A FIXED-BUDGET QUANTIZED KERNEL LEAST MEAN SQUARE ALGORITHM

Songlin Zhao, Badong Chen and José C. Príncipe

University of Florida, Electrical and Computer Engineering Department, Gainesville, Florida, 32611

ABSTRACT

We present a quantization-based kernel least mean square (QKLMS) algorithm with a fixed memory budget. In order to deal with the growing support inherent in online kernel methods, the proposed method utilizes a growing and pruning combined technique and defines a criterion, significance, based on weighted statistical contribution of a data center. This method doesn't need any apriori information and its computational complexity is acceptable, linear with the center number. As we show theoretically and experimentally, the introduced algorithm successfully quantifies the least 'significant' datum and preserves the most important ones resulting in less system error.

Index Terms— Kernel method, Quantized kernel least mean square, Fixed budget, Growing and pruning

1. INTRODUCTION

During the last few years, there has been more and more attention on the kernel learning methods which utilize Mercer kernels to map the data from the input space to a high-dimensional feature space. These methods achieve powerful classification and regression performance in complicated nonlinear problems. The successful examples of kernel methods including support vector machines [1], kernel principal component analysis (Kernel PCA) [2], kernel least mean square error (KLMS) [3], kernel recursive least square algorithm (KRLS) [4] among others. However, kernel methods have an obvious bottleneck: the system structure grows linearly with the number of processed data. This drawback results in high computational burden as well as memory especially for continuous scenarios which hindered so far online applications.

To deal with the linear growing structure problem, a variety of sparsification techniques have been proposed to restrain growth: the novelty criterion [5], approximation linear dependency (ALD) criterion [4] and the surprise criterion [6], etc. Recently, quantization techniques have been introduced to KLMS, called quantized KLMS (QKLMS) [7]. Unlike conventional sparsification approaches, this new method doesn't purely discard the 'redundant' data. It utilizes the so-called 'redundant' data to update the coefficient of the closest center, and hence achieves a more compact network with better accuracy. Even though these sparsification algorithms curb the growth of the system structure, they do not allow us to fix in advance the exact network size especially in non-stationary learning scenarios.

There are few methods have been proposed to solve the fixed-budget problem. Generalized growing and pruning (G-GAP) [8], sliding-window methods [9], the forgetron [10] and fixed-budget KRLS [11] combine growing and pruning procedures to memorize the most 'important' data and discard the 'unimportant' data.

In this paper, we propose a growing and pruning method based on QKLMS. Inspired by [8], the concept of *significance* of a center data is introduced so as to realize a fixed-budget QKLMS algorithm, called QKLMS-FB. Significance of a center gives a measure of the important content about the system to be learned and is defined as the contribution made by that center to the network output weighted averaged over the input data received so far. Different from [8], the proposed criterion doesn't require any aprior information and is more practical especially in nonstationary condition. QKLMS-FB achieves a sparse structure and after the network size reaches its upper bound, the center with the smallest significance will be discarded. Moreover, a recursive approach decreases the computation complexity of significance to linearity with the center number.

The rest of this paper is organized as follow. After a brief overview of QKLMS in section 2, section 3 introduces the definition of significance based on QKLMS and then describes a simple estimation scheme. The results are reported in section 4 and, finally, the conclusion remarks and future lines of research are listed in section 5.

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 $(\boldsymbol{u}(1), d(1)), (\boldsymbol{u}(2), d(2)), \ldots, (\boldsymbol{u}(N), d(N)) \in \mathbb{Z}^N$ where $\boldsymbol{u}(i)$ 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 compact the input space, and then to curb the network size of the kernel adaptive filter. Every time a new sample arrives, the QKLMS quantizes it according to the

This work was partially supported by NSF grant ECCS 0856441.

