Multikernel Adaptive Filters With Multiple Dictionaries and Regularization

Taichi Ishida* and Toshihisa Tanaka*
*Department of Electrical and Electronic Engineering,
Tokyo University of Agriculture and Technology,
2-24-16, Nakacho, Koganei-shi, Tokyo, Japan
E-mail: tishida@sip.tuat.ac.jp, tanakat@cc.tuat.ac.jp
Tel: +81-42-388-7123

***Abstract*—We discuss a method of regularization and a construction method of dictionary which have a high degree of freedom within the framework of multikernel adaptive filtering. The multikernel adaptive filter is an extension of the kernel adaptive filter using multiple kernels. Hence, it has offers higher performance than the kernel adaptive filter. In this paper, we focus on the fact that the multikernel adaptive filter determines a subspace in the sum of multiple reproducing kernel Hilbert spaces (RKHSs) associated with different kernels. Based on this, we propose a novel method to individually select appropriate input signals in order to construct dictionary which determines the subspace. Furthermore, based on the fact that any unknown filter is an element in RKHS, we propose L_2 regularization in order to avoid overadaptation. Also, we derive an algorithm that fixes the dictionary size in order to construct an efficient adaptive algorithm. Numerical examples show the efficiency of the proposed method.**

I. Introduction

Filters that approximate or track unknown systems that change from time to time are known as adaptive filters [1]. Adaptive filtering is a challenging technique which has been applied to system identification, noise or echo cancellation, and signal prediction. Depending on whether the system (to which the filter attempts to adapt) is linear or nonlinear, the hypothesized model changes. There are a number of research results regarding nonlinear adaptive filters. In particular, a well-known filter is the adaptive Volterra filter [2]–[4]. Moreover, in recent years, the efficiency of kernel adaptive filters has become well-known [5]–[13].

The kernel adaptive filter is a nonlinear adaptive filter that also makes effective use of kernel methods, which are one of the techniques to construct effective nonlinear systems with a reproducing kernel Hilbert space (RKHS) [14], [15] and a positive definite kernel [14], [15]. In this case, the filter is an element in the RKHS and output of the system is modelled as inner product of the filter and nonlinear mapping of the input signal. By using kernel methods, we maintain the advantage of using linear methods, while applying them to nonlinear mapping of the input signal. That is, even though the inner product in a higher dimensional feature space to which this mapping belongs cannot be computed explicitly, it is possible to compute the inner product through the use of a RKHS as a higher dimensional feature space, with its corresponding kernel.

Since the kernel adaptive filter is represented by the linear sum of kernels corresponding to observed input signals, the adaptive algorithm is intended to estimate coupling coefficients of kernels. Hence, as the number of observed input signals increases, the computational load increases due to linearly growing dimension of the subspace. Furthermore, the filter will be prone to overadaptation (or overfitting) due to increasing model dimension. Therefore, it has been proposed [8] to a reduction method of a number of kernels to design the filter by constructing a set of functions called a dictionary, using a coherence threshold. This method judges whether to add the kernel corresponding to an observed input signal to the dictionary at each time instant. Thus, the observed input signal similar to a signal in the dictionary is discarded. This method makes it possible to prevent overadaptation, and to reduce the computation time in updating the filter.

A multikernel adaptive filter is a modified version of the kernel adaptive filter [16], [17]. The multikernel adaptive filter estimates unknown systems adaptively using multiple different kernels. It offers higher performance than the kernel adaptive filter. Moreover it has been reported that the multikernel adaptive filter can be applied to nonstationary nonlinear systems [16].

Considering the multikernel adaptive filter within the framework of RKHS, it can be seen that the output of the multikernel adaptive filter is represented by the inner product in RKHS determined by a sum of multiple kernels. However, in spite of using the multiple different kernels, a common dictionary is used for all kernels [16]. In this paper, we focus on the fact that the output signal is estimated by the inner product in RKHS determined by the sum of kernels. Based on this, we propose to construct a dictionary independently in each RKHS. Hence, adaptation performance is improved since each subspace is constructed in each RKHS. Moreover we define the cost function with L_2 regularization, and weight coefficients of the filter are updated by this cost function, because the filter may cause overadaptation by increasing degrees of freedom of the filter. Furthermore we propose an algorithm that fixes the dictionary size in order to effectively adapt nonstationary nonlinear systems. Numerical examples show the efficiency of the proposed method.

II. Kernel Adaptive Filters

In this section, we describe kernel adaptive filters and a multikernel adaptive filter.

A. Kernel Adaptive Filters

Kernel adaptive filters are an extension of linear adaptive filters by kernel methods. In what follows, we describe the kernel least mean square (KLMS) algorithm [6] and the kernel normalized least mean square (KNLMS) algorithm [8].

1) The KLMS Algorithm: Let $\mathcal{U} \subset \mathbb{R}^L$, $\boldsymbol{u} \in \mathcal{U}$, and $\mathcal{F}$ denote the input space, the input signal, and the higher feature space respectively. Suppose that the output of system is given by the inner product $f(\boldsymbol{u}) = \langle \Omega, \phi(\boldsymbol{u}) \rangle$ of the filter $\Omega \in \mathcal{F}$ with a nonlinear mapping of an input signal $\phi(\boldsymbol{u}) \in \mathcal{F}$. We consider the problem of adaptively estimating filter Ω. The output signal at time instant $k \in \mathbb{N}$ can be written with filter $\Omega_k \in \mathcal{F}$ as

$$y_k = f(\boldsymbol{u}_k) = \langle \Omega_k, \phi(\boldsymbol{u}_k) \rangle. \tag{1}$$

