Density-dependent Quantized Kernel Least Mean Square

Bao Xi¹, Lei Sun², Badong Chen¹, Jianji Wang¹, Nanning Zheng¹, José C. Príncipe¹
¹School of Electrical and Information Engineering, Xi’an Jiaotong University, Xi’an, PR China, 710049
²School of Information and Electronics, Beijing Institute of Technology, Beijing, PR China, 100081
E-mail address: xibao525@163.com (B. Xi), sunlei@ntu.edu.sg (L. Sun),
chenbd@mail.xjtu.edu.cn (B. Chen), wangjianji@mail.xjtu.edu.cn (J. Wang),
nnzheng@mail.xjtu.edu.cn (N. Zheng), principe@cnel.ufl.edu (J. Príncipe).

Abstract—Kernel least mean square is a simple and effective adaptive algorithm, but dragged by its unlimited growing network size. Many schemes have been proposed to reduce the network size, but few takes the distribution of the input data into account. Input data distribution is generally important in view of both model sparsification and generalization performance promotion. In this paper, we introduce an online density-dependent vector quantization scheme, which adopts a shrinkage threshold to adapt its output to the input data distribution. This scheme is then incorporated into the quantized kernel least mean square (QKLMS) to develop a density-dependent QKLMS (DQKLMS). Experiments on static function estimation and short-term chaotic time series prediction are presented to demonstrate the desirable performance of DQKLMS.

Index Terms—Kernel adaptive filter, Quantized kernel least mean square, Density-dependent, Vector quantization.

I. INTRODUCTION

Kernel methods, which use Mercer kernels, Gaussian kernel at most of the time, to map low dimensional input space to high dimensional feature space to get the rich expressiveness, achieve great success in both batch kernel learning, such as support vector machine [1] [2], kernel principle component analysis [3], and online kernel learning (OKL) [4]. OKL learns the target function incrementally, which is particularly suitable for situations where the data arrives sequentially.

Kernel adaptive filters [5] are among the most celebrated subfield of OKL, which include kernel least mean square (KLMS) [6] [7], kernel affine projection algorithms (KAPAs) [8], kernel recursive least squares (KRLS) [9], extended KRLS [10], kernel adaptive filtering with maximum correntropy criterion (KMC) [11] and kernel recursive maximum correntropy (KRMC) [12], etc. However, a common problem for kernel adaptive filters is that the size of their network keeps growing with the continuously coming of new data. Unlimited growth of network size poses both computational and memory challenges. What’s more, large network size also leads to overfitting, making algorithms sensitive to outliers. Both computational burden and overfitting call for imposing sparseness on kernel adaptive filters.

So far, a variety of sparsification techniques have been applied to kernel adaptive filters to curb the growth of network size. These techniques use sparsification criteria to accept only important input data as new centers. Typical sparsification criteria include novelty criterion [13], prediction variance criterion [14], surprise criterion [15], and approximate linear dependency [9]. These criteria can reduce network size of kernel adaptive filters dramatically. However, these methods purely discard the redundant input data without using its information to update the coefficients of the current centers. Consequently, it takes more input data to train an accurate model. Platt proposed RANs [13], which uses redundant input to update all the coefficients of existing centers while keep network structure unchanged. Updating all the coefficients at every iteration is computational expensive and is also sensitive to initial parameter setting and outliers. A novel quantization scheme was proposed in [16] to curtail the network size and was applied to quantized kernel adaptive filters, including quantized least mean square (QKLMS) [16], Quantized kernel least mean square with desired signal smoothing (QKLMS-S) [17], modified quantized kernel least mean square (MQKLMS) [18] and quantized kernel recursive least squares (QKRLS) [19]. These algorithms reduce the network size significantly, and even get a better performance than KLMS.

Apart from the network size, the input data representation also influences the generalization performance. The relationship between the input data density and prediction error has been studied in [20]. In QKLMS, the centers are approximately uniformly distributed in the input domain. It is shown in many works [21] [22] that the input data probability density dependent learning algorithms can improve the generalization performance significantly. In [23], A density-dependent vector quantization scheme (DQS) was applied to LS-SVM, which can attain good performance on large data sets. Motivated by similar idea, we incorporate DQS with QKLMS [16] in this work to develop a density dependent QKLMS, called DQKLMS hereafter.

The organization of the rest of this paper is as follows. QKLMS is briefly reviewed in section II. In section III, we discuss the influence of the distribution of centers to the generalization performance of algorithm. Subsequently, an input density-dependent quantization scheme is applied to QKLMS to develop DQKLMS. In section IV, simulations on static function estimation and chaotic time-series prediction are presented. Finally, a conclusion is given in section V.
II. QKLMS

The goal of adaptive filters is to learn a continuous input-output mapping \( f : U \rightarrow \mathbb{R} \) based on a sequence of input-output pairs \( \{u(i), d(i), i = 1, 2, \ldots, N\} \). \( U \) is the input space and is assumed as a subspace of \( \mathbb{R}^m \). The output is assumed to be one-dimensional but it is easy to generalize to multi-dimensional applications.

