NONLINEAR ADAPTIVE FILTERING TECHNIQUES WITH MULTIPLE KERNELS

Masahiro Yukawa

Department of Electrical and Electronic Engineering, Niigata University
2-8050 Ikarashi, Nishi-ku, Niigata, 950-2181 Japan
phone/fax: +81 (25) 262 7549, email: yukawa@eng.niigata-u.ac.jp

ABSTRACT

In this paper, we propose a novel approach using multiple kernels to nonlinear adaptive filtering problems. We present two types of multi-kernel adaptive filtering algorithms, both of which are based on the kernel normalized least mean square (KNLMS) algorithm (Richard *et al.*, 2009). One is a simple generalization of KNLMS, adopting the coherence criterion for dictionary selection. The other is derived by applying the adaptive proximal forward-backward splitting method to a certain squared distance function penalized by a weighted block ℓ_1 norm. The latter algorithm operates the weighted block soft-thresholding which encourages the sparsity of dictionary at the block level. Numerical examples demonstrate the efficacy of the proposed approach.

1. INTRODUCTION

Kernel has been proven an attractive tool in adaptive filtering and online learning when the desired response is a nonlinear function of input data [1–7]; see [8] for a comprehensive introduction to kernel adaptive filtering. The major advantages of the kernel-based approach include that it involves (i) no local minima unlike the neural network approach, and (ii) a marginal number of parameters unlike the Volterra series based approach. One of its central issues is that the size of *dictionary*[1] grows linearly with the number of input data observed. This obviously conflicts with the limitations of memory and computational resource/time. Several *sparsification* techniques have been proposed and investigated for updating the dictionary in such a way that only dominant ones remain among basis vectors.

Kernel design is another important issue, as witnessed by the successive studies on kernel selection; see [9–11] among many others. Those previous studies have been done exclusively for batch processing, and the techniques developed are not suitable for online processing. In the literature of kernel adaptive filtering or online learning, a reasonable kernel has been assumed available prior to adaptation, which is however not always possible.

This paper proposes efficient nonlinear adaptive filtering techniques using a set of kernels, typically Gaussian with different kernel parameters. Although there are other possible approaches to exploiting multiple kernels, we restrict ourselves to extending the kernel normalized least mean square (KNLMS) algorithm [6] in the current study. First, we derive a multi-kernel NLMS algorithm adopting a coherence-based criterion for the dictionary construction. Second, we present another multi-kernel NLMS algorithm based on iterative use of *metric projection* and *weighted block soft-thresholding*. The latter algorithm is derived by applying the adaptive proximal forward-backward splitting method [12] to the following cost function: a certain squared distance (*smooth*) plus a weighted block ℓ_1 norm (*nonsmooth*). The weighted block soft-thresholding promotes *sparsity of the dictionary at the block level*. Thanks to this property, the algorithm keeps its computational efficiency when applied to the problems with high dimensional inputs. The estimator of the proposed approach can be characterized as an element of a Cartesian product of the reproducing kernel Hilbert spaces associated respectively with the kernels employed. Numerical examples support the advantages of the proposed algorithms.

[1] The term "dictionary" stands for the set of basis vectors that are used to construct a nonlinear estimator.

2. KERNEL NLMS ALGORITHM

Let $\mathbf{u}_n \in \mathscr{U}$ be an input vector at time instant $n \in \mathbb{N}$, and $d_n \in \mathbb{R}$ the desired response depending *nonlinearly* on $\mathbf{u}_n$. Here $\mathscr{U}$ is a compact subset of the L dimensional Euclidean space $\mathbb{R}^L$. The task is to find the nonlinear dependency in an online fashion with the measurements $(\mathbf{u}_n, d_n)$ arriving sequentially.

In the kernel adaptive filtering approach, a filter (or an estimator) is modeled as an element of the reproducing kernel Hilbert space (RKHS) associated with a *kernel* $\kappa : \mathscr{U} \times \mathscr{U} \to \mathbb{R}$ [8, 13–15]. One of the celebrated examples is the *Gaussian kernel* defined as follows: $\kappa(\mathbf{x},\mathbf{y}) := \exp\left(-\alpha \|\mathbf{x}-\mathbf{y}\|^2\right)$, $\forall \mathbf{x},\mathbf{y} \in \mathscr{U}$, for the kernel parameter $\alpha > 0$. For the sake of simple notation, we denote by $\|\cdot\|$ the Euclidean norm in any dimension space. It is known that the Gaussian kernel has the universal approximation property [8]. *Data sparsification* is necessary to limit the computational costs to a manageable amount. Let $\mathscr{J}_{n-1} := \{\omega_1, \omega_2, \cdots, \omega_{r_{n-1}}\} \subset \{0, 1, \cdots, n-1\}$ indicate the dictionary $(\kappa(\cdot, \mathbf{u}_j))_{j \in \mathscr{J}_{n-1}}$ that forms a nonlinear estimator at time n, where r_{n-1} denotes the size of dictionary. The estimator producing an estimate of d_n is given as follows:

$$\psi_n(\mathbf{u}) := \sum_{j \in \mathscr{J}_{n-1}} h_{j,n} \kappa(\mathbf{u}, \mathbf{u}_j), \ \mathbf{u} \in \mathscr{U}, \tag{1}$$