Let $d_k \in \mathbb{R}$ denote the desired signal, as well as the LMS algorithm [1], the estimation error is represented as

$$e_k = d_k - \langle \Omega_k, \phi(\boldsymbol{u}_k) \rangle, \tag{2}$$

hence, the filter is updated as follows:

$$\Omega_{k+1} = \Omega_k + \mu e_k \phi(\boldsymbol{u}_k), \tag{3}$$

where, if $\Omega_0 := 0$, (1) can be rewritten as

$$y_k = \langle \Omega_k, \phi(\boldsymbol{u}_k) \rangle = \mu \sum_{i=1}^{k-1} e_i \langle \phi(\boldsymbol{u}_i), \phi(\boldsymbol{u}_k) \rangle. \tag{4}$$

Generally, it is difficult to explicitly compute the inner product. However, it is possible to compute the inner product easily through the use of a RKHS $\mathcal{H}$ as higher dimensional feature space $\mathcal{F}$. Let $\kappa(\cdot,\cdot) : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$ denote the kernel of $\mathcal{H}$ and inner product $\langle \phi(\boldsymbol{u}_i), \phi(\boldsymbol{u}_k) \rangle_{\mathcal{H}}$ is given as

$$\langle \phi(\boldsymbol{u}_i), \phi(\boldsymbol{u}_k) \rangle_{\mathcal{H}} = \kappa(\boldsymbol{u}_i, \boldsymbol{u}_k). \tag{5}$$

In terms of the kernel, the output is rewritten as

$$y_k = \langle \Omega_k, \phi(\boldsymbol{u}_k) \rangle_{\mathcal{H}} = \mu \sum_{i=1}^{k-1} e_i \kappa(\boldsymbol{u}_i, \boldsymbol{u}_k). \tag{6}$$

For example, polynomial kernel and Gaussian kernel are used as the kernel [6]. Figure 1 shows a conceptual diagram of the kernel adaptive filter. As shown in Fig. 1, the input $\boldsymbol{u}_k$ is mapped by $\phi(\cdot)$, and the output is given by the inner product with Ω_k. Then, the filter Ω_k is updated to reduce the error.

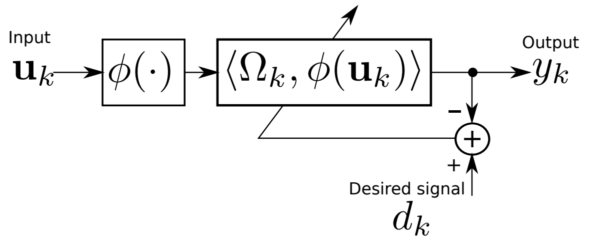

Fig. 1. Conceptual diagram of kernel adaptive filters

2) The KNLMS Algorithm: (6) can be rewritten as

$$y_k = \sum_{i=1}^{k} h_i \kappa(\boldsymbol{u}_i, \boldsymbol{u}_k) = \boldsymbol{h}_k^\top \boldsymbol{\kappa}_k, \tag{7}$$

where, $\boldsymbol{h}_k := [h_1, \ldots, h_k]^\top$ is weight vector, $\boldsymbol{\kappa}_k := [\kappa(\boldsymbol{u}_1, \boldsymbol{u}_k), \ldots, \kappa(\boldsymbol{u}_k, \boldsymbol{u}_k)]^\top$. Also, Ω_k is represented as

$$\Omega_k = \sum_{i=1}^{k} h_k \kappa(\boldsymbol{u}_i, \cdot). \tag{8}$$

Essentially, updating Ω_k is equivalent to updating $\boldsymbol{h}_k$. Therefore, the updating equation can be written by using the NLMS algorithm [1] as follows:

$$\boldsymbol{h}_{k+1} = \boldsymbol{h}_k + \frac{\eta}{\rho + \|\boldsymbol{\kappa}_k\|^2} (d_k - \boldsymbol{h}_k^\top \boldsymbol{\kappa}_k) \boldsymbol{\kappa}_k, \tag{9}$$

where η and ρ are a step size parameter and a stabilization parameter respectively.

3) Coherence-Based Sparsification: As seen from (8), the filter is represented by the sum of kernels associated with all of past input signals. Hence, as the number of observed input signals increases, not only does the computational burden grow, but also infinite memory size is required in principle. Therefore, the filter may lead to overadaptation. For this problem, in [8], the number of kernels restricted by using coherence-based sparsification. In what follows, we describe KNLMS algorithm with coherence-based sparsification.

The coherence-based sparsification constructs a set of functions called a dictionary using a coherence threshold in order to restrict a number of kernels. Let $\mathcal{J}_k := \{j_1^{(k)}, j_2^{(k)}, \ldots, j_{r_k}^{(k)}\} \subset \{0, 1, \ldots, k-1\}$ indicate the dictionary $\{\kappa(\cdot, \boldsymbol{u}_j)\}_{j \in \mathcal{J}_k}$. Here, $r_k := |\mathcal{J}_k|$ is the dictionary size with $|\cdot|$ denoting the cardinality of a set. Then, the output signal is given by

$$y_k = \sum_{j \in \mathcal{J}_k} h_{j,k} \kappa(\boldsymbol{u}_j, \boldsymbol{u}_k). \tag{10}$$