The least mean square (LMS) [24] is an incremental linear algorithm, which uses the following procedure to update the weight vector

\[
\begin{align*}
    w(0) &= 0 \\
    e(i) &= d(i) - w(i-1)^T u(i) \\
    w(i) &= w(i-1) + \eta e(i) u(i)
\end{align*}
\]

where \( e(i) \) represents the prediction error at iteration \( i \), \( \eta \) is the step size of stochastic gradient descend, and \( T \) denotes transpose operation. At iteration \( i \), given an input data \( u_* \), the output of LMS is \( \hat{f}(u_*) = w(i)^T u_* \). \( \hat{f} \) is a linear estimator of the target mapping \( f \). Since the LMS estimator is a linear combination of the input samples, it could fail if the target mapping is highly nonlinear.

To overcome the limitation of linearity but to preserve the simplicity of LMS, the kernel-induced mapping \( \varphi \) is employed to transform the input \( u(i) \) into a high-dimensional feature space \( F \) [6] as \( \varphi(u(i)) \). The inner product in space \( F \) is

\[
\varphi(u)^T \varphi(u') = \kappa(u, u')
\]

where \( \kappa \) is a Mercer kernel. When Gaussian kernel with kernel width \( \sigma \) is adopted, the inner product in feature space \( F \) is expressed as [25]

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

Mapping \( \{u(i)\} \) to \( \{\varphi(u(i))\} \) and then applying LMS in the high-dimensional feature space \( F \), then we get the learning rule of KLMS

\[
\begin{align*}
    f(0) &= 0 \\
    e(i) &= d(i) - f_{i-1}(u(i)) \\
    f_i &= f_{i-1} + \eta e(i) \kappa(u(i), \cdot).
\end{align*}
\]

As we can see that the network size of KLMS is as large as the size of the training set, which brings heavy computational burden and memory requirement.

QKLMS prunes the redundant centers in KLMS by using a simple vector quantization scheme (called primitive quantization scheme (PQS) [23] hereafter) to compress the input data set \( \{u(i)\} \) to a quantization dictionary \( \{c(i)\} \), and reduces the network size significantly. At every iteration, an input is accepted as a new center in the quantization dictionary if and only if the distances between the input and all existing centers in the quantization dictionary are larger than a predefined fixed quantization threshold, else just update the coefficient of the closest center. In this situation, the learning rule for QKLMS is

\[
\begin{align*}
    f(0) &= 0 \\
    e(i) &= d(i) - f_{i-1}(u(i)) \\
    f_i &= f_{i-1} + \eta e(i) \kappa(Q[u(i)], \cdot).
\end{align*}
\]

where \( Q[\cdot] \) is a quantization operator in input domain and the output of \( Q[u(i)] \) is \( u(i) \) itself or the closest center. As a result of the fixed quantization threshold, the distribution of the centers in the dictionary is approximately uniform in the region of input data. Uniform distribution implicates the quantization output of dense regions are the same with sparse regions, which fails to reflect the distribution of input data.

III. DENSITY-DEPENDENT QKLMS

In this section, the relationship between input data representation and generalization performance is discussed. Based on this discussion, we introduce a density-dependent quantization scheme (DQS) [23] and incorporate it into QKLMS to develop an input density-dependent quantization kernel least mean square (DQKLMS).

The test square error on a test sequence, which subjects to the same distribution as the training set, is defined as

\[
E_{te} = \int p(x)(f(x) - \hat{f}(x))^2 dx = \int p(x)\varepsilon(x) dx
\]

where \( \hat{f}(x) \) denotes an estimation of the desired mapping \( f(x) \) and \( \varepsilon(x) \) denotes the square error with respect to input \( x \). In order to find the distribution of \( \varepsilon(x) \) that minimize the expectation of test square error, we define a intermediate measure as

\[
\zeta(x) = \exp(-\varepsilon(x)).
\]

This measure is negatively correlated to the square error \( \varepsilon(x) \). We assume \( \int \zeta(x) dx < \infty \) is bounded and we further assume that \( \zeta(x) \) is normalized by \( \int \zeta(x) dx \) such that \( \zeta(x) \) satisfies \( \int \zeta(x) dx = 1 \). Therefore, the test error can be rewritten as

\[
E_{te} = \int p(x)\varepsilon(x) dx = -\int p(x) \log \zeta(x) dx.
\]

This expression get the minimum if and only if \( p(x) = \zeta(x) \) holds almost everywhere. Since \( \zeta(x) \) is negatively correlated with \( \varepsilon(x) \), thus \( \varepsilon(x) \) is inversely correlated with \( p(x) \). Therefore, in order to get a small \( E_{te} \), \( \varepsilon(x) \) should be small in dense regions where \( p(x) \) are large while in sparse regions \( \varepsilon(x) \) can be relatively larger (s.t. \( \int \zeta(x) dx = 1 \)).

