Quantized mixture kernel least mean square

Quantized Mixture Kernel Least Mean Square

Rosha Pokharel, Sohan Seth and Jose C. Principe

***Abstract*—Use of multiple kernels in the conventional kernel algorithms addresses the kernel selection problem as well as improves the performance, and is therefore, gaining much popularity recently. Kernel least mean square (KLMS) has been extended to multiple kernels recently using different approaches, one of which is mixture kernel least mean square (MxKLMS). Although this method addresses the kernel selection problem, and improves the performance, it suffers from a problem of linearly growing dictionary like in KLMS. In this paper, we present the quantized MxKLMS (QMxKLMS) algorithm to achieve sub-linear growth in dictionary. This method quantizes the input space based on the conventional criteria using Euclidean distance in input space as well as a new criteria using Euclidean distance in RKHS induced by the sum kernel. The empirical results suggest that QMxKLMS using the later metric is suitable in a non-stationary environment with abruptly changing modes as they are able to utilize the information regarding the relative importance of kernels. Moreover, the QMxKLMS using both metrics are compared with the QKLMS and the existing multi-kernel methods MKLMS and MKNLMS-CS, showing an improved performance over these methods.**

I. Introduction

THE kernel least mean square (KLMS) algorithm has been recently extended to multiple kernels to address the issue of kernel selection. Some of the multi-kernel adaptive filtering methods include multi-kernel least mean square (MKLMS) [9], multi-kernel normalized least mean square (MKNLMS) [1] and mixture kernel least mean square (MxKLMS) [2] algorithm. In addition to tackling this issue of kernel selection, the use of multiple kernels with KAF have shown fast convergence compared to the mono-kernel methods along with an ability to perform better in non-stationary environment.

Given the various advantages of using multiple kernels, this also adds a memory and computational burden. As it is well known that the dictionary size of KLMS grows linearly with the incoming samples, it is evident that the computational burden as well as memory requirement increases with the use of multiple kernels. There have been several work in the kernel adaptive filtering (KAF) literature to constrain the growth of the dictionary in KLMS. Some of them include, the use of novelty criterion (NC) [3] in [4] to check if the newly arrived datum is informative enough. Similar method called coherence criterion(CC) [5] has also been explored in [4]. The idea of approximate linear dependency test (ALD) [6] was used with KLMS in [7]. Recently, the idea of quantizing the input space by representing a certain location with a fewer samples, was used with KLMS in [8], showing a significantly improved performance over the NC and CC. Some of these methods including NC and CC have also been used to constrain the linear dictionary growth in multiple kernel adaptive filtering, like in [9] and [1], turning it into a sub-linear growth. However, when it comes to the multiple kernels, a criterion based on a metric defined in the input space might not always be justifiable. This is because, such a metric would disregard the information pertaining to the relative distances of the data in different spaces projected by different kernels. And since multiple kernel adaptive filtering weighs a pool of predefined kernels, it would therefore, be more sensible to build a criterion that gives a decision based on a metric defined in different spaces along with the the information about the importance of respective kernels. Keeping this in mind, in this paper, we have proposed a new criterion suitable for MxKLMS, that implements the quantization method for KLMS [8]. This criteria uses the Euclidean distance in the RKHS induced by sum of kernels, which in fact, can also be interpreted as the weighted sum of distances in the multiple RKHSs. Since the Euclidean distance in an RKHS corresponds to the correntropy metric in the input space [10], [11], this procedure effectively is equivalent to a correntropy metric based in the sum space. We also implement quantization criteria for MxKLMS by computing euclidean distances in the input space like in the original method [8] and compare their performances for stationary and non-stationary data.

R. Pokharel and J. C. Principe are with the Computational NeuroEngineering Laboratory, Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611 USA (email: rosha@cnel.ufl.edu; principe@cnel.ufl.edu).

S. Seth is with Helsinki Institute for Information Technology, Department of Information and Computer Science, Aalto University, Finland (email: sohan.seth@hiit.fi).

II. Background

A. KLMS and Quantized KLMS

Let, $\boldsymbol{u}_n \in \mathbb{U} \subset \mathbb{R}^l$ be an input vector at time n, and $y_n \in \mathbb{Y} \subset \mathbb{R}$ be the desired response, which is a non-linear function of input $\boldsymbol{u}_n$. The goal is to learn a continuous input-output mapping $f : \mathbb{U} \to \mathbb{Y}$, based on the incoming input-output pair $\{\boldsymbol{u}_i, y_i\}_{i=1}^N$ in the reproducing kernel Hilbert space (RKHS) $\mathbb{F}$, induced by the positive-definite kernel $\kappa : \mathbb{U} \times \mathbb{U} \to \mathbb{R}$ [12]. Any Mercer kernel induces a functional mapping $\boldsymbol{\varphi} : \mathbb{U} \to \mathbb{F}$, such that $\kappa(\boldsymbol{u}, \boldsymbol{u}') = \boldsymbol{\varphi}(\boldsymbol{u})^\top \boldsymbol{\varphi}(\boldsymbol{u}')$ holds. The commonly used Gaussian kernel is $\kappa(\boldsymbol{u}, \boldsymbol{u}') = \exp\left(\frac{-\|\boldsymbol{u}-\boldsymbol{u}'\|^2}{2\sigma^2}\right)$ where $\sigma > 0$ is the kernel width.

KLMS can simply be explained as an LMS performed in an RKHS. Given the latest input-output pair $\{\boldsymbol{\varphi}(\boldsymbol{u}_n), y_n\}$, the current weight vector $\boldsymbol{\Omega}_n$ in RKHS is given by using the gradient descent on the cost $e_n^2 = \left(y_n - \boldsymbol{\Omega}_{n-1}^\top \boldsymbol{\varphi}(\boldsymbol{u}_n)\right)^2$, such

that,

$$\boldsymbol{\Omega}_n = \boldsymbol{\Omega}_{n-1} + \eta e_n \boldsymbol{\varphi}(\boldsymbol{u}_n) \tag{1}$$

where, η is the learning rate for the gradient update. Now, according to the reproducing property, $f(\boldsymbol{u}) = \langle \boldsymbol{\Omega}, \boldsymbol{\varphi}(\boldsymbol{u}) \rangle$ and so, $f_n = \sum_{i=1}^{n} \eta e_i \kappa(\boldsymbol{u}_i, \cdot)$.

