Mixture Kernel Least Mean Square

Rosha Pokharel   Sohan Seth   Jose C. Principe

Abstract—Instead of using single kernel, different approaches of using multiple kernels have been proposed recently in kernel learning literature, one of which is multiple kernel learning (MKL). In this paper, we propose an alternative to MKL in order to select the appropriate kernel given a pool of predefined kernels, for a family of online kernel filters called kernel adaptive filters (KAF). The need for an alternative is that, in a sequential learning method where the hypothesis is updated at every incoming sample, MKL would provide a new kernel, and thus a new hypothesis in the new reproducing kernel Hilbert space (RKHS) associated with the kernel. This does not fit well in the KAF framework, as learning a hypothesis in a fixed RKHS is the core of the KAF algorithms. Hence, we introduce an adaptive learning method to address the kernel selection problem for the KAF, based on competitive mixture of models. We propose mixture kernel least mean square (MxKLMS) adaptive filtering algorithm, where the kernel least mean square (KLMS) filters learned with different kernels, act in parallel at each input instance and are competitively combined such that the filter with the best kernel is an expert for each input regime. The competition among these experts is created by using a performance based gating, that chooses the appropriate expert locally. Therefore, the individual filter parameters as well as the weights for combination of these filters are learned simultaneously in an online fashion. The results obtained suggest that the model not only selects the best kernel, but also significantly improves the prediction accuracy.

I. INTRODUCTION

Kernel based methods are widely used in the field of machine learning because of their ability to solve non-linear problems by implicitly projecting the input samples to high dimensional feature space, where the problem can be solved using linear algorithms. Support vector machines (SVM) for classification [1] and for time series prediction [2], Gaussian processes [3], [4], kernel principal component analysis [5] are some of the popular kernel based methods. Recently, adaptive filtering methods have been kernelized, called kernel adaptive filtering (KAF) algorithms [6] which are shown to be very effective in solving system identification problems. However, like other kernel based methods, KAF suffers from the difficult task of selecting an appropriate kernel.

Coming up with ways to select the best kernel has always been a challenging issue in kernel methods, and a single widely accepted solution remains to be explored. In the recent years, the techniques of multiple kernel learning (MKL) has

received much popularity to address the kernel selection problem, which is based on using some combination of multiple kernels than using single kernel. Different approaches of using multiple kernels can be found in the MKL literature, that are based on the linear combination such as weighted or unweighted sum and non-linear combination such as multiplication or exponentiation. For example, Lanckriet et al. [7] and Sonnenburg et al. [8] have shown that the simplest way is to use an unweighted sum of kernel functions. On the contrary, Tanabe et al. [9] have used linear combination of kernels where kernel weights are selected based on the performance values obtained by each kernel separately, in order to avoid learning weights using some optimization techniques like in [10]. Using the notion of similarity between two kernels, called kernel alignment [11], [12], is also popular in the field of MKL.

However, the MKL methods might not fit the framework of every kernel based method, specially the sequential or online methods such as KAF. We show later in the paper that the standard MKL framework cannot be applied in the context of KAF, based on the kernel least mean square (KLMS) algorithm [13]. This is because MKL effectively searches for a hypothesis over several reproducing kernel Hilbert spaces, each defined by a particular combination of weights over the kernels, whereas in KAF it is assumed that the underlying RKHS, i.e., the respective kernel is fixed. Unfortunately, when the kernel weights change over iterations, the corresponding RKHS also changes, thus, computing these inner products becomes impractical. In order to address this problem, we describe an alternative approach to select the appropriate kernel from a pool of predefined kernels, in an online fashion. We call it mixture kernel least mean square algorithm (MxKLMS). While this method can be generalized for all KAF algorithms, here we focus only on the kernel least mean square (KLMS) algorithm.

