AN EFFICIENT KERNEL ADAPTIVE FILTERING ALGORITHM USING HYPERPLANE PROJECTION ALONG AFFINE SUBSPACE

Masahiro Yukawa and Ryu-ichiro Ishii

Department of Electrical and Electronic Engineering, Niigata University, JAPAN

ABSTRACT

We propose a novel kernel adaptive filtering algorithm that selectively updates a few coefficients at each iteration by projecting the current filter onto the zero instantaneous-error hyperplane along a certain time-dependent affine subspace. Coherence is exploited for selecting the coefficients to be updated as well as for measuring the novelty of new data. The proposed algorithm is a natural extension of the normalized kernel least mean squares algorithm operating iterative hyperplane projections in a reproducing kernel Hilbert space. The proposed algorithm enjoys low computational complexity. Numerical examples indicate high potential of the proposed algorithm.

Index Terms— kernel adaptive filter, projection algorithms, reproducing kernel Hilbert space, normalized kernel least mean square algorithm

1. INTRODUCTION

Kernel adaptive filtering has received considerable attention as an attractive approach to nonlinear function estimation tasks [1–9]. The existing algorithms can be classified into two general categories [9]: the reproducing kernel Hilbert space (RKHS) approach and the parameter-space approach (see Section 2). The RKHS approach would be more reasonable from the function approximation point of view. Its major drawback is however that the filter is updated only when a new datum is added into the dictionary, although a certain amount of computation is required at every iteration for dictionary construction.

In this paper, we propose *the hyperplane projection along affine subspace (HYPASS) algorithm* which falls into the RKHS approach. The key is that the filter is updated at every iteration because such data that are discarded in the dictionary-construction process are exploited to adjust the coefficients of the present dictionary elements so that the instantaneous error for that specific data becomes nearly zero. This is accomplished by projecting the current filter onto the zero instantaneous-error hyperplane along the subspace spanned by the dictionary elements. When new datum is added into the dictionary, the algorithm is automatically reduced to the the normalized kernel least mean square algorithm [7, Chapter 2]. The proposed algorithm is thus systematic unlike a heuristic way of combining the RKHS and parameter-space approaches. To reduce the computational complexity of the algorithm, a low-complexity version is derived by introducing the mechanism of selectively updating only a few coefficients associated with the dictionary elements that are maximally coherent to the newly observed data. The selective update is realized by replacing the subspace used in the fully-updating algorithm by its subset which is an affine subspace. The proposed selectively-updating algorithm (HYPASS) includes the fully-updating algorithm as its particular case. The major benefits of the proposed algorithm include that a simple criterion can be adopted for dictionary construction and that high estimation accuracy can be achieved with a reasonably small size of dictionary. This is because each coefficient is polished many times according to the incoming data in the best way in the sense of the minimal disturbance in RKHS. We therefore adopt the simple coherence criterion [6] for the dictionary construction and the selection of coefficients to be updated. Numerical examples indicate high potential of the proposed algorithm.

This work was supported by KDDI Foundation. The author would like to thank Prof. C. Richard of the University of Nice Sophia-Antipolis, France, for offering information about the paper [1].

2. KERNEL ADAPTIVE FILTER AND GENERAL CLASSIFICATION OF EXISTING ALGORITHMS

We address the general problem of estimating an unknown nonlinear function $\psi : \mathcal{U} \to \mathbb{R}$ adaptively according to the input vector $\boldsymbol{u}_n \in \mathcal{U}$ and the output $d_n := \psi(\boldsymbol{u}_n) \in \mathbb{R}$ which are observed sequentially. Here $\mathcal{U} \subset \mathbb{R}^L$ denotes the input space. In kernel adaptive filtering, the nonlinear function ψ is estimated in the following form:

$$\varphi_n(\boldsymbol{u}) := \sum_{j \in \mathcal{J}_n} h_{j,n} \kappa(\boldsymbol{u}, \boldsymbol{u}_j),\ \boldsymbol{u} \in \mathcal{U}, \quad n \in \mathbb{N}, \qquad (1)$$