where $h_{j,n} \in \mathbb{R}$, $j \in \mathscr{J}_{n-1}$, are the parameters determined by an adaptive algorithm with the set of input-output data $(\mathbf{u}_j, d_j)_{j=0}^{n-1}$. Define $\mathbf{h}_n := [h_{\omega_1,n}, h_{\omega_2,n}, \cdots, h_{\omega_{r_{n-1}},n}]^\mathsf{T} \in \mathbb{R}^{r_{n-1}}$ and $\mathbf{k}_n := [\kappa(\mathbf{u}_n, \mathbf{u}_{\omega_1}), \kappa(\mathbf{u}_n, \mathbf{u}_{\omega_2}), \cdots, \kappa(\mathbf{u}_n, \mathbf{u}_{\omega_{r_{n-1}}})]^\mathsf{T} \in \mathbb{R}^{r_{n-1}}$, where $(\cdot)^\mathsf{T}$ stands for *transpose*. An estimate of d_n is then given as $\widehat{d}_n = \psi_n(\mathbf{u}_n) = \mathbf{k}_n^\mathsf{T} \mathbf{h}_n$, $n \in \mathbb{N}$. The point here is *how to design the dictionary* and *how to update the coefficients* $h_{j,n}$.

In [6], a coherence-based dictionary-designing scheme has been proposed. Suppose for simplicity that the kernel has a unit Hilbertian norm, i.e., $\kappa(\mathbf{x},\mathbf{x}) = 1$, $\forall \mathbf{x} \in \mathscr{U}$; the Gaussian kernel satisfies this condition. With the initial dictionary $\{\kappa(\cdot, \mathbf{u}_0)\}$ indicated by $\mathscr{J}_{-1} := \{0\}$, the scheme inserts $\kappa(\cdot, \mathbf{u}_n)$, $n \in \mathbb{N}$, into the dictionary if the following condition is satisfied:

$$\max_{j \in \mathscr{J}_{n-1}} \left|\kappa(\mathbf{u}_n, \mathbf{u}_j)\right| \le \delta, \ n \in \mathbb{N}, \tag{2}$$

where $\delta > 0$ is the threshold that governs the degree of sparsity and the coherence of the dictionary. Letting $\zeta_\mathrm{a} > 0$ and $\zeta_\mathrm{b} > 0$ be kernel parameters of Gaussian kernels, the same dictionary size is

obtained by setting their corresponding thresholds $\delta_{\mathrm{a}} > 0$ and $\delta_{\mathrm{b}} > 0$ such that $\delta_{\mathrm{a}}^{\zeta_{\mathrm{b}}} = \delta_{\mathrm{b}}^{\zeta_{\mathrm{a}}}$; this idea is used to determine the values of δ in Section 4.

With the initial $\mathbf{h}_0 = h_{0,0} := 0$, the step size $\eta \in [0,2]$, and the regularization parameter $\rho > 0$, the update rule of KNLMS is given as follows.

(i) If (2) is unsatisfied, $\mathscr{J}_n := \mathscr{J}_{n-1}$, and

$$\mathbf{h}_{n+1} := \mathbf{h}_n + \eta \frac{d_n - \mathbf{k}_n^{\mathsf{T}} \mathbf{h}_n}{\|\mathbf{k}_n\|^2 + \rho} \mathbf{k}_n. \tag{3}$$

(ii) If (2) is satisfied, $\mathscr{J}_n := \mathscr{J}_{n-1} \cup \{n\}$, and

$$\mathbf{h}_{n+1} := \bar{\mathbf{h}}_n + \eta \frac{d_n - \bar{\mathbf{k}}_n^{\mathsf{T}} \bar{\mathbf{h}}_n}{\|\bar{\mathbf{k}}_n\|^2 + \rho} \bar{\mathbf{k}}_n, \tag{4}$$

where $\bar{\mathbf{k}}_n := [\mathbf{k}_n^{\mathsf{T}}, \kappa(\mathbf{u}_n, \mathbf{u}_n)]^{\mathsf{T}}$ and $\bar{\mathbf{h}}_n := [\mathbf{h}_n^{\mathsf{T}}, 0]^{\mathsf{T}}$.

Although its affine projection version is proposed in [6], we solely consider the NLMS version to present our idea as simply as possible.

3. MULTI-KERNEL NLMS ALGORITHMS

We present multi-kernel NLMS algorithms based on two different sparsification techniques. Let $\kappa_m : \mathscr{U} \times \mathscr{U} \to \mathbb{R}$, $m \in \mathscr{M} := \{1, 2, \cdots, M\}$, be a set of M distinct kernels to be employed.

3.1 Algorithm with Coherence-based Sparsification

With $\mathscr{J}_{n-1}^{\mathrm{CS}} := \{\omega_1, \omega_2, \cdots, \omega_{r_{n-1}}\} \subset \{0, 1, \cdots, n-1\}$ indicating a dictionary, our estimator takes the following form:

$$\psi_n(\mathbf{u}) := \sum_{m \in \mathscr{M}} \sum_{j \in \mathscr{J}_{n-1}^{\mathrm{CS}}} h_{j,n}^{(m)} \kappa_m(\mathbf{u}, \mathbf{u}_j), \ \mathbf{u} \in \mathscr{U}, \tag{5}$$

where $h_{j,n}^{(m)} \in \mathbb{R}$, $m \in \mathscr{M}$, $j \in \mathscr{J}_{n-1}^{\mathrm{CS}}$. The estimates $\widehat{d}_n := \psi_n(\mathbf{u}_n)$ of d_n can be rewritten in a vector form as follows:

$$\psi_n(\mathbf{u}_n) = \sum_{m \in \mathscr{M}} \mathbf{h}_{m,n}^{\mathsf{T}} \mathbf{k}_{m,n}, \tag{6}$$

where $\mathbf{h}_{m,n} := [h_{\omega_1,n}^{(m)}, h_{\omega_2,n}^{(m)}, \cdots, h_{\omega_{r_{n-1}},n}^{(m)}]^{\mathsf{T}} \in \mathbb{R}^{r_{n-1}}$ and $\mathbf{k}_{m,n} := [\kappa_m(\mathbf{u}_n, \mathbf{u}_{\omega_1}), \kappa_m(\mathbf{u}_n, \mathbf{u}_{\omega_2}), \cdots, \kappa_m(\mathbf{u}_n, \mathbf{u}_{\omega_{r_{n-1}}})]^{\mathsf{T}} \in \mathbb{R}^{r_{n-1}}$. Moreover, it can be written in a matrix form as

$$\psi_n(\mathbf{u}_n) = \mathrm{tr}\left(\mathbf{H}_n^{\mathsf{T}} \mathbf{K}_n\right), \tag{7}$$

where $\mathbf{H}_n := [\mathbf{h}_{1,n}\ \mathbf{h}_{2,n} \cdots \mathbf{h}_{M,n}]$, $\mathbf{K}_n := [\mathbf{k}_{1,n}\ \mathbf{k}_{2,n} \cdots \mathbf{k}_{M,n}]$, and $\mathrm{tr}(\cdot)$ stands for the trace of matrix.

The dictionary is constructed based on the following criterion:

$$\|\mathbf{K}_n\|_{\max} := \max_{m \in \mathscr{M}} \max_{j \in \mathscr{J}_{n-1}^{\mathrm{CS}}} \left|\kappa_m(\mathbf{u}_n, \mathbf{u}_j)\right| \le \delta_{\mathrm{CS}}, \ n \in \mathbb{N}, \tag{8}$$

where $\delta_{\mathrm{CS}} > 0$ is the threshold. In the same way as [6, Proposition 2], it can readily be proved that the dictionary size under (8) is finite for any sequence $(\mathbf{u}_n)_{n=1}^{\infty} \subset \mathscr{U}$ due to the compactness of $\mathscr{U}$.

Let $\mathbf{H}_0 = [h_{0,0}^{(1)}, h_{0,0}^{(2)}, \cdots, h_{0,0}^{(M)}] := \mathbf{0}_M^{\mathsf{T}}$ be the initial row vector, where $\mathbf{0}_M \in \mathbb{R}^M$ denotes the length M zero vector. With the step size $\eta_{\mathrm{CS}} \in [0,2]$ and the regularization parameter $\rho_{\mathrm{CS}} > 0$, the update rule is given as follows.

(i) If (8) is unsatisfied, $\mathscr{J}_n^{\mathrm{CS}} := \mathscr{J}_{n-1}^{\mathrm{CS}}$ and

$$\mathbf{H}_{n+1} := \mathbf{H}_n + \eta_{\mathrm{CS}} \frac{d_n - \mathrm{tr}\left(\mathbf{H}_n^{\mathsf{T}} \mathbf{K}_n\right)}{\|\mathbf{K}_n\|_{\mathrm{F}}^2 + \rho_{\mathrm{CS}}} \mathbf{K}_n, \tag{9}$$

where $\|\cdot\|_{\mathrm{F}}$ denotes the Frobenius norm.

(ii) If (8) is satisfied, $\mathscr{J}_n^{\mathrm{CS}} := \mathscr{J}_{n-1}^{\mathrm{CS}} \cup \{n\}$ and

$$\mathbf{H}_{n+1} := \bar{\mathbf{H}}_n + \eta_{\mathrm{CS}} \frac{d_n - \mathrm{tr}\left(\bar{\mathbf{K}}_n^{\mathsf{T}} \bar{\mathbf{H}}_n\right)}{\|\bar{\mathbf{K}}_n\|_{\mathrm{F}}^2 + \rho_{\mathrm{CS}}} \bar{\mathbf{K}}_n, \tag{10}$$

where $\bar{\mathbf{H}}_n := [\mathbf{H}_n^{\mathsf{T}}\ \mathbf{0}_M]^{\mathsf{T}}$ and $\bar{\mathbf{K}}_n := [\mathbf{K}_n^{\mathsf{T}}\ \bar{\mathbf{k}}_n]^{\mathsf{T}}$ with $\bar{\mathbf{k}}_n := [\kappa_1(\mathbf{u}_n, \mathbf{u}_n), \kappa_2(\mathbf{u}_n, \mathbf{u}_n), \cdots, \kappa_M(\mathbf{u}_n, \mathbf{u}_n)]^{\mathsf{T}}$.

We name the above algorithm *the multi-kernel NLMS algorithm with coherence-based sparsification (MKNLMS-CS)*. For $M = 1$, the MKNLMS-CS algorithm coincides with the KNLMS algorithm, and hence it is a generalization of KNLMS.

