DICTIONARY-BASED ONLINE KERNEL PRINCIPAL SUBSPACE ANALYSIS WITH DOUBLE ORTHOGONALITY PRESERVATION

Toshihisa Tanaka

Department of Electrical and Electronic Engineering, Tokyo University of Agriculture and Technology, Koganei-shi, Tokyo, 184–8588, Japan
Email: tanakat@cc.tuat.ac.jp

ABSTRACT

An adaptive online algorithm with a dictionary of observed signals for kernel principal subspace analysis is presented. A coefficient matrix for eigenfunctions is updated by a recursive least squares (RLS)-type algorithm and entries in the dictionary are adaptively added / removed preserving orthogonality of the eigenfunctions. It is shown that the orthogonalization can be implemented by analytically solvable (generalized) eigenvalues of 2×2 matrices, instead of the computation of the inverse squared root of matrix having the size of the dictionary. Numerical example is then illustrated to support the analysis.

Index Terms— Recursive least squares, kernel principal component analysis, subspace tracking

1. INTRODUCTION

Principal component analysis (PCA) is a powerful statistical tool in areas of signal processing, machine learning, communications, and biomedical engineering. Principal component (PC) is the one that maximizes its variance over a set of multivariate signals, and the problem to find the PC is reduced to the eigendecomposition of the correlation matrix of signals. The PCA enables us to represent the signals in a subspace of dimension much lower than the number of variables.

The PC can be regarded as the output of a linear system, where the system parameter is given as the eigenvector. In other words, the traditional PCA assumes that an observed signal is a linearly generated stochastic process. However, real-world signals and data are inherently nonlinear and linear PCA sometimes cannot capture efficient features of the data.

To deal with nonlinear multivariate signals, an efficient and successful approach is to use the kernel PCA (KPCA) [1], which is the PCA constructed in a reproducing kernel Hilbert space (RKHS). In the standard KPCA, all the observed sample signals are mapped to the RKHS induced by a reproducing kernel, $\kappa(\cdot,\cdot)$, which is a symmetric positive definite map, $\boldsymbol{R}^d \times \boldsymbol{R}^d \to \boldsymbol{R}$, called Mercer kernel. In such a setting, the inner produce of two elements in the RKHS is given as a value of the kernel function. It has been shown that the KPCA is given as the eigendecomposition of signal of the Gram matrix, $\boldsymbol{K}_{ij} = \kappa(\boldsymbol{u}_i, \boldsymbol{u}_j)$. This readily implies that the more we observe signals, the larger the size of the Gram matrix is. This means that KPCA may require very high computational load when we have a large number of samples. To avoid the large scale of the batch processing, several solutions have been proposed. Incremental KPCA [2, 3] is an extension of the incremental PCA. This needs an eigenvalue decomposition at every iteration. The kernel Hebbian algorithm (KHA) [4] is an online PCA that is an extension of the generalized Hebbian algorithm (GHA) [5]. Also the learning rate of KHA has been studied [6], where an annealing-type learning rate is introduced to accelerate the update speed. Ding et al. addressed this problem and developed an adaptive KPCA algorithm [7]. However, this algorithm still needs to solve an eigenvalue problem at each update, which demands high computational load. Washizawa [8] addressed the problem to extend KHA to an adaptive algorithm, which can be classified into the steepest descent type or LMS-type. However, the algorithm is derived in Hilbert space, and the projection onto a subspace is necessary to stabilize the algorithm. A recursive least-squares (RLS) type algorithm has also been developed by Tanaka and coworkers [9].

Another aspect of online kernel methods is the increase of dimensions at every iteration. An eigenfunction is a superposition of kernel functions corresponding to observed sample signals. Thus, to reduce the computational complexity and the memory size, constructing a dictionary of sample signals is necessary [2,8]. However, updating the dictionary can violate the orthogonality of eigenfunctions. This paper develops an efficient dictionary update algorithm with preserving the orthogonality. The proposed orthogonalization is applied to a RLS-type algorithm [9] for principal subspace tracking.

This work was supported in part by KAKENHI.

2. KERNEL PRINCIPAL COMPONENT/SUBSPACE ANALYSIS

2.1. Kernel PCA and Its Batch Algorithm

Let $\mathscr{H}$ be a RKHS of functions on $\boldsymbol{C}^d$, and denote the inner product by $\langle\cdot,\cdot\rangle$ and the reproducing kernel by $\kappa(\cdot,\cdot)$. The principal component analysis in $\mathscr{H}$ is to find the r functions in $\mathscr{H}$ that 'compress' observed data as much as possible. Let us denote the set of these r functions by $\mathscr{S} = \{\varphi_i \in \mathscr{H}\}_{i=1}^r$. These functions can be obtained by solving an eigenvalue problem given as

$$R\varphi = \lambda\varphi, \tag{1}$$

where R is called a correlation operator.