where $\kappa : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$ is a positive definite kernel, $\mathcal{J}_n := \{j_1^{(n)}, j_2^{(n)}, \cdots, j_{r_n}^{(n)}\} \subset \{0, 1, \cdots, n\}$ is an index set indicating the *dictionary* $\{\kappa(\cdot, \boldsymbol{u}_j)\}_{j \in \mathcal{J}_n}$ (r_n is the dictionary size at time n), and $h_{j,n} \in \mathbb{R}$ is a coefficient of $\kappa(\cdot, \boldsymbol{u}_j)$ at time instant n. (Due to the limitation in memory and computational resources, it is practically infeasible to exploit all the data and hence a selection of data is required to form a dictionary; see Section 3.) Assume for simplicity that a Gaussian kernel $\kappa(\boldsymbol{x}, \boldsymbol{y}) := \exp\left(-\zeta \|\boldsymbol{x} - \boldsymbol{y}\|^2\right)$, $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{U}$, is employed, although any other kernel can be employed. Here, $\zeta > 0$ is the kernel parameter and $\|\cdot\|$ stands for the Euclidean norm which is induced by the standard inner product $\langle \cdot, \cdot \rangle$. In this case, $\kappa(\cdot, \boldsymbol{u}_j)$ is a Gaussian function centered at $\boldsymbol{u}_j$. We denote by $\mathcal{H}$ the reproducing kernel Hilbert space (RKHS) associated with κ and $\mathcal{U}$. Also denote by $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ and $\|\cdot\|_{\mathcal{H}}$ the inner product and its induced norm defined in $\mathcal{H}$, respectively.

Definition 1 (Metric projection)**.** *Let $\mathcal{X}$ be a real Hilbert space and $\|\cdot\|_{\mathcal{X}}$ the norm defined in $\mathcal{X}$. Also let $C \subset \mathcal{X}$*

be a closed convex subset of $\mathcal{X}$. Then, for any point $x \in \mathcal{X}$, there exists the unique point $x^* \in C$ such that $\|x - x^*\|_{\mathcal{X}} \leq \|y - x^*\|_{\mathcal{X}}$, $\forall y \in C$ [10]. The point x^* is called the metric projection of x onto C and denoted by $P_C(x)$.

From the vector space projection viewpoint, we can classify the existing kernel adaptive filtering algorithms into two general categories, each of which is represented by the following update equations:

$$\varphi_{n+1} = \varphi_n + \mu\left(P_{\Pi_n}(\varphi_n) - \varphi_n\right), \tag{2}$$

$$\boldsymbol{h}_{n+1} = \boldsymbol{h}_n + \mu\left(P_{H_n}(\boldsymbol{h}_n) - \boldsymbol{h}_n\right). \tag{3}$$

Here, $\mu \in (0, 2)$ is the step size and

$$\Pi_n := \{g \in \mathcal{H} : g(\boldsymbol{u}_n) = \langle g, \kappa(\cdot, \boldsymbol{u}_n)\rangle_{\mathcal{H}} = d_n\}, \tag{4}$$

$$H_n := \{\boldsymbol{h} \in \mathbb{R}^{r_n} : \langle \boldsymbol{h}, \boldsymbol{k}_n\rangle = d_n\}, \tag{5}$$

are hyperplanes in $\mathcal{H}$ and $\mathbb{R}^{r_n}$, respectively, with $\boldsymbol{h}_n := [h_{j_1^{(n)},n}, h_{j_2^{(n)},n}, \cdots, h_{j_{r_n}^{(n)},n}]^{\mathsf{T}}$ and $\boldsymbol{k}_n := [\kappa(\boldsymbol{u}_{j_1^{(n)}}, \boldsymbol{u}_n), \kappa(\boldsymbol{u}_{j_2^{(n)}}, \boldsymbol{u}_n), \cdots, \kappa(\boldsymbol{u}_{j_{r_n}^{(n)}}, \boldsymbol{u}_n)]^{\mathsf{T}}$; the superscript $(\cdot)^{\mathsf{T}}$ stands for transposition. The algorithm in (2) is the normalized kernel least mean square algorithm presented in [7, Chapter 2], and the algorithm in (3) is the kernel normalized least mean square algorithm proposed in [6]. The algorithm in (2) is based on the RKHS-inner-product expression $\varphi_n(\boldsymbol{u}_n) := \langle \varphi_n, \kappa(\cdot, \boldsymbol{u}_n)\rangle_{\mathcal{H}}$ of the filter output, and we refer to those algorithms based on this expression as *the RKHS approach*. The algorithms presented in [1,2,4,5,8] fall into this approach. On the other hand, the algorithm in (3) is based on the parameter-space- (Euclidean-space-) inner-product expression $\varphi_n(\boldsymbol{u}_n) := \langle \boldsymbol{h}_n, \boldsymbol{k}_n\rangle$, and we refer to those algorithms based on this another inner-product expression as *the parameter-space approach*. The algorithms presented in [3,6,9] fall into this approach.