Rather than selecting the best kernel as a weighted combination of the base kernels for the KAF, our approach addresses the kernel selection problem using weighted sum of filters learned with different kernels, each in separate RKHS associated with the respective kernel. That is, we perform the least mean square (LMS) adaptive filtering [14] simultaneously in different RKHSs, and the final filter is obtained at each instance from the weighted combination of these individual filters. This requires learning the filter parameters along with the weights for the filter combination, in an online fashion. We use a competitive gated model to learn the combination weights, such that the filters compete among each other to explain the data at each time instance. This is done by using softmax function as a gating function. Hence, this method is

Rosha Pokharel and Jose C. Principe are with the Computational Neuro-Engineering Laboratory, Department of Electrical and Computer Engineering, University of Florida (email: rosha.principe@cnel.ufl.edu).

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

This work was supported by NSF IIS 0964197

a *gated competition of experts* [15], where each expert is a filter learned in the RKHS induced by each kernel. Here, the gate is a function of the performance of the filter, as the gate parameters are updated based on the gradient descent on the squared error, keeping the individual filter parameters constant. In addition, the individual filter parameters are adapted sequentially based on the gradient descent on the squared error keeping gate parameters constant. The gating function is calculated to forecast which expert should be given more weight. By using such a gate, we are effectively imposing sparsity constraint on these combination weights. The fact that this is an online algorithm, is very beneficial for solving practical problems.

Competitive gating was first introduced in the mixture of experts (MOE) model [16]. It has been popularly used to combine models learned with different realizations of unknown system parameters such as process and measurement noise [17], [18], [15] or different subsets of input data [19], in order to address issues related to non-stationarity, accuracy, computational burden, etc. The common aspect on these methods and our method is that, the experts compete to represent the input at each time instance. This gives the filter a unique ability to adaptively adjust to the abrupt changes in the system by changing the best kernel at different instance of time. Such an ability of being able to switch between models at different instance of time would be very beneficial for non-stationary system identification. This method is particularly different from the multikernel adaptive filtering [20], which assumes the functional form consisting of weighted kernel combination apriori, and uses standard optimization tools to learn the weights with appropriate constraints. Whereas, the proposed method arrives at the functional form as a result of combining the outputs of KAF learned on different RKHSs through a nonlinear gating function.

The rest of the paper proceeds as follows: Section II provides a brief overview on the KLMS algorithm. Section III explains why the conventional MKL formulation does not fit the KLMS framework. Section IV describes the proposed approach, MxKLMS and derives the learning algorithm for the approach. Section V describes the working of the method based on the prediction of a synthetic data, along with short time prediction of a chaotic time series and compares it with the KLMS algorithm. Finally, section VI concludes the paper with relevant ideas for improvement and future work.

II. KERNEL LEAST MEAN SQUARE ALGORITHM

Let $\mathbf{u}_n \in \mathbb{U}$ be an input vector at time instant n , and $d_n \in \mathbb{R}$ the desired response, which is non-linearly dependent on input $\mathbf{u}_n$. Bold symbols will be used to denote a vector in this paper.

Kernel Least Mean Square (KLMS) adaptive filtering algorithm is an online non-linear filtering method which can be seen as an application of least mean square (LMS) algorithm in feature space [21]. This algorithm maps the input data $\mathbf{u}_n$ into a high dimensional feature space using a Mercer's kernel, where LMS can be solved to obtain d_n . Mercer's kernel is defined as a continuous, symmetric, positive-definite function

$\kappa : \mathbb{U} \times \mathbb{U} \rightarrow \mathbb{R}$ [22]. One of the commonly used Mercer's kernels is Gaussian kernel :

$$\kappa(\mathbf{u}, \mathbf{u}') = \exp\left(-\frac{\|\mathbf{u} - \mathbf{u}'\|^2}{2\sigma^2}\right) \quad (1)$$

where $\sigma > 0$ is the kernel width. According to the Mercer's theorem, any Mercer's kernel $\kappa(\mathbf{u}, \mathbf{u}')$ induces a mapping φ from the input space $\mathbb{U}$ to a high dimensional feature space $\mathbb{F}$ such that the following relation holds:

$$\kappa(\mathbf{u}, \mathbf{u}') = \varphi(\mathbf{u})^T \varphi(\mathbf{u}') \quad (2)$$