In a RKHS, the kernel principal component analysis (KPCA) [1] may be formulated in the following way. Suppose that we have a set of N observed sample signals (vectors) denoted by $\{\boldsymbol{u}_i \in \mathbb{C}^d\}_{i=1}^N$. Define a function in $\mathscr{H}$ associated with the ith observed signal as

$$\phi_i = \phi(\boldsymbol{u}_i) = \kappa(\cdot, \boldsymbol{u}_i) \in \mathscr{H}. \tag{2}$$

The empirical correlation operator is given as $R = \frac{1}{N}\sum_{i=1}^N \phi_i \phi_i^*$, where $\cdot^*$ denotes the operation defined such that for arbitrary $f, g \in \mathscr{H}$,

$g^* f = \langle f, g \rangle$. The KPCA is indeed the PCA for the set of functions ϕ_i defined in (2). If all the functions in $\{\phi_i\}_{i=1}^N$ are linearly independent, the empirical correlation operator has N eigenvalues: $\lambda_1 \geq \cdots \geq \lambda_N \neq 0$, and the corresponding eigenfunctions denoted by φ_j. The range of the operator defined as

$$W = [\varphi_1, \ldots, \varphi_r] \tag{3}$$

is called the principal subspace of rank r. Inner product $\langle \varphi_j, \phi_i \rangle$ is called the jth kernel principal component (KPC) of observed data $\boldsymbol{u}_i$,

From the representer theorem [10], φ in (1) is given as a superposition of the functions associated with the observed signals:

$$\varphi = \sum_{i=1}^{N} a_i \phi_i = \Phi \boldsymbol{a}, \tag{4}$$

where $\boldsymbol{a} = [a_1, \ldots, a_N]^T$ and $\Phi = [\phi_1, \ldots, \phi_N]$. Define the adjoint as $\Phi^* = [\phi_1^*, \ldots, \phi_N^*]^T$. By left-multiplying Φ^* on both sides, (1) with (4) becomes $\boldsymbol{K}^2 \boldsymbol{a} = \lambda \boldsymbol{K} \boldsymbol{a}$, where $(\boldsymbol{K})_{ij} = \langle \phi_i, \phi_j \rangle = \kappa(\boldsymbol{u}_i, \boldsymbol{u}_j)$ is called the Gram matrix. Thus, the KPCA is inherently equivalent to the eigendecomposition of a matrix of size $N \times N$ [1].

2.2. KPCA With Sample Dictionary

The size of $\boldsymbol{K}$ is determined by the number of observed signals. Moreover, $\boldsymbol{K}$ is basically dense. This implies that for a large number of samples, the computational complexity for solving the eigenvalue problem is high. To scale down the problem, we can consider the PCA in the subspace spanned by a subset of samples, $\{\phi_{\pi(i)}\}_{i=1}^M$ ($M \ll N$), where $\pi(i) \in \{1, \ldots, N\}$ is a permutation [3, 11]. An eigenfunction is then given as $\varphi = \sum_{i=1}^M a_i \phi_{\pi(i)} = \Psi \boldsymbol{a}$, where $\Psi = [\phi_{\pi(1)}, \ldots, \phi_{\pi(M)}]$. Indeed, $\boldsymbol{a}$ is the eigenvector with respect to φ. By left-multiplying Ψ^* on both sides of (1), we obtain

$$\boldsymbol{H}\boldsymbol{H}^H \boldsymbol{a} = \lambda \boldsymbol{M} \boldsymbol{a}, \tag{5}$$

where $\boldsymbol{M} = \Psi^* \Psi$ and $\boldsymbol{H} = \Psi^* \Phi$. It should be noted that solving the above equation amounts to the generalized eigenvalue problem of matrix pair $(\boldsymbol{H}\boldsymbol{H}^H, \boldsymbol{M})$.

The jth eigenvector is described with the coefficient vector denoted by $\boldsymbol{a}_j = [a_{1j}, \ldots, a_{Mj}]^T$ as

$$\varphi_j = \sum_{i=1}^{M} a_{ij} \psi_i = \Psi \boldsymbol{a}_j. \tag{6}$$

Therefore, the eigenspace spanned by the first r eigenfunctions is represented with $\boldsymbol{A} = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_r] \in \mathbb{C}^{M \times r}$ as $W = \Psi \boldsymbol{A}$. We call $\boldsymbol{A}$ the *coefficient matrix* for W.