3.2 Algorithm with Block Soft-thresholding

In $\mathbb{R}^{p \times q}$ for any $p, q \in \mathbb{N} \setminus \{0\}$, define an inner product as $\langle \mathbf{A}, \mathbf{B} \rangle := \mathrm{tr}\left(\mathbf{A}^{\mathsf{T}} \mathbf{B}\right)$, $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{p \times q}$, and its induced norm as $\|\mathbf{A}\| := \sqrt{\langle \mathbf{A}, \mathbf{A} \rangle} = \|\mathbf{A}\|_{\mathrm{F}}$, $\mathbf{A} \in \mathbb{R}^{p \times q}$. If in particular $q = 1$, the inner product and norm are reduced respectively to the standard inner product and the Euclidean norm. With $\mathscr{J}_{n-1}^{\mathrm{BT}} := \{\omega_1, \omega_2, \cdots, \omega_{r_{n-1}}\} \subset \{0, 1, \cdots, n-1\}$ indicating a dictionary, our estimator produces an estimate of d_n in the following form:

$$\psi_n(\mathbf{u}_n) = \langle \mathbf{H}_n, \mathbf{K}_n \rangle, \tag{11}$$

where $\mathbf{H}_n$ and $\mathbf{K}_n$ are defined as in Section 3.1.

Define the metric distance $d(\mathbf{H}, \Pi_n) := \min_{\mathbf{G} \in \Pi_n} \|\mathbf{H} - \mathbf{G}\|$ between $\mathbf{H} \in \mathbb{R}^{(r_{n-1}+1) \times M}$ and

$$\Pi_n := \left\{ \mathbf{X} \in \mathbb{R}^{(r_{n-1}+1) \times M} : \left\langle \bar{\mathbf{K}}_n, \mathbf{X} \right\rangle = d_n \right\}, \tag{12}$$

where $\bar{\mathbf{K}}_n := [\mathbf{K}_n^{\mathsf{T}}\ \bar{\mathbf{k}}_n]^{\mathsf{T}} \in \mathbb{R}^{(r_{n-1}+1) \times M}$ is the same as defined in Section 3.1. Our cost function is then given as follows:

$$\Theta_n(\mathbf{H}) := \underbrace{\frac{1}{2} d^2(\mathbf{H}, \Pi_n)}_{=: \Theta_n^{(1)}(\mathbf{H})} + \underbrace{\lambda \sum_{i=1}^{r_{n-1}+1} w_{i,n} \|\mathbf{h}_i\|}_{=: \Theta_n^{(2)}(\mathbf{H})}, \ n \in \mathbb{N}, \tag{13}$$

$$\mathbf{H} = \left[\mathbf{h}_1^{\mathsf{T}}\ \mathbf{h}_2^{\mathsf{T}} \cdots \mathbf{h}_{r_{n-1}+1}^{\mathsf{T}}\right]^{\mathsf{T}} \in \mathbb{R}^{(r_{n-1}+1) \times M}. \tag{14}$$

Here $\lambda > 0$ and

$$w_{i,n} := \begin{cases} \varepsilon, & \text{if } \|\mathbf{h}_i\| > \tau, \\ 1, & \text{otherwise,} \end{cases} \qquad i = 1, 2, \cdots, r_{n-1} + 1, \tag{15}$$

with $\varepsilon > 0$ and $\tau > 0$. *The squared distance term* $\Theta_n^{(1)}$ in (13) contributes to reducing the empirical risks. Meanwhile, *the (weighted) block ℓ_1 regularization term* $\Theta_n^{(2)}$ promotes the sparsity of the estimator at the block level, where the (block) sparsity is controlled by the parameters λ, ε, and τ. The block ℓ_1 norm has been used, as an alternative to the nowadays popular ℓ_1 regularization, for an estimator involving a block structure, such as in the grouped Lasso estimator [16], the block-sparse signal recovery [17], and the multiple kernel learning [18]. If the block size is $M = 1$, it coincides with the (weighted) ℓ_1 norm.

Since the block ℓ_1 norm is nonsmooth, the stochastic gradient approach cannot be applied to Θ_n. Noticing however that Θ_n is the

sum of smooth and nonsmooth convex functions, we can apply *the adaptive proximal forward-backward splitting method* [12].[2] Define the augmented matrix $\bar{\mathbf{H}}_n := [\mathbf{H}_n^\mathsf{T}\ \mathbf{0}_M]^\mathsf{T} \in \mathbb{R}^{(r_{n-1}+1)\times M}$. The dictionary is initialized as $\mathscr{J}_{-1}^{\mathrm{BT}} := \emptyset$, which makes $\bar{\mathbf{K}}_0 = \bar{\mathbf{k}}_0^\mathsf{T} := [\kappa_1(\mathbf{u}_0,\mathbf{u}_0), \kappa_2(\mathbf{u}_0,\mathbf{u}_0), \cdots, \kappa_M(\mathbf{u}_0,\mathbf{u}_0)]$ and $\bar{\mathbf{H}}_0 = \mathbf{0}_M^\mathsf{T}$. The proposed algorithm is given as follows:

$$\mathbf{H}_{n+1} := T\left[\mathrm{prox}_{\mu\Theta_n^{(2)}}\left(\bar{\mathbf{H}}_n - \mu\nabla\Theta_n^{(1)}(\bar{\mathbf{H}}_n)\right)\right],\ n\in\mathbb{N}. \tag{16}$$

where $\mu\in(0,2)$ is the step size parameter[3] and $\nabla\Theta_n^{(1)}$ denotes the gradient of $\Theta_n^{(1)}$. The algorithm, including the operators T and $\mathrm{prox}_{\mu\Theta_n^{(2)}}$, is elaborated below in three steps.