defined distance between the new sample and the quantized input space. If the current quantized input sample has already been allocated to an existing center, the network size remains unchanged, but the coefficient of that center will be still updated. For online kernel adaptive methods, most of existing VQ algorithms are not suitable because the codebook is usually trained based on an offline data set and the computational burden is heavy. Therefore, a very simple online VQ method in which the Euclidean distance in the input space is the quantization rule is proposed. In conclusion, the summary of the QKLMS algorithm with online VQ is as follows:

Algorithm 1 QKLMS Algorithm

Initialization: stepsize η, kernel width $\sigma_{_M}$, quantization threshold $\varepsilon_{_{\mathbb{U}}} > 0$, center dictionary $\boldsymbol{C}(1) = \{\boldsymbol{u}(1)\}$ and coefficient vector $\boldsymbol{a}(1) = [\eta d(1)]$
while $\{\boldsymbol{u}(1), d(1)\}$ is avaiable **do**
$e(i) = d(i) - \sum_{j=1}^{K(i-1)} \boldsymbol{a}_j(i-1)\kappa_{\sigma_{_M}}(\boldsymbol{u}(i), \boldsymbol{C}_j(i-1))$
$dis(\boldsymbol{u}(i), \boldsymbol{C}(i-1)) = \min_{1\le j\le K(i-1)} \|\boldsymbol{u}(i) - \boldsymbol{C}_j(i-1)\|$
$j^* = \arg\min_{1\le j\le K(i-1)} \|\boldsymbol{u}(i) - \boldsymbol{C}_j(i-1)\|$
if $dis(\boldsymbol{u}(i), \boldsymbol{C}(i-1)) \le \varepsilon_u$ **then**
$\boldsymbol{C}(i) = \boldsymbol{C}(i-1)$, $\boldsymbol{a}_{j^*}(i) = \boldsymbol{a}_{j^*}(i-1) + \eta e(i)$
else
$\boldsymbol{C}(i) = \{\boldsymbol{C}(i-1)\boldsymbol{u}(i)\}$, $\boldsymbol{a}(i) = [\boldsymbol{a}(i-1), \eta e(i)]$
end if
end while

(where $\kappa_{\sigma_{_M}}$ is the Mercer kernel controlled by kernel size $\sigma_{_M}$, $\boldsymbol{C}_j(i-1)$ and $K(i-1)$ are the jth element and size of $\boldsymbol{C}(i-1)$ respectively, and $\|.\|$ denotes the Euclidean norm in the input space.)

The sufficient condition for mean-square convergence, and a lower and upper bound on the theoretical value of the steady-state excess mean square error (EMSE) are discussed in [7].

3. DEFINITION AND ESTIMATION OF SIGNIFICANCE OF CENTERS

This section first introduces the concept of significance of a center based on a weighted average contribution over all input data, even though some of them are quantized to existing centers. An important aspect of an online algorithm is moderate computational complexity at every iteration. Therefore, a recursive method to estimate significance is also developed. To simplify the notation, we always refer to time index i, and therefore express $\boldsymbol{C}_j(i-1)$, $\boldsymbol{a}_j(i-1)$ and $K(i-1)$ as $\boldsymbol{c}_j$, $\boldsymbol{a}_j$ and K.

3.1. Definition of Significance

The output of the QKLMS for a new input vector $\boldsymbol{u}(i)$ is:

$$y(i) = \eta \sum_{j=1}^{K} \boldsymbol{a}_j \kappa_{\sigma_{_M}}(\boldsymbol{u}(i), \boldsymbol{c}_j) \tag{1}$$

If the center $\boldsymbol{c}_k$ is removed, the output of the system with remaining centers for $\boldsymbol{u}_i$ is

$$\hat{y}(i) = \eta \sum_{j=1}^{k-1} \boldsymbol{a}_j \kappa_{\sigma_{_M}}(\boldsymbol{u}(i), \boldsymbol{c}_j) + \eta \sum_{j=k+1}^{K} \boldsymbol{a}_j \kappa_{\sigma_{_M}}(\boldsymbol{u}(i), \boldsymbol{c}_j) \tag{2}$$