The QKLMS algorithm can then be obtained by simply quantizing the feature vector $\boldsymbol{\varphi}(\boldsymbol{u}_n)$ in the weight update equation (1) such that,

$$\boldsymbol{\Omega}_n = \boldsymbol{\Omega}_{n-1} + \eta e_n \boldsymbol{\varphi} \mathbb{Q}[(\boldsymbol{u}_n)] \tag{2}$$

where, $\mathbb{Q}[.]$ denotes a quantization operator in $\mathbb{F}$ [8]. However, since the feature space is very high dimensional and possibly infinite dimensional, the quantization is performed in the input space $\mathbb{U}$ so that the learning rule for QKLMS is,

$$\left.\begin{aligned} f_0 &= 0 \\ e_n &= y_n - f_{n-1}(\boldsymbol{u}_n) \\ f_n &= f_{n-1} + \eta e_n \kappa(Q[\boldsymbol{u}_n], \cdot) \end{aligned}\right\} \tag{3}$$

where, $Q[\cdot]$ is a quantization operator in $\mathbb{U}$. Now, an online VQ method is implemented to quantize the input space $\mathbb{U}$ in which the dictionary is obtained directly from online samples by computing Euclidean distance in $\mathbb{U}$, given by,

$$\|\boldsymbol{u}_i - \boldsymbol{u}_j\|_{\mathbb{U}} = \{(\boldsymbol{u}_i - \boldsymbol{u}_i)^\top (\boldsymbol{u}_i - \boldsymbol{u}_j)\}^{\frac{1}{2}} \tag{4}$$

Here, the quantization criteria is based on the Euclidean distance in $\mathbb{U}$ and not in the RKHS $\mathbb{F}$, since the later is the monotonically increasing function of the former, given a Gaussian kernel, and hence, the quantization obtained in both the cases would be very similar.

B. MxKLMS

Consider P different positive definite kernels $\kappa_m, m \in \{1, \ldots, P\}$ inducing P respective RKHSs $\mathbb{F}_m, m \in \{1, \ldots, P\}$. Let, $\boldsymbol{\varphi}_m : \mathbb{U} \rightarrow \mathbb{F}_{\kappa_m}$ be the corresponding mapping functions. MxKLMS is based on learning functions f_m in individual RKHS $\mathbb{F}_m$ and then linearly combining the functions, thus, $f = \sum_{m=1}^{P} \beta_m f_m$. This allows us to sequentially learn $\boldsymbol{\beta}$ together with f_m in their respective RKHSs. We learn the functional f_m by solving KLMS in each $\mathbb{F}_{\kappa_m}$ simultaneously to minimize the cost $\left(y_n - \sum_{m=1}^{P} \beta_m \langle f_{m,n-1}, \boldsymbol{\varphi}_m(\boldsymbol{u}_n) \rangle\right)^2$ with respect to β_m and f_m. Therefore, this is a biconvex problem in f_m and β_m and can be solved by using alternating minimization.

Using the KAF notation, we will be representing f by the weight vector $\boldsymbol{\Omega}$ and thus, f_m by $\boldsymbol{\Omega}_m$ and will be switching between the notations as required. Now, given the input-output pair $\{\boldsymbol{\varphi}_k(\boldsymbol{u}_n), y_n\}$, at n^{th} iteration, the weight vector $\boldsymbol{\Omega}_{k,n}$ is given by the following LMS update rule $\mathbb{F}_{\kappa_m}$:

$$\left.\begin{aligned} \boldsymbol{\Omega}_{k,0} = 0, e_n &= y_n - \sum_{m=1}^{P} \beta_m \langle \boldsymbol{\Omega}_{m,n-1} \boldsymbol{\varphi}_m(\boldsymbol{u}_n) \rangle \\ \boldsymbol{\Omega}_{k,n} &= \boldsymbol{\Omega}_{k,n-1} + \eta e_n \beta_k \boldsymbol{\varphi}_k(\boldsymbol{u}_n) \\ \boldsymbol{\Omega}_{k,n} &= \sum_{i=1}^{n} \alpha_{k,n} \varphi_k(\boldsymbol{u}_i) \end{aligned}\right\} \tag{5}$$

where, $\alpha_{k,i} = \eta e_i \beta_{k,i} = [\alpha_{k,1}, \ldots, \alpha_{k,n}]$ is the coefficient vector for linear combination. In addition to learning $\boldsymbol{\Omega}_k$, the non-negative weights β_k are learned using a non-linear gating function,

$$\beta_k = \frac{\exp(v_k)}{\sum_{j=1}^{P} \exp(v_j)} \tag{6}$$

where, the gate parameter v_k is the intermediate weight at the n^{th} iteration. The gating function (6) maintains convexity in $\boldsymbol{\beta}$, induces sparsity and creates competition among individual filters. By imposing convexity in $\boldsymbol{\beta}$, the relative importance of each filter $\boldsymbol{\Omega}_\kappa$ and thus, the respective kernel can be assessed. This requires learning v_k using and additional update rule:

$$v_{k,n+1} = v_{k,n} - \mu \nabla v_{k,n} \tag{7}$$

where, μ is the learning rate. Therefore, the final function estimation $f(\boldsymbol{u})$ is given by,

$$\begin{aligned} f(\boldsymbol{u}) = \boldsymbol{\Omega}(\boldsymbol{u}) &= \sum_{m=1}^{P} \beta_m \boldsymbol{\Omega}_m(\boldsymbol{u}) = \sum_{m=1}^{P} \beta_m \langle \boldsymbol{\Omega}_{m,n}, \boldsymbol{\varphi}_m(\boldsymbol{u}_{n+1}) \rangle \\ &= \sum_{m=1}^{P} \sum_{i=1}^{n} \beta_m \alpha_{m,i} \kappa_m(\boldsymbol{u}_i, \boldsymbol{u}_{n+1}) \end{aligned} \tag{8}$$

Hence. we arrive at the multi-kernel adaptive filtering formulation in (8) as a result of inherent optimization in multiple RKHSs and the properties of sum space. We call this the mixture kernel LMS (MxKLMS) [2].