Step 1: Compute

$$\bar{\mathbf{H}}_n^{(1)} := \bar{\mathbf{H}}_n - \mu\nabla\Theta_n^{(1)}(\bar{\mathbf{H}}_n) = \bar{\mathbf{H}}_n + \mu\left(P_{\Pi_n}(\bar{\mathbf{H}}_n) - \bar{\mathbf{H}}_n\right), \tag{17}$$

where

$$P_{\Pi_n}(\mathbf{H}) := \underset{\mathbf{G}\in\Pi_n}{\mathrm{argmin}}\,\|\mathbf{H}-\mathbf{G}\|,\ \mathbf{H}\in\mathbb{R}^{(r_{n-1}+1)\times M}, \tag{18}$$

denotes the projection of $\mathbf{H}$ onto Π_n, and it is readily verified that

$$P_{\Pi_n}(\bar{\mathbf{H}}_n) = \bar{\mathbf{H}}_n + \frac{d_n - \langle\bar{\mathbf{K}}_n, \bar{\mathbf{H}}_n\rangle}{\|\bar{\mathbf{K}}_n\|^2}\bar{\mathbf{K}}_n. \tag{19}$$

Step 2: Compute

$$\bar{\mathbf{H}}_n^{(2)} := \mathrm{prox}_{\mu\Theta_n^{(2)}}(\bar{\mathbf{H}}_n^{(1)}), \tag{20}$$

where

$$\mathrm{prox}_{\mu\Theta_n^{(2)}} : \mathbb{R}^{(r_{n-1}+1)\times M} \to \mathbb{R}^{(r_{n-1}+1)\times M}$$
$$\mathbf{H} \mapsto \underset{\mathbf{G}\in\mathbb{R}^{(r_{n-1}+1)\times M}}{\mathrm{argmin}}\ \Theta_n^{(2)}(\mathbf{G}) + \frac{1}{2\mu}\|\mathbf{H}-\mathbf{G}\|^2 \tag{21}$$

is called *the proximity operator of* $\Theta_n^{(2)}$ *of index* μ [19]. The ith row of $\bar{\mathbf{H}}_n^{(2)}$ is computed as [20]

$$(\bar{\mathbf{H}}_n^{(2)})_i = \max\left\{1 - \frac{\lambda\mu w_{i,n}}{\left\|(\bar{\mathbf{H}}_n^{(1)})_i\right\|}, 0\right\}(\bar{\mathbf{H}}_n^{(1)})_i \in \mathbb{R}^{1\times M},$$
$$i = 1,2,\cdots,r_{n-1}+1, \tag{22}$$

where $(\cdot)_i$ denotes the ith row of matrix. The operator in (22) is essentially equivalent to the operator exploited in [21, Eq. (2.4)]; we call it *weighted block soft-thresholding*. The weighted block soft-thresholding operator eliminates the blocks (i.e., the row vectors) $(\bar{\mathbf{H}}_n^{(1)})_i$ having small norms, since such blocks make no significant contribution in estimating d_n. Accordingly the parameters ε and τ in (15) should be reasonably small so that the blocks $(\bar{\mathbf{H}}_n^{(1)})_i$ having sufficiently large norms are not seriously affected.

Step 3: Compute

$$\mathbf{H}_{n+1} = T(\bar{\mathbf{H}}_n^{(2)}) \in \mathbb{R}^{r_n\times M}, \tag{23}$$

[2]The method is an adaptive extension of the proximal forward-backward splitting method [19], which is a powerful tool for various problems including, among many others, constrained least-squares problems, multiresolution sparse regularization problems, and total variation problems.

[3]The range of step size is generally $(0, 2/\gamma)$, where $\gamma > 0$ is the Lipschitz constant of $\nabla\Theta_n^{(1)}$. In our case, the Lipschitz constant is $\gamma = 1$.

Table 1: Computational complexity and memory requirements of KNLMS, MKNLMS-CS, and MKNLMS-BT with Gaussian kernels.

	KNLMS	MKNLMS-CS	MKNLMS-BT
Multiplication	$(L+3)r_n$	$(L+3M)r_n$	$(L+5M)r_n$
Exponential	r_n	Mr_n	Mr_n
Memory	$(L+1)r_n$	$(L+M)r_n$	$(L+M)r_n$

where T is an operator that eliminates all the zero row vectors, and $r_n\in\mathbb{N}$ denotes the number of nonzero row vectors of $\bar{\mathbf{H}}_n^{(2)}$. Namely, $\mathbf{H}_{n+1}$ consists of all the nonzero row vectors of $\bar{\mathbf{H}}_n^{(2)}$. The set $\mathscr{J}_n^{\mathrm{BT}} \subset \mathscr{J}_{n-1}^{\mathrm{BT}} \cup \{n\}$ is obtained by eliminating from $\mathscr{J}_{n-1}^{\mathrm{BT}} \cup \{n\}$ all the indices that correspond to the zero row vectors of $\bar{\mathbf{H}}_n^{(2)}$.

We name the algorithm in (16) *the multi-kernel NLMS algorithm with block soft-thresholding (MKNLMS-BT).*

3.3 Computational Complexity and Remarks

We discuss the computational complexity and memory requirements of the proposed algorithms. Suppose that M Gaussian kernels with different kernel parameters α_m, $m = 1,2,\cdots,M$, are employed, as the overall complexity depends on the kernels. The complexity and memory requirements of the proposed and KNLMS algorithms are summarized in Table 1. Note that, given a $\mathbf{u}_j$, $j\in\mathscr{J}_{n-1}^{\mathrm{CS}}$, the computation of $\kappa_m(\mathbf{u}_j,\mathbf{u}_n)$ for all $m = 1,2,\cdots,M$ requires no more than $L+M$ multiplications. This is because (i) once $\|\mathbf{u}_j-\mathbf{u}_n\|^2$ is computed, it can be used for evaluating all $\kappa_m(\mathbf{u}_j,\mathbf{u}_n)$s, and (ii) $\exp(-\alpha_m)$, $m = 1,2,\cdots,M$, can be calculated and stored prior to adaptation.

