2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

A Fast Incremental Kernel Principal Component Analysis for Data Streams

Annie anak Joseph and Seiichi Ozawa

Abstract— **Kernel Principal Component Analysis (KPCA) is widely used feature extraction as it have been proven that KPCA is powerful in many areas in pattern recognition. Considering that the conventional KPCA should decompose a kernel matrix of all training data, this would be an unrealistic assumption for data streams in real-world applications. Therefore, in this paper, we propose an online feature extraction called Chunk Incremental Kernel Principal Component Analysis (CIKPCA) that can handle data streams in an incremental mode. In the proposed method, the training data are assumed to be given in a chunk of multiple data at one time. In CIKPCA, an eigen-feature space is updated by solving the eigenvalue decomposition once whenever a chunk of data is given. However, if a chunk size is large, a kernel matrix to be decomposed is also large, resulting in high computational time. Considering that not all the data are useful for the eigen-feature space learning, the data in a chunk are first selected based on the importance. Several benchmark data sets in the UCI Machine Learning Repository are used to evaluate the performance of the proposed method. The experimental results show that our proposed method can accelerate the learning of the eigen-feature space compared to Takeuchi et al.'s IKPCA without reducing the recognition accuracy.**

I. Introduction

Along with the rapid development of the technologies in real-world applications, data are often continuously generated in our life. This kind of data are called "data streams". Therefore, the importance of incremental learning is increasing due to the demand of online learning in real-time recognition is increased. An Incremental learning is not only important in classifier but also for feature extraction part. Principal Component Analysis (PCA) [1] has been one of the well-known feature extraction method in machine learning. Therefore, there have been proposed many extended works on PCA [2], [3], [4], [5], [6], [7]. One of them is Kernel Principal Component Analysis (KPCA) [8] which performs a nonlinear form of PCA. The main idea of KPCA is to map an input data into a higher dimensional feature space by using some specially designed mapping function ϕ. Then, principal components are computed in a very high or infinity dimensional feature space to approximate the data distribution by a low dimensional subspace called "eigen-feature space". However, the traditional batch KPCA has its own limitation where the computation time is increased with the increasing number of training data. This is because the size of a kernel matrix to be decomposed is increased for the larger size of the training data. Hence, the larger the training data are, the higher the computation time are. In this case, an incremental learning algorithm, which can update an eigenspace model without keeping all the training data, is solicited.

Annie anak Joseph and Seiichi Ozawa are with the Graduate School of Engineering, Kobe university, Kobe 657-8501, Japan (email: 097t805t@stu.kobe-u.ac.jp, ozawasei@kobe-u.ac.jp).

Recently, Chin and Suter [5] proposed an incremental KPCA that can update an eigen-feature space incrementally by using a singular value decomposition. However, their proposed IKPCA requires large computation and memory costs for the high-dimensional data. On the other hand, Takeuchi et al. then proposed another incremental KPCA (IKPCA) [9] where they can approximate an eigen-feature space using linearly independent data for every incoming data. However, their proposed method still remain an open problem. Considering that the training data could appear in a chunk, IKPCA should be repeatedly applied to the individual data in a chunk. Therefore, the update of the eigen-feature space should be conducted for every incoming data. In order to overcome this problem, when a chunk of multiple data is given, these data should be learned at the same time. However, when a large chunk of data is given, the computational time of the independent data selection could dominate over those of the eigenvalue decomposition. Hence, the large computation and memory costs still remained an issue for the large chunk data [10].

To tackle this problem, we propose an extended approach to Takeuchi et al.'s IKPCA. Here, we assume that not all of the data are useful for the eigen-feature space learning. Therefore, when a chunk of multiple data is given, the data are first subdivided into small chunks and the useful data are selected from the small chunks. It is expected that the computation and memory costs could be reduced when the training data in a chunk are reduced. Then, the linearly independent data are selected from the reduced data. The eigenvectors are represented by the linear sum of the linearly independent data which are selected from the reduced data in a chunk. This is because the eigenvectors are not obtained directly in a high-dimensional feature space.

The rest of this paper is organized as follows: In Section II, we briefly explain KPCA and Takeuchi et al.'s IKPCA. Then, we proposed another incremental KPCA by extending Takeuchi et al.'s IKPCA called Chunk IKPCA (CIKPCA) in Section III. In Section IV, the performance of the proposed CIKPCA is evaluated by using several benchmark data sets from the UCI machine learning repository [11], and we give conclusions in Section V.