Thus, the error resulting from removing $\boldsymbol{c}_k$ is given by

$$E_{k,i}(i) = \|y(i) - \hat{y}(i)\|_1 = \|\boldsymbol{a}_k\| \kappa_{\sigma_{_M}}(\boldsymbol{u}(i), \boldsymbol{c}_k) \tag{3}$$

where $\|.\|_1$ is the 1-norm of vectors. Therefore, the average error for all sequentially learned observation cased by removing c_k is

$$\begin{aligned} E_k(i) &= \frac{1}{i} \left\| (E_k(1), E_k(2) \ldots, E_k(i))^T \right\|_1 \\ &= \frac{1}{i} \|\boldsymbol{a}_k\| \sum_{j=1}^{i} \kappa_{\sigma_{_M}}(\boldsymbol{u}(j), \boldsymbol{c}_k) \end{aligned} \tag{4}$$

Suppose that the input of the observation sequence are drawn from a domain $\mathbb{U}$ with a sampling density function $p(\boldsymbol{u})$. According to [8], we have

$$\lim_{i \to +\infty} E_k(i) = \|\boldsymbol{a}_k\| \int_{\mathbb{U}} \kappa_{\sigma_{_M}}(\boldsymbol{u}, \boldsymbol{c}_k) p(\boldsymbol{u}) d\boldsymbol{u} \tag{5}$$

In practice, the distribution of $\boldsymbol{u}$ is unknown. Here we propose to apply the popular kernel density estimation (Parzen window) to non-parametrically estimate the probability density function of a random variable, to yield,

$$\hat{p}(\boldsymbol{u}) = \frac{1}{i} \sum_{j=1}^{i} \kappa_{\sigma_{_P}}(\boldsymbol{u}, \boldsymbol{u}_j) \tag{6}$$

where $\kappa_{\sigma_{_P}}$ is the kernel function with kernel size $\sigma_{_P}$. Therefore, Equ (4) can be further written as

$$E_k(i) = \frac{\|\boldsymbol{a}_k\|}{i} \sum_{j=1}^{i} \int_{\mathbb{U}} \kappa_{\sigma_{_M}}(\boldsymbol{u}, \boldsymbol{c}_k) \kappa_{\sigma_{_P}}(\boldsymbol{u}, \boldsymbol{u}_j) d\boldsymbol{u} \tag{7}$$

In QKLMS, some of the input data are quantized to the center dictionary $\boldsymbol{C}$ and merely a part of data are memorized. Moreover, the value of i doesn't influence the rank of $E_k(i)$ and the decision which center will be pruned. Hence, Equ (7) is inappropriate to QKLMS and must be modified into:

$$E_k(i) = \|\boldsymbol{a}_k\| \sum_{j=1}^{K} \int_{\mathbb{U}} \lambda_k(i) \kappa_{\sigma_{_M}}(\boldsymbol{u}, \boldsymbol{c}_k) \kappa_{\sigma_{_P}}(\boldsymbol{u}, \boldsymbol{c}_j) d\boldsymbol{u} \tag{8}$$

where $\lambda_k(i)$ is the influence factor for center $\boldsymbol{c}_k$ at time i and includes the number and the time of data merged into $\boldsymbol{c}_k$. $E_k(i)$ is the statistical contribution of the center $\boldsymbol{c}_k$ to the

overall output, and thus we define this as the significance of $\boldsymbol{c}_k$ and represent it as $E_k^{sig}(i)$.

Influence factor is very important for pruning, especially in nonstationary situations. It should take into account the information of how many data are quantized into a specific center when a new data appears in this neighborhood. Therefore, the expression of influence factor is

$$\lambda_k(i) = \begin{cases} \beta\lambda_k(i-1) & \text{If no merge for } \boldsymbol{c}_k \\ \beta\lambda_k(i-1) + \kappa(\boldsymbol{u}_i, \boldsymbol{c}_k) & \text{otherwise} \end{cases} \tag{9}$$

where $0 \leq \beta \leq 1$ is the memory factor which helps the system track the input data statistic change. The smaller the β value, the more influence the current data has.