III. Quantized MxKLMS

The quantized MxKLMS (QMxKLMS) algorithm can be obtained by quantization of the feature vector $\boldsymbol{\varphi}_k(\boldsymbol{u}_n)$ in the weight-update equation $\boldsymbol{\Omega}_{k,n} = \boldsymbol{\Omega}_{k,n-1} + \eta e_n \beta_k \boldsymbol{\varphi}_k(\boldsymbol{u}_n), \forall k \in \{1, \ldots, P\}$, in (5) which can be expressed as

$$\left.\begin{aligned} \boldsymbol{\Omega}_{k,0} = 0, e_n &= y_n - \sum_{m=1}^{P} \beta_m \langle \boldsymbol{\Omega}_{m,n-1} \boldsymbol{\varphi}_m(\boldsymbol{u}_n) \rangle \\ \boldsymbol{\Omega}_{k,n} &= \boldsymbol{\Omega}_{k,n-1} + \eta e_n \beta_k \mathcal{Q}\left[\boldsymbol{\varphi}_k(\boldsymbol{u}_n)\right] \\ \boldsymbol{\Omega}_n &= \sum_{m=1}^{P} \beta_m \alpha_{m,i} \mathcal{Q}\left[\boldsymbol{\varphi}_m(\boldsymbol{u}_n)\right] \end{aligned}\right\} \tag{9}$$

where $\mathcal{Q}[\cdot]$ is a quantization operator in $\mathbb{F}_{\kappa_m}$. Also, β_m is learned along with $\boldsymbol{\Omega}_m$, for which the learning rule is same as described in the section II-B. Following the same reasoning as in section II-A, the quantization is performed in the input space $\mathbb{U}$ such that the learning rule for QMxKLMS is

$$\left.\begin{aligned} f_{k,0} = 0, e_n &= y_n - f_{k,n-1}(\boldsymbol{u}_n) \\ f_{k,n} &= f_{k,n-1} + \eta e_n \beta_k \kappa_k\left(Q\left[\boldsymbol{u}_n\right], \cdot\right) \\ f_n &= \sum_{m=1}^{P} \beta_m \alpha_{m,i} \kappa_m\left(Q[\boldsymbol{u}_n], \cdot\right) \end{aligned}\right\} \tag{10}$$

where, $Q[\cdot]$ is a quantization operator in $\mathbb{U}$. In the rest of the paper, the notations are simplified using $\boldsymbol{\varphi}_{m,n} = \boldsymbol{\varphi}(\boldsymbol{u}_n)$, $\boldsymbol{\varphi}^q_{m,n} = \mathcal{Q}[\boldsymbol{\varphi}(\boldsymbol{u}_n)]$, and $\boldsymbol{u}^q_n = Q[\boldsymbol{u}_n]$.

Algorithm 1 Online VQ in $\mathbb{U}$

Input: $\{\boldsymbol{u}_i \in \mathbb{U}\} \forall i \in \{1, \ldots, n\}$
Initialization: Choose quantization size $\delta \geq 0$, and initialize dictionary $\boldsymbol{D}_1 = \{\boldsymbol{u}_1\}$
Computation:
while $\{\boldsymbol{u}_i\}(i > 1)$ available **do**
1. Compute the distance between $\boldsymbol{u}_i$ and $\boldsymbol{D}_{i-1}$: $dis(\boldsymbol{u}_i, \boldsymbol{D}_{i-1}) = \min_{1 \leq j \leq size(\boldsymbol{D}_{i-1})} dis(\boldsymbol{u}_i, \boldsymbol{u}_{j,i})$
2. If $dis(\boldsymbol{u}_i, \boldsymbol{D}_{i-1}) \leq \delta$, keep the dictionary unchanged: $\boldsymbol{D}_i = \boldsymbol{D}_{i-1}$, and quantize $\boldsymbol{u}_i$ to the closest dictionary element vector $\boldsymbol{u}_i^q = \boldsymbol{D}_{j^*,i-1}$, where, $j^* = \arg\min_{1 \leq j \leq size(\boldsymbol{D}_{i-1})} dis(\boldsymbol{u}_i, \boldsymbol{u}_{j,i-1})$
3. Else, update the dictionary : $\boldsymbol{D}_i = \{\boldsymbol{D}_{i-1}, \boldsymbol{u}_i\}$, and quantize $\boldsymbol{u}_i$ as itself: $\boldsymbol{u}_i^q = \boldsymbol{u}_i$

end while

The online vector quantization (VQ) method required to quantize the input space $\mathbb{U}$, presented in [8] was based on building the dictionary directly from online samples by computing Euclidean distance in $\mathbb{U}$. However, when we have combination of multiple kernels incorporating certain weights $\boldsymbol{\beta}$, quantization based on the distance in the corresponding sum space might not always be same as one based on the distance in single kernel, depending upon how the $\boldsymbol{\beta}$ changes over time. Intuitively, it makes sense to consider the relevance of a kernel (given by $\boldsymbol{\beta}$) along with the distance measured in its induced RKHS. Therefore, we propose to compute the Euclidean distance in the sum space $\mathbb{F}$ at every in coming sample, given by,

$$\|\boldsymbol{\varphi}_i - \boldsymbol{\varphi}_j\|_{\mathbb{F}} = \left\{\left(\boldsymbol{\varphi}_i - \boldsymbol{\varphi}_j\right)^\top \left(\boldsymbol{\varphi}_i - \boldsymbol{\varphi}_j\right)\right\}^{\frac{1}{2}} = \sqrt{2 - 2\kappa(\boldsymbol{u}_i, \boldsymbol{u}_j)} \tag{11}$$

where, $\boldsymbol{\varphi} : \mathbb{U} \rightarrow \mathbb{F}$ be the corresponding mapping function. We can notice that (11) is in fact, the correntropy induced metric [10], [11] defined in the RKHS $\mathbb{F}$.