2.3. Online KPCA Algorithms

The KPCA needs a set of observed sample signals for the eigendecomposition of the Gram matrix. A possible extension of this batch KPCA is to develop incremental or online algorithms for the KPCA, where the kernel eigensubspace is updated every time when a new sample is observed. This type of algorithms can be classified into three categories: incremental KPCA [2, 3], online Hebbian KPCA [4], and adaptive KPCA [9, 12]. The underlying idea behind incremental KPCA algorithms is to apply an eigenvalue decomposition of augmented operator $[W, \phi_{N+1}]$. On the other hand, Hebbian and adaptive algorithms need no eigenvalue decomposition, which lead to heavy load of computation. However, the adaptive update algorithms do not consider the preservation of orthogonality of kernel eigenvectors, say, $W^*W = I$. In particular, as we will see, the update of dictionary can violate the orthogonality. The straightforward orthogonalization is to use $W(W^*W)^{-1/2}$ instead of W. The present paper shows that this operation can be accomplished without the computation of the inverse of matrix square root.

3. ADAPTIVE KPA WITH DOUBLE ORTHOGONALIZATION

Again, the operator that gives principal subspace is given as

$$W = \Psi \boldsymbol{A}. \tag{7}$$

Our proposed method is doubly adaptive to the observed samples, as summarized in the following:

1. Update the dictionary, Ψ, and augment/shrink $\boldsymbol{A}$, accordingly;
2. Update the coefficient matrix, $\boldsymbol{A}$;

while the orthogonality, $W^*W = I$ is preserved at each step.

3.1. Updating the Dictionary

To determine a subspace that represents a signal, basis functions, Ψ_i, are needed. We will call a set of the basis functions a *dictionary* [13], and the entries of the dictionary are chosen from observed signals. If statistics of the signal source is time-variant, the past signal that has been a member of the dictionary will reduce its significance of representing samples as the time index increases. Let $\boldsymbol{u}[k]$ be a input signal at time instance k, and define $\phi[k] = \phi(\boldsymbol{u}[k])$. In the following, when the dictionary is updated by adding or eliminating an entry, $\phi[k] = \phi(\boldsymbol{u}[k])$, we address how efficiently the Gram matrix and its inverse are calculated preserving the orthogonality. Consider the case where the number of functions in the dictionary does not exceed L even though a new basis function is added. For simplicity, the norm of the kernel in a RKHS is assumed to be unity, that is, we assume that $\kappa(\boldsymbol{u}, \boldsymbol{u}) = 1$, $\boldsymbol{u} \in \mathbb{R}^d$.

3.1.1. Case of Adding an Entry

We consider the following augmented dictionary operator:

$$\Psi[k] = [\Psi[k-1], \phi[k]]. \tag{8}$$

In this case, $W[k-1] = \Psi[k-1]\boldsymbol{A}[k-1]$ can be also described as $W[k-1] = \Psi[k]\boldsymbol{B}[k]$, where

$$\boldsymbol{B}[k] = \begin{bmatrix} \boldsymbol{A}[k-1] \\ \boldsymbol{0}_{1\times r} \end{bmatrix}. \tag{9}$$

Next we discuss the update for $\boldsymbol{M}[k-1]$ and $\boldsymbol{P}[k-1] = \boldsymbol{M}^{-1}[k-1]$. The update rules will be used for the update of $\boldsymbol{A}[k-1]$, as will be mentioned later.

First of all, we define

$$\boldsymbol{h}[k] = \Psi^*[k]\phi[k], \tag{10}$$

which gives the representation in the Euclidean space for $\phi[k]$ restricted to the subspace determined by $\Psi[k]$. Note that by definition of the kernel, the last element of $\boldsymbol{h}[k]$ is always unity. Moreover, $\boldsymbol{h}[k]$ can be represented with subvector $\hat{\boldsymbol{h}}[k]$ by $\boldsymbol{h}[k] = [\hat{\boldsymbol{h}}^T[k], 1]^T$. Together with the fact that $\boldsymbol{M}[k] = \Psi^*[k]\Psi[k]$, and the block matrix inversion formula to reduce the computational complexity, the following relations are immediately derived:

Gram matrix

$$\boldsymbol{M}[k] = \begin{bmatrix} \boldsymbol{M}[k-1] & \hat{\boldsymbol{h}}[k] \\ \hat{\boldsymbol{h}}^H[k] & 1 \end{bmatrix}. \quad (11)$$

The inverse

$$\boldsymbol{P}[k] = \begin{bmatrix} \boldsymbol{P}[k-1] & \boldsymbol{0} \\ \boldsymbol{0} & 0 \end{bmatrix} + \frac{1}{1-\hat{\boldsymbol{h}}^H[k]\bar{\boldsymbol{h}}[k]} \begin{bmatrix} \bar{\boldsymbol{h}}[k]\bar{\boldsymbol{h}}^H[k] & -\bar{\boldsymbol{h}}[k] \\ -\bar{\boldsymbol{h}}^H[k] & 1 \end{bmatrix}, \quad (12)$$

where $\bar{\boldsymbol{h}}[k] = \boldsymbol{P}[k-1]\hat{\boldsymbol{h}}[k]$.

Moreover, the computation of $\boldsymbol{P}[k]\boldsymbol{h}[k]$ is greatly reduced to $\boldsymbol{P}[k]\boldsymbol{h}[k] = [\boldsymbol{0}_{L[k-1]}, 1]^T$. It should be emphasized that the orthogonality of $\boldsymbol{B}[k]$ is maintained.

Proposition 1 (Orthogonality) *If $W^*[k-1]W[k-1] = I$, then*

$$\boldsymbol{A}^H[k-1]\boldsymbol{M}[k-1]\boldsymbol{A}[k-1] = \boldsymbol{I}_r \quad (13)$$

and $\boldsymbol{B}^H[k]\boldsymbol{M}[k]\boldsymbol{B}[k] = \boldsymbol{I}_r$.

Proof is omitted because it is straightforward.

3.1.2. Case of Eliminating an Entry

Define $\psi_i[k]$ as the ith entry of $\Psi[k]$. Without loss of generality, assume that the first entry, $\psi_1[k]$, is eliminated from the dictionary. Let l be the number of basis functions in the dictionary before the elimination, and then the update formula is $\Psi[k] = [\psi_2[k-1], \ldots, \psi_l[k-1]]$:

Gram matrix

$$\boldsymbol{M}[k] = \bar{\boldsymbol{M}}, \quad (14)$$

which is the submatrix of size $(l-1)\times(l-1)$ defined in

$$\boldsymbol{M}[k-1] = \begin{bmatrix} 1 & \boldsymbol{m}^H \\ \boldsymbol{m} & \bar{\boldsymbol{M}} \end{bmatrix}. \quad (15)$$

Its inverse

$$\boldsymbol{P}[k] = \boldsymbol{P}_{2:l,2:l}[k-1] - \frac{1}{p}\boldsymbol{p}\boldsymbol{p}^H,$$

where $\boldsymbol{p}$ is the vector defined in

$$\boldsymbol{P}[k-1] = \begin{bmatrix} p & \boldsymbol{p}^H \\ \boldsymbol{p} & \boldsymbol{P}_{2:l,2:l}[k-1] \end{bmatrix}, \quad (16)$$

Eliminating an entry of the dictionary causes the violation of orthogonality. Note that $W'[k-1] = \Psi[k]\bar{\boldsymbol{A}}[k-1]$ where $\bar{\boldsymbol{A}}$ is the submatrix defined as

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{a}^1 \\ \bar{\boldsymbol{A}} \end{bmatrix}, \quad (17)$$

where $\boldsymbol{a}^i$ is the ith row vector of $\boldsymbol{A}$. $W'[k-1]$ is generally non-orthogonal, $(W'[k-1])^*W'[k-1] \neq I$, even though $W[k-1]$ is orthogonal.

Due to lack of space, we provide only the main result in the following. We start with defining vectors as $\boldsymbol{a} = (\boldsymbol{a}^1)^H$, and $\boldsymbol{b} = \frac{1}{2}\boldsymbol{a} + \bar{\boldsymbol{A}}^H\boldsymbol{m}$, and matrices $\boldsymbol{C} = [\boldsymbol{a}\ \boldsymbol{b}]$ and $\boldsymbol{E} = \boldsymbol{a}\boldsymbol{b}^H + \boldsymbol{b}\boldsymbol{a}^H$. Let $\boldsymbol{t}_1, \boldsymbol{t}_2 \in \mathbb{C}^2$ be the generalized eigenvectors of matrix pencil $(\boldsymbol{C}^H\boldsymbol{E}\boldsymbol{C}, \boldsymbol{C}^H\boldsymbol{C})$.

