ADAPTIVE KERNEL PRINCIPAL COMPONENTS TRACKING

Toshihisa Tanaka[1]*, Yoshikazu Washizawa*[2]*, and Anthony Kuh*[3]

[1]Tokyo University of Agriculture and Technology, Koganei-shi, Tokyo
[2]University of Electro-Communications, Chofu-shi, Tokyo
[3]University of Hawai'i at Manoa, Honolulu, HI
Emails: tanakat@cc.tuat.ac.jp, washizawa@uec.ac.jp, kuh@hawaii.edu

ABSTRACT

Adaptive online algorithms for simultaneously extracting nonlinear eigenvectors of kernel principal component analysis (KPCA) are developed. KPCA needs all the observed samples to represent basis functions, and the same scale of eigenvalue problem as the number of samples should be solved. This paper reformulates KPCA and deduces an expression in the Euclidean space, where an algorithm for tracking generalized eigenvectors is applicable. The developed algorithm here is least mean squares (LMS)-type and recursive least squares (RLS)-type. 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 crucial technology in areas of statistical signal processing, such as machine learning, communications, and image processing. 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 one to find the first eigenvector of the correlation matrix of signals. Even though the signal is observed in a high dimensional space, that is, the signal vector consists of a large number of elements, PCA enables us to represent the signals in a much lower dimensional subspace. This leads to efficient data compression and feature extraction.

Nevertheless, many signals and data are inherently nonlinear in nature and linear methods such as PCA do not do a good job in capturing features of the data. The traditional PCA only fits a set of signals that are linearly generated, since PCA, by definition, assumes that an observed signal is a linearly generated stochastic process. To deal with nonlinear multivariate signals, an efficient approach is to make use of the Mercer kernels, which behaves as the inner product in the space of higher dimensional functions onto which the observed signals are mapped. More specifically, let $\{\boldsymbol{u}_i \in \mathbb{R}^d\}_{i=1}^N$ be a set of multivariate signals, and $\phi(\cdot)$ be a nonlinear map from $\mathbb{R}^d$ to the reproducing kernel Hilbert space (RKHS) denoted by $\mathcal{H}$. The kernel approach defines the inner product in $\mathcal{H}$ as $\langle \phi(\boldsymbol{u}_i), \phi(\boldsymbol{u}_j)\rangle = \kappa(\boldsymbol{u}_i, \boldsymbol{u}_j)$, where $\kappa(\cdot,\cdot)$ is a symmetric positive definite map, $\boldsymbol{R}^d \times \boldsymbol{R}^d \to \boldsymbol{R}$, called Mercer kernel.

PCA constructed in this kernel-induced space is called kernel PCA (KPCA), which was first formulated by Schölkopf et al. [1]. The problem of KPCA is to find eigenvectors of Gram matrix, which is given as $\boldsymbol{K}_{ij} = \kappa(\boldsymbol{u}_i, \boldsymbol{u}_j)$. 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. A solution to this problem is to apply the kernel Hebbian algorithm (KHA) [2], where an online PCA called the generalized Hebbian algorithm (GHA) [3] is extended. Also the learning rate of KHA has been studied [4], where an annealing-type learning rate is introduced to accelerate the update speed.

Even though these iterative algorithms are shown to be computationally effective, they are not fully "online" or "adaptive" in the context of adaptive signal processing. Ding et al. addressed this problem and developed an adaptive KPCA algorithm [5]. However, this algorithm needs to solve an eigenvalue problem at each update, which demands high computational load and brings about the over-adaptation problem (see papers on eigenvector tracking [6, 7], for example). A recent study by 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.

This paper develops a novel adaptive KPCA algorithm that can track eigenvectors in nonlinear space with kernels. The main feature is that the established algorithm is structures similar to LMS and RLS, which are standard in adaptive signal processing. This means that the algorithms do not need eigenvalue decomposition. Moreover, we show the reformulation of kernel PCA problem in the standard Euclidean space. This means that it is not necessary any longer to consider higher dimensional Hilbert space. This makes the formation simpler and clearer.