Moreover, it can be written simply as

$$y_k = \boldsymbol{h}_k^\top \boldsymbol{\kappa}_k, \tag{11}$$

where

$$\boldsymbol{h}_k := [h_{j_1^{(k)},k}, h_{j_2^{(k)},k}, \ldots, h_{j_{r_k}^{(k)},k}]^\top \in \mathbb{R}^{r_k}, \tag{12}$$

$$\boldsymbol{\kappa}_k := [\kappa(\boldsymbol{u}_{j_1^{(k)}}, \boldsymbol{u}_k), \kappa(\boldsymbol{u}_{j_2^{(k)}}, \boldsymbol{u}_k), \ldots, \kappa(\boldsymbol{u}_{j_{r_k}^{(k)}}, \boldsymbol{u}_k)]^\top \in \mathbb{R}^{r_k}. \tag{13}$$

If the initial value of weight vector is $\boldsymbol{h}_0 := 0$ and the coherence threshold is $\delta > 0$, the update rule is then given as follows:

1) If $\max_{j \in \mathcal{J}_k} |\kappa(\boldsymbol{u}_k, \boldsymbol{u}_j)| > \delta$

$$\mathcal{J}_{k+1} = \mathcal{J}_k, \tag{14}$$

$$\boldsymbol{h}_{k+1} = \boldsymbol{h}_k + \frac{\eta}{\rho + \|\boldsymbol{\kappa}_k\|^2}(d_k - \boldsymbol{h}_k^\top \boldsymbol{\kappa}_k)\boldsymbol{\kappa}_k; \tag{15}$$

2) If $\max_{j \in \mathcal{J}_k} |\kappa(\boldsymbol{u}_k, \boldsymbol{u}_j)| \le \delta$

$$\mathcal{J}_{k+1} = \mathcal{J}_k \cup \{k\}, \tag{16}$$

$$\boldsymbol{h}_{k+1} = \bar{\boldsymbol{h}}_k + \frac{\eta}{\rho + \|\bar{\boldsymbol{\kappa}}_k\|^2}(d_k - \bar{\boldsymbol{h}}_k^\top \bar{\boldsymbol{\kappa}}_k)\bar{\boldsymbol{\kappa}}_k, \tag{17}$$

where η and ρ are a step size parameter and a stabilization parameter respectively. Also, $\bar{\boldsymbol{\kappa}}_k := [\boldsymbol{\kappa}_k^\top, \kappa(\boldsymbol{u}_k, \boldsymbol{u}_k)]^\top$, $\bar{\boldsymbol{h}}_k := [\boldsymbol{h}_k^\top, 0]^\top$. Namely, this rule with the coherence threshold judges whether to add the kernel generated by the observed input signal to the dictionary at each time instant. The kernel is added to the dictionary if necessary, or discarded if not needed. Consequently, the filter is an element in the subspace that is spanned by the dictionary.

B. Multikernel Adaptive Filters

A multikernel adaptive filter is a modified version of the kernel adaptive filter by using multiple different kernels, it offers higher performance than the single kernel adaptive filter and moreover it can be applied to nonstationary nonlinear systems. In what follows, we describe the multikernel NLMS algorithm with coherence-based sparsification (MKNLMS-CS) [16] that is an extension of the KNLMS algorithm described in Section II-A3.

Assume that M different kernels $\{\kappa_m(\cdot,\cdot)\}_{m \in \mathcal{M}}$ are given. Here, $\mathcal{M} := \{1, 2, \ldots, M\}$. Let $\mathcal{J}_k := \{j_1^{(k)}, j_2^{(k)}, \ldots, j_{r_k}^{(k)}\} \subset \{0, 1, \ldots, k-1\}$ indicate the dictionary $\{\kappa_m(\cdot, \boldsymbol{u}_j)\}_{m \in \mathcal{M}, j \in \mathcal{J}_k}$. Here, $r_k := |\mathcal{J}_k|$ is dictionary size. In this case, the output signal is defined as

$$y_k = \sum_{m \in \mathcal{M}} \sum_{j \in \mathcal{J}_k} h_{m,k}^{(j)} \kappa_m(\boldsymbol{u}_k, \boldsymbol{u}_j). \tag{18}$$

Furthermore we estimate $\{h_{m,k}^{(j)}\}$, where, we define $\boldsymbol{h}_{m,k}$ and $\boldsymbol{\kappa}_{m,k}$ as follows:

$$\boldsymbol{h}_{m,k} := [h_{m,k}^{(j_1^{(k)})}, h_{m,k}^{(j_2^{(k)})}, \ldots, h_{m,k}^{(j_{r_k}^{(k)})}]^\top \in \mathbb{R}^{r_k}, \tag{19}$$

$$\boldsymbol{\kappa}_{m,k} := [\kappa_m(\boldsymbol{u}_{j_1^{(k)}}, \boldsymbol{u}_k), \kappa_m(\boldsymbol{u}_{j_2^{(k)}}, \boldsymbol{u}_k), \ldots, \kappa_m(\boldsymbol{u}_{j_{r_k}^{(k)}}, \boldsymbol{u}_k)]^\top \in \mathbb{R}^{r_k}. \tag{20}$$

The output is rewritten as

$$y_k = \sum_{m \in \mathcal{M}} \boldsymbol{h}_{m,k}^\top \boldsymbol{\kappa}_{m,k}. \tag{21}$$

If the initial value of weight vector is $\boldsymbol{h}_{m,0} := 0$ and the coherence threshold is $\delta > 0$, the update rule is then given as follows:

1) If $\max_{m \in M} \max_{j \in \mathcal{J}_k} |\kappa_m(\boldsymbol{u}_k, \boldsymbol{u}_j)| > \delta$

$$\mathcal{J}_{k+1} = \mathcal{J}_k, \tag{22}$$

$$\boldsymbol{h}_{m,k+1} = \boldsymbol{h}_{m,k} + \frac{\eta}{\rho + \sum_{m \in \mathcal{M}} \|\boldsymbol{\kappa}_{m,k}\|^2}(d_k - y_k)\boldsymbol{\kappa}_{m,k}; \tag{23}$$