As will be seen in the following section, the use of multiple kernels allows us to achieve the same amount of error as the single kernel case with a smaller value of r_n. Let $\beta\in(0,1)$ denote the ratio of the r_n value of MKNLMS-CS, or MKNLMS-BT, to that of KNLMS. Then, MKNLMS-CS, or MKNLMS-BT, has lower complexity than KNLMS provided that $L > (4\beta M-3)/(1-\beta)$, or $L > (6\beta M-3)/(1-\beta)$. Here the complexity for an exponential calculation is counted as that for a multiplication. Moreover the memory requirements of both MKNLMS-CS and MKNLMS-BT are lower than that of KNLMS provided that $L > (\beta M-1)/(1-\beta)$. We emphasize that the efficiency of MKNLMS-BT in computation and memory requirements is due to the block soft-thresholding operator. Some remarks on the proposed algorithms are given below.

Remark 1

(a) *A kernel adaptive filter is characterized as an element of RKHS associated with a kernel employed. In the multi-kernel approach, however, we have multiple RKHSs associated with the kernels. How can we characterize our estimators? Indeed the estimator of MKNLMS-CS in* (5) *is characterized as an element of the Cartesian product of the M RKHSs as* $\left(\sum_{j\in\mathscr{J}_{n-1}^{\mathrm{CS}}} h_{j,n}^{(1)}\kappa_1(\cdot,\mathbf{u}_j), \cdots, \sum_{j\in\mathscr{J}_{n-1}^{\mathrm{CS}}} h_{j,n}^{(M)}\kappa_M(\cdot,\mathbf{u}_j)\right)$. *The inner product in the product space is defined as the sum of the inner products in each RKHS; the same applies to MKNLMS-BT. It has been shown in* [20] *that a positive definite kernel can be defined in the product space and that the reproducing property of RKHS is handed down to the product space as well as the representer theorem. The proposed approach is more general than the one trying to design an optimal kernel as a convex combination of multiple kernels* [9–11].

(b) *When the data are nonstationary, kernel functions that make significant contribution in estimation tend to change from time*

to time. In such a scenario, the MKNLMS-BT algorithm discards wasted kernel functions from the dictionary. In contrast, the MKNLMS-CS algorithm keeps all the kernel functions during the whole adaptation process once they are inserted into the dictionary.

(c) *From a theoretical point of view, it would be more appropriate to present the MKNLMS-BT algorithm in the M-fold Cartesian product of the ℓ_2 space* [20]. *We take the present form as it is more convenient for implementation. The algorithm enjoys the monotone approximation property: if* $\Omega_n := \operatorname{argmin}_{\mathbf{H}\in\mathbb{R}^{(r_{n-1}+1)\times M}} \Theta_n(\mathbf{H}) \neq \emptyset$ *and* $\bar{\mathbf{H}}_n \notin \Omega_n$, *then* $\left\|\bar{\mathbf{H}}_n^{(2)} - \mathbf{H}^*\right\| < \left\|\bar{\mathbf{H}}_n - \mathbf{H}^*\right\|$, $\forall \mathbf{H}^* \in \Omega_n$.

4. NUMERICAL EXAMPLES

We compare the performance of the proposed algorithms with KNLMS [6] in online prediction of highly nonlinear time series generated by $d_n := [0.8 - 0.5\exp(-d_{n-1}^2)]d_{n-1} - [0.3 + 0.9\exp(-d_{n-1}^2)]\, d_{n-2} + 0.1\sin(d_{n-1}\pi)$ with $d_{-2} := d_{-1} := 0.1$; this is a benchmark problem described in [6]. The signals d_n are corrupted by a zero-mean Gaussian noise with variance 0.01. Each datum d_n is predicted with $\mathbf{u}_n := [d_{n-1}, d_{n-2}]^\mathsf{T}$, $n \in \mathbb{N}$, in an online fashion (i.e., $L = 2$).

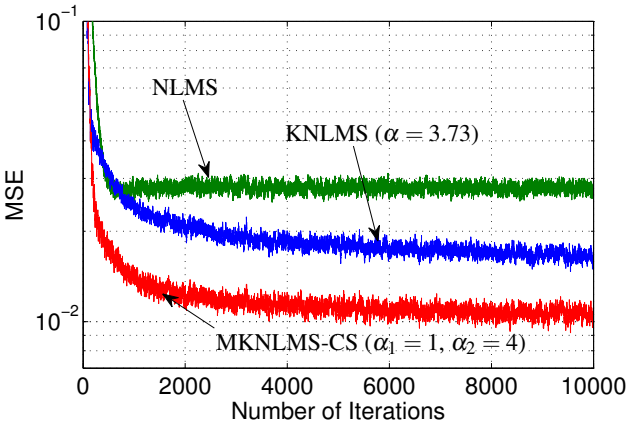
In all the simulations, we fix the step size of KNLMS, MKNLMS-CS, and MKNLMS-BT to $\eta = \eta_{\mathrm{CS}} = \mu = 9.0 \times 10^{-2}$, and the regularization parameter of KNLMS and MKNLMS-CS to $\rho = 3.0 \times 10^{-2}$ and $\rho_{\mathrm{CS}} = 3.0M \times 10^{-2}$, respectively. For the proposed algorithms, we adopt Gaussian kernels with different kernel parameters $\alpha_1, \alpha_2, \cdots, \alpha_M$. For KNLMS, we also adopt a Gaussian kernel.