3. FULLY-UPDATING HYPASS ALGORITHM

In the parameter-space algorithm in (3), the update direction is given by the normal vector $\boldsymbol{k}_n$ of H_n (or its negative), meaning that the projection $P_{H_n}(\boldsymbol{h}_n)$ is always feasible and that all the coefficients $h_{j,n}$ are updated at every iteration. In contrast, in the RKHS algorithm in (2), the update direction is given by the normal vector $\kappa(\cdot, \boldsymbol{u}_n)$ of Π_n (or its negative), meaning that the projection $P_{\Pi_n}(\varphi_n)$ is feasible (and thus the filter is updated) only when the new datum is added into the dictionary. Another remarkable difference from the parameter-space algorithm is that only the coefficient for the newly added function $\kappa(\cdot, \boldsymbol{u}_n)$ is updated. In the RKHS algorithm for $\mu = 1$, the updated vector $\varphi_{n+1} = P_{\Pi_n}(\varphi_n)$ is the closest point in $\mathcal{H}$ from the current filter φ_n that makes an instantaneous error be zero. The projection viewpoint presented above brings a natural idea to extend the RKHS algorithm so that the projection becomes always feasible as explained below.

Let $\mathcal{J}_{-1} := \emptyset$. The dictionary index set $\mathcal{J}_n$ is defined as

$$\mathcal{J}_n := \begin{cases} \mathcal{J}_{n-1} \cup \{n\} & \text{if new datum } \boldsymbol{u}_n \text{ is sufficiently novel} \\ \mathcal{J}_{n-1} & \text{otherwise.} \end{cases}$$

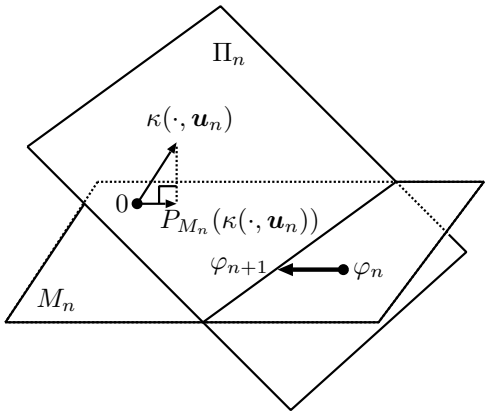

Fig. 1. A geometric interpretation of Algorithm 1 for $\mu = 1$.

For the novelty criterion, we adopt the coherence criterion [6] due to its simplicity, although another criterion can be exploited for better performance. We define the subspace spanned by the dictionary elements at time instant n as follows:

$$M_n := \operatorname{span}\{\kappa(\cdot, \boldsymbol{u}_j)\}_{j \in \mathcal{J}_n},$$

which may or may not include the Gaussian function $\kappa(\cdot, \boldsymbol{u}_n)$ centered at the current input vector $\boldsymbol{u}_n$. Since the filter is restricted to the subspace M_n due to the limitation in memory and computational resources, our primitive idea is given as follows: find the closest point, from the current filter φ_n in M_n, that makes an instantaneous error be zero (following the minimal disturbance principle). The problem is formulated as follows: $\text{minimize}_{f \in M_n} \|f - \varphi_n\|_{\mathcal{H}}$ subject to $f(\boldsymbol{u}_n) = d_n$, or equivalently

$$\min_{f \in M_n \cap \Pi_n} \|f - \varphi_n\|_{\mathcal{H}}, \tag{6}$$

where Π_n is defined in (4). The solution of the problem in (6) is given by $P_{M_n \cap \Pi_n}(\varphi_n)$ which is the orthogonal projection of φ_n onto the intersection $M_n \cap \Pi_n$. Due to the use of the coherence criterion, the problem in (6) is always feasible since the intersection $M_n \cap \Pi_n$ is ensured to be a nonempty affine subspace. Based on this orthogonal projection, the proposed algorithm is given as follows.

Algorithm 1. *For the initial estimate $\varphi_0 := 0$, update the nonlinear filter φ_n at each time instant $n \in \mathbb{N}$ by*

$$\varphi_{n+1} := \varphi_n + \mu\left(P_{M_n \cap \Pi_n}(\varphi_n) - \varphi_n\right), \quad n \in \mathbb{N}, \tag{7}$$

where $\mu \in (0, 2)$ is the step size.