Among the various possible ways in which κ can be decomposed as the weighted sum of κ_m, where the weights are convex [12], [13], we choose to represent κ as $\sum_m \beta_m \kappa_m$ in order to be consistent with the functional decomposition $f = \sum_{m=1}^{P} \beta_m f_m$. Moreover, this allows us to use the relative importance of spaces given directly by $\boldsymbol{\beta}$. Therefore, the distance in $\mathbb{F}$ can be written as,

$$\begin{aligned} \|\boldsymbol{\varphi}_i - \boldsymbol{\varphi}_j\|_{\mathbb{F}} &= \sqrt{2\left(1 - \sum_{m=1}^{P} \beta_m \kappa_m(\boldsymbol{u}_i, \boldsymbol{u}_j)\right)} \\ &= \sqrt{\sum_{m=1}^{P} \beta_m \left(1 - \exp\left(\frac{-\|\boldsymbol{u}_i - \boldsymbol{u}_j\|^2}{2\sigma_m^2}\right)\right)} \end{aligned} \tag{12}$$

where, the constant $\sqrt{2}$ is ignored. Clearly, the distance in $\mathbb{F}$ can be represented as the weighted sum of distances in $\mathbb{F}_m$. Now, a general online VQ method is presented in Algorithm 1, that can perform online quantization of input space based on the distance in $\mathbb{U}$ or $\mathbb{F}$. In the algorithm, $\boldsymbol{D}_{j,i-1}$ denotes the j^{th} element of the dictionary $\boldsymbol{D}$ at $(i-1)^{\text{th}}$ iteration and $dis(.)$ operator can be the Euclidean distance in $\mathbb{U}$ or $\mathbb{F}$. The Euclidean distance in $\mathbb{U}$ is given by 4 and in $\mathbb{F}$ is given by12. Clearly, the distance in $\mathbb{F}$ can be represented as the weighted sum of distances in $\mathbb{F}_m$.

It is true that (12) is also a monotonically increasing function of distance in the input space $\mathbb{U}$ and using either might not make a difference in quantization when it comes to static signal leading to static values of $\boldsymbol{\beta}$. But, if there is variability in the values of $\boldsymbol{\beta_m}$, for example, in case of abruptly changing data, where the relative importance of kernels would change with time, this distance will make a difference which we will see later in the sectionIV. In this paper, we present the QMxKLMS results based on the euclidean distances in both $\mathbb{U}$ and $\mathbb{F}$ and call them QMxKLMS-Eu and QMxKLMS-KEu respectively.

Algorithm 2 Quantized MxKLMS Algorithm

Input: $\{\boldsymbol{u}_i \in \mathbb{U}, y_i\} \forall i \in \{1, \cdots, n\}$
P Kernels
Initialization: $\hat{y}_1 = 0$,
Gate parameter $v_{m,1} = \frac{1}{P}, \beta_{m,1} = \frac{1}{P} \forall m \in \{1, \cdots, P\}$,
Learning rates: $\eta, \mu > 0$,
Quantization threshold $\delta \geq 0$,
Initialize dictionary $\boldsymbol{D}_1 = \{\boldsymbol{u}_1\}$
Computation:
while $\{\boldsymbol{u}_i\}(i > 1)$ available **do**
1. Evaluate: $\hat{y}_i = \sum_{m=1}^{P} \sum_{j=1}^{size(\boldsymbol{D}_{i-1})} \beta_{m,i} \alpha_{m,i} \kappa_m(\boldsymbol{D}_{j,i-1}, \boldsymbol{u}_i)$
2. Compute error: $e_i = y_i - \hat{y}_i$
3. Update gate parameter: $v_{m,i} = v_{m,i-1} - \mu \nabla v_{m,i}$
4. Compute $\beta_{m,i}$
5. Compute the distance between $\boldsymbol{u}_i$ and $\boldsymbol{D}_{i-1}$: $dis(\boldsymbol{u}_i, \boldsymbol{D}_{i-1}) = \min_{1 \leq j \leq size(\boldsymbol{D}_{i-1})} dis(\boldsymbol{u}_i, \boldsymbol{u}_{j,i})$
6. If $dis(\boldsymbol{u}_i, \boldsymbol{D}_{i-1}) \leq \delta$, keep the dictionary unchanged: $\boldsymbol{D}_i = \boldsymbol{D}_{i-1}$, and quantize $\boldsymbol{u}_i$ to the closest dictionary element vector through updating the coefficient of that vector : $\alpha_{m,j^*,i} = alpha_{m,j^*,i-1} + \eta \beta_{m,i} e_i$ where, $j^* = \arg\min_{1 \leq j \leq size(\boldsymbol{D}_{i-1})} dis(\boldsymbol{u}_i, \boldsymbol{u}_{j,i-1})$
7. Else, add a new center and corresponding new coefficient: $\boldsymbol{D}_i = \{\boldsymbol{D}_{i-1}, \boldsymbol{u}_i\}$, and $\alpha_{m,i} = [\alpha_{m,i-1}, \eta \beta_{m,i} \eta_i]$

end while

IV. Experiments and Results

In this section, we present experiments to demonstrate the working and robustness of QMxKLMS based on two datasets: synthetic non-stationary data and Lorenz chaotic series. We will also compare the method with the two existing multi-kernel adaptive filtering techniques MKLMS and MKNLMS, that implement NC and CC respectively, to constrain the dictionary growth.