2) If $\max_{m \in M} \max_{j \in \mathcal{J}_k} |\kappa_m(\boldsymbol{u}_k, \boldsymbol{u}_j)| \le \delta$

$$\mathcal{J}_{k+1} = \mathcal{J}_k \cup \{k\}, \tag{24}$$

$$\boldsymbol{h}_{m,k+1} = \bar{\boldsymbol{h}}_{m,k} + \frac{\eta}{\rho + \sum_{m \in \mathcal{M}} \|\bar{\boldsymbol{\kappa}}_{m,k}\|^2}(d_k - y_k)\bar{\boldsymbol{\kappa}}_{m,k}, \tag{25}$$

where η and ρ are a step size parameter and a stabilization parameter respectively. Also, $\bar{\boldsymbol{\kappa}}_{m,k} := [\boldsymbol{\kappa}_{m,k}^\top, \kappa_m(\boldsymbol{u}_k, \boldsymbol{u}_k)]^\top$, $\bar{\boldsymbol{h}}_{m,k} := [\boldsymbol{h}_{m,k}^\top, 0]^\top$. Note that, if $M = 1$, the MKNLMS-CS algorithm coincides with the KNLMS algorithm.

III. Multikernel Adaptive Filters With Multiple Dictionaries and Regularization

In this section, we give a representation of multikernel adaptive filters in terms of a RKHS and propose the multikernel adaptive filter with multiple dictionaries and regularization. Then we derive the associated algorithm. First, we indicate that the multikernel adaptive filter in [16] is an element in a RKHS associated with a sum of different kernels, Moreover, we describe the RKHS as a sum space of RKHS associated with each kernel. Based on this, we show that it is possible to design a filter that has a high degree of freedom by constructing the dictionary in each RKHS. Furthermore, we discuss L_2 regularization in order to prevent overadaptation. Finally, we derive an algorithm that fixes the dictionary size in order to construct an efficient adaptive algorithm.

A. The Sum Space of RKHS

We describe the sum space of RKHS [18] in order to discuss a space in which a multikernel adaptive filter exist. We consider the case of sum space of two RKHS, for the sake of ease without loss of generality.

1) Norm: Let $\mathcal{H}$ and $\|\cdot\|_{\mathcal{H}}$ denote RKHS and the norm of $\mathcal{H}$ respectively. Moreover, $\mathcal{H}_1 \oplus \mathcal{H}_2$ denotes the direct sum of RKHS of $\mathcal{H}_1$ and $\mathcal{H}_2$, it is represented as H. In this case, the norm of direct sum $f = (f_1, f_2) \in H$ of $f_1 \in \mathcal{H}_1$ and $f_2 \in \mathcal{H}_2$ is represented as follows [18]:

$$\|f\|_H^2 := \|f_1\|_{\mathcal{H}_1}^2 + \|f_2\|_{\mathcal{H}_2}^2. \tag{26}$$

In particular, if $\mathcal{H}_1 \cap \mathcal{H}_2 = \{0\}$, the sum space $\tilde{\mathcal{H}} := \{f = f_1 + f_2 \mid f_1 \in \mathcal{H}_1, f_2 \in \mathcal{H}\}$ is isomorphic to the direct space H [18]. Consequently, the norm in $\tilde{\mathcal{H}}$ is represented as

$$\|f\|_{\tilde{\mathcal{H}}}^2 := \|f_1\|_{\mathcal{H}_1}^2 + \|f_2\|_{\mathcal{H}_2}^2. \tag{27}$$

2) Reproducing Property: Let the kernel of $\mathcal{H}_1$ and $\mathcal{H}_2$ denote κ_1 and κ_2 respectively. The value of any $f \in \tilde{\mathcal{H}}$ can be evaluated by the kernel $\kappa = \kappa_1 + \kappa_2$ [18]

$$\begin{aligned} f(u) &= \langle f, \kappa(\cdot, u) \rangle_{\tilde{\mathcal{H}}} \\ &= \langle f_1, \kappa_1(\cdot, u) \rangle_{\mathcal{H}_1} + \langle f_2, \kappa_2(\cdot, u) \rangle_{\mathcal{H}_2}. \end{aligned} \tag{28}$$

B. Multikernel Adaptive Filters With Multiple Dictionaries and Regularization

Assume that M different kernels $\{\kappa_m(\cdot,\cdot)\}_{m\in\mathcal{M}}$ are given. Here, $\mathcal{M} := \{1,2,\dots,M\}$. Also, let $\mathcal{H}_m$ and $\mathcal{H}$ denote RKHS determined by the mth kernel and the sum space. In this case, from (28), the output is represented with the filter $\Omega_k \in \mathcal{H}$ and nonlinear mapping of input $\phi(\boldsymbol{u}_k) \in \mathcal{H}$ as

$$\begin{aligned} y_k &= \langle \Omega_k, \phi(\boldsymbol{u}_k)\rangle_{\mathcal{H}} \\ &= \sum_{m\in\mathcal{M}} \langle \Omega_{m,k}, \phi_m(\boldsymbol{u}_k)\rangle_{\mathcal{H}_m}. \end{aligned} \tag{29}$$

$\Omega_{m,k}$ can be constructed in each $\mathcal{H}_m$. Hence, Ω_k is the sum of $\Omega_{m,k}$ in $\mathcal{H}_m$. Therefore, we can consider

$$\Omega_{m,k} = \sum_{j\in\mathcal{J}_{m,k}} h_{m,k}^{(j)} \kappa_m(\cdot,\boldsymbol{u}_j). \tag{30}$$