We can show that (see Appendix)

$$P_{M_n \cap \Pi_n}(\varphi_n) = \varphi_n + \beta_n P_{M_n}(\kappa(\cdot, \boldsymbol{u}_n)) \tag{8}$$

for some $\beta_n \in \mathbb{R}$. The projection $P_{M_n}(\kappa(\cdot, \boldsymbol{u}_n))$ is written as

$$P_{M_n}(\kappa(\cdot, \boldsymbol{u}_n)) = \sum_{j \in \mathcal{J}_n} \alpha_j \kappa(\cdot, \boldsymbol{u}_j), \quad \alpha_j \in \mathbb{R}. \tag{9}$$

By (7)–(9), we obtain

$$\varphi_{n+1} := \sum_{j \in \mathcal{J}_n} (h_{j,n} + \mu\beta_n\alpha_j)\kappa(\cdot, \boldsymbol{u}_j). \tag{10}$$

The coefficient vector $\boldsymbol{\alpha} := \left[\alpha_{j_1^{(n)}}, \alpha_{j_2^{(n)}}, \cdots, \alpha_{j_{r_n}^{(n)}}\right]^{\mathsf{T}} \in \mathbb{R}^{r_n}$ is characterized as a solution to the following normal equation [10]:

$$\boldsymbol{K}_n \boldsymbol{\alpha} = \boldsymbol{y}_n, \tag{11}$$

where

$$\boldsymbol{K}_n := \begin{bmatrix} \kappa\left(\boldsymbol{u}_{j_1^{(n)}}, \boldsymbol{u}_{j_1^{(n)}}\right) & \cdots & \kappa\left(\boldsymbol{u}_{j_1^{(n)}}, \boldsymbol{u}_{j_{r_n}^{(n)}}\right) \\ \vdots & \ddots & \vdots \\ \kappa\left(\boldsymbol{u}_{j_{r_n}^{(n)}}, \boldsymbol{u}_{j_1^{(n)}}\right) & \cdots & \kappa\left(\boldsymbol{u}_{j_{r_n}^{(n)}}, \boldsymbol{u}_{j_{r_n}^{(n)}}\right) \end{bmatrix}, \tag{12}$$

$$\boldsymbol{y}_n := \left[\kappa\left(\boldsymbol{u}_{j_1^{(n)}}, \boldsymbol{u}_n\right), \cdots, \kappa\left(\boldsymbol{u}_{j_{r_n}^{(n)}}, \boldsymbol{u}_n\right)\right]^{\mathsf{T}}. \tag{13}$$

Substituting (9) into (8) and then substituting $g = P_{M_n \cap \Pi_n}(\varphi_n)$ into $g(\boldsymbol{u}_n) = d_n$ appearing in (4), we obtain with simple manipulations

$$\beta_n = \frac{d_n - \varphi_n(\boldsymbol{u}_n)}{\sum_{j \in \mathcal{J}_n} \alpha_j \kappa(\boldsymbol{u}_n, \boldsymbol{u}_j)}. \tag{14}$$

A geometric interpretation of Algorithm 1 is presented in Fig. 1. One can intuitively understand (8) by observing that the displacement vector from the current filter φ_n to its projection φ_{n+1} onto the intersection $M_n \cap \Pi_n$ is given by scaling the projection of the normal vector $\kappa(\cdot, \boldsymbol{u}_n)$ of Π_n onto the subspace M_n. In the special case that $\kappa(\cdot, \boldsymbol{u}_n) \in M_n$, Algorithm 1 is reduced to the algorithm in (2). This implies that Algorithm 1 is a natural extension of the algorithm in (2) so that the coefficients are updated at every iteration no matter if observed data are added into the dictionary or not.

Although Algorithm 1 exhibits excellent performance (as will be seen in Section 5), it involves the inversion of the $r_n \times r_n$ matrix $\boldsymbol{K}_n$ which could be prohibitive when the dictionary size r_n is large. In the following section, we introduce a selectively-updating mechanism to Algorithm 1, which turns out to reduce the computational complexity remarkably while maintaining reasonable performance.

4. THE HYPASS ALGORITHM

The key idea for the selective update is the following: pick up only a few, say Q, coefficients that are maximally coherent to the current data and update only the selected coefficients. To be precise, we choose the subset $\mathcal{I}_n := \{\iota_1^{(n)}, \iota_2^{(n)}, \cdots, \iota_Q^{(n)}\} \subset \mathcal{J}_n$ such that $\kappa(\cdot, \boldsymbol{u}_\iota)$, $\iota \in \mathcal{I}_n$, has the largest coherence to $\kappa(\cdot, \boldsymbol{u}_n)$; i.e.,

$$\kappa(\boldsymbol{u}_\iota, \boldsymbol{u}_n) \geq \kappa(\boldsymbol{u}_j, \boldsymbol{u}_n), \quad \forall \iota \in \mathcal{I}_n, \quad \forall j \in \mathcal{J}_n \setminus \mathcal{I}_n, \tag{15}$$

provided that $r_n > Q$ (i.e., the dictionary size is larger than Q). If $r_n \leq Q$, all the coefficients are updated; i.e., $\mathcal{I}_n := \mathcal{J}_n$. To update the coefficients associated with the index set $\mathcal{I}_n$ and keep the other coefficients unchanged, the displacement vector from the current filter φ_n to the next one φ_{n+1} should lie in the subspace

$$\tilde{M}_n := \operatorname{span}\{\kappa(\cdot, \boldsymbol{u}_\iota)\}_{\iota \in \mathcal{I}_n} \subset M_n. \tag{16}$$