A. Non-Stationary Signal Prediction

The purpose of this experiment is to demonstrate the difference in the working of QMxKLMS-Eu and QMxKLMS-KEu in learning the non-stationary system with a constrained

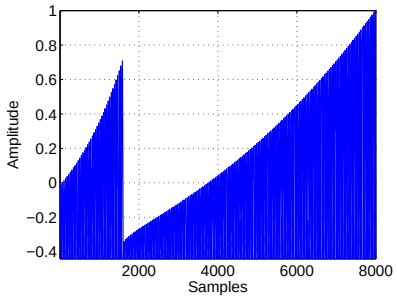
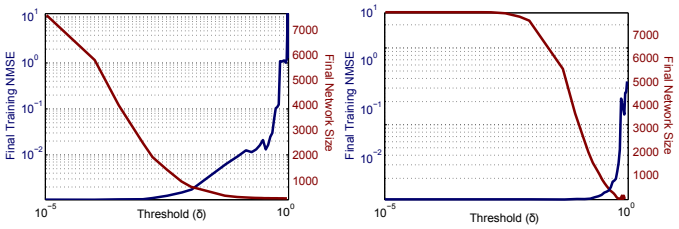


Fig. 1: Synthetic non-stationary input signal after removing mean and normalizing amplitude.



(a) (b)

Fig. 2: The figure shows the effect of quantization threshold (δ) on the final normalized MSE and final network size on synthetic non-stationary (a) QMxKLMS-Eu (b) QMxKLMS-KEu

dictionary. A signal consisting of two different modes is created using the following function [2]:

$$\left.\begin{array}{ll} 5\exp(0.5t)|\sin(2\pi ft)| & 0<t<1600 \\ \exp(\sqrt{t})|\sin(2\pi ft)| & t>1600 \end{array}\right\} \quad (13)$$

With this setup, we hope to attain variability in the values of $\boldsymbol{\beta}$ due to the changing modes and thus change in the relative importance of kernels over time, as mentioned in the previous section. This setup is also useful in testing the ability of QMxKLMS to effectively learn the changes in a non-stationary system, by competitively combining the kernel weights. Such a unique ability of creating competition among kernels adds to its robustness compared to the mono-kernel methods, thus, giving a better overall performance.

The data is preprocessed by removing the mean and normalizing it. Here, the past 10 samples are used to predict the next sample. The multiple kernels used are Gaussian kernels with different kernel sizes, $\sigma = \{0.1, 1\}$. The input signal obtained after preprocessing is shown in Fig. 1. First, the effect of quantization threshold (δ) is evaluated on the normalized MSE (NMSE) and the network size at the final stage of adaptation, for QMxKLMS-Eu and QMxKLMS-KEu. These effects can be seen in Fig.2. As expected, the final NMSE increases with increasing value of δ, and thus, the decreasing final network size for both the methods. However, in case of QMxKLMS-KEu, the final NMSE obtained even with minimum network size is much smaller than in case

TABLE I: Comparison of QMxKLMS-Eu, QMxKLMS-KEu, MxKLMS and KLMS based on prediction of synthetic non-stationary signal, to obtain (I)smallest network size (centers) (II)minimum NMSE.

Method	Centers	NMSE	Gain	Parameters
QMxKLMS-Eu	9	1.082 ± 3.25	-0.34	$\eta = 2, \mu = 0.5, \delta = 0.73$
QMxKLMS-KEu	**7**	$\mathbf{0.071 \pm 0.24}$	**11.15**	$\eta = 2, \mu = 0.5, \delta = 0.81$
QKLMS,$\sigma = 0.1$	9	2.195 ± 7.78	-3.45	$\delta = 0.73, \eta = 0.5$
QKLMS,$\sigma = 1$	9	0.105 ± 0.26	9.76	$\delta = 0.73, \eta = 0.5$
QMxKLMS-Eu	1660	0.001 ± 0.004	29.30	$\eta = 2, \mu = 0.5, \delta = 0.001$
QMxKLMS-KEu	**1285**	$\mathbf{0.001 \pm 0.004}$	**29.30**	$\eta = 2, \mu = 0.5, \delta = 0.25$
QKLMS,$\sigma = 0.1$	1650	0.020 ± 0.059	16.86	$\delta = 0.001, \eta = 0.5$
QKLMS,$\sigma = 1$	1650	0.004 ± 0.01	23.82	$\delta = 0.001, \eta = 0.5$
MxKLMS	8000	0.001 ± 0.0004	29.61	$\eta = 2, \mu = 0.5$

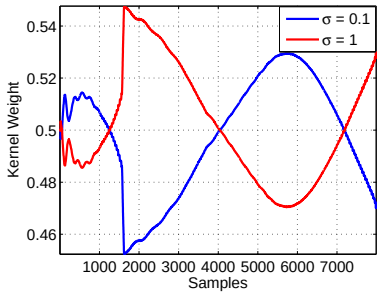


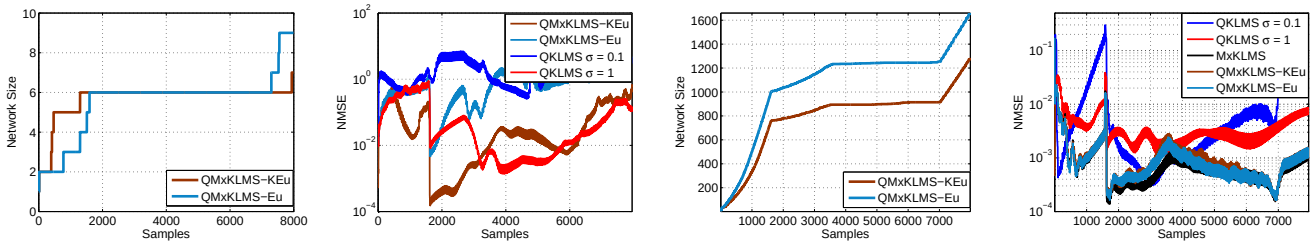
Fig. 4: Kernel weight evolution across samples for synthetic non-stationary signal

of QMxKLMS-Eu. Therefore, for a fair comparison, the two methods have been evaluated for two cases: (I) small network size (II) minimum NMSE.