Thus, there is no need for the index set of dictionary in each RKHS to equalize. Let $\mathcal{J}_{m,k} := \{j_1^{(k)}, j_2^{(k)}, \dots, j_{r_{m,k}}^{(k)}\} \subset \{0,1,\dots,k-1\}$ indicate the mth dictionary $\{\kappa_m(\cdot,\boldsymbol{u}_j)\}_{j\in\mathcal{J}_{m,k}}$. Here, $r_{m,k} := |\mathcal{J}_{m,k}|$ is the mth dictionary size. Then in the MKNLMS-CS algorithm,

$$\mathcal{J}_{n,k} = \mathcal{J}_{l,k} \quad (\forall n, \forall l \in \mathcal{M}, n \neq l), \tag{31}$$

while, in the case described in (30),

$$\mathcal{J}_{n,k} \neq \mathcal{J}_{l,k} \quad (\forall n, \forall l \in \mathcal{M}, n \neq l). \tag{32}$$

Consequently, it can be said that the filter Ω_k given by (30) has a high degree of freedom and generality. The output is rewritten by using the Ω_k as

$$y_k = \sum_{m\in\mathcal{M}} \sum_{j\in\mathcal{J}_{m,k}} h_{m,k}^{(j)} \kappa_m(\boldsymbol{u}_k, \boldsymbol{u}_j). \tag{33}$$

Moreover, it can be written simply as

$$y_k = \sum_{m\in\mathcal{M}} \boldsymbol{h}_{m,k}^\top \boldsymbol{\kappa}_{m,k}, \tag{34}$$

where,

$$\boldsymbol{h}_{m,k} := [h_{m,k}^{(j_1^{(k)})}, h_{m,k}^{(j_2^{(k)})}, \dots, h_{m,k}^{(j_{r_{m,k}}^{(k)})}]^\top \in \mathbb{R}^{r_{m,k}}, \tag{35}$$

$$\boldsymbol{\kappa}_{m,k} := [\kappa_m(\boldsymbol{u}_{j_1^{(k)}}, \boldsymbol{u}_k), \kappa_m(\boldsymbol{u}_{j_2^{(k)}}, \boldsymbol{u}_k), \dots, \kappa_m(\boldsymbol{u}_{j_{r_{m,k}}^{(k)}}, \boldsymbol{u}_k)]^\top \in \mathbb{R}^{r_{m,k}}. \tag{36}$$

Next, we discuss the cost function used to determine the filter. For simplicity, we choose L_2 norm, and apply regularization for $\Omega_{m,k}$ in order to avoid overadaptation. We define the cost function in order to update weight vector $\boldsymbol{h}_{m,k}$ as follows:

$$\begin{aligned} C_k &:= |d_k - \langle \Omega_k, \phi(\boldsymbol{u}_k)\rangle_{\mathcal{H}}|^2 + \sum_{m\in\mathcal{M}} \lambda_m \|\Omega_{m,k}\|_{\mathcal{H}_m}^2 \\ &= \left| d_k - \sum_{m\in\mathcal{M}} \langle \Omega_{m,k}, \phi_m(\boldsymbol{u}_k)\rangle_{\mathcal{H}_m} \right|^2 + \sum_{m\in\mathcal{M}} \lambda_m \|\Omega_{m,k}\|_{\mathcal{H}_m}^2 \\ &= \left| d_k - \sum_{m\in\mathcal{M}} \langle \Omega_{m,k}, \phi_m(\boldsymbol{u}_k)\rangle_{\mathcal{H}_m} \right|^2 \\ &\quad + \sum_{m\in\mathcal{M}} \lambda_m \left\langle \sum_{j\in\mathcal{J}_{m,k}} h_{m,k}^{(j)} \kappa_m(\cdot,\boldsymbol{u}_j), \sum_{j\in\mathcal{J}_{m,k}} h_{m,k}^{(j)} \kappa_m(\cdot,\boldsymbol{u}_j) \right\rangle_{\mathcal{H}_m} \\ &= \left| d_k - \sum_{m\in\mathcal{M}} \langle \Omega_{m,k}, \phi_m(\boldsymbol{u}_k)\rangle_{\mathcal{H}_m} \right|^2 \\ &\quad + \sum_{m\in\mathcal{M}} \lambda_m \sum_{j\in\mathcal{J}_{m,k}} \sum_{j\in\mathcal{J}_{m,k}} h_{m,k}^{(j)} h_{m,k}^{(j)} \langle \kappa_m(\cdot,\boldsymbol{u}_j), \kappa_m(\cdot,\boldsymbol{u}_j)\rangle_{\mathcal{H}_m} \\ &= \left| d_k - \sum_{m\in\mathcal{M}} \boldsymbol{h}_{m,k}^\top \boldsymbol{\kappa}_{m,k} \right|^2 + \sum_{m\in\mathcal{M}} \lambda_m \boldsymbol{h}_{m,k}^\top \boldsymbol{K}_{m,k} \boldsymbol{h}_{m,k}, \end{aligned} \tag{37}$$

where the second term in the right-hand side of the cost function is the L_2 regularization term, and also, $\boldsymbol{K}_{m,k}$ and λ_m are a Gram matrix and a regularization parameter of the mth kernel respectively.