As such, φ_{n+1} is constrained in the affine subspace

$$V_n := \varphi_n + \tilde{M}_n := \left\{\varphi_n + f : f \in \tilde{M}_n\right\}. \tag{17}$$

The primitive idea of the proposed selectively-updating algorithm is quite similar to that of Algorithm 1: find the closest point, from the current filter φ_n in the affine subspace V_n, that makes an instantaneous error be zero. Such a point is obviously given by the projection of φ_n onto the intersection $V_n \cap \Pi_n$, leading to the following algorithm.

Algorithm 2. *For the initial estimate $\varphi_0 := 0$, update the nonlinear filter φ_n at each time instant $n \in \mathbb{N}$ by*

$$\varphi_{n+1} := \varphi_n + \mu \left(P_{V_n \cap \Pi_n}(\varphi_n) - \varphi_n\right), \tag{18}$$

where $\mu \in (0, 2)$ is the step size.

We can show that (see Appendix)

$$P_{V_n \cap \Pi_n}(\varphi_n) = \varphi_n + \tilde{\beta}_n P_{\tilde{M}_n}(\kappa(\cdot, \boldsymbol{u}_n)) \tag{19}$$

for some constant $\tilde{\beta}_n \in \mathbb{R}$. The projection $P_{\tilde{M}_n}(\kappa(\cdot, \boldsymbol{u}_n))$ is written in the following form:

$$P_{\tilde{M}_n}(\kappa(\cdot, \boldsymbol{u}_n)) = \sum_{\iota \in \mathcal{I}_n} \tilde{\alpha}_\iota \kappa(\cdot, \boldsymbol{u}_\iota), \quad \tilde{\alpha}_\iota \in \mathbb{R}. \tag{20}$$

By (18)–(20), we obtain

$$\varphi_{n+1} := \sum_{\iota \in \mathcal{I}_n} (h_{\iota,n} + \mu \tilde{\beta}_n \tilde{\alpha}_\iota) \kappa(\cdot, \boldsymbol{u}_\iota) + \sum_{j \in \mathcal{J}_n \setminus \mathcal{I}_n} h_{j,n} \kappa(\cdot, \boldsymbol{u}_j). \tag{21}$$

The coefficient vector $\tilde{\boldsymbol{\alpha}} := \left[\tilde{\alpha}_{\iota_1^{(n)}}, \tilde{\alpha}_{\iota_2^{(n)}}, \cdots, \tilde{\alpha}_{\iota_{p_n}^{(n)}}\right]^{\mathsf{T}}$, where $p_n := \min\{r_n, Q\}$, is obtained by solving

$$\tilde{\boldsymbol{K}}_n \tilde{\boldsymbol{\alpha}} = \tilde{\boldsymbol{y}}_n, \tag{22}$$

where

$$\tilde{\boldsymbol{K}}_n := \begin{bmatrix} \kappa\left(\boldsymbol{u}_{\iota_1^{(n)}}, \boldsymbol{u}_{\iota_1^{(n)}}\right) & \cdots & \kappa\left(\boldsymbol{u}_{\iota_1^{(n)}}, \boldsymbol{u}_{\iota_{p_n}^{(n)}}\right) \\ \vdots & \ddots & \vdots \\ \kappa\left(\boldsymbol{u}_{\iota_{p_n}^{(n)}}, \boldsymbol{u}_{\iota_1^{(n)}}\right) & \cdots & \kappa\left(\boldsymbol{u}_{\iota_{p_n}^{(n)}}, \boldsymbol{u}_{\iota_{p_n}^{(n)}}\right) \end{bmatrix}, \tag{23}$$

$$\tilde{\boldsymbol{y}}_n := \left[\kappa\left(\boldsymbol{u}_{\iota_1^{(n)}}, \boldsymbol{u}_n\right), \cdots, \kappa\left(\boldsymbol{u}_{\iota_{p_n}^{(n)}}, \boldsymbol{u}_n\right)\right]^{\mathsf{T}}. \tag{24}$$

The constant $\tilde{\beta}_n$ is given by

$$\tilde{\beta}_n = \frac{d_n - \varphi_n(\boldsymbol{u}_n)}{\sum_{\iota \in \mathcal{I}_n} \tilde{\alpha}_\iota \kappa(\boldsymbol{u}_n, \boldsymbol{u}_\iota)}. \tag{25}$$