Generally, the more centers are located in a region, the smaller the test square error will be in there. As analyzed in last section, the distribution of centers is approximately uniform in input domain in QKLMS. Therefore, the magnitude of test error of QKLMS is approximately equivalent in different regions of input domain. We can expect that \( E_{te} \) will decrease largely if the test error drops greatly in the dense regions of input domain. Consequently, more centers should be placed in the dense regions.

There are two methods to fulfil this goal. The one is choosing a smaller quantization threshold in overall input domain,
the other is choosing smaller quantization thresholds in dense regions only while keeping relatively larger ones in sparse regions. The first one will get a better performance at the expense of larger computational burden. However, compared to dense region, the sparse region is less important. Thus, the second one is more practical. Since the distribution of centers is controlled by the quantization threshold, the quantization threshold should vary corresponding to the probability density of the input data. In DQS [23], each center has its own quantization threshold, which is adjusted adaptively by a shrinkage threshold so that the quantization thresholds in dense regions are smaller than those in sparse regions.

The output of PQS and DQS are shown in Fig.1, in which the cardinalities of the two dictionaries are tuned to 68 by adjusting the quantization threshold and the shrinkage threshold. The distribution of centers in (a) is approximately uniform, while the distribution in (b) is closer to the distribution of input data (blue dots). Therefore, DQS can represent the input data distribution roughly.

Replacing PQS in QKLMS by DQS and keeping other parts in QKLMS unchanged, we obtain a modified QKLMS, called density-dependent QKLMS (DQKLMS). The pseudocode is given in Algorithm 1.

### Algorithm 1: DQKLMS

**Input:** \( \{ u(i) \in U, d(i) \} \), \( i = 1, 2, \ldots, N \)

**Output:** \( C_N, a_N \)

**Initialize:** learning rate \( \eta > 0 \), kernel width \( \sigma > 0 \), initial dictionary \( C_1 = \{ c_1 \} \), initial quantization threshold \( \varepsilon_0 \geq 0 \), \( c_1 = u_1 \), \( a_1 = [\eta d(1)] \), \( l = [1] \), \( \varepsilon = \varepsilon_0 \)

for \( i = 2 \) To \( N \) do

\[
q^* = \arg\min_{1 \leq j \leq \text{size}(C_{i-1})} \| u_i - c_j \|_2
\]

\[
y_i = \sum_{j=1}^{\text{size}(C_{i-1})} a_j(i-1) \kappa(C_{i-1}(j), u_i)
\]

\[
e_i = d_i - y_i
\]

if \( \| u_i - c_{q^*} \|_2 > \varepsilon q^* \) then

\[
C_i = \{ C_{i-1}, u_i \}, a_i = [a_{i-1}; \eta e_i], \varepsilon = [\varepsilon; \varepsilon_0], l(q^*) = 1, l = [l; 1].
\]

else if \( \| u_i - c_{q^*} \|_2 > \frac{\varepsilon q^*}{2} \) and \( l(q^*) > \delta \) then

\[
C_i = \{ C_{i-1}, u_i \}, a_i = [a_{i-1}; \eta e_i], \varepsilon q^* = \frac{\varepsilon q^*}{2}, l(q^*) = 1, l = [l; 1].
\]

else

\[
a_i(q^*) = a_{i-1}(q^*) + \eta e_i,
\]

\[
l(q^*) = l(q^*) + 1.
\]

end

end

In Algorithm 1, \( C_i \) and \( a_i \) denote the quantization dictionary at the \( i \)th iteration and the corresponding coefficients of centers respectively. The vector \( l \) contains \( \text{size}(C) \) elements, which records the number of input data falling into the quantization region of each center. Only in two circumstances can an input be accepted as a center to the dictionary. In the first case, if the minimum distance between input and centers exceeds the corresponding element of the quantization threshold vector \( \varepsilon \), then this input will be added to the dictionary as a new center and the quantization threshold of this new center is set at \( \varepsilon_0 \). In the second case, even though the minimum distance is smaller than the quantization threshold of the closest center, it is larger than half of this threshold and the corresponding element in \( l \) exceeds the shrinkage threshold \( \delta \). Then this input is included into the dictionary and the quantization threshold of the closest center shrinks to half of its original value, which is also assigned to the new added center as quantization threshold at the same time. DQKLMS also uses stochastic gradient descent to update the coefficients of centers.