The result of prediction for case (I) and case (II) are presented in Fig.3. The figure compares the network size evolution as well as the normalized squared error across samples for QMxKLMS- Eu and QMxKLMS-KEu. The results have been quantified in Table.I for both the cases (I) and (II). in terms of mean NMSE, prediction gain and the final network size. The prediction gain is defined as, $G_p = 10\log_{10}\left(\frac{\sum_{i=T_0}^{T}\|y_i\|^2}{\sum_{i=T_0}^{T}\|y_i-\hat{y}_i\|_2^2}\right)$.

In case (I), QMxKLMS-Eu is clearly unable to learn, while QMxKLMS-KEu still gives a reasonable performance with only 7 samples. Where as, in case (II), both the methods are able to attain same minimum NMSE but QMxKLMS-KEu still attains smaller network size than QMxKLMS-Eu.

We can notice that both QMxKLMS are able to attain the same NMSE as MxKLMS with a network size that is as small as $\frac{1}{6}^{\text{th}}$ the total, without any compromise in the performance, showing a significant advantage over MxKLMS. Moreover, even with reduced network size QMxKLMS-Eu and QMxKLMS-KEu depict similar kernel weight evolution as in MxKLMS as shown in Fig.4. It is also evident from the



(a) (b) (c) (d)

Fig. 3: The figures compare the performances of QMxKLMS-Eu, QMxKLMS-KEu and QKLMS($\sigma = 0.1, 1$) on prediction of synthetic non-stationary signal in terms of (a) Network size evolution across samples (QMxKLMS-Eu and QKLMS($\sigma = 0.1, 1$) overlap) (b) Normalized squared error across samples to obtain smallest network size (c) Network size evolution across samples (QMxKLMS-Eu and QKLMS($\sigma = 0.1, 1$) overlap) (d) Normalized squared error across samples, to obtain minimum NMSE

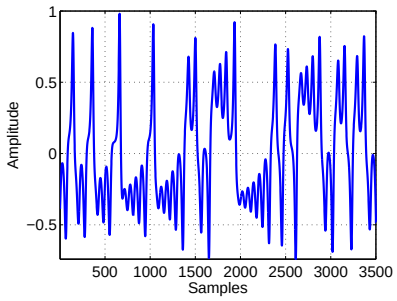
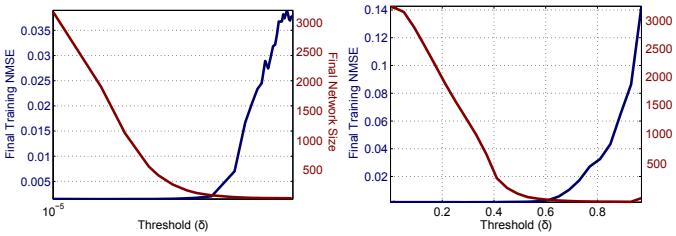


Fig. 5: Lorenz series x-component



(a) (b)

Fig. 6: The figures show the final training normalized square error and final network size on lorenz time series data plotted against values of threshold (δ) for (a) QMxKLMS-Eu (b) QMxKLMS-KEu

results that QMxKLMS is able to show effective prediction performance, better than mono kernel QMxKLMS, even when the system switches between two modes, by selecting the best kernel at each region.

In Fig. 3d, QKLMS with $\sigma = 0.1$ gives less prediction error until the 800^{th} sample, compared to QKLMS with $\sigma = 1$. This is reflected in the kernel weights for QMxKLMS and MxKLMS in Fig.4, where emphasis is given to the kernel with $\sigma = 0.1$ compared to kernel with $\sigma = 1$. As the performance of QKLMS with $\sigma = 0.1$ starts to degrade compared to $\sigma = 1$, the kernel weights start to switch. Thus, QMxKLMS, like MxKLMS, is able to select different kernels for different regions. Similar behavior can be noticed in the regions around 2000, 4000, 7000 samples.

B. Short-term Prediction of Lorenz Chaotic Time Series

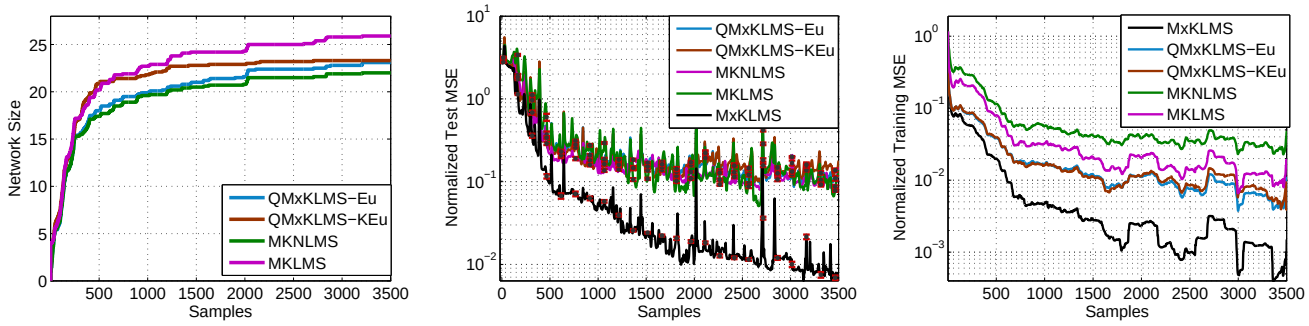
This experiment compares the QMxKLMS-Eu and QMxKLMS-KEu with the two existing multi-kernel methods: MKLMS and MKNLMS-CS for short-term prediction of Lorenz chaotic time series, in terms of the convergence speed, accuracy of prediction and their network size evaluated for the two cases: (I) small network size (II) minimum NMSE. The Lorenz chaotic system is described by the following differential equations:

$$\frac{dx}{dt} = \gamma x + yz,\ \frac{dy}{dt} = \delta(z - y),\ \frac{dz}{dt} = -zy + \rho y - z.$$

The parameters $\gamma = 8/3$, $\delta = 10$, $\rho = 28$, are used to obtain sample data using first-order approximation with step size 0.01. We utilize the x-dimension of the time series, and predict the current value $u(i)$ from past 5 values $\boldsymbol{u}_i = [u(i-5), u(i-4), ..., u(i-1)]^\top$. The signal is further preprocessed by removing mean and normalizing its amplitude and is shown in Fig.5.