Here $\mathbb{F}$ is regarded same as the RKHS induced by kernel κ , such that $\varphi(\mathbf{u}) = \kappa(\mathbf{u}, \cdot)$. Thus, the kernel induced mapping φ maps the input data $\mathbf{u}_n$ into $\mathbb{F}$ as $\varphi(\mathbf{u}_n)$ where the filter weights Ω are updated by using gradient descent on the cost function. The cost function here is the mean squared difference between the desired response d and the predicted response $\hat{y}$. Thus, given the new pair wise sample $\{\varphi(\mathbf{u}_n), d_n\}$, Ω_n , the estimate of weight vector in $\mathbb{F}$ is obtained by using LMS algorithm:

$$\begin{aligned} \Omega_0 &= 0 \\ e_n &= d_n - \Omega_{n-1}^T \varphi(\mathbf{u}_n) \\ \Omega_n &= \Omega_{n-1} + \eta e_n \varphi(\mathbf{u}_n) \end{aligned}$$

where, η is the learning rate for gradient update. Then,

$$\Omega_n = \sum_{i=1}^n \eta e_i \varphi(\mathbf{u}_i). \quad (3)$$

and also,

$$\hat{y}_{n+1} = \Omega_n^T \varphi(\mathbf{u}_{n+1}) = \sum_{i=1}^n \eta e_i \kappa(\mathbf{u}_i, \mathbf{u}_{n+1}). \quad (4)$$

KLMS has been established as a self-regularized algorithm in [23], with the use of an appropriate learning rate and initial condition.

III. WEIGHTED MKL FORMULATION IN KLMS FRAMEWORK

MKL methods use multiple kernels instead of selecting one specific kernel function and its corresponding parameters. Use of multiple kernels can be defined by:

$$\kappa_\psi(\mathbf{u}_i, \mathbf{u}_j) = f_\psi(\{\kappa_m(\mathbf{u}_i, \mathbf{u}_j)\}_{m=1}^P | \psi) \quad (5)$$

where the combination function $f_\psi : \mathbb{R}^P \rightarrow \mathbb{R}$, can be a linear or non-linear function and ψ parameterizes the combination function for P different kernels. A more common implementation is where the parameters are used to combine a set of predefined kernels [10]. In this paper we will only consider the MKL algorithms that combine predefined kernels by learning the combination parameters.

In weighted combination, the combination function is linearly parametrized such that:

$$\kappa_\psi(\mathbf{u}_i, \mathbf{u}_j) = \sum_{m=1}^P \psi_m \kappa_m(\mathbf{u}_i, \mathbf{u}_j) \quad (6)$$

where ψ denotes the kernel weights. Different versions of this approach depend on the way ψ is restricted: the linear sum (i.e., $\psi \in \mathbb{R}^P$), the conic sum (i.e., $\psi \in \mathbb{R}_+^P$), or the convex sum (i.e., $\sum_{m=1}^P \psi_m = 1$ and $\psi \in \mathbb{R}_+^P$) [24]. Here, the combined feature representation is defined as:

$$\varphi_\psi(\mathbf{u}) = \left[\sqrt{\psi_1} \varphi_1(\mathbf{u}^1), \sqrt{\psi_2} \varphi_2(\mathbf{u}^2), \dots, \sqrt{\psi_P} \varphi_P(\mathbf{u}^P) \right]^T \quad (7)$$

Then, the product in the combined feature space gives the combined final kernel:

$$\begin{aligned} & \langle \varphi_\psi(\mathbf{u}_i), \varphi_\psi(\mathbf{u}_j) \rangle \\ &= \left[\sqrt{\psi_1} \varphi_1(\mathbf{u}_i^1), \sqrt{\psi_2} \varphi_2(\mathbf{u}_i^2), \dots, \sqrt{\psi_P} \varphi_P(\mathbf{u}_i^P) \right] \\ & \quad \left[\sqrt{\psi_1} \varphi_1(\mathbf{u}_j^1), \sqrt{\psi_2} \varphi_2(\mathbf{u}_j^2), \dots, \sqrt{\psi_P} \varphi_P(\mathbf{u}_j^P) \right]^T \\ &= \sum_{m=1}^P \psi_m \kappa_m(\mathbf{u}_i, \mathbf{u}_j) \end{aligned} \quad (8)$$

With the update of the weight ψ , a new RKHS is produced as a result of new kernel combination, where a new solution is obtained.