Figure 1(a) depicts the learning curves of MKNLMS-CS for $M = 2$, $\alpha_1 = 1$, $\alpha_2 = 4$, and KNLMS for $\alpha = 3.73$; this choice of α is also used in [6]. The thresholds for KNLMS and MKNLMS-CS are set respectively to $\delta = 0.24$ and $\delta_{\mathrm{CS}} = 0.68$ so that the average values of r_n of both algorithms are approximately 12. MSE is calculated by taking an arithmetic average over 200 experiments. For reference, the learning curve of NLMS for the step size 0.012 is also plotted.

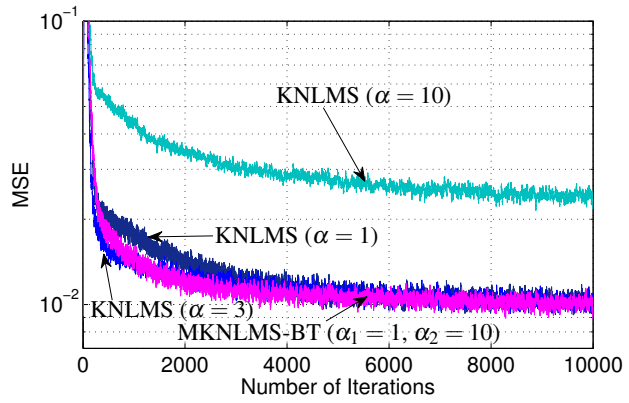
Figure 1(b) depicts the learning curves of MKNLMS-BT for $M = 2$, $\alpha_1 = 1, \alpha_2 = 10$, and KNLMS for $\alpha = 1, 3, 10$. The threshold for KNLMS is set to $\delta = 0.8,\ 0.55,\ 0.13$ so that the average values of r_n of both algorithms are approximately 20. For the proposed algorithm, we set the parameters to $\lambda = 5.0 \times 10^{-2}$, $\varepsilon = 1.0 \times 10^{-5}$, $\tau = 1.5 \times 10^{-2}$; the same values of λ and ε are used in all the experiments in this paper.

Figure 2 plots MSE against the average value of r_n for KNLMS with $\alpha = 3.73$, and MKNLMS-CS and MKNLMS-BT with $M = 2$, $\alpha_1 = 1$, $\alpha_2 = 4$. The marked points are obtained by changing δ, δ_{CS}, and τ as follows. δ is varied from 0.15 to 0.2 in increments of 0.01 and from 0.2 to 0.7 in increments of 0.02. δ_{CS} is varied from 0.3 to 0.9 in increments of 0.02. τ is varied from 0.7×10^{-2} to 3.7×10^{-2} in increments of 0.1×10^{-2}. The MSE values are obtained by averaging over the last 2,000 samples of 10,000 samples for 200 experiments.

Figure 3 plots the MSEs of KNLMS, MKNLMS-CS, and MKNLMS-BT for different choices of kernel parameters. For KNLMS, we change α within the range of $[0.5, 10]$. For MKNLMS-CS and MKNLMS-BT, we fix $\alpha_1 = 1.0$ for $M = 2$, and change α_2 within the range of $[0.5, 10]$. For comparison, we also plot the MSE of NLMS for the step size 0.012. The value of δ, δ_{CS}, or τ is chosen for each α or α_2 so that the average value of r_n becomes 20 approximately. The MSE values are calculated in the same way as in Fig. 2.



(a) The average values of r_n are 12 approximately



(b) The average values of r_n are 20 approximately

Figure 1: Learning curves of NLMS (green), KNLMS (light/normal/dark blue), MKNLMS-CS (red), and MKNLMS-BT (magenta).

The simulation results obtained suggest the following advantages of the proposed algorithms.

1. Given a value of MSE, the multi-kernel approach could reduce the average value of r_n compared to KNLMS. This implies the computational efficiency of the proposed approach when L becomes large (see Section 3.3).
2. Given an average value of r_n, the multi-kernel approach could improve the MSE performance compared to KNLMS. The improvement is considerable particularly when the average value of r_n is low. For instance, the difference in decibel between the MSEs of KNLMS and MKNLMS-CS averaged over the last 2,000 samples in Fig. 1(a) is approximately 1.8 [dB].
3. Compared to the single kernel approach with a 'good' kernel parameter, the multi-kernel approach could achieve comparable (or better) performance without specifying a good kernel parameter. In other words, the proposed algorithm is *insensitive to the choice of kernel parameters (hence reducing the efforts for the kernel-parameter selection)*. This implies that the proposed approach is advantageous when data are nonstationary and a good kernel parameter changes from time to time. In such scenarios, in addition to the expected advantage in performance, MKNLMS-BT will bring a particular advantage in computational complexity and memory requirements (see Remark 1(b)).