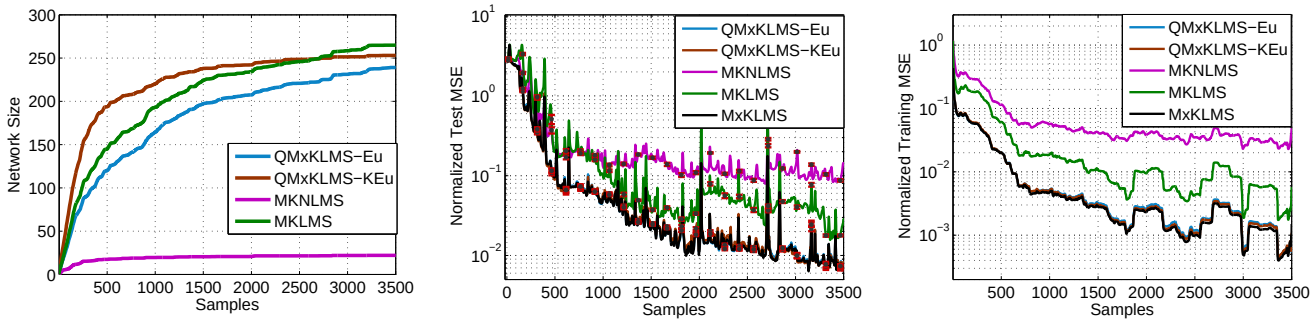
Same set of Gaussian kernels with kernel sizes $\sigma \in \{0.01, 0.1, 0.5, 1\}$ are used for all methods. The MKLMS and MKNLMS-CS have been implemented with the NC and CC sparsification criteria that they have been presented with in [9] and [1], respectively. The sparsification parameters for these methods include the input distance threshold δ and the error threshold δ_e for MKLMS and the input distance threshold δ for MKNLMS-CS. These parameters, along with the threshold parameter δ for both QMxKLMS were assigned different values to obtain the two cases (I) and (II). The other parameters including, learning rates η, $\hat{\eta}$ and the regularization parameter ρ for MKLMS, the learning rate η for MKNLMS-CS and the learning rates η and μ for QMxKLMS, MxKLMS, were fixed for both the cases. The parameters are listed in Table. II.

The results of prediction are generated by averaging over 10 independent simulations, each using different input segments of 3500 sample length. Fig. 7 compares the average network size, average test and training normalized MSE (NMSE) of



(a) (b) (c)

Fig. 7: The figures compare performances of QMxKLMS-Eu, QMxKLMS-KEu, MKNLMS-CS, and MKNLMS on short term prediction of chaotic Lorenz series based on the (a) mean network size (b) mean test NMSE (all methods except MxKLMS overlap) (c)mean training NMSE, obtained for case (I) small network size



(a) (b) (c)

Fig. 8: The figures compare performances of QMxKLMS-Eu, QMxKLMS-KEu, MKNLMS-CS, and MKNLMS on short term prediction of chaotic Lorenz series based on the (a) mean network size (b) mean test NMSE (MxKLMS, QMxKLMS-Eu and QMxKLMS-KEu overlap) (c) mean training NMSE, obtained for case (II) minimum test NMSE

QMxKLMS-Eu, QMxKLMS-KEu, MKLMS and MKNLMS-CS across training samples, for the case (I). Each ensemble learning curve is obtained by plotting the ensemble MSE on test data of 100 sample length (keeping the filter fixed during testing phase) at each training instance. Notice that, for a network size as small as 25 centers, the test NMSE for the four methods are similar although they are much less compared to that of MxKLMS which uses all the available centers. Similarly, Fig. 8 compares the average network size and average test and training normalized MSE (NMSE) of QMxKLMS-Eu, QMxKLMS-KEu, MKLMS and MKNLMS-CS across training samples, for the case (II). Notice that, MKNLMS-CS gives minimum test NMSE when the network size is as small as 25 centers, as in case (I) but the other methods require around 250 samples to give their best performances in terms of test NMSE. However, the best performances of QMxKLMS-Eu and QMxKLMS-KEu out perform the best performances of MKLMS and MKNLMS. In fact, QMxKLMS-Eu and QMxKLMS-KEu are able to give as good a performance as MxKLMS that uses all the centers. These results are quantified in Table. II in terms of final network sizes, final test and training NMSE and the final test gain for both the cases (I) and (II). The final network size is calculated as an average of final network size obtained at the end of training for 10 different simulations. The final normalized test MSE is calculated as an average over last 1000 samples in the average ensemble learning curves for 10 simulations. Also, the final normalized training MSE is calculated as an average over last 1000 samples in the average normalized training MSE curves for 10 different simulations. Finally, the test gain is obtained as an average prediction gain of test data from 10 different simulations. As we can see, QMxKLMS-KEu and QMxKLMS-Eu are able to outperform all the other methods. Also, it should be noted that, both QMxKLMS are able to attain as good a performance as MxKLMS with as less as close to 250 centers. Hence, we are able to cut down the memory requirement and computational complexity by a very large amount without compromising in the performance of the system.