Now let us try to fit this in KLMS framework. KLMS is an online algorithm in which the weight function Ω_n is updated with every incoming new sample. Let, Ω_n be the weight function obtained at iteration n , that is learned in RKHS $\mathbb{F}_{\psi(n)}$, induced by kernel $\kappa_{\psi(n)}$. When $\psi(n)$ is updated to $\psi(n+1)$, a new kernel $\kappa_{\psi(n+1)}$ is obtained as a result of new kernel combination. Thus, $\mathbb{F}_{\psi(n+1)}$ induced by $\kappa_{\psi(n+1)}$ is different from $\mathbb{F}_{\psi(n)}$ and so are their functional mappings $\varphi_{\psi(n+1)}(\mathbf{u}_{n+1})$ and $\varphi_{\psi(n)}(\mathbf{u}_n)$, respectively. Using KLMS update rule from (3), new weight function is given by,

$$\Omega_n = \sum_{i=1}^n \eta e_i \varphi_{\psi(n)}(\mathbf{u}_i) \quad (9)$$

Then, using (4), the output is,

$$\hat{y}_{n+1} = \Omega_n^T \varphi_{\psi(n+1)}(\mathbf{u}_{n+1}) \quad (10)$$

Clearly, from (9) and (10), the inner product between two feature vectors $\varphi_{\psi(n)}(\mathbf{u}_n)$ and $\varphi_{\psi(n+1)}(\mathbf{u}_{n+1})$ belonging to two different RKHS $\mathbb{F}_{\psi(n)}$ and $\mathbb{F}_{\psi(n+1)}$, respectively, cannot be replaced by any Mercer's kernel, and thus, MKL formulation does not trivially hold for KLMS. Certainly, it can no longer be framed as a kernel adaptive filter, the core of which lies in learning a function in a specific RKHS induced by a Mercer kernel.

IV. MIXTURE KERNEL LEAST MEAN SQUARE ALGORITHM

In this section, we propose a different formulation where the ultimate goal is to be able to learn the final hypothesis by projecting each input sample into multiple RKHS. Let, P be a number of feature spaces $\mathbb{F}_1, \dots, \mathbb{F}_P$ into which data is mapped using mapping functions $\varphi_1, \dots, \varphi_P$, respectively. In order to avoid the issue of MKL for KAF as mentioned in section III, we reformulate the problem by expressing the final

filter as a weighted sum of individual filters, $\sum_{m=1}^P \psi_m \Omega_m$, where $\psi_k \Omega_k \in \mathbb{F}_k \forall k \in 1, \dots, P$ are the estimate of weight function in each feature space, under the assumption $\psi_m^2 < \infty$. Then the output is $\hat{y}_n = \sum_{m=1}^P \psi_m \langle \Omega_m(n-1), \varphi_m(\mathbf{u}_n) \rangle$, where $\Omega_m(n)$ is Ω_m at the n^{th} instance. We aim to jointly optimize both Ω_k and ψ_k under the constraint $\sum_{m=1}^P \psi_m = 1$ and $\psi_k \geq 0 \forall k \in 1, \dots, P$, to impose sparsity. Then, for a new sample pair $\{\varphi(\mathbf{u}_n), d_n\}$, $\Omega_k(n)$ in the k^{th} RKHS can be obtained by using gradient descent on the cost function $J(\psi, \Omega) = e_n^2 = (d_n - \sum_{m=1}^P \psi_m \langle \Omega_m(n-1), \varphi_m(\mathbf{u}_n) \rangle)^2$, with respect to Ω_k , keeping ψ constant. The update rule for Ω_k is as follows:

$$\begin{aligned} \Omega_k(0) &= 0 \\ e_n &= d_n - \sum_{m=1}^P \psi_m(n) \langle \Omega_m(n-1), \varphi_m(\mathbf{u}_n) \rangle \\ \Omega_k(n) &= \Omega_k(n-1) + \eta e_n \psi_k(n) \varphi_k(\mathbf{u}_n) \end{aligned}$$