2. KERNEL PCA AND ITS EUCLIDEAN REPRESENTATION

Let $\mathcal{H}$ be a RKHS. Principal component analysis in $\mathcal{H}$ is to find a set of r functions in $\mathcal{H}$ that can "compress" observed data. We denote this set by $\mathcal{S} = \{\varphi_i \in \mathcal{H}\}_{i=1}^r$, which minimizes the approximation cost:

$$J_0[\{\varphi_i\}_{i=1}^r] = \sum_{i=1}^N \left\| \phi_i - \sum_{j=1}^r \langle \varphi_j, \phi_i \rangle \varphi_j \right\|^2,$$

where $\phi_i \in \mathcal{H}$ is a function associated with the ith observed signal, that is, it is defined in RKHS as $\phi_i = \phi(\boldsymbol{u}_i) = \kappa(\cdot, \boldsymbol{u}_i)$. Inner product $\langle \varphi_j, \phi_i \rangle$ is called the jth principal component (PC) of observed data, and φ_j is called the jth kernel principal eigenvector.

Each element in $\mathcal{S}$ can be represented by the superposition of linearly independent functions in $\mathcal{H}$. Therefore, the problem is reduced to finding an operator with lower rank that approximates elements in

This work was supported in part by KAKENHI, Grant-in Aid for Scientific Research (B), 23300069.

the space in the sense of mean squared error:

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

where $\cdot^*$ is its adjoint,

$$W = [\varphi_1, \ldots, \varphi_r] = \Psi \boldsymbol{A}, \tag{1}$$

$\boldsymbol{A} \in \mathbb{R}^{M\times r}$, and $\Psi = [\psi_1, \ldots, \psi_M]$, $\psi_i \in \mathcal{H}$, $i = 1, \ldots, M$. Ψ determines the subspace where the approximation is conducted. It should be noted that if Ψ is assumed fixed, optimizing W is equivalent to optimizing $\boldsymbol{A}$. We also define $\Phi = [\phi_1, \ldots, \phi_N]$, which is a collection of functions associated with observed samples. Let $R(\cdot)$ be the range of an operator. Traditionally, KPCA gives in the space spanned by $\{\phi_i\}_{i=1}^N$ the r-dimensional subspace that gives approximation, that is, $R(\Psi) = R(\Phi)$, implying that $M = N$.

However, this paper addresses more general case, say, $R(\Psi) \neq R(\Phi)$ and $M \gg N$. Note that $R(\Psi)$ is not necessarily a subspace of $R(\Phi)$, which is the case considered in [9]. This setting is more natural in the context of adaptive signal processing, since in an on-line system, it is not realistic that all the observed samples are buffered. Several kernel adaptive filtering algorithms has been developed with this assumption (see [10], for example).

In the rest of this section, we will confirm that the problem of finding KPCA is reduced to a generalized eigenvalue problem in the Euclidean space. Let $P_{R(\Psi)}$ be the orthogonal projector onto $R(\Psi)$. From the Pythagorean theorem, the original cost function becomes

$$J_0[W] = \sum_{i=1}^{N} \left(\|P_{R(\Psi)}(\phi_i - WW^*\phi_i)\|^2 + \|P_{R(\Psi)^\perp}(\phi_i - WW^*\phi_i)\|^2 \right). \tag{2}$$

Since it is verified (see [11], for example) that $P_{R(\Psi)} = \Psi(\Psi^*\Psi)^{-1}\Psi^*$, $P_{R(\Psi)}\Psi = \Psi$. Then, the first term of (2) reads

$$\begin{aligned} \|P_{R(\Psi)}(\phi_i - WW^*\phi_i)\|^2 &= \|(\Psi^*\Psi)^{-1/2}\Psi^* P_{R(\Psi)}(\phi_i - \Psi \boldsymbol{A}\boldsymbol{A}^T\Psi^*\phi_i)\|^2 \\ &= \|\boldsymbol{M}^{-1/2}(\Psi^*\phi_i - \boldsymbol{M}\boldsymbol{A}\boldsymbol{A}^T\Psi^*\phi_i)\|^2 \\ &= \|\boldsymbol{M}^{-1/2}(\boldsymbol{h}_i - \boldsymbol{M}\boldsymbol{A}\boldsymbol{A}^T\boldsymbol{h}_i)\|^2, \end{aligned}$$

where $\boldsymbol{M} = \Psi^*\Psi$ and $\boldsymbol{h}_i = \Psi^*\phi_i$. Since $P_{R(\Psi)^\perp}\Psi = 0$, together with (1), the second term in the summation of (2) can be written as

$$\|P_{R(\Psi)^\perp}(\phi_i - WW^*\phi_i)\|^2 = \|P_{R(\Psi)^\perp}\phi_i\|^2,$$

which is indeed a constant. Therefore, the original cost function can be rewritten as $J_0[W] = J_1[\boldsymbol{A}] + J_2$, where

$$J_1[\boldsymbol{A}] = \sum_{i=1}^{N} \|\boldsymbol{M}^{-1/2}(\boldsymbol{h}_i - \boldsymbol{M}\boldsymbol{A}\boldsymbol{A}^T\boldsymbol{h}_i)\|^2, \quad J_2 = \sum_{i=1}^{N} \|P_{R(\Psi)^\perp}\phi_i\|^2.$$

Again, J_2 is a constant, which is ignored.

Define $\boldsymbol{H} = \Psi^*\Phi$ and $\boldsymbol{R} = \sum_{i=1}^{N} \boldsymbol{h}_i\boldsymbol{h}_i^T = \boldsymbol{H}\boldsymbol{H}^T$. Then, $J_1[\boldsymbol{A}]$ can be written as

$$J_1[\boldsymbol{A}] = \mathrm{tr}[\boldsymbol{M}^{-1}\boldsymbol{R}] - 2\mathrm{tr}[\boldsymbol{A}^T\boldsymbol{R}\boldsymbol{A}] + \mathrm{tr}[\boldsymbol{A}^T\boldsymbol{R}\boldsymbol{A}\boldsymbol{A}^T\boldsymbol{M}\boldsymbol{A}]. \tag{3}$$

The minimizer of this cost function can be obtained by solving the generalized eigenvalue problem of symmetric matrix pair $(\boldsymbol{R}, \boldsymbol{M})$, as studied in [6]. When $\Psi = \Phi$, $\boldsymbol{H}$ is the same as the Gram matrix, that is, $\boldsymbol{H} = \boldsymbol{K}$; therefore, the KPCA problem becomes a standard eigenvalue problem of $\boldsymbol{K}$.

As pointed out in [7,12], it should be noted that $J_1[\boldsymbol{A}] = J_1[\boldsymbol{A}\boldsymbol{V}]$, where $\boldsymbol{V}$ is any $r \times r$ orthogonal matrix. This implies that by minimizing J_1, we can only track the subspace spanned by $\boldsymbol{A}$, not the eigenvectors. This basis ambiguity problem can be solved by using a so-called weighted updating rules (see [13], for example of PCA). As suggested in [12], instead of using (3), we use the modified weighted cost given as

$$J[\boldsymbol{A}] = \mathrm{tr}[\boldsymbol{M}^{-1}\boldsymbol{R}] - 2\mathrm{tr}[\boldsymbol{D}\boldsymbol{A}^T\boldsymbol{R}\boldsymbol{A}] + \mathrm{tr}[\boldsymbol{A}^T\boldsymbol{R}\boldsymbol{A}\boldsymbol{A}^T\boldsymbol{M}\boldsymbol{A}], \tag{4}$$

where $\boldsymbol{D}$ is a diagonal matrix with positive entries in descending order.

3. ADAPTIVE TRACKING ALGORITHM

To develop the updating rule at time k, we introduce time-varying parameters. Let $\boldsymbol{u}[k]$ be the kth input signal and define $\phi[k] = \phi(\boldsymbol{u}[k])$. Let $\Phi[k] = [\phi[1], \ldots, \phi[k]] : \mathcal{H} \to \mathbb{R}^k$. We assume that the number of entries in Ψ varies with time, thus we denote it by $L[k]$. Define $\Psi[k] : \mathcal{H} \to \mathbb{R}^{L[k]}$ as an operator consisting of $L[k]$ functions in $\Phi[k]$. Let $\boldsymbol{R}[k]$ be the estimation of $\boldsymbol{R}$ at time k.

Using (4) and the above defined quantities, the kth update of $\boldsymbol{A}$ is obtained by minimizing the following time-varying cost function:

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

where $\boldsymbol{M}[k] = \Psi^*[k]\Psi[k]$ and $\boldsymbol{A}[k] \in \mathbb{R}^{L[k]\times r}$. It should be noted that $\phi[k]$ represented in terms of $\Psi[k]$ is given by $\boldsymbol{h}[k] = \Psi^*[k]\phi[k]$. Let us introduce the notation, $\boldsymbol{c}[k] = \boldsymbol{A}^T[k-1]\boldsymbol{h}[k]$, which will be used later on.

3.1. LMS-Type Algorithm

By direct differentiation of $J_k[\boldsymbol{A}[k]]$ with respect to $\boldsymbol{A}[k]$, we get the gradient given as $\partial_{\boldsymbol{A}[k]}J_k = -2(\boldsymbol{R}[k]\boldsymbol{A}[k]\boldsymbol{D} - \boldsymbol{M}[k]\boldsymbol{A}[k]\boldsymbol{A}^T[k]\boldsymbol{R}[k]\boldsymbol{A}[k])$. To develop an LMS-like algorithm, in a way similar to the derivation of LMS, $\boldsymbol{R}[k]$ is replaced by $\boldsymbol{h}[k]\boldsymbol{h}^T[k]$. Therefore, the update rule can be the following. In the algorithm, $F_1(\cdot)$, $F_2(\cdot)$, and $F_3(\cdot)$ will be defined and discussed later.

Algorithm 1 LMS-type online KPCA algorithm

1: $\Psi[k] = F_1(\Psi^*[k-1])$
2: $\boldsymbol{h}[k] = \Psi^*[k]\phi[k]$
3: $\tilde{\boldsymbol{A}}[k-1] = F_2(\boldsymbol{A}[k-1])$
4: $\boldsymbol{c}[k] = \tilde{\boldsymbol{A}}^T[k-1]\boldsymbol{h}[k]$
5: $\boldsymbol{M}[k] = F_3(\boldsymbol{M}[k-1])$
6: $\boldsymbol{g}[k] = \tilde{\boldsymbol{A}}[k-1]\boldsymbol{c}[k]$
7: $\boldsymbol{A}[k] = \tilde{\boldsymbol{A}}[k-1] + \eta(\boldsymbol{h}[k]\boldsymbol{c}^T[k]\boldsymbol{D} - \boldsymbol{M}[k]\boldsymbol{g}[k]\boldsymbol{c}^T[k])$

3.2. RLS-Type Algorithm

By defining $\boldsymbol{R}_{hc}[k] = \boldsymbol{R}[k]\boldsymbol{A}[k]$ and $\boldsymbol{R}_c[k] = \boldsymbol{A}^T[k]\boldsymbol{R}[k]\boldsymbol{A}[k]$, the cost function can be rewritten as [6]

$$J_k[\boldsymbol{A}[k]] = \mathrm{tr}[\boldsymbol{M}^{-1}[k]\boldsymbol{R}[k]] - 2\mathrm{tr}[\boldsymbol{D}\boldsymbol{A}^T[k]\boldsymbol{R}_{hc}[k]] + \mathrm{tr}[\boldsymbol{R}_c[k]\boldsymbol{A}^T[k]\boldsymbol{M}[k]\boldsymbol{A}[k]],$$

which is quadratic, therefore, it is minimized by

$$\boldsymbol{A}[k] = \boldsymbol{M}^{-1}[k]\boldsymbol{R}_{hc}[k]\boldsymbol{D}\boldsymbol{R}_c^{-1}[k]. \tag{5}$$

Assume that $\Psi[k] = \Psi[k-1]$[1]. Then the update of $\boldsymbol{R}[k]$ can be made as

$$\boldsymbol{R}[k] = \beta\boldsymbol{R}[k-1] + \boldsymbol{h}[k]\boldsymbol{h}^T[k]. \tag{6}$$

[1]The argument of the case when $\Psi[k] \neq \Psi[k-1]$ will be given in another paper.

We apply the so-called projection approximation [14] described as $\boldsymbol{R}[k]\boldsymbol{A}[k] \approx \boldsymbol{R}[k]\boldsymbol{A}[k-1]$. With this approximation and (6), we obtain the following iterative updates:

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

Moreover, we define $\boldsymbol{P}[k] = \boldsymbol{M}^{-1}[k]$, and $\boldsymbol{Q}[k] = \boldsymbol{R}_c^{-1}[k]$. Based on the above preparation, RLS-type updating rule can be derived as follows. In the algorithm, $F_4(\cdot)$ will be defined later.

Algorithm 2 RLS-type online KPCA algorithm

1: Do 1 to 4 of Algorithm 1
2: $\boldsymbol{g}[k] = \dfrac{\boldsymbol{Q}[k-1]\boldsymbol{c}[k-1]}{\beta + \boldsymbol{c}^T[k]\boldsymbol{Q}[k-1]\boldsymbol{c}[k]}$
3: $\boldsymbol{P}[k] = F_4(\boldsymbol{P}[k-1])$
4: $\boldsymbol{Q}[k] = \beta^{-1}\left(\boldsymbol{Q}[k-1] - \boldsymbol{g}[k]\boldsymbol{c}^T[k]\boldsymbol{Q}[k-1]\right)$
5: $\hat{\boldsymbol{d}}[k] = \boldsymbol{Q}[k]\boldsymbol{D}\boldsymbol{c}[k]$
6: $\hat{\boldsymbol{A}}[k] = \boldsymbol{A}[k-1] + \boldsymbol{P}[k]\boldsymbol{h}[k]\hat{\boldsymbol{d}}^T[k] - \hat{\boldsymbol{c}}[k]\boldsymbol{g}^T[k]$
7: $\boldsymbol{A}[k] = \boldsymbol{R}$-orthonormalize$(\hat{\boldsymbol{A}}[k])$

3.3. Update of Dictionary Operator

In the context of kernel adaptive filtering, a coherence-based dictionary design method has been proposed [10]. We apply this design method for the proposed KPCA tracking algorithms.

For simplicity, suppose that the norm of the kernel in RKHS is unity, that is, $\kappa(\boldsymbol{u},\boldsymbol{u}) = 1, \boldsymbol{u} \in \mathbb{R}^d$, which is satisfied by the Gaussian kernel. The scheme adds $\phi[k] = \kappa(\cdot, \boldsymbol{u}[k])$ into Ψ if the following condition holds:

$$\max_{1\le j\le L[k-1]} (\Psi^*[k-1]\phi[k])_j \le \delta, \tag{7}$$

where $\delta > 0$ is the positive coherence threshold and $(\cdot)_j$ denotes the jth element of a vector. Thus, we obtain the following update rule for $\Psi[k]$ and $\boldsymbol{A}[k]$, If (7) is satisfied.

$$F_1(\Psi[k-1]) = [\Psi[k-1], \phi[k]],$$
$$F_2(\boldsymbol{A}[k]) = \begin{bmatrix} \boldsymbol{A}[k] \\ \boldsymbol{0}_{1\times r} \end{bmatrix}$$

Moreover, $\boldsymbol{P}[k] = \boldsymbol{M}^{-1}[k]$ should be updated. Note that the number of vectors added in the dictionary is finite as discussed in [10]. In the following, we develop the update rule of $\boldsymbol{P}[k]$ based on the block matrix inversion formula to reduce the computational complexity. Note that the last element of $\boldsymbol{h}[k]$ is always unity because of the assumption for the kernel, therefore, $\boldsymbol{h}[k]$ can be denoted with subvector $\hat{\boldsymbol{h}}[k]$ by $\boldsymbol{h}[k] = [\hat{\boldsymbol{h}}^T[k], 1]^T$. Define $\boldsymbol{m}[k] = \boldsymbol{P}[k-1]\hat{\boldsymbol{h}}[k]$. Finally, we get

$$F_4(\boldsymbol{P}[k-1]) = \begin{bmatrix} \boldsymbol{P}[k-1] & \boldsymbol{0} \\ \boldsymbol{0} & 0 \end{bmatrix} + \frac{1}{1 - \hat{\boldsymbol{h}}^T[k]\boldsymbol{m}[k]} \begin{bmatrix} \boldsymbol{m}[k]\boldsymbol{m}^T[k] & -\boldsymbol{m}[k] \\ -\boldsymbol{m}^T[k] & 1 \end{bmatrix}$$

Furthermore, it can be verified that the computation of $\boldsymbol{P}[k]\boldsymbol{h}[k]$ is significantly simplified to

$$\boldsymbol{P}[k]\boldsymbol{h}[k] = \begin{bmatrix} \boldsymbol{0}_{L[k-1]\times 1} \\ 1 \end{bmatrix}.$$

Proof is omitted due to lack of the space.

4. NUMERICAL EXAMPLE

For the evaluation of performance, we use the benchmark used in [10], which is the nonlinear system described by the difference equation:

$$s_k = (0.8 - 0.5\exp(-s_{k-1}^2))s_{k-1} - (0.3 + 0.9\exp(-s_{k-1}^2))s_{k-2} + 0.1\sin(s_{k-1}\pi)$$

The data were generated from the initial condition, $s_0 = 0.1$ and $s_1 = 0.1$. Outputs s_k were corrupted by a zero-mean Gaussian noise with variance equal to 0.01. We define a data vector stacked with these nonlinear signals. More specifically, $\boldsymbol{u}[k] = [s_{k-d+1}, \ldots, s_k]^T \in \mathbb{R}^d$.

We compared the following three algorithms:

1. LMS-type algorithm,
2. RLS-type algorithm,
3. SubKHA algorithm [8] given as the following update:

$$\boldsymbol{A}[k] = \tilde{\boldsymbol{A}}[k-1] + \eta(\boldsymbol{P}[k]\boldsymbol{h}[k]\boldsymbol{c}^T[k]\boldsymbol{D} - \boldsymbol{g}[k]\boldsymbol{c}^T[k]).$$

In the numerical simulation, the size of sample vectors was set to 6 ($d = 6$) and the first two kernel eigenvectors ($r = 2$) were tracked. The number of generated sample vectors was 5000 ($N = 5000$). We employed here the Gaussian kernel given as $\kappa(\boldsymbol{u}_i, \boldsymbol{u}_j) = \exp(-0.1\|\boldsymbol{u}_i - \boldsymbol{u}_j\|^2)$. The coherence threshold was set to 0.95 ($\delta = 0.95$). For all the algorithms, the following initial parameters were used: $\Psi[0] = [\phi[0], \ldots, \phi[r-1]]$, $\boldsymbol{P}[0] = \boldsymbol{M}^{-1}[0]$, and $\boldsymbol{A}[0] = (\boldsymbol{P}^{1/2}[0])_{1:r}$, where $(\cdot)_{1:r}$ represents the r left columns. The learning rate and forgetting factor were set: $\eta = 0.02$ and $\beta = 0.98$, respectively.

The target kernel principal eigenvectors, φ_i^* was obtained with all the sample vectors through the eigenvalue decomposition of the corresponding Gram matrix of size 5000×5000, whereas we tracked the kernel eigenvectors, $\varphi_i[k]$, with the above algorithms. The similarity between φ_i^* and $\varphi_i[k]$ was evaluated by direction cosine:

$$\text{direction cosine}[k] = \frac{|\langle \varphi_i^*, \varphi_i[k]\rangle|}{\|\varphi_i^*\|\|\varphi_i[k]\|}.$$

Moreover, the approximation ability of this adaptive KPCA was evaluated by the MSE at each time instance:

$$\begin{aligned}\text{MSE}[k] =& N^{-1}\sum_{i=1}^{N}\|\phi_i - W[k]W^*[k]\phi_i\|^2 \\ =& N^{-1}\left\{\text{tr}[\boldsymbol{K}] - 2\text{tr}[\boldsymbol{A}^T[k]\boldsymbol{H}[k]\boldsymbol{H}^T[k]\boldsymbol{A}[k]]\right. \\ &\left. + \text{tr}[(\boldsymbol{A}^T[k]\boldsymbol{M}[k]\boldsymbol{A}[k])(\boldsymbol{A}^T[k]\boldsymbol{H}[k]\boldsymbol{H}^T[k]\boldsymbol{A}[k])]\right\},\end{aligned}$$

where $\boldsymbol{H}[k] = \Psi^*[k]\Phi$.

Figures 1 and 2 depicts the evolution of direction cosines of the first and second kernel principal eigenvectors. We can see that RLS-type algorithm gives the fastest convergence to the 1st kernel eigenvector represented by the all 5000 samples, while the proposed algorithms as well as SubKHA chose only eighteen (18) samples to compose the dictionary operator (Ψ) in the end of iterations. Moreover, we observe that the 1st eigenvector is tracked faster than the 2nd. This is the same phenomena as standard eigenvector tracking algorithms exhibit [6, 7].

To evaluate the behavior of algorithms from another aspect, MSE is calculated and plotted in Fig. 3. In the sense of MSE, RLS-type gives the fastest convergence and indeed the MSE of RLS-type after convergence is very close to the MSE of KPCA. We see a jump at around the 10th iteration, which may be from oscillation of the 2nd eigenvectors observed in Fig. 2.

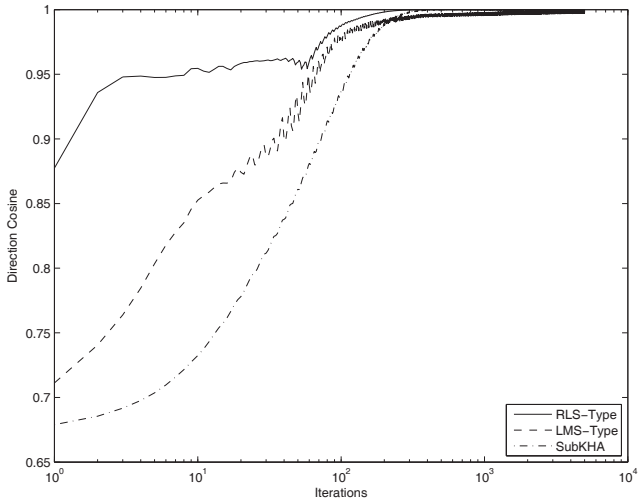

Fig. 1. Direction cosine of the 1st principal kernel eigenvector

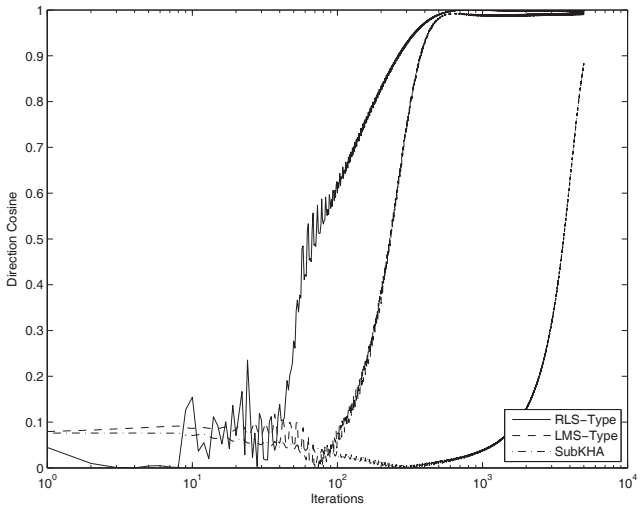

Fig. 2. Direction cosine of the 2nd principal kernel eigenvector

5. CONCLUSION

Motivated by recursive least squares algorithms, we have developed a fast online algorithm for simultaneously extracting kernel principal eigenvectors. It has been proven that the problem to find kernel principal components is reduced to a generalized eigenvalue problem. The adaptive algorithm can be derived from the mean squares error criterion. We have shown that the tracking problem of kernel eigenvectors is reduced to the one of generalized eigenvectors.

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] 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.

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

[4] 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.

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

[6] 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.

[7] T. Tanaka, "Fast generalized eigenvector tracking based on the power method," *IEEE Signal Processing Letters*, vol. 16, pp. 969–972, Nov. 2009.

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

[9] Y. Washizawa, "Subset kernel principal component analysis," in *Proc. IEEE Int. Workshop on Machine Learning for Signal Processing (MLSP 2009)*, pp. 1–6, Oct. 2009.

[10] 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.

[11] A. Ben-Israel and T. N. E. Greville, *Generalized Inverse: Theory and Applications*. New York: John Wiley & Sons, 1974.

[12] J. Yang, Y. Zhao, and H. Xi, "Weighted rule based adaptive algorithm for simultaneously extracting generalized eigenvectors," *IEEE Trans. Neural Networks*, vol. 22, pp. 800–806, May 2011.

[13] T. Tanaka, "Generalized weighted rules for principal components tracking," *IEEE Trans. Signal Processing*, vol. 53, pp. 1243–1253, Apr. 2005.

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

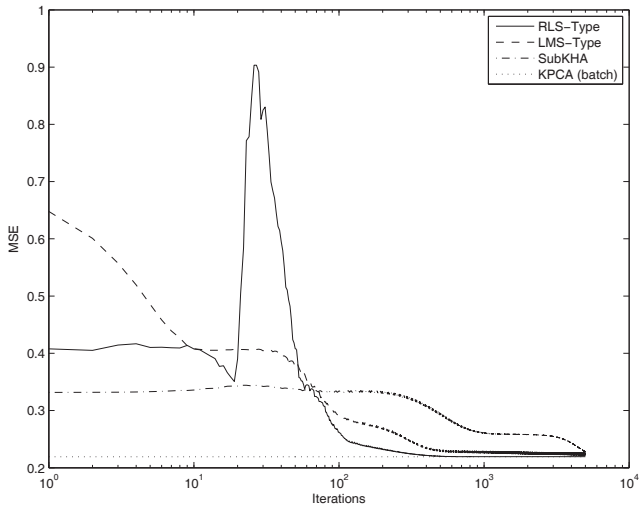

Fig. 3. Mean square error (MSE)