V. Conclusion

In this paper, we presented MxKLMS with a method to obtain sub-linear growth in dictionary by quantizing the input space, and called it quantized MxKLMS (QMxKLMS). We quantized the input space based on the conventional criteria

TABLE II: Comparison of QMxKLMS-Eu, QMxKLMS-KEu, MxKLMS, MKNLMS-CS, MKLMS, and QKLMS based on prediction of chaotic Lorenz series to obtain (I)smallest network size (II)minimum MSE.

Experiment	Method	Network Size	Training MSE	Test MSE	Test Gain	Parameters
I	QMxKLMS-Eu	$\mathbf{23.14 \pm 1.79}$	$\mathbf{0.007 \pm 0.003}$	$\mathbf{0.124 \pm 0.102}$	$\mathbf{14.75 \pm 5.89}$	$\eta = 1, \mu = 1, \delta = 0.06$
	QMxKLMS-KEu	23.31 ± 2.54	0.008 ± 0.004	0.150 ± 0.144	14.82 ± 7.55	$\eta = 1, \mu = 1, \delta = 0.67$
	MKLMS	25.90 ± 1.66	0.014 ± 0.006	0.126 ± 0.122	14.99 ± 4.21	$\hat{\eta} = 0.05, \eta = 0.08, \delta_d = 0.22, \delta_e = 0.1$
	MKNLMS-CS	22.05 ± 1.63	0.033 ± 0.004	0.110 ± 0.082	13.79 ± 2.61	$\rho = 10^{-4}, \eta = 0.1, \delta = 0.97$
II	QMxKLMS-Eu	$\mathbf{239.10 \pm 3.98}$	$\mathbf{0.001 \pm 0.001}$	$\mathbf{0.013 \pm 0.011}$	$\mathbf{25.38 \pm 3.44}$	$\eta = 1, \mu = 1, \delta = 0.0031$
	QMxKLMS-KEu	253.10 ± 20.34	0.001 ± 0.001	0.013 ± 0.009	25.27 ± 3.48	$\eta = 1, \mu = 1, \delta = 0.45$
	MKLMS	264.9 ± 17.85	0.006 ± 0.004	0.044 ± 0.057	22.43 ± 4.99	$\hat{\eta} = 0.05, \eta = 0.08, \delta_d = 0.04, \delta_e = 0.1$
	MKNLMS-CS	22.05 ± 1.63	0.033 ± 0.004	0.110 ± 0.082	13.79 ± 2.61	$\rho = 10^{-4}, \eta = 0.1, \delta = 0.97$
	MxKLMS	3500	0.001 ± 0.001	0.012 ± 0.01	25.47 ± 3.51	$\eta = 1, \mu = 1$

using the Euclidean distance in the input space, as well as using a new criteria based on the Euclidean distance in the final sum space and called them QMxKLMS-Eu and QMxKLMS-KEu respectively. We showed that QMxKLMS-KEu is able to incorporate the distances in different spaces based on their relative importance given by the $\boldsymbol{\beta}$. We demonstrated its usefulness over QMxKLMS-Eu in a non-stationary environment containing rapidly switching modes. In addition, we also demonstrated that both QMxKLMS-Eu and QMxKLMS-KEu perform equally well compared to each other and much better compared to the existing multi-kernel adaptive filtering methods MKLMS and MKNLMS-CS based in the short term prediction of Lorenz chaotic time series. Therefore, QMxKLMS over all, is a much efficient method for addressing the issue of kernel selection and improving the performance of a system. Moreover, when it comes to non-stationary system, QMxKLMS-KEu is more suitable than QMxKLMS-Eu. The convergence behavior of QMxKLMS and its steady state analysis and tracking properties can be possible future avenues for research.

REFERENCES

[1] M. Yukawa, "Multikernel adaptive filtering," *Signal Processing, IEEE Transactions on*, vol. 60, no. 9, pp. 4672–4682, 2012.

[2] R. Pokharel, S. Seth, and J. C. Principe, "Mixture kernel least mean square," in *Neural Networks (IJCNN), The 2013 International Joint Conference on*. IEEE, 2013, pp. 1–7.

[3] J. Platt, "A resource-allocating network for function interpolation," *Neural computation*, vol. 3, no. 2, pp. 213–225, 1991.

[4] J. C. Príncipe, W. Liu, and S. Haykin, *Kernel Adaptive Filtering: A Comprehensive Introduction*. John Wiley & Sons, 2011, vol. 57.

[5] C. Richard, J. C. M. Bermudez, and P. Honeine, "Online prediction of time series data with kernels," *Signal Processing, IEEE Transactions on*, vol. 57, no. 3, pp. 1058–1067, 2009.

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

[7] P. P. Pokharel, W. Liu, and J. C. Principe, "Kernel least mean square algorithm with constrained growth," *Signal Processing*, vol. 89, no. 3, pp. 257–265, 2009.

[8] B. Chen, S. Zhao, P. Zhu, and J. C. Principe, "Quantized kernel least mean square algorithm," *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 23, no. 1, pp. 22–32, 2012.

[9] F. A. Tobar, S.-Y. Kung, and D. P. Mandic, "Multikernel least mean square algorithm," *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 25, pp. 265–277, 2013.

[10] W. Liu, P. P. Pokharel, and J. C. Príncipe, "Correntropy: properties and applications in non-gaussian signal processing," *Signal Processing, IEEE Transactions on*, vol. 55, no. 11, pp. 5286–5298, 2007.

[11] S. Seth and J. C. Príncipe, "Compressed signal reconstruction using the correntropy induced metric," in *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*. IEEE, 2008, pp. 3845–3848.

[12] N. Aronszajn, "Theory of reproducing kernels," *Transactions of the American mathematical society*, vol. 68, no. 3, pp. 337–404, 1950.

[13] C. Scovel, D. Hush, I. Steinwart, and J. Theiler, "Radial kernels and their reproducing kernel hilbert spaces," *Journal of Complexity*, vol. 26, no. 6, pp. 641–660, 2010.