-1}$.

Denote $P(i) = (K(i) + \lambda I)^{-1}$. The recursive algorithm for computing $P(i)$ will be:

a) If $\text{dis}(u(i),C(i-1)) > \varepsilon$

$$P(i) = r(i)^{-1}\begin{bmatrix} P(i-1)r(i) + z(i)z(i)^T - z(i)z(i)^T & -z(i)z(i)^T \\ -z(i)z(i)^T & 1 \end{bmatrix}$$

In the following, we give a summary of the proposed sparse QKLMS algorithm.

### Sparse QKLMS

**Initialization:**

Choose step-size $\eta$, kernel size $\sigma$, quantization size $\varepsilon$, weight factor $\gamma$, parameter $\lambda$, and initialize the codebook and coefficient vector: $C(i) = \{u(1)\}$, $\alpha(i) = [\eta\epsilon(1)]$.

**Computation:**

While $\{u(i),d(i)\} (i > 1)$ available do

1) Compute the prediction error:

$$e(i) = d(i) - \sum_{j=1}^{\infty} a_j(i)\kappa(C_j(i-1),u(i))$$

2) Compute the distance between $u(i)$ and $C(i-1)$:

$$\text{dis}(u(i),C(i-1)) = \min_{C_j(i-1)} |u(i) - C_j(i-1)|$$

3) If $\text{dis}(u(i),C(i-1)) \leq \varepsilon$, then update the coefficient vector $\alpha$ using (14), and let $C(i) = C(i-1)$, $P(i) = P(i-1)$.

Else

Update the coefficient vector $\alpha$ using (13), update the codebook $C(i) = \{C(i-1),u(i)\}$, and update

$$P(i) = P(i-1)$$

End if

End while

**Remark:** It is easy to observe that if there is no quantization ($\varepsilon = 0$), the computational complexity of this new algorithm is just the same as that of the kernel recursive least squares (KRLS) [4]. With quantization, however, the computational burden of the proposed algorithm is usually much less than that of the KRLS, because in this case the network size will be reduced dramatically.

### IV. SIMULATION RESULTS

Now we present simulation examples to demonstrate the performances of the proposed sparse QKLMS algorithm.
A. Static function estimation

In the first example, we consider the case in which the underlying mapping is a mix-Gaussian function, and the desired output is related to the input via

\[
d(i) = \exp\left(\frac{(ui(i)-5.0)^2}{20}\right) + 0.2\exp\left(\frac{(ui(i)+5.0)^2}{20}\right) + \nu(i)
\]

where \(\nu(i)\) is an additive noise independent of the input. In the simulations below, the noise is Gaussian-distributed with variance 0.0001, the input data are uniformly distributed over \([-10,10]\).

First, we set the kernel size as \(\sigma = 1.0\), such that the underlying mapping is a very sparse mapping since it can be expressed as a linear combination of only two centers. The quantization size \(\varepsilon\) and the step-size \(\eta\) are respectively 0.2 and 0.5. For the sparse QKLMS algorithm, the weight factor \(\gamma\) is set as \(\gamma = 0.000002\), and the regularization parameter \(\lambda\) is set as \(\lambda = 0.0001\). The convergence curves (in terms of the excess mean square error, EMSE), averaged over 200 Monte Carlo runs, are shown in Fig. 1. Evidently both the QKLMS and sparse QKLMS algorithms achieve almost the same convergence performance. However, the sparse QKLMS can produce a sparser network, of which many coefficients are negligibly small (very close to zero). This can be clearly seen from Fig. 2, where the coefficient values for each center are plotted. If we prune those centers whose coefficients are smaller than 0.01, the network size will reduce to only seven. Fig. 3 illustrates the learned mapping by sparse QKLMS with and without pruning compared with the desired mapping. One can see pruning the negligible centers has little effect on the learned mapping.

Next, we perform another simulation with the same settings. This time we set the kernel size as \(\sigma = 0.5\), such that the sparsity of the underlying mapping decreases. In this case, as shown in Fig. 4, the sparse QKLMS algorithm can still yield many negligible centers.
B. Lorenz Time Series Prediction

The second example is about short-term chaotic time series prediction. Consider the Lorenz chaotic system [6]:

\[
\begin{align*}
\frac{dx}{dt} &= -\beta x + yz \\
\frac{dy}{dt} &= \delta (z - y) \\
\frac{dz}{dt} &= -xy + \rho y - z
\end{align*}
\]  

(18)

where parameters $\beta$, $\delta$, $\rho$ are set as $\beta=8/3$, $\delta=10$ and $\rho=28$. The state $x$ is used for short-term prediction task.

The signal is preprocessed to be zero-mean and unit-variance, followed by the addition of a white Gaussian noise with variance 0.0001. The goal is to predict the value of the current sample using the previous five consecutive samples.

In this example, the parameters in QKLMS are set as $\sigma=1.0$, $\varepsilon=0.3$ and $\eta=0.5$. For the sparse QKLMS, the weight factor equals $\gamma=0.00001$, and the regularization parameter is $\lambda=0.0001$. Fig. 5 shows the average convergence curves (in terms of MSE) over 100 Monte Carlo simulations. Once again, the QKLMS and the sparse QKLMS yield almost the same convergence performance. The histograms of the coefficients values (at final iteration) are illustrated in Fig. 6.

As expected, the sparse QKLMS produces a sparser network, since its coefficients are much more concentrated around zero. The network size evolution curve (which depends only on the quantization size) is shown in Fig. 7. The final network size lies between 90 and 100. The network size can be further reduced by pruning the negligible centers. Table 1 presents the testing MSE (over 100 test data) and network size with and without pruning (the threshold for pruning is set at 0.1). We observe that, with pruning the sparse QKLMS yields a very compact model that consists of only 21 centers, while with original QKLMS, the pruned network size equals 50. Moreover, the sparse QKLMS can achieve smaller testing MSE.

<table>
<thead>
<tr>
<th></th>
<th>Testing MSE</th>
<th>Network Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>QKLMS</td>
<td>0.0089±0.0123</td>
<td>94</td>
</tr>
<tr>
<td>QKLMS (pruned)</td>
<td>0.0188±0.0189</td>
<td>50</td>
</tr>
<tr>
<td>Sparse QKLMS</td>
<td>0.0080±0.0112</td>
<td>94</td>
</tr>
<tr>
<td>Sparse QKLMS (pruned)</td>
<td>0.0122±0.0173</td>
<td>21</td>
</tr>
</tbody>
</table>

Table. 1. The testing MSE and network size with and without pruning.
V. CONCLUSION

By incorporating a $l_1$ norm penalty of the coefficients into the cost function, we derive a sparse QKLMS algorithm. Simulation results show the proposed algorithm can produce many negligible centers, which can be pruned with little loss in performance.

Compared with the QKLMS, the new algorithm has two extra free parameters: the weight factor $\gamma$, and the regularization parameter $\lambda$. Usually both parameters should be set to relatively small values. The determination of the best values, however, depends on the application and needs more analytical and numerical study.

In future studies, an online pruning method should be developed with this new algorithm. The $l_1$ norm penalty can also be incorporated into the KRLS algorithm, and can be extended to include other sparsity inducing penalties, such as the reweighted $l_1$ norm.

REFERENCES