3.2. Estimation of significance

The original calculation of significance in part 3.1 is time consuming, and scales by $O(K^2)$. Therefore, some appropriate simplification should be applied to speed up its calculation: a recursive method is an option. According to the pruning and growing demand, there are three conditions:

1) A new center $\boldsymbol{c}_{K+1}$ is added and the size of center dictionary can still grow.

$$E_k(i) = \beta E_k(i-1) + \|\boldsymbol{a}_{K+1}\| \int_{\mathbb{U}} \kappa_{\sigma_M}(\boldsymbol{u}, \boldsymbol{c}_k)\kappa_{\sigma_P}(\boldsymbol{u}, \boldsymbol{c}_{K+1})d\boldsymbol{u} \tag{10}$$

2) If $\boldsymbol{u}(i)$ should be quantized to $\boldsymbol{c}_{j^*}$, the update process is as shown in Equ (11).

3)when an center $\boldsymbol{c}_L$ is pruned, the influence of this center should be eliminated, yielding,

$$E_k(i) = E_k(i) - \|\boldsymbol{a}_k\| \lambda_L(i-1) \int_{\mathbb{U}} \kappa_{\sigma_M}(\boldsymbol{u}, \boldsymbol{c}_k)\kappa_{\sigma_P}(\boldsymbol{u}, \boldsymbol{c}_L)d\boldsymbol{u} \tag{12}$$

Most interestingly, if the Mercer kernel κ_{σ_M} and density estimation kernel κ_{σ_P} both are the normalized gaussian kernel, the integral is easy to estimate. And in this paper, for simplification, we assume $\sigma_M = \sigma_P = \sigma$. Such that, for example,

$$\begin{aligned} &\int_{\mathbb{U}} \kappa_{\sigma_M}(\boldsymbol{u}, \boldsymbol{c}_k)\kappa_{\sigma_P}(\boldsymbol{u}, \boldsymbol{c}_L)d\boldsymbol{u} \\ &= \int_{\mathbb{U}} exp(-\frac{(\boldsymbol{u}-\boldsymbol{c}_k)^2}{\sigma_M^2} - \frac{(\boldsymbol{u}-\boldsymbol{c}_L)^2}{\sigma_P^2})dx \\ &= exp(-\frac{(\boldsymbol{c}_k-\boldsymbol{c}_L)^2}{\sigma_M^2+\sigma_P^2}) \times \int_{\mathbb{U}} exp(-\frac{\sigma_M^2+\sigma_P^2}{\sigma_M^2\sigma_P^2} \\ &\quad \times (\boldsymbol{u} - \frac{\boldsymbol{c}_k + \frac{\sigma_M^2}{\sigma_P^2}\boldsymbol{c}_L}{1+\frac{\sigma_M^2}{\sigma_P^2}})^2)d\boldsymbol{u} \\ &\xrightarrow{\sigma_M=\sigma_P=\sigma} \sqrt{\frac{\pi\sigma^2}{2}} exp(-\frac{(\boldsymbol{c}_k-\boldsymbol{c}_L)^2}{2\sigma^2}) \end{aligned} \tag{13}$$

4. SIMULATION

Consider the Lorenz chaotic system. We generate two time systems, one with $\sigma = 10, \gamma = 28, B = \frac{8}{3}$, another one with $\sigma = 16, \gamma = 45.62, B = 4$. The x dimension is selected to build two time series with a sampling period of 0.01s. 3000 samples are segmented from the two sequences respectively and a DC component with magnitude 1 is added to the second sequence. Then concatenate them to create an example of a nonstationary time series. The problem setting for short term prediction is as follows: the previous five points $\boldsymbol{u}(i) = [x(i-5), ..., x(i-1)]^T$ are used as the input vector to predict the current value $x(i)$ which is the desired response here. In the simulations below, the kernel size is set as $\sigma = 0.707$, and the stepsizes involved are set as $\eta = 0.9$. The superiority of QKLMS has already been proved in [7]. Therefore, merely the performance comparisons between QKLMS-FB and QKLMS is shown here. We also did simulation for QKLMS with significance defined in [8], called QKLMS-GGAP. Table 1 lists the specific parameters setting. Several conventional distributions (Gaussian, Rayleigh and Exponential) were tested in QKLMS-GGAP and have similar performances. Fig.3 merely shows the performance of Gaussian distribution.