Proposition 2 (Orthogonality) *If $W^*[k-1]W[k-1] = I$, then $\Psi[k]\boldsymbol{B}[k]$ is orthogonal, where*

$$\boldsymbol{B}[k] = \bar{\boldsymbol{A}}[k-1] + \bar{\boldsymbol{A}}[k-1]\boldsymbol{C}\left\{\sum_{i=1}^{2}\left(\frac{1}{\sqrt{1-\lambda_i}} - 1\right)\boldsymbol{t}_i\boldsymbol{t}_i^H\right\}\boldsymbol{C}^H. \quad (18)$$

It should be emphasized that the original problem of orthogonalization is reduced to a generalized eigenvalue problem for matrices of size 2×2.

3.1.3. Dictionary Updating Criterion

When we observe signal $\boldsymbol{u}[k]$, we determine whether the signal is added to the dictionary or not, based on the criterion that is called the coherence-based sparsification rule [13]. In the coherence criteria, if the following condition

$$\max_{1\le j\le L[k-1]} (\Psi^*[k-1]\phi[k])_j \le \delta, \quad (19)$$

where $0 \le \delta < 1$ is a threshold that determines the level of sparcity and the coherence of the dictionary, and $(\cdot)_j$ is the jth entry of the vector. This criterion does not consider eliminating a basis function from the dictionary. To save the memory, we proposed to fix the size of dictionary, say, the number of entries of the dictionary, denoted by L. If the dictionary already has L entries, we remove the 'oldest' entry from the dictionary when a new sample is added based on the coherence criteria.

3.2. Updating the Coefficients

3.2.1. Square Error Formulation

Principal component analysis in $\mathscr{H}$ can be also formulated as the problem to find the subspace (or basis functions) that give the best approximation a set of observed samples. Specifically, the solution of the eigenvalue problem given as in (1) minimizes the approximation cost:

$$J_0[W] = \sum_{i=1}^{N} \|\phi_i - WW^*\phi_i\|^2,$$

with $W = \Psi\boldsymbol{A}$ as defined in (7).

Therefore, the optimization problem to find $\mathscr{S}$ is reduced to the one that finds matrix $\boldsymbol{A}$. Define a 'correlation matrix' in the Euclidean space as $\boldsymbol{R} = \sum_{i=1}^{N}\boldsymbol{h}_i\boldsymbol{h}_i^H = \boldsymbol{H}\boldsymbol{H}^H$. The minimization of J_0 is equivalent to the minimization of $J_1[\boldsymbol{A}]$ given [9] as

$$J_1[\boldsymbol{A}] = \mathrm{tr}[\boldsymbol{M}^{-1}\boldsymbol{R}] - 2\mathrm{tr}[\boldsymbol{A}^H\boldsymbol{R}\boldsymbol{A}] + \mathrm{tr}[\boldsymbol{A}^H\boldsymbol{R}\boldsymbol{A}\boldsymbol{A}^H\boldsymbol{M}\boldsymbol{A}],$$

where $\boldsymbol{H} = \Psi^*\Phi$ was defined in (5). It has been claimed in [14] that the optimization of this functional amounts to solving the generalized eigenvalue problem of matrix pencil $(\boldsymbol{R}, \boldsymbol{M})$.

3.2.2. Adaptive Tracking Algorithm

To derive a updating rule at time instance k, we make $\boldsymbol{A}$, $\boldsymbol{R}$, and $\boldsymbol{M}$ all time-variant and denote them by $\boldsymbol{A}[k]$, $\boldsymbol{R}[k]$, and $\boldsymbol{M}[k]$, respectively. Next we define $\Phi[k] = [\phi[1], \ldots, \phi[k]] : \mathbb{R}^k \to \mathscr{H}$, and Let $\Psi[k] : \mathbb{R}^{L[k]} \to \mathscr{H}$ be an operator consisting of $L[k]$ functions from $\Phi[k]$. Note that the time-varying $\boldsymbol{R}[k]$ is the 'correlation matrix' in the subspace represented by $\Psi[k]$.