Therefore,

$$\Omega_k(n) = \sum_{i=1}^n \alpha_{k,i} \varphi_k(\mathbf{u}_i) \quad (11)$$

where $\alpha_{k,i} = \eta \psi_k(i) e_i$. Thus, the weight function Ω_k learned in each RKHS $\mathbb{F}_k$ is in fact scaled by ψ_k , and by restricting the value of ψ_k such that $\psi_k \geq 0$ and $\sum_{m=1}^P \psi_m = 1$, we are able to assess the relative importance of Ω_k , and thus, the respective kernel. This restriction is imposed in optimization of ψ_k by using the softmax gating function defined as:

$$\psi_k(n) = \frac{\exp(v_k(n))}{\sum_{j=1}^P \exp(v_j(n))} \quad (12)$$

where the gate parameter $v_k(n)$, $\forall k \in \{1, \dots, P\}$ is the intermediate weight at the n^{th} instance. Then, $v_k(n)$ is obtained by minimizing the cost function $J(\psi, \Omega) = e_n^2$ with respect to v_k , keeping Ω constant.

$$\begin{aligned} & \frac{\partial J(\psi, \Omega)}{\partial v_k(n)} \\ &= -2e_n \frac{\partial}{\partial v_k} \left[\sum_{m=1}^P \frac{-\exp(v_m(n))}{\left(\sum_{j=1}^P \exp(v_j(n)) \right)} \Omega_m(n)^T \varphi_m(\mathbf{u}_n) \right] \\ &= -2e_n \frac{\sum_{j=1}^P \exp(v_j(n)) (\exp(v_k(n)) y_k(n))}{\left(\sum_{j=1}^P \exp(v_j(n)) \right)^2} \\ & \quad - 2e_n \frac{\sum_{m=1}^P (\exp(v_m(n)) y_m(n)) \exp(v_k(n))}{\left(\sum_{j=1}^P \exp(v_j(n)) \right)^2} \end{aligned} \quad (13)$$

where $y_k(n) = \langle \Omega_k(n-1), \varphi_k(\mathbf{u}_n) \rangle$. Therefore, the new intermediate weight update is:

$$v_k(n+1) = v_k(n) - \mu \nabla v_k(n) \quad (14)$$

where, μ is the learning rate for the gradient descent update. Hence, we are efficiently solving a biconvex problem by using alternating minimization approach where, in the first step, the coefficients ψ_k are updated such that the function Ω_k that most

Fig. 1: Mixture Kernel LMS filter structure

closely approximates the desired response gets more emphasis. In the second step, the function Ω_k is updated in proportion to ψ_k , producing the net effect such that the function that closely approximates the desired response changes most in an online fashion. Now, the final output $\hat{y}_{n+1}$ is given by,

$$\begin{aligned}\hat{y}_{n+1} &= \sum_{m=1}^P \psi_m \sum_{i=1}^n \alpha_{m,i} \varphi_m(\mathbf{u}_i)^T \varphi_m(\mathbf{u}_{n+1}) \\ &= \sum_{m=1}^P \sum_{i=1}^n \psi_m \alpha_{m,i} \kappa_m(\mathbf{u}_i, \mathbf{u}_{n+1})\end{aligned}\quad (15)$$

This approach actually gives $P(n+1)$ coefficients α , at the n^{th} instance, compared to the $n+P$ coefficients in the usual MKL formulations. The final filter structure is shown in Fig. 1. This is a form of *gated competition of experts* [15], where each KLMS can be regarded as an expert and by implementing ψ as a gating function, responsibility of each expert can be assessed at every n^{th} instance. The experts are all given the same input $\mathbf{u}_n$, and the gating function ψ determines which expert gets more importance in an online fashion. It can be seen that by updating the gating function parameters and the parameters of each expert with gradient descent on the cost, the solution is local. At the n^{th} instance, whenever an expert gives less error than the weighted average of the errors of all the experts, its responsibility for that case will be increased, and whenever it does worse than the weighted average its responsibility will be decreased. Therefore, so we can expect different kernel combinations at different locations of input space.