If the initial value of weight vector is $\boldsymbol{h}_{m,0} := 0$ and the coherence threshold is $\delta_m > 0$, by using (37), the update rule is then given as follows:

1) If $\max_{j\in\mathcal{J}_{m,k}} |\kappa_m(\boldsymbol{u}_k, \boldsymbol{u}_j)| > \delta_m$

$$\mathcal{J}_{m,k+1} = \mathcal{J}_{m,k}, \tag{38}$$

$$\boldsymbol{h}_{m,k+1} = \boldsymbol{h}_{m,k} + \eta \frac{(d_k - y_k)\boldsymbol{\kappa}_{m,k} - \lambda_m \boldsymbol{K}_{m,k} \boldsymbol{h}_{m,k}}{\rho + \sum_{m\in\mathcal{M}} \|\boldsymbol{\kappa}_{m,k}\|^2}; \tag{39}$$

2) If $\max_{j\in\mathcal{J}_{m,k}} |\kappa_m(\boldsymbol{u}_k, \boldsymbol{u}_j)| \le \delta_m$

$$\mathcal{J}_{m,k+1} = \mathcal{J}_{m,k} \cup \{k\}, \tag{40}$$

$$\boldsymbol{h}_{m,k+1} = \bar{\boldsymbol{h}}_{m,k} + \eta \frac{(d_k - y_k)\bar{\boldsymbol{\kappa}}_{m,k} - \lambda_m \bar{\boldsymbol{K}}_{m,k} \bar{\boldsymbol{h}}_{m,k}}{\rho + \sum_{m\in\mathcal{M}} \|\bar{\boldsymbol{\kappa}}_{m,k}\|^2}, \tag{41}$$

where η and ρ are a step size parameter and a stabilization parameter respectively. Also, $\bar{\boldsymbol{\kappa}}_{m,k} := [\boldsymbol{\kappa}_{m,k}^\top, \kappa_m(\boldsymbol{u}_k, \boldsymbol{u}_k)]^\top$, $\bar{\boldsymbol{h}}_{m,k} := [\boldsymbol{h}_{m,k}^\top, 0]^\top$. $\bar{\boldsymbol{K}}_{m,k}$ is a Gram matrix of $\bar{\boldsymbol{\kappa}}_{m,k}$. We name the above proposed algorithm the multikernel NLMS algorithm with multiple dictionaries and coherence-based sparsification (MKNLMS-MDCS).

Finally, we discuss an update rule if the dictionary size is fixed. Let L_m denote a dictionary size in $\mathcal{H}_m$. If $\max_{j\in\mathcal{J}_{m,k}} |\kappa_m(\boldsymbol{u}_k, \boldsymbol{u}_j)| \le \delta_m$ is satisfied, and the dictionary exceeds the size L_m (thus $|\mathcal{J}_{m,k}| = L_m$), we delete the oldest kernel in the dictionary. In this case, the above update rule 2) is replaced by what follows:

2)' If $\max_{j\in\mathcal{J}_{m,k}} |\kappa_m(\boldsymbol{u}_k, \boldsymbol{u}_j)| \leq \delta_m$

$$\mathcal{J}_{m,k+1} = \begin{cases} \mathcal{J}_{m,k} \cup \{k\} & (|\mathcal{J}_{m,k}| < L_m) \\ (\mathcal{J}_{m,k} \cup \{k\}) - \min \mathcal{J}_{m,k} & (|\mathcal{J}_{m,k}| = L_m), \end{cases} \tag{42}$$

$\boldsymbol{h}_{m,k+1}$ is updated by (41) with

$$\bar{\boldsymbol{\kappa}}_{m,k} := [\tilde{\boldsymbol{\kappa}}_{m,k}^{\top}, \kappa_m(\boldsymbol{u}_k, \boldsymbol{u}_k)]^{\top}, \tag{43}$$

$$\bar{\boldsymbol{h}}_{m,k} := [\tilde{\boldsymbol{h}}_{m,k}^{\top}, 0]^{\top}, \tag{44}$$

where $\tilde{\boldsymbol{\kappa}}_{m,k}$ and $\tilde{\boldsymbol{h}}_{m,k}$ are the deleted first element $\boldsymbol{\kappa}_{m,k}$ and $\boldsymbol{h}_{m,k}$ respectively. We name this algorithm the multikernel NLMS algorithm with multiple dictionaries fixed dictionary size (MKNLMS-MDF). This rule deletes unnecessary kernels (kernels generated by the system prior to the switch). Hence, the algorithm is practical because memory size can be fixed, and it can be applied effectively for nonstationary nonlinear systems.

Note that, as the number of different kernels, M, increases, the memory usage of the proposed algorithms increases unlike the MKNLMS-CS algorithm. However, the computational complexity of the proposed and MKNLMS-CS algorithms can get almost similar by adjustment of coherence thresholds.

IV. Numerical examples

This section demonstrates the efficiency of the proposed method through two computer experiments. One demonstrates the pure effect of using multiple dictionaries and L_2 regularization compared to the existing methods on online prediction of a nonlinear system. The other demonstrates effect of the update rule with fixed dictionary sizes on online prediction of a nonstationary nonlinear system. In the experiments, we adopt a Gaussian kernel as

$$\kappa(\boldsymbol{u}_i, \boldsymbol{u}_k) = \exp(-\zeta \|\boldsymbol{u}_i - \boldsymbol{u}_k\|^2) \quad (\zeta \geq 0). \tag{45}$$

The MKNLMS-CS, the MKNLMS-MDCS, and the MKNLMS-MDF algorithms employ two Gaussian kernels. The parameters of two Gaussian kernels $\kappa_1(\cdot,\cdot)$ and $\kappa_2(\cdot,\cdot)$ are ζ_1 and ζ_2 respectively. In this case, $\mathcal{H}_1 \cap \mathcal{H}_2 = \{0\}$ is obvious because different Gaussian kernels are linearly independent.