With the defined symbols, the update for $\boldsymbol{A}[k]$ at time instance k can be derived by minimizing the following time-variant cost function:

$$\begin{aligned} J_k[\boldsymbol{A}[k]] =& \mathrm{tr}[\boldsymbol{M}^{-1}[k]\boldsymbol{R}[k]] - 2\mathrm{tr}[\boldsymbol{A}^H[k]\boldsymbol{R}[k]\boldsymbol{A}[k]] \\ &+ \mathrm{tr}[\boldsymbol{A}^H[k]\boldsymbol{R}[k]\boldsymbol{A}[k]\boldsymbol{A}^H[k]\boldsymbol{M}[k]\boldsymbol{A}[k]], \end{aligned}$$

where $\boldsymbol{M}[k] = \Psi^*[k]\Psi[k]$ and $\boldsymbol{A}[k] \in \mathbb{R}^{L[k]\times r}$. It should be noted that the representation of $\phi[k]$ with respect of $\Psi[k]$ is given by $\boldsymbol{h}[k] = \Psi^*[k]\phi[k]$, and we define $\boldsymbol{c}[k] = \boldsymbol{A}^H[k-1]\boldsymbol{h}[k]$.

By differentiating $J_k[\boldsymbol{A}[k]]$ with respect to $\boldsymbol{A}[k]$, we obtain the gradient, $\partial_{\boldsymbol{A}[k]}J_k = -2(\boldsymbol{R}[k]\boldsymbol{A}[k] - \boldsymbol{M}[k]\boldsymbol{A}[k]\boldsymbol{A}^H[k]\boldsymbol{R}[k]\boldsymbol{A}[k])$. Define

Algorithm 1 RLS-type online KPCA algorithm

1: **if** $\max_{1\le j\le L[k-1]}(\Psi^*[k-1]\phi[k])_j \le \delta$ **then**
2: $\quad\Psi[k] \leftarrow [\Psi[k-1], \phi[k]]$
3: $\quad$Update $\boldsymbol{M}[k]$, $\boldsymbol{P}[k]$, and $\boldsymbol{B}[k-1]$ by (11), (12), and (9).
4: $\quad$**if** # of $\Psi[k]$ more than L **then**
5: $\quad\quad\Psi[k] \leftarrow [\psi_2, \ldots, \psi_L]$.
6: $\quad\quad$Update $\boldsymbol{M}[k]$, $\boldsymbol{P}[k]$, and $\boldsymbol{B}[k-1]$ by (14), (16), and (18).
7: $\quad$**end if**
8: **else**
9: $\quad\boldsymbol{M}[k] = \boldsymbol{M}[k-1]$, $\boldsymbol{P}[k] = \boldsymbol{P}[k-1]$, and $\boldsymbol{B}[k-1] = \boldsymbol{A}[k-1]$.
10: **end if**
11: $\boldsymbol{h}[k] = \Psi^*[k]\phi[k]$
12: $\boldsymbol{c}[k] = \boldsymbol{B}^H[k-1]\boldsymbol{h}[k]$
13: $\boldsymbol{d}[k] = \boldsymbol{Q}[k-1]\boldsymbol{c}[k]$
14: $\boldsymbol{g}[k] = \boldsymbol{d}[k]/(\beta + \boldsymbol{c}^H[k]\boldsymbol{d}[k])$
15: $\boldsymbol{Q}[k] = \beta^{-1}\left(\boldsymbol{Q}[k-1] - \boldsymbol{g}[k]\boldsymbol{c}^H[k]\boldsymbol{Q}[k-1]\right)$
16: $\boldsymbol{e}[k] = \boldsymbol{P}[k]\boldsymbol{h}[k] - \boldsymbol{B}[k-1]\boldsymbol{c}[k]$
17: $\boldsymbol{A}[k] = \boldsymbol{B}[k-1] + \boldsymbol{e}[k]\boldsymbol{g}^H[k]$

$\boldsymbol{R}_{hc}[k] = \boldsymbol{R}[k]\boldsymbol{A}[k]$ and $\boldsymbol{R}_c[k] = \boldsymbol{A}^H[k]\boldsymbol{R}[k]\boldsymbol{A}[k]$. Then the optimizer is given as

$$\boldsymbol{A}[k] = \boldsymbol{M}^{-1}[k]\boldsymbol{R}_{hc}[k]\boldsymbol{R}_c^{-1}[k]. \tag{20}$$

Assuming that the dictionary operator does not change at time k, say $\Psi[k] = \Psi[k-1]$, we update $\boldsymbol{R}[k]$ by

$$\boldsymbol{R}[k] = \beta\boldsymbol{R}[k-1] + \boldsymbol{h}[k]\boldsymbol{h}^H[k]. \tag{21}$$