The case that $Q = 1$ is of particular interest because the algorithm becomes particularly simple as follows:

$$\varphi_{n+1} = \varphi_n + \mu \frac{d_n - \varphi_n(\boldsymbol{u}_n)}{\kappa(\boldsymbol{u}_n, \boldsymbol{u}_\iota)} \kappa(\cdot, \boldsymbol{u}_\iota), \tag{26}$$

where $\kappa(\cdot, \boldsymbol{u}_\iota)$ has the maximum coherence to $\kappa(\cdot, \boldsymbol{u}_n)$ among $\{\kappa(\cdot, \boldsymbol{u}_j)\}_{j\in\mathcal{J}_n}$ ($n \in \mathcal{J}_n \Rightarrow \iota = n$). Despite its simplicity, the algorithm works well reasonably as shown in Section 5.

The summary of Algorithm 2 is presented in Table 1. Clearly, Algorithm 1 is a particular case of Algorithm 2 for $Q = \infty$. We name Algorithm 2 *the HYperplane Projection along Affine SubSpace (HYPASS) algorithm*. A geometric interpretation of Algorithm 2 is presented in Fig. 2. As we choose the set of vectors $\{\kappa(\cdot, \boldsymbol{u}_\iota)\}_{\iota\in\mathcal{I}_n}$ that are maximally coherent to $\kappa(\cdot, \boldsymbol{u}_n)$, the projection $P_{\tilde{M}_n}(\varphi_n)$ employed in Algorithm 2 is expected to be a good approximation of the projection $P_{M_n}(\varphi_n)$ which is employed in Algorithm 1. This intuition is supported by the numerical examples presented in Section 5. We emphasize that coherence is an efficient but not the only criterion of selecting $\mathcal{J}_n$ and $\mathcal{I}_n$ in HYPASS, and a better criterion could be devised.

Relation to some previous works: Algorithms related to HYPASS have been proposed in [1,8]. The algorithm in [1] updates the nonlinear filter in the same direction as HYPASS for $Q = \infty$ but HYPASS has a wider range of step size in addition that the criterion of designing the dictionary is different. On the other hand, the quantized kernel least mean square (QKLMS) algorithm in [1] is related to HYPASS for $Q = 1$, and the difference is such as LMS and normalized LMS; QKLMS has no denominator $\kappa(\boldsymbol{u}_n, \boldsymbol{u}_\iota)$ in (26).

Computational complexity: The number of multiplications involved in the filter update by Algorithm 2 is $(Q^2-Q)L/2+O(Q^3)+Q^2+2Q$. When Q is sufficiently small, this is negligible compared to the r_nL multiplications required for computing the filter output. (A typical value of Q for achieving reasonable performance is $Q \leq 3$.) In addition, the algorithm requires comparison operations for the dictionary construction and the $\mathcal{I}_n$ construction both of which are based on the coherence. When $Q = 1$, the total number of comparisons to be performed is only r_n, which is the same as the number of comparisons required solely for the dictionary construction. This is because one can operate $r_n - 1$ comparisons to find such a $\kappa(\cdot, \boldsymbol{u}_{j^*})$, $j^* \in \mathcal{J}_n$, that has the largest coherence to $\kappa(\cdot, \boldsymbol{u}_n)$ and then compare the value of $\kappa(\boldsymbol{u}_{j^*}, \boldsymbol{u}_n)$ with the threshold δ. If $\kappa(\boldsymbol{u}_{j^*}, \boldsymbol{u}_n) \leq \delta$, then $\kappa(\cdot, \boldsymbol{u}_n)$ is added into the dictionary and $\mathcal{I}_n := \{n\}$. Otherwise $\kappa(\cdot, \boldsymbol{u}_n)$ is not added into the dictionary and $\mathcal{I}_n := \{j^*\}$. In the general case, the algorithm requires no more than $r_n+(Q-1)\,(r_n - (Q+2)/2)$ comparisons.