978-1-4799-1484-5/14/$31.00 ©2014 IEEE

II. Kernel Principal Component Analysis (KPCA)

Assume that N training data $\boldsymbol{X} = \{(\boldsymbol{x}_i, d_i)\}_{i=1}^{N}$ are given initially where $\boldsymbol{x}_i$ and d_i are the ith input vector and its class label, respectively. These data are mapped into a feature space using a mapping function $\phi(\cdot)$. Here, $\phi(\boldsymbol{x})$ is the l-dimensional data. A covariance matrix $\boldsymbol{Q}$ of data in the feature space is given by

$$\boldsymbol{Q} = \frac{1}{N}\sum_{i=1}^{N}(\phi(\boldsymbol{x}_i) - \boldsymbol{c})(\phi(\boldsymbol{x}_i) - \boldsymbol{c})^T \tag{1}$$

where $\boldsymbol{c}$ is a mean vector of the mapped data and it is defined as

$$\boldsymbol{c} = \frac{1}{N}\sum_{i=1}^{N}\phi(\boldsymbol{x}_i). \tag{2}$$

In KPCA, we have to find eigenvalues $\boldsymbol{\Lambda}$ and the associated eigenvectors $\boldsymbol{Z}$. These are obtained by solving the following eigenvalue problem:

$$\boldsymbol{Q}\boldsymbol{Z} = \boldsymbol{Z}\boldsymbol{\Lambda}. \tag{3}$$

Since the dimensionality l could be infinity, the eigenvalue problem is not solved directly. Therefore, the so called kernel trick is applied to Eq. (3) by using the following kernel function:

$$k(\boldsymbol{x}, \boldsymbol{x}') = \phi(\boldsymbol{x})^T\phi(\boldsymbol{x}'). \tag{4}$$

Eigenvector matrix $\boldsymbol{Z}$ is represented as follows:

$$\boldsymbol{Z} = [\phi(\boldsymbol{x}_1), \cdots, \phi(\boldsymbol{x}_m)]\begin{bmatrix} \alpha_{1,1} & \cdots & \alpha_{1,m} \\ \vdots & \ddots & \vdots \\ \alpha_{m,1} & \cdots & \alpha_{m,m} \end{bmatrix} = \boldsymbol{\Phi}_m\boldsymbol{A}_m \tag{5}$$

where m is a set of linearly independent data that span a feature space. The kernel matrix can be represented by

$$\boldsymbol{H}_{ij} = \begin{bmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & \cdots & k(\boldsymbol{x}_1, \boldsymbol{x}_j) \\ \vdots & \ddots & \vdots \\ k(\boldsymbol{x}_i, \boldsymbol{x}_1) & \cdots & k(\boldsymbol{x}_i, \boldsymbol{x}_j) \end{bmatrix}. \tag{6}$$

Finally, a kernel eigenvalue problem [12] is given by

$$\frac{1}{N}\boldsymbol{L}^{-1}\boldsymbol{H}_{Nm}^T(\boldsymbol{I}_N - \boldsymbol{1}_N)\boldsymbol{H}_{Nm}(\boldsymbol{L}^{-1})^T(\boldsymbol{L}^T\boldsymbol{\alpha}_i) = \lambda_i(\boldsymbol{L}^T\boldsymbol{\alpha}_i). \tag{7}$$

Here, λ_i is an eigenvalue ($\lambda_i \geq 0$) and $\boldsymbol{L}^T\boldsymbol{\alpha}_i$ is the associated ith eigenvector spanning a feature space. $\boldsymbol{I}_N$ is an $N \times N$ unit matrix and $\boldsymbol{1}_N$ is an $N \times N$ matrix with all elements are $1/N$. $\boldsymbol{L}$ is a lower triangular matrix and it is obtained by the Cholesky factorization, $\boldsymbol{H}_{mm} = \boldsymbol{L}\boldsymbol{L}^T$.

From the m eigenvectors, only the first d largest eigenvalues are selected to form a low dimensional eigen-feature space. The criterion to determine the augmentation of the eigen-feature space is by measuring the accumulation ratio below:

$$C(d) = \frac{\sum_{i=1}^{d}\lambda_i}{\sum_{i=1}^{m}\lambda_i}. \tag{8}$$

The accumulation ratio $C(d)$ indicates how much the information remains in the feature space after the d principal components are selected.

In Takeuchi et al.'s incremental KPCA, the learning is carried out for every incoming data. In this case, a kernel eigenvalue problem needs to be solved one by one. Hence, the computation time are considered to be high. Therefore, more efficient algorithm is needed for data streams under realistic environments.