A. Online Prediction of a Nonlinear System

We consider the nonlinear system as follows [8]:

$$\begin{aligned} d_k = {} & (0.8 - 0.5\exp(-d_{k-1}^2))d_{k-1} \\ & - (0.3 + 0.9\exp(-d_{k-1}^2))d_{k-2} + 0.1\sin(d_{k-1}\pi), \end{aligned} \tag{46}$$

where d_k is the desired signal, whose initial value is $[d_{-1}, d_{-2}]^{\top} = [0.1, 0.1]^{\top}$, and data size is 3000. Moreover, d_k is corrupted by noise sampled from a zero-mean Gaussian distribution with standard deviation equal to 0.1. Input signals are $\boldsymbol{u}_k = [d_{k-1}, d_{k-2}]^{\top}$. In the adaptation of the system, we compare NLMS, KNLMS, MKNLMS-CS, and MKNLMS-MDCS algorithms. The evaluation criteria adopted mean square error (MSE). The MSE is calculated by taking an arithmetic average over 100 independent realizations. Parameters of each filter are given in Table I. Here, the coherence thresholds of MKNLMS-CS and KNLMS were manually adjusted such that the numbers of kernels forming the filter are almost the same as that in the MKNLMS-MDCS.

TABLE I
Parameters

NLMS	$\alpha = 1.0\times10^{-2}, \beta = 3.0\times10^{-2}$
KNLMS1	$\eta = 9.0\times10^{-2}, \rho = 3.0\times10^{-2}$ $\delta = 0.95, \zeta = 0.5$
KNLMS2	$\eta = 9.0\times10^{-2}, \rho = 3.0\times10^{-2}$ $\delta = 0.36, \zeta = 10$
MKNLMS-CS	$\eta = 9.0\times10^{-2}, \rho = 3.0\times10^{-2}$ $\delta = 0.885, \zeta_1 = 0.5, \zeta_2 = 10$
MKNLMS-MDCS	$\eta = 9.0\times10^{-2}, \rho = 3.0\times10^{-2}$ $\delta_1 = 0.8, \delta_2 = 0.2, \zeta_1 = 0.5, \zeta_2 = 10$ $\lambda_1 = 1.0\times10^{-10}, \lambda_2 = 2.0\times10^{-3}$

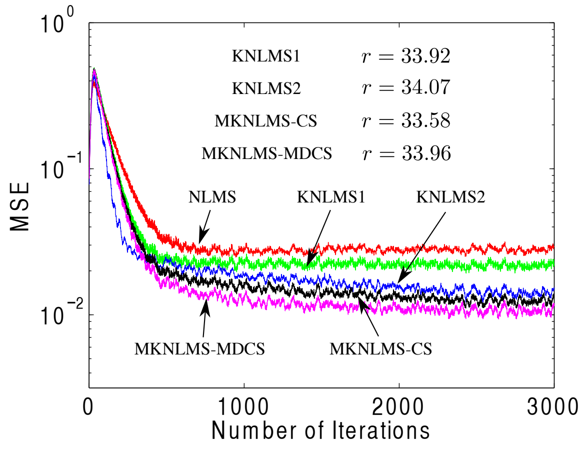

Fig. 2. Learning curves of NLMS, KNLMS, MKNLMS-CS and MKNLMS-MDCS for the nonlinear system

Figure 2 shows the MSE at each iteration, where r denotes the mean of the number of generated kernels. From Fig. 2, it is seen that, thanks to the use of multiple dictionaries and L_2 regularization, the MKNLMS-MDCS algorithm shows smaller MSE than the others.

B. Online Prediction of a Nonstationary Nonlinear System

We use the following nonlinear model [16] to generate a test signal:

$$\begin{aligned} d'_k = {} & (0.2 - 0.7\exp(-d'^2_{k-1}))d'_{k-1} \\ & - (0.8 + 0.8\exp(-d'^2_{k-1}))d'_{k-2} + 0.2\sin(d'_{k-1}\pi), \end{aligned} \tag{47}$$

where, the initial value is $[d'_{-1}, d'_{-2}]^{\top} = [0.1, 0.1]^{\top}$, and data size is 21000. Moreover, d'_k is corrupted by noise sampled from a zero-mean Gaussian distribution with standard deviation equal to 0.1. The nonlinear system used in this simulation, with d'_k, is defined as

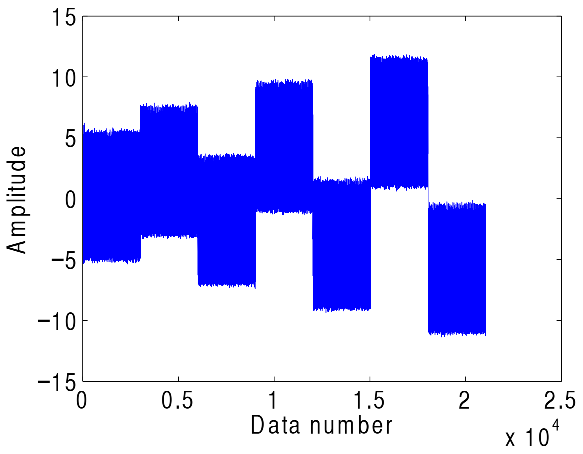

Fig. 3. The generated signal by nonstationary nonlinear system

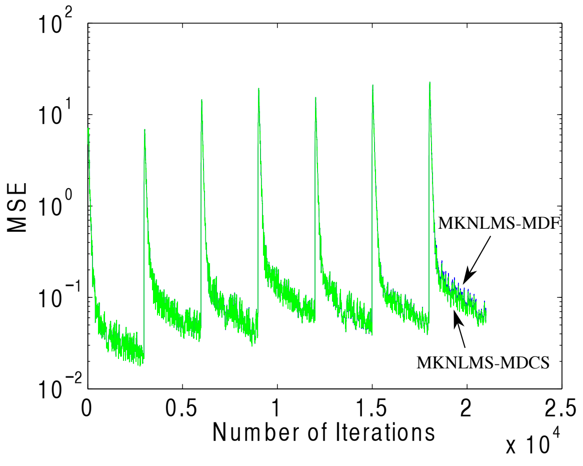

Fig. 4. Learning curves of MKNLMS-MDCS and MKNLMS-MDF for the nonstationary nonlinear system