5. NUMERICAL EXAMPLES

The performance of the proposed algorithm is evaluated in the application to online prediction of the time-series data generated by $d_n := [0.8 - 0.5\exp(-d_{n-1}^2)]d_{n-1} - [0.3 + 0.9\exp(-d_{n-1}^2)]d_{n-2} + 0.1\sin(d_{n-1}\pi)$ for $d_{-2} := d_{-1} := 0.1$. We predict each datum d_n by a kernel adaptive filter with its input $\boldsymbol{u}_n := [\hat{d}_{n-1}, \hat{d}_{n-2}]^\mathsf{T} \in \mathcal{U} \subset \mathbb{R}^L$ ($L = 2$), where $\hat{d}_n := d_n + \nu_n$, $n \in \mathbb{N}$, where $\nu_n \sim \mathcal{N}(0, 0.01)$. The proposed algorithm is compared with the RKHS hyperplane-projection algorithm [7, Chapter 2] in (2), the parameter-space hyperplane-projection algorithm [6] in (3), and another RKHS algorithm, QKLMS [8]. The first two conventional algorithms are referred to simply as the RKHS algorithm and the parameter-space algorithm, respectively. We adopt the Platt's criterion for the RKHS algorithm and the coherence criterion for the proposed and parameter-space algorithms. The Euclidean-distance criterion of QKLMS is equivalent to the coherence criterion in the case of Gaussian kernel. Throughout the simulations, the kernel parameter is set to $\zeta = 2.0$.

The step size is set to $\mu = 0.1$ for the proposed algorithm, $\mu = 1.1$ for the parameter-space and QKLMS algorithms, and $\mu = 0.5$ for the RKHS algorithm.[1] The step size values were chosen in such a way that a further decrease of step size brings little improvements of steady-state performance. The coherence threshold $\delta > 0$ is set to $\delta = 0.7$ and its equivalent distance-threshold 0.4223 is used for QKLMS. The distance threshold $\delta_1 > 0$ and the error threshold $\delta_2 > 0$ for the Platt's criterion are set respectively to $\delta_1 = 0.4$ and $\delta_2 = 0.2$. The threshold values were chosen in such a way that a further increase of dictionary size brings little improvements of performance. We test 300 independent runs by generating the noise randomly and the mean squared error (MSE) is computed by averaging the instantaneous squared errors over the

Table 1. Summary of Algorithm 2.

The HYPASS Algorithm
Requirement : step size $\mu \in [0, 2]$
Initialization : $\mathcal{J}_{-1} := \emptyset$
Filter output : $\varphi_n(\boldsymbol{u}_n) := \sum_{j\in\mathcal{J}_n} h_{j,n}\kappa(\boldsymbol{u}_n, \boldsymbol{u}_j)$
Filter update :
1. Define $\mathcal{J}_n$ based, e.g., on the coherence criterion.
2. If $n \in \mathcal{J}_n$, let $h_{n,n} := 0$.
3. If $r_n > Q$, define $\mathcal{I}_n := \{\iota_1^{(n)}, \iota_2^{(n)}, \cdots, \iota_Q^{(n)}\} \subset \mathcal{J}_n$ based, e.g., on (15). Otherwise, let $\mathcal{I}_n := \mathcal{J}_n$.
4. Compute $\tilde{\alpha}_\iota$, $\iota \in \mathcal{I}_n$, by (22)–(24).
5. Compute $\tilde{\beta}_n$ by (25).
6. Update the coefficients by $h_{\iota,n+1} := h_{\iota,n} + \mu\tilde{\beta}_n\tilde{\alpha}_\iota$ for all $\iota \in \mathcal{I}_n$ (see (21)).

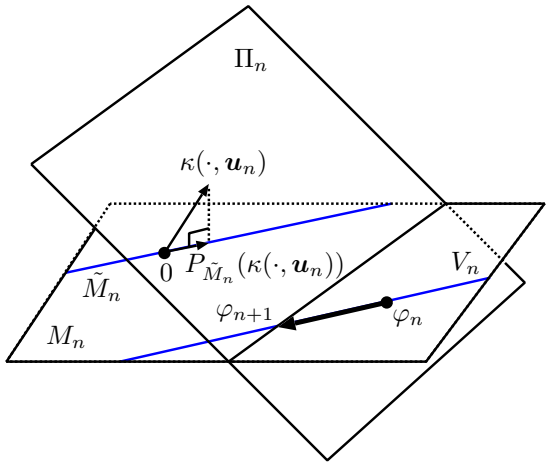

Fig. 2. A geometric interpretation of Algorithm 2 for $\mu = 1$.

[1] The step size of the RKHS algorithm is larger than the other algorithms because it updates the coefficients only when the new datum is added into the dictionary, and thus the use of smaller step size yields extremely poor performance.