III. Proposed Chunk IKPCA

The novelty of this paper lies in the reduction of computation and memory costs by introducing a mechanism of data selection in a chunk. Before an eigen-feature space augmentation, a chunk of data should be first subdivided into smaller chunks and the useful data are selected from the smaller chunks. In the following, the five main operations of the proposed method are described in details. In the final part of this section, the algorithm of the proposed method is summarized.

A. Data Selection

Before an incremental learning is carried out, batch KPCA is applied to the initial data $\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1}^{N}$. Then an eigen-feature space is constructed based on the initial data. The incremental learning is conducted to update the eigen-feature space when a chunk of new data $\boldsymbol{Y} = \{\boldsymbol{y}_i\}_{i=1}^{L}$ is given. The data in a chunk should be first selected based on the accumulation ratio before updating the eigen-feature space. Here, the accumulation ratio is used to judge whether a new eigenaxis should be added. It is the ratio of amount of information between eigenaxes and the original feature space. However, large computation and memory costs are needed to obtain an accumulation ratio if a chunk size is large. Takaomi et al. [10] investigated the influence of chunk size to the learning time. They found that more time is required for the large chunk data unless the large chunk data are divided into small chunks. Therefore, to overcome the problem of large computational time, when a new chunk of data is given, the data are first subdivided into smaller chunks. For the notational simplicity, let us denote a divided chunk as $\boldsymbol{y} = \{\boldsymbol{y}_i\}_{i=1}^{\tilde{l}}$. Then, the useful data are selected from $\boldsymbol{y}$ based on the accumulation ratio. Here, the accumulation ratio is updated with the data in a smaller chunk as follows:

$$C'(d) = \frac{\sum_{i=1}^{d}\lambda_i'}{\sum_{i=1}^{m}\lambda_i'}, \tag{9}$$

where

$$\sum_{i=1}^{d}\lambda_i' = \frac{N}{N+\tilde{l}}\sum_{i=1}^{d}\lambda_i + \frac{N\tilde{l}}{(N+\tilde{l})^2}\sum_{i=1}^{d}\left\{\boldsymbol{\alpha}_i^T(\tilde{\boldsymbol{c}} - \tilde{\boldsymbol{c}_y})\right\}^2 + \frac{1}{N+\tilde{l}}\sum_{i=1}^{d}\sum_{j=1}^{\tilde{l}}\left\{\boldsymbol{\alpha}_i^T(\boldsymbol{K}_m(\boldsymbol{y}_j) - \tilde{\boldsymbol{c}})\right\}^2 \tag{10}$$

$$\sum_{i=1}^{m}\lambda_i' = \frac{N}{N+\tilde{l}}\sum_{i=1}^{m}\lambda_i + \frac{N\tilde{l}}{(N+\tilde{l})^2}\boldsymbol{c}^T\boldsymbol{c} - \frac{\tilde{l}}{(N+\tilde{l})^2}\sum_{i=1}^{\tilde{l}}\sum_{j=1}^{\tilde{l}}k(\boldsymbol{y}_i,\boldsymbol{y}_j) - \frac{2N}{(N+\tilde{l})^2}\sum_{i=1}^{\tilde{l}}\boldsymbol{\beta}_i^T\tilde{\boldsymbol{c}} + \frac{1}{N+\tilde{l}}\sum_{j=1}^{\tilde{l}}k(\boldsymbol{y}_j,\boldsymbol{y}_j). \quad (11)$$

Here, $\tilde{\boldsymbol{c}}$, $\tilde{\boldsymbol{c}_y}$, $\boldsymbol{K}_m(\boldsymbol{y}_j)$ and $\boldsymbol{\beta}_i$ are defined by

$$\boldsymbol{\Phi}_m^T\boldsymbol{c} = \frac{1}{N}\left[\sum_{i=1}^{N}k(\hat{\boldsymbol{x}}_1,\boldsymbol{x}_i),\cdots,\sum_{i=1}^{N}k(\hat{\boldsymbol{x}}_m,\boldsymbol{x}_i)\right]^T \equiv \tilde{\boldsymbol{c}} \quad (12)$$

$$\boldsymbol{\Phi}_m^T\boldsymbol{c}_y = \frac{1}{\tilde{l}}\left[\sum_{i=1}^{\tilde{l}}k(\hat{\boldsymbol{x}}_1,\boldsymbol{y}_i),\ \cdots,\sum_{i=1}^{\tilde{l}}k(\hat{\boldsymbol{x}}_m,\boldsymbol{y}_i)\right]^T \equiv \tilde{\boldsymbol{c}}_y \quad (13)$$

$$\boldsymbol{\Phi}_m^T\phi(\boldsymbol{y}_j) = \left[k(\hat{\boldsymbol