Even though the performance of QKLMS-FB is slightly worse (QKLMS: -17.7dB, QKLMS-FB: -17.82dB) in stationary conditions, it decreases the network size from 41 to 32 (network size reduction is 22%), as shown in Fig.1. Fig.2 compares the network size of QKLMS and QKLMS-FB with respect to the same final MSE. Because QKLMS-FB discards the old data in the first zone after the change at iteration 3000, it obtains better tracking ability and faster convergence speed in the second zone. In order to achieve the same MSE at iteration 6000, QKLMS requires the smaller quantization factor, which speeds up the network size increment. In Fig.3, all of three methods have the same network size, 31, and the performance of QKLMS-FB is superior. The obvious difference between conventional distributions, like Gaussian, and the actual input distribution leads to the impropriety of center discard in QKLMS-GGAP and the divergence in the second zone.

Table 1: Parameters setting for different algorithms

	Stationary	Same MSE	Same network size
QKLMS	$\gamma^* = 0.65$	$\gamma = 0.56$	$\gamma = 1$
QKLMS-FB	$\gamma = 0.65$	$\gamma = 0.74$	$\gamma = 0.74$
	$\beta = 0.998$	$\beta = 0.998$	$\beta = 0.998$
QKLMS-GGAP			$\gamma = 0.74$

*$\gamma = \varepsilon_u/\sigma$ is the quantization factor [7] .

$$E_k(i) = \begin{cases} \beta E_k(i-1) + \kappa(\boldsymbol{u}_i, \boldsymbol{c}_{j^*}) \|\boldsymbol{a}_k\| \int_{\mathbb{U}} \kappa_{\sigma_M}(\boldsymbol{u}, \boldsymbol{c}_k)\kappa_{\sigma_P}(\boldsymbol{u}, \boldsymbol{c}_{j^*})d\boldsymbol{u} & k \neq j^* \\ \frac{\|\boldsymbol{a}_k + e(i)\|}{\|\boldsymbol{a}_k\|}\beta E_k(i-1) + \kappa(\boldsymbol{u}_i, \boldsymbol{c}_{j^*}) \|\boldsymbol{a}_k + e(i)\| \int_{\mathbb{U}} \kappa_{\sigma_M}(\boldsymbol{u}, \boldsymbol{c}_k)\kappa_{\sigma_P}(\boldsymbol{u}, \boldsymbol{c}_{j^*})d\boldsymbol{u} & k = j^* \end{cases} \tag{11}$$

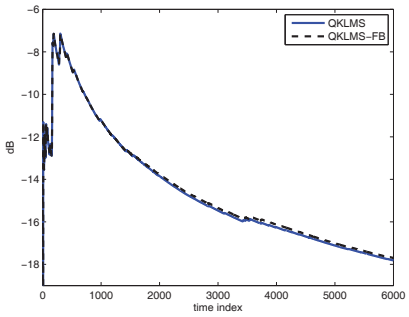
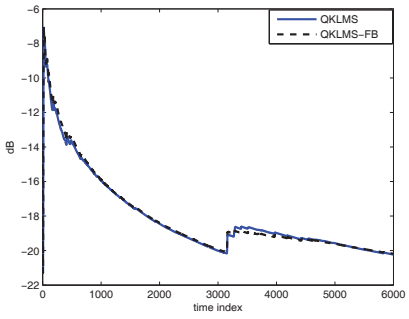
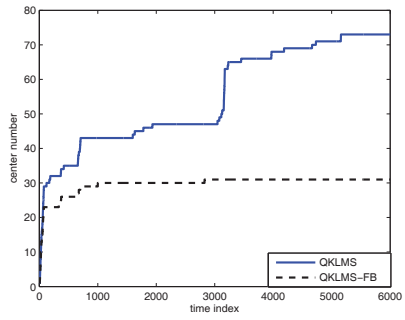


Fig. 1: Performance comparison in stationary conditon. All of 6000 data belong to the first time series.