The base kernels used for the combination can be any positive definite kernels with different hyper-parameters (e.g., kernel size in Gaussian kernel) or can be different types of kernels (e.g. Gaussian, polynomial, linear, etc). In this paper, we focus on using Gaussian kernels with different kernel sizes

as the base kernels, however, different types of kernels can also be implemented with this algorithm.

V. EXPERIMENTS AND RESULTS

In this section, we evaluate the performance of the proposed method based on the prediction of a non-stationary synthetic dataset and on the short-term prediction of a chaotic time series. We also compare the performance in each case with that of KLMS using each of the kernel sizes used for MxKLMS, and show that MxKLMS has an ability to select the best kernel as required and also perform significantly better than KLMS.

A. Synthetic Non-stationary Data

For this experiment, a non-stationary data is generated with two main modes, consisting of abrupt and very slow change in signal amplitude. The purpose of this experiment is to show that the proposed method is able to learn the changes in the system, by adaptively tuning to the appropriate kernel at different instances of time and can give a better overall performance. The data was created using following function:

$$\begin{aligned}5 \exp(0.5t) |\sin(2\pi ft)| & \quad 0 < t < 1600 \\ \exp(\sqrt{t}) |\sin(2\pi ft)| & \quad t > 1600\end{aligned}$$

The data was preprocessed by removing the mean and normalizing it. The preprocessed data is shown in Fig. 2. The prediction problem is such that 10 previous samples are used to predict the next new sample. Simple case is shown by using two Gaussian kernels with kernel sizes $\sigma = [0.1, 1]$ and the learning rate η and μ were selected as 0.9 and 2 respectively. The squared prediction error at every instant is plotted in Fig. 3 for MxKLMS and is compared with that of KLMS using each Gaussian kernel. The learning rate for KLMS is also same as η . The weights for each kernel is plotted in Fig. 4. From these results, it is clear that, MxKLMS can effectively do the prediction even when the system switches between two regimes, by selecting the best kernel at each regime. For example, below 800^{th} sample, we can see that KLMS with $\sigma = 0.1$ does better but after that it starts to do worse, and so the weight slowly decreases. Whereas, KLMS with $\sigma = 1$ starts to do better and so the weight for $\sigma = 1$ gradually increases. Thus, MxKLMS is able to emphasize these two different kernels in two different regions. Similarly, in case of samples after 2000, we can see that MxKLMS is slowly increasing the weight for $\sigma = 0.1$ and decreasing it for $\sigma = 1$ and so we can see that the error is still decreasing. The normalized mean squared error for the synthetic data is listed in Table I.

B. Short-term Chaotic Time Series Prediction

In this section, we perform the short term chaotic time series prediction for Lorenz series. Let us consider the Lorenz chaotic system whose states are governed by the following

Fig. 2: Synthetic data after removing mean and normalization

Fig. 3: Squared prediction error for synthetic data at every sample for MxKLMS and KLMS using different Gaussian kernels

differential equations [25]:

$$\begin{aligned} \frac{dx}{dt} &= \beta x + yz \\ \frac{dy}{dt} &= \delta(z - y) \\ \frac{dz}{dt} &= -zy + \rho y - z \end{aligned} \quad (16)$$

where the parameters are set as $\beta = 8/3$, $\delta = 10$, $\rho = 28$. The sample data are obtained using first-order approximation with step size 0.01. The first component, x , is used in this short-term prediction task after preprocessing such that it has zero mean and is normalized. A segment of the processed first component is shown in Fig. 5. Here, short-term prediction is done such that the current value $x(i)$, or the desired response is predicted by using the previous five