In DQS, the initial quantization threshold \( \varepsilon_0 \) and shrinkage threshold \( \delta \) are two important parameters. Generally, the initial quantization threshold in DQS is set slightly larger than that in PQS, and set the shrinkage threshold at about 1% of the size of the total training data [23].
IV. SIMULATION RESULTS

In this section, two experiments are implemented to compare the performance of KLMS, QKLMS, and DQKLMS on static function estimation and short-term chaotic time series prediction. We conduct Monte-Carlo simulations to get the average results of these three algorithms.

A. Static function estimation

The adopted static function is 2-dimentional sinc function, and the training data is generated by the following method:

\[ d_i = \text{sinc}(u_i) + v_i, \]  

where the input sequence \( \{u_i\} \) subjects to a 2-dimensional Gaussian distribution which centered at origin. The noise sequence \( \{v_i\} \) is 0 mean 0.1 standard variance white Gaussian noise. Testing data are generated in the same way but with no noise sequence added. In all experiments, a Gaussian kernel is adopted and its kernel width is set at \( \sigma = 0.7 \), the learning step in these three algorithms is set at \( \eta = 0.5 \).

In simulations, the size of training set is 1000, and the testing MSE is calculated on the basis of 500 noise-free test data in every iteration. The final network size and final testing MSE are calculated as an average over 100 Monte-Carlo simulations. The parameter settings are listed in table I below. The performance comparisons between QKLMS and DQKLMS in static function estimation are presented in Fig.2 and Fig.3. In Fig.2, the parameters are chosen such that QKLMS and DQKLMS produce almost the same network size. Due to the larger initial quantization size, DQKLMS converges slower than QKLMS. However, after 300 iterations,
DQKLMS gets a better steady performance than QKLMS. In Fig.3, QKLMS and DQKLMS produce almost the same MSE. We can see from Fig.3 (b) that the network size of DQKLMS is about 100 less than that of QKLMS. From Fig.2 and Fig.3, it can be observed that the performance of DQKLMS is better than that of QKLMS, as DQKLMS uses less centers to achieve a similar steady state error or uses almost the same amount of centers but gets a lower steady state error.

B. Short-Term Chaotic Time-Series Prediction

![Convergence curves in terms of the testing MSE](image)

![Network size evolution curves](image)

Fig. 4. Comparison between QKLMS and DQKLMS in short-term chaotic sequence prediction.

Chaotic time series has been widely used as a benchmark data for sequential prediction. The adopted chaotic sequence is generated from the following differential equations [16]:

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

(10)

where the parameters are set at \(\beta = 8/3\), \(\gamma = 10\), and \(\rho = 28\). \(x\) is used for short-term prediction task after preprocessed to zero mean and unit variance. The problem setting for short-term Chaotic time series prediction is as follows: the previous 10 points are used as input vector \(u(i) = [x(i - 10), x(i - 9), \ldots, x(i - 1)]\) to predict the current value \(x(i)\). The training data is also added by a 0 mean 0.1 standard variance Gaussian white noise while the test data is noise free. In this experiment, the kernel width is set at \(\sigma = 0.707\) and the learning rate is set at \(\eta = 0.2\). In QKLMS, the quantization threshold is set at \(\varepsilon = 0.2\). In DQKLMS, the initial quantization threshold is set at \(\varepsilon_0 = 0.33\) and the shrinkage threshold is set at \(\delta = 5\).

The performance comparisons between KLMS, QKLMS, and DQKLMS in short-term chaotic time series prediction are presented in Fig.4, where all the simulation results are averaged over 100 Monte-Carlo simulations with different segments of \(x\). As is shown in Fig.4 (a) that DQKLMS converges slightly slower than QKLMS and KLMS, however, its steady-state error is the smallest. In Fig.4 (b), we can see that the network size of DQKLMS is about 100 less than that of QKLMS. Therefore, DQKLMS uses the least kernels but attains the best generalization performance.

V. Conclusion

Regarding sparse representations of the input samples using quantization techniques, recent studies showed that a kernel learning method could achieve desirable performance if the distribution of the quantization centers is consistent with the input data distribution. It was shown that the output of PQS, which is used by quantized kernel least mean square (QKLMS), were uniformly distributed in the input domain, because of its fixed quantization threshold. In this work, an online input data density-dependent quantization scheme (DQS) was introduced to fix this problem and was incorporated into QKLMS. With the advantage of DQS over PQS, DQKLMS achieved better performance than KLMS and QKLMS. Experiments on static function estimation and chaotic time-series prediction validated the good performance of the new algorithm.

ACKNOWLEDGMENT

This work was supported by 973 Program (No. 2015CB351703) and National NSF of China (No. 61372152).

REFERENCES