Applying the projection approximation [15] given as $\boldsymbol{R}[k]\boldsymbol{A}[k] \approx \boldsymbol{R}[k]\boldsymbol{A}[k-1]$, we obtain

$$\boldsymbol{R}_{hc}[k] \approx \beta\boldsymbol{R}_{hc}[k-1] + \boldsymbol{h}[k]\boldsymbol{c}^H[k], \boldsymbol{R}_c[k] \approx \beta\boldsymbol{R}_c[k-1] + \boldsymbol{c}[k]\boldsymbol{c}^H[k].$$

Moreover, defining $\boldsymbol{P}[k] = \boldsymbol{M}^{-1}[k]$ and $\boldsymbol{Q}[k] = \boldsymbol{R}_c^{-1}[k]$, we can derive a RLS-type algorithm given in Algorithm 1.

3.2.3. *Update Preserving Orthogonality*

In Algorithm 1, $\boldsymbol{A}[k] = \boldsymbol{B}[k-1] + \boldsymbol{e}[k]\boldsymbol{g}^H[k]$ obtained in line 17 is not always $\boldsymbol{M}[k]$-orthogonal even though $\boldsymbol{B}[k-1]$ is orthogonal. We may replace $\boldsymbol{A}[k]$ by $\boldsymbol{A}[k](\boldsymbol{A}^H[k]\boldsymbol{M}[k]\boldsymbol{A}[k])^{-1/2}$ for orthogonalization. The basic idea behind this is an extension of the orthonormal PAST (OPAST) algorithm [16].

Proposition 3 *It holds that* $(\boldsymbol{A}^H[k]\boldsymbol{M}[k]\boldsymbol{A}[k])^{-1/2} = \boldsymbol{I} + \gamma[k]\boldsymbol{g}[k]\boldsymbol{g}^H[k]$, *where*

$$\gamma[k] = \frac{1}{\|\boldsymbol{g}[k]\|^2}\left(\frac{1}{\sqrt{1+\alpha[k]\|\boldsymbol{g}[k]\|^2}} - 1\right)\boldsymbol{g}[k]\boldsymbol{g}^H[k]$$

By using this lemma and defining,

$$\boldsymbol{f}[k] = (\gamma[k]\|\boldsymbol{g}[k]\|^2 + 1)\boldsymbol{e}[k]\gamma[k]\boldsymbol{B}[k]\boldsymbol{g}[k]$$

we obtain a simple recursive formula given as

$$\boldsymbol{A}[k] = \boldsymbol{B}[k-1] + \boldsymbol{f}[k]\boldsymbol{g}^H[k]. \tag{22}$$

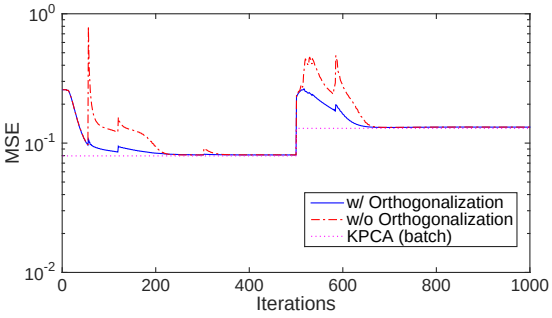

Fig. 1. Mean squares error

4. NUMERICAL EXAMPLES

For evaluation, we use benchmark signals that are described by nonlinear difference equations. The signal model is changed during the iteration to observe the cabability of tracking.

- For $k = 2, \ldots, 500$ [13]:
  $s_k = (0.8 - 0.5\exp(-s_{k-1}^2))s_{k-1} - (0.3 + 0.9\exp(-s_{k-1}^2))s_{k-2} + 0.1\sin(s_{k-1}\pi)$
- For $k = 501, \ldots, 1000$:
  $s_k = (0.7 - 0.5\exp(-s_{k-1}^2))s_{k-1} - (0.4 + 0.9\exp(-s_{k-1}^2))s_{k-2} + 0.1\sin(s_{k-1}\pi)$

Let $s_0 = 0.1$ and $s_1 = 0.1$. We assume that the zero-men Gaussian noise with the variance of 0.01 is added to output s_k. By using such a generated nonlinear signal, we define the input vector as $\boldsymbol{u}[k] = [s_{k-d+1}, \ldots, s_k]^T \in \mathbb{R}^d$.