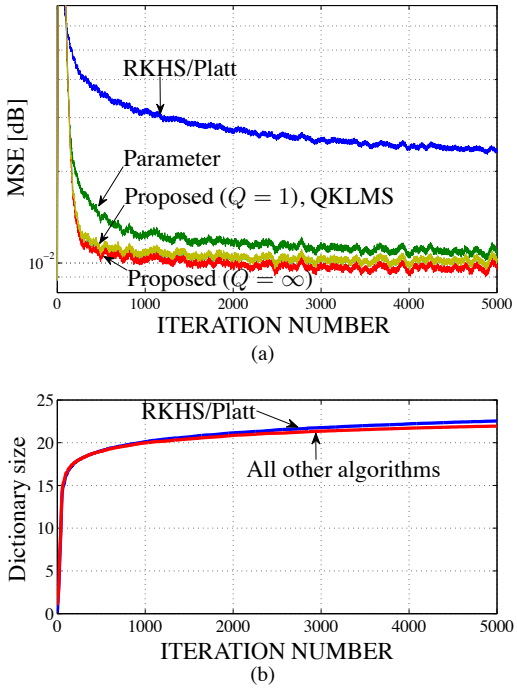

Fig. 3. MSE learning curves and dictionary-size evolutions.

300 runs. Figure 3 depicts the MSE learning curves and the evolution of the dictionary size r_n. It is seen that the proposed algorithm outperforms the RKHS and parameter-space algorithms. In this specific case, moreover, we observe that the proposed algorithm for $Q = 1$ exhibits (i) overall MSE nearly identical to QKLMS and (ii) steady-state MSE slightly higher than the fully-updating version ($Q = \infty$) despite its reasonable complexity. The reason for the former observation is that $\kappa(\boldsymbol{u}_n, \boldsymbol{u}_\iota) \approx 1$ in (26) because the Gaussian kernel is used and because the maximally coherent $\boldsymbol{u}_\iota$ (which has the minimum Euclidean distance to $\boldsymbol{u}_n$) is chosen (see Section 4). The performance would be significantly different between the proposed algorithm for $Q = 1$ and QKLMS if we employ another kernel and/or another criterion of designing $\mathcal{I}_n$. The dictionary sizes for all the coherence-based algorithms are exactly the same, and that for RKHS/Platt is also nearly the same.

6. CONCLUSION

This paper presented a natural extension of the normalized kernel least mean squares algorithm presented in [7, Chapter 2]. The proposed HYPASS algorithm selectively updates a few coefficients at each iteration by projecting the current filter onto the zero instantaneous-error hyperplane along the selected affine subspace. Coherence is exploited for the coefficients-to-be-updated selection as well as for the dictionary construction. The proposed algorithm enjoys low computational complexity. Numerical examples indicated high potential of the proposed algorithm. The proposed algorithm serves as a nice framework encompassing the Dodd's algorithm with a wider step-size-range and the normalized version of the QKLMS algorithm. It will also serve as a basis in devising further advanced algorithms based on, for instance, data reusing.

APPENDIX: PROOFS OF (8) AND (19)

Lemma 1. *Let M be a subspace of a real Hilbert space $(\mathcal{X}, \langle\cdot,\cdot\rangle_{\mathcal{X}})$, and $\Pi := \{x \in \mathcal{X} : \langle a, x\rangle_{\mathcal{X}} = 0\}$, $(0 \neq) a \in \mathcal{X}$, be a hyperplane. Then, there exists $\beta \in \mathbb{R}$ such that*

$$P_{M\cap\Pi}(x) = x + \beta P_M(a), \quad \forall x \in M. \tag{27}$$

Proof: Suppose that $a \in M^\perp$. In this case, we can show that (27) holds for any $\beta \in \mathbb{R}$ because $M \cap \Pi = M$ and $P_M(a) = 0$. Suppose now that $a \notin M^\perp$. In this case, the existence of β satisfying $x + \beta P_M(a) \in M \cap \Pi$ is ensured by $\langle a, P_M(a)\rangle_{\mathcal{X}} = \|P_M(a)\|_{\mathcal{X}}^2 \neq 0$. By the orthogonal projection theorem [10], it is therefore sufficient to show that $P_M(a) = a - P_{M^\perp}(a) \perp M \cap \Pi$, which is verified by $a \perp \Pi$ and $P_{M^\perp}(a) \perp M$. □

Lemma 1 verifies the equations (8) and (19) by translation.

7. REFERENCES

[1] T. J. Dodd, V. Kadirkamanathan, and R. F. Harrison, "Function estimation in Hilbert space using sequential projections," in *IFAC Conf. Intell. Control Syst. Signal Process.*, 2003, pp. 113–118.

[2] 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.

[3] 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.

[4] 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.

[5] 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, Dec. 2009.

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

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

[9] M. Yukawa, "Multi-kernel adaptive filtering," *IEEE Trans. Signal Processing*, to appear.

[10] D. G. Luenberger, *Optimization by Vector Space Methods*, New York: Wiley, 1969.