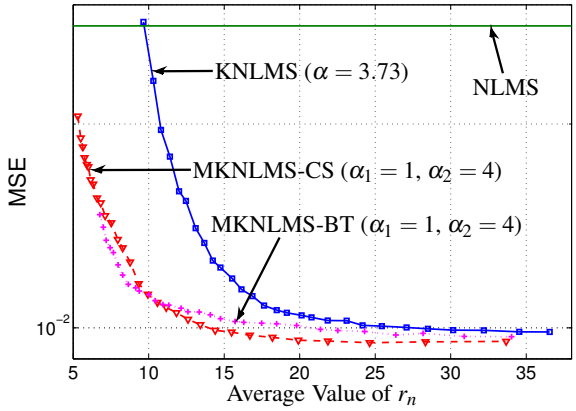


Figure 2: MSE against the average value of r_n.

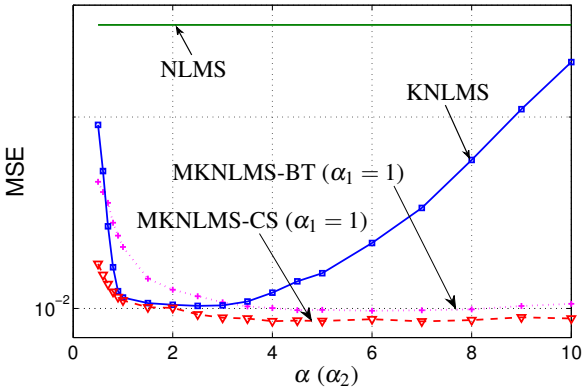


Figure 3: MSE against kernel parameters. The average values of r_n are 20 approximately.

5. CONCLUSION

The present study has exemplified that the use of multiple kernels leads to efficient adaptive filtering for nonlinear systems. We have presented two nonlinear adaptive filtering algorithms using multiple kernels. The first algorithm (MKNLMS-CS) exploits the coherence-based criterion for dictionary designing, while the second algorithm (MKNLMS-BT) exploits the weighted block soft-thresholding operator. The potential advantages of the proposed algorithms have been discussed on the ground of the numerical examples. Further investigations will be required to clarify comparative merits and demerits of the MKNLMS-CS and MKNLMS-BT algorithms. We repeat finally that there are other possible approaches, to be investigated, to exploiting multiple kernels in adaptive filtering.

Acknowledgements: This work was partially supported by KDDI Foundation.

REFERENCES

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

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

[3] A. V. 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.

[4] W. Liu and J. Príncipe, "Kernel affine projection algorithms," *EURASIP J. Adv. Signal Process.*, vol. 2008, pp. 1–12, 2008, Article ID 784292.

[5] K. Slavakis, S. Theodoridis, and I. Yamada, "Online kernel-based classification using adaptive projection algorithms," *IEEE Trans. Signal Process.*, vol. 56, no. 7, pp. 2781–2796, July 2008.

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

[7] S. Theodoridis, K. Slavakis, and I. Yamada, "Adaptive learning in a world of projections: a unifying framework for linear and nonlinear classification and regression tasks," *IEEE Signal Processing Magazine*, vol. 28, no. 1, pp. 97–123, Jan. 2011.

[8] W. Liu, J. Príncipe, and S. Haykin, *Kernel Adaptive Filtering*, Wiley, New Jersey, 2010.

[9] N. Cristianini, A. Elisseeff, J. Shawe-Taylor, and J. Kandla, "On kernel target alignment," *Adv. Neural Information Processing Systems (NIPS)*, vol. 13, pp. 367–373, 2001.

[10] G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M. I. Jordan, "Learning the kernel matrix with semi-definite programming," *J. Mach. Learn. Res.*, vol. 5, pp. 27–72, 2004.

[11] C. S. Ong, A. J. Smola, and R. C. Williamson, "Learning the kernel with hyperkernels," *J. Mach. Learn. Res.*, vol. 6, pp. 1043–1071, 2005.

[12] Y. Murakami, M. Yamagishi, M. Yukawa, and I. Yamada, "A sparse adaptive filtering using time-varying soft-thresholding techniques," in *Proc. IEEE ICASSP*, 2010, pp. 3734–3737.

[13] N. Aronszajn, "Theory of reproducing kernels," *Trans. Amer. Math. Soc.*, vol. 68, no. 3, pp. 337–404, May 1950.

[14] V. N. Vapnik, *Statistical Learning Theory*, Wiley, New York, 1998.

[15] B. Schölkopf and A. J. Smola, *Learning with Kernels*, MIT Press, Cambridge, MA, 2001.

[16] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, Springer, New York, 2nd edition, 2009.

[17] M. Stojnic, F. Parvaresh, and B. Hassibi, "On the reconstruction of block-sparse signals with an optimal number of measurements," *IEEE Trans. Signal Process.*, vol. 57, no. 8, pp. 3075–3085, Aug. 2009.

[18] F. R. Bach, "Consistency of the group lasso and multiple kernel learning," *J. Mach. Learn. Res.*, vol. 9, pp. 1179–1225, 2008.

[19] P. L. Combettes and V. R. Wajs, "Signal recovery by proximal forward-backward splitting," *SIAM Journal on Multiscale Modeling and Simulation*, vol. 4, pp. 1168–1200, 2005.

[20] M. Yukawa, "On use of multiple kernels in adaptive learning —Extended reproducing kernel Hilbert space with Cartesian product," in *Proc. IEICE Signal Processing Symposium*, Nov. 2010, pp. 59–64.

[21] M. Yuan and Y. Lin, "Model selection and estimation in regression with grouped variables," *J. R. Statist. Soc. B*, vol. 68, no. 1, pp. 49–67, 2006.