We compare the following algorithms: 1) Proposed RLS-type algorithm with (22); 2) Case where the orthogonalization is not applied when updating the basis and $\boldsymbol{A}[k]$. In the numerical test, the size of vectors is set to 10 ($d = 10$), and we tracked the principal kernel subspace of rank 2 ($r = 2$). We use the Gaussian kernel given by $\kappa(\boldsymbol{u}_i, \boldsymbol{u}_j) = \exp(-0.1\|\boldsymbol{u}_i - \boldsymbol{u}_j\|^2)$. The coherence threshold is set to 0.96 ($\delta = 0.96$), and the volume of dictionary is set to 25 ($L = 25$). Moreover, for all the algorithms, we use the following initial parameters: $\Psi[0] = [\phi[0], \ldots, \phi[r-1]]$, $\boldsymbol{P}[0] = \boldsymbol{M}^{-1}[0]$, and $\boldsymbol{A}[0] = (\boldsymbol{P}^{1/2}[0])_{1:r}$. We use the forgetting factor $\beta = 0.98$. We eveluate the approximation performance of the proposed adaptive KPCA by the mean squared error (MSE) at every time instance: $\mathrm{MSE}[k] = N^{-1}\sum_{i=1}^N \|\phi_i - W[k]W^*[k]\phi_i\|^2$ where $\boldsymbol{H}[k] = \Psi^*[k]\Phi$.

Figure 1 illustrates the evolution of MSE. We can confirm that the orthogonalization works very efficiently in stabilizing the tracking algorithm. The MSE after convergence is very close to the MSE with KPCA. Moreover, we can observe that before the 100th iteration, the algorithm without orthgonalization gets nearly diverged. On the other hand, the proposed orthogonalization can contribute to stabilization of adaptive kernel principal subspace tracking.

5. CONCLUSION

It has been shown that the proposed adaptive online method with orthogonalization efficiently works in terms of MSE. Note that the result obtained in Proposition 2 is applicable to any other online or incremental kernel principal component algorithms. Numerical exmples with practical applications will be reported in future.

6. REFERENCES

[1] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, vol. 10, no. 5, pp. 1299–1319, 1998.

[2] T.-J. Chin and D. Suter, "Incremental kernel principal component analysis," *Image Processing, IEEE Transactions on*, vol. 16, no. 6, pp. 1662–1674, 2007.

[3] S. Kimura, S. Ozawa, and S. Abe, "Incremental kernel PCA for online learning of feature space," in *Computational Intelligence for Modelling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, International Conference on*, vol. 1, pp. 595–600, Nov 2005.

[4] K. I. Kim, M. O. Franz, and B. Schölkopf, "Iterative kernel principal component analysis for image modeling," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 27, pp. 1351–1366, Sept. 2005.

[5] E. Oja, "Neural networks, principal components, and subspaces," *Int. J. Neural Systems*, vol. 1, pp. 61–68, Apr. 1989.

[6] S. Günter, N. N. Schraudolph, and S. Vishwanathan, "Fast iterative kernel principal component analysis," *Journal of Machine Learning Research*, vol. 8, pp. 1893–1918, Aug. 2007.

[7] M. Ding, Z. Tian, and H. Xu, "Adaptive kernel principal component analysis," *Signal Processing*, vol. 90, pp. 1542–1553, 2010.

[8] Y. Washizawa, "Adaptive subset kernel principal component analysis for time-varying patterns," *Submitted*, 2011.

[9] T. Tanaka, Y. Washizawa, and A. Kuh, "Adaptive kernel principal components tracking," in *Proc. of 2012 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2012)*, pp. 1905–1908, Mar. 2012.

[10] B. Schölkopf, R. Herbrich, and A. J. Smola, "A generalized representer theorem," in *Computational learning theory*, pp. 416–426, Springer, 2001.

[11] Y. Washizawa, "Subset kernel principal component analysis," in *Machine Learning for Signal Processing, 2009. MLSP 2009. IEEE International Workshop on*, pp. 1–6, Sept 2009.

[12] Y. Washizawa, "Adaptive subset kernel principal component analysis for time-varying patterns," *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 23, pp. 1961–1973, Dec 2012.

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

[14] J. Yang, H. Xi, F. Yang, and Y. Zhao, "RLS-based adaptive algorithms for generalized eigen-decomposition," *IEEE Trans. Signal Processing*, vol. 54, pp. 1177–1188, Apr. 2006.

[15] B. Yang, "Projection approximation subspace tracking," *IEEE Trans. Signal Processing*, vol. 43, pp. 95–107, Jan. 1995.

[16] K. Abed-Meraim, S. Attallah, A. Chkeif, and Y. Hua, "Orthogonal Oja algorithm," *IEEE Signal Processing Letters*, vol. 7, pp. 116–119, May 2000.