Fig. 4: Weight track for different Gaussian kernels across samples for synthetic data

TABLE I: Normalized mean square training error for synthetic data at each instant and test error for Lorenz time series averaged over last 1000 iterations for 10 trials

Dataset	Method	Normalized MSE
Synthetic Data	KLMS ($\sigma = 0.1$)	8.62×10^{-4}
	KLMS ($\sigma = 1$)	3.4×10^{-4}
	MxKLMS($\sigma = [0.1, 1]$)	2.5×10^{-4}
Lorenz Time Series	KLMS ($\sigma = 0.01$)	1.3443 ± 0.3526
	KLMS ($\sigma = 0.1$)	0.0686 ± 0.0161
	KLMS ($\sigma = 0.5$)	0.3268 ± 0.1157
	KLMS ($\sigma = 1$)	2.7187 ± 0.9199
	MxKLMS($\sigma = [0.01, 0.1, 0.5, 1]$)	0.0037 ± 0.0015

values as input, i.e., $\mathbf{u}_i = [x(i-5), x(i-4), \dots, x(i-1)]^T$. A group of Gaussian kernels are used with different values of $\sigma = \{0.01, 0.1, 0.5, 1\}$. The learning rates η and μ are selected to be 0.8 and 4 respectively. The performance of prediction is quantified based on the normalized mean squared error of predicting 100 test samples after every iteration. Fig. 6 shows the learning curve computed for 3000 samples averaged over 10 trials with a different test sequence every time. The performance of MxKLMS (proposed method) is compared with that of KLMS using each of the Gaussian kernels used for MxKLMS and learning rate same as η . We can see that MxKLMS is able to out-perform KLMS. The kernel selected by MxKLMS based on training error is shown in Fig. 7. The mean of normalized test MSE over last 1000 iterations over 10 trials are listed in Table I. It is clear that the proposed method not only selects the best kernel from a group, but also gives a significantly better performance compared to KLMS.

VI. CONCLUSION

We proposed a different method unlike the conventional MKL formulation, for adaptively selecting the appropriate kernel, which can be used in online learning scenarios. We showed that the conventional MKL formulation does not necessarily fit all the kernel based methods, one of which

Fig. 5: Processed Lorenz series after removing mean and normalization

Fig. 7: Weight track for different Gaussian kernels across samples for short term Lorenz time series prediction

Fig. 6: Normalized test MSE after every training iteration for Lorenz time series prediction

being KAF and if we apply MKL to KAF, then the theoretical model of learning in RKHS cannot be justified. Hence we derived the mixture kernel LMS (MxKLMS) algorithm to be able to learn in multiple RKHS simultaneously, as an alternative to the MKL approach. We showed that MxKLMS can adaptively select the appropriate kernel and can also switch between different kernels in response to abrupt changes in the time series. Not only can the proposed method select the appropriate kernel, it also significantly reduces prediction error compared to KLMS. Such a property of being able to learn the appropriate kernel in an online fashion, thus reducing the prediction error, can be very beneficial for practical problems related to system identification. One of the important issues

of this method is its computational complexity as it uses multiple RKHSs, which in principle implies a higher number of operations. It would be interesting to come up with a way to address this issue. Moreover, detailed study of this approach and its application for different practical problems such as non-stationary system identification can be potential future work.