(a) Learning curves



(b) Network size evolution curves

Fig. 2: Performance comparison for the same MSE

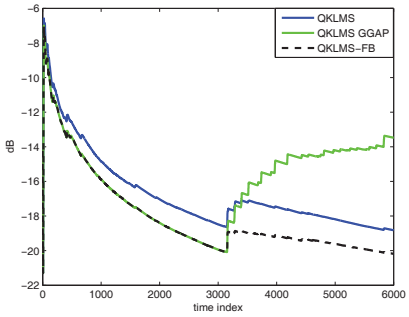


Fig. 3: Learning curves comparision on the same network size

5. CONCLUSION

Fixed-budget kernel methods are crucial in online applications, since practical computing devices have limited physical memory. In this paper, we present a new efficient pruning and growing strategy in designing a QKLMS-FB system. To maintain its network size, we utilize a discarding criterion called significance, first introduced in GGAP-RBF [8], but our implementation is more practical because of smaller computational burden and does not require any apriori information.

Although the QKLMS-FB obtains good performance, the relation between the budget upper bound, free parameters, such as quantization factor, and system performance are still unknown and will be subject to further research. These relations will help us choose appropriate parameters to fix the center number and is useful to determine the effect of the budget upper bound on the expect accuracy.

6. REFERENCES

[1] V. Vapnik, *The nature of statistical learning theory*, Springer, New York, 1995.

[2] B. Scholkopf, A. Smola, and K. Muller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Compuatation*, vol. 10, no. 5, pp. 1299–1319, 1998.

[3] W. Liu, P. Pokharel, and J. Principe, "The kernel least mean quare algorithm," *IEEE Transactions on Signal Processing*, vol. 56, pp. 543–554, 2008.

[4] S. Mannor Y. Engel and R. Meir, "The kernel recursive least-squares algorithm," *IEEE Transactions on signal processing*, vol. 52, no. 8, pp. 2275–2285, 2004.

[5] J. Platt, "A resource-allocating network for function interpolation," *Neural Compuatation*, vol. 3, pp. 213–225, 1991.

[6] W. Liu, I. Park, and J. Principe, "An information theoretic approach of designing sparse kernel adaptive filters," *IEEE Transactions on Neural Networks*, vol. 20, pp. 1950–1961, 2009.

[7] B. Chen, S. Zhao, P. Zhu, and J. Principe, "Quantized kernel least mean square algorithm," *IEEE Transactions on Neural Networks*, pp. 22–32, Jan. 2012.

[8] G. Huang, P. Saratchandran, and N. Sundararajan, "A generalized growing and pruning rbf (ggap-rbf) neural network for function approximation," *IEEE Transactions on Neural Networks*, vol. 16, no. 1, pp. 57–67, 2005.

[9] S. Van Vaerenbergh, J. Via, and I. Santamaria, "Nonlinear system identification using a new sliding-window kernel rls algorithm," *Journal of Communications*, vol. 2, no. 3, pp. 1–8, 2007.

[10] O. Dekel, S. Shalev-Shwartz, and Y. Singer, "The forgetron: A kernel-based perceptron on a budget," *SIAM Journal on Computing*, vol. 37, no. 5, pp. 1342–1372, 2008.

[11] S. Van Vaerenbergh, I. Santamaria, W. Liu, and J. Principe, "Fixed-budget kernel recursive least-squares," in *IEEE International Conference on Acoustics Speech and Signal Processing*, Dallas, USA, March 2010.