$$d_k = \begin{cases} d'_k & (k \leq 3000) \\ d'_k + 2 & (3000 < k \leq 6000) \\ d'_k - 2 & (6000 < k \leq 9000) \\ d'_k + 4 & (9000 < k \leq 12000) \\ d'_k - 4 & (12000 < k \leq 15000) \\ d'_k + 6 & (15000 < k \leq 18000) \\ d'_k - 6 & (18000 < k \leq 21000). \end{cases} \tag{48}$$

Figure 3 shows a generated signal from the system. In this experiment, we compare the MKNLMS-MDCS and the MKNLMS-MDF algorithms in order to investigate efficiency of the update rule with fixed dictionary sizes. The evaluation criteria and parameters of filters are the same as Section IV-A. Moreover, in the MKNLMS-MDF algorithm, the dictionary sizes of κ_1 and κ_2 fixed 120 and 100 respectively.

Figure 4 shows the MSE at each iteration. From Fig. 4, it is seen that the adaptation ability of the two approaches is almost unchanged. However, an increase in efficiency (i.e. decreased dictionary size for equal MSE) was noted with the MKNLMS-MDF algorithm. The total of dictionary size in the MKNLMS-MDF algorithm is fixed at 220 (the dictionary sizes of κ_1 and κ_2 are 120 and 100 respectively), while the total of dictionary size in the MKNLMS-MDCS algorithm was 427 (the dictionary sizes of κ_1 and κ_2 are 146 and 281 respectively) on average. Hence, the MKNLMS-MDF algorithm is effective for nonstationary nonlinear systems.

V. Conclusion

This paper proposed a multikernel adaptive filter with multiple dictionaries and regularization. By using multiple dictionaries and regularization, we showed that adaptation ability can be improved by numerical examples. Moreover, we proposed an algorithm that fixes the dictionary size and we showed its effectiveness by numerical examples.

Acknowledgment

This work is supported in part by JSPS KAKENHI Grant Number 23300069.

References

[1] S. Haykin, *Adaptive Filter Theory*. Upper Saddle River, NJ: Prentice-Hall, 2002.

[2] J. Lee and J. V. Mathews, "A fast recursive least squares adaptive second order Volterra filter and its performance analysis," *IEEE Trans. Signal Process.*, vol. 41, no. 3, pp. 1087–1102, 1993.

[3] R. D. Nowak and B. D. Van Veen, "Random and pseudorandom inputs for Volterra filter identification," *IEEE Trans. Signal Process.*, vol. 42, no. 8, pp. 2124–2135, 1994.

[4] ——, "Volterra filter equalization: A fixed point approach," *IEEE Trans. Signal Process.*, vol. 45, no. 2, pp. 377–388, 1997.

[5] W. Liu, J. Principe, and S. Haykin, *Kernel Adaptive Filtering*. Hoboken, NJ: Wiley, 2010.

[6] J. Kivinen, A. J. Smola, and R. C. Williamson, "Online learning with kernels," *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2165–2176, 2004.

[7] W. Liu, P. P. Pokharel, and J. C. Principe, "The kernel least-mean-square algorithm," *IEEE Trans. Signal Process.*, vol. 56, no. 2, pp. 543–554, 2008.

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

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

[10] K. Slavakis, S. Theodoridis, and I. Yamada, "Adaptive constrained learning in reproducing kernel Hilbert spaces: The robust beamforming case," *IEEE Trans. Signal Process.*, vol. 57, no. 12, pp. 4744–4764, 2009.

[11] A. Malipatil, Y.-F. Huang, S. Andra, and K. Bennett, "Kernelized set-membership approach to nonlinear adaptive filtering," in *Proc. IEEE ICASSP*, 2005, pp. 149–152.

[12] J. Gil-Cacho, T. van Waterschoot, M. Moonen, and S. Jensen, "Nonlinear acoustic echo cancellation based on a parallel-cascade kernel affine projection algorithm," in *Proc. IEEE ICASSP*, 2012, pp. 33–36.

[13] P. Bouboulis and S. Theodoridis, "Extension of Wirtinger's calculus to reproducing kernel Hilbert spaces and the complex kernel LMS," *IEEE Trans. Signal Process.*, vol. 59, no. 3, pp. 964–978, 2011.

[14] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY: Springer, 2006.

[15] T. Tanaka, Y. Washizawa, and A. Kuh, "Adaptive kernel principal components tracking," in *Proc. IEEE ICASSP*, 2012, pp. 1905–1908.

[16] M. Yukawa, "Multikernel adaptive filtering," *IEEE Trans. Signal Process.*, vol. 60, no. 9, pp. 4672–4682, 2012.

[17] Y. Nakajima and M. Yukawa, "Nonlinear channel equalization by multi-kernel adaptive filter," in *Proc. IEEE SPAWC*, 2012, pp. 384–388.

[18] N. Aronszajn, "Theory of reproducing kernels," *Trans. Amer. Math. Soc.*, vol. 68, no. 9, pp. 337–404, 1950.