REFERENCES

- [1] C. J. Burges, "A tutorial on support vector machines for pattern recognition," *Data mining and knowledge discovery*, vol. 2, no. 2, pp. 121–167, 1998.
- [2] K. Müller, A. Smola, G. Rätsch, B. Schölkopf, J. Kohlmorgen, and V. Vapnik, "Predicting time series with support vector machines," *Artificial Neural Networks ICANN'97*, pp. 999–1004, 1997.
- [3] C. K. Williams and C. E. Rasmussen, "Gaussian processes for regression," 1996.
- [4] D. J. MacKay, "Introduction to gaussian processes," *NATO ASI Series F Computer and Systems Sciences*, vol. 168, pp. 133–166, 1998.
- [5] B. Schölkopf, A. Smola, and K.-R. Müller, "Kernel principal component analysis," *Artificial Neural Networks ICANN'97*, pp. 583–588, 1997.
- [6] J. Principe, W. Liu, and S. Haykin, *Kernel Adaptive Filtering: A Comprehensive Introduction*. Wiley, 2011, vol. 57.
- [7] G. Lanckriet, N. Cristianini, P. Bartlett, L. Ghaoui, and M. Jordan, "Learning the kernel matrix with semidefinite programming," *The Journal of Machine Learning Research*, vol. 5, pp. 27–72, 2004.
- [8] S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf, "Large scale multiple kernel learning," *The Journal of Machine Learning Research*, vol. 7, pp. 1531–1565, 2006.
- [9] H. Tanabe, T. Ho, C. Nguyen, and S. Kawasaki, "Simple but effective methods for combining kernels in computational biology," in *Research, Innovation and Vision for the Future, 2008. RIVF 2008. IEEE International Conference on*. IEEE, 2008, pp. 71–78.
- [10] M. Gönen and E. Alpaydm, "Multiple kernel learning algorithms," *Journal of Machine Learning Research*, vol. 12, pp. 2211–2268, 2011.
- [11] J. Kandola, J. Shawe-Taylor, and N. Cristianini, "Optimizing kernel alignment over combinations of kernel," 2002.
- [12] S. Qiu and T. Lane, "A framework for multiple kernel support vector regression and its applications to sirna efficacy prediction," *Computational Biology and Bioinformatics, IEEE/ACM Transactions on*, vol. 6, no. 2, pp. 190–199, 2009.

- [13] P. Pokharel, W. Liu, and J. Principe, "Kernel lms," in *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, vol. 3. IEEE, 2007, pp. III-1421.
- [14] S. S. Haykin, B. Widrow, and J. Wiley, *Least-mean-square adaptive filters*. Wiley Online Library, 2003, vol. 31.
- [15] C. L. Fancourt, "Gated competitive systems for unsupervised segmentation and modeling of piecewise stationary signals," Ph.D. dissertation, Citeseer, 1998.
- [16] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural computation*, vol. 3, no. 1, pp. 79-87, 1991.
- [17] W. S. Chaer, R. H. Bishop, and J. Ghosh, "A mixture-of-experts framework for adaptive kalman filtering," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 27, no. 3, pp. 452-464, 1997.
- [18] N. Huh, J. Oh, and K. Kang, "On-line learning of a mixture-of-experts neural network," *Journal of Physics A: Mathematical and General*, vol. 33, no. 48, p. 8663, 2000.
- [19] R. Collobert, S. Bengio, and Y. Bengio, "A parallel mixture of svms for very large scale problems," *Neural computation*, vol. 14, no. 5, pp. 1105-1114, 2002.
- [20] M. Yukawa, "Multikernel adaptive filtering," *Signal Processing, IEEE Transactions on*, vol. 60, no. 9, pp. 4672-4682, 2012.
- [21] P. Pokharel, W. Liu, and J. Principe, "Kernel least mean square algorithm with constrained growth," *Signal Processing*, vol. 89, no. 3, pp. 257-265, 2009.
- [22] N. Aronszajn, "Theory of reproducing kernels," *Trans. Amer. Math. Soc.*, vol. 68, no. 3, pp. 337-404, 1950.
- [23] W. Liu, P. Pokharel, and J. Principe, "The kernel least-mean-square algorithm," *Signal Processing, IEEE Transactions on*, vol. 56, no. 2, pp. 543-554, 2008.
- [24] B. Schölkopf and A. Smola, *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. MIT press, 2001.
- [25] E. N. Lorenz, "Deterministic nonperiodic flow," *Journal of the atmospheric sciences*, vol. 20, no. 2, pp. 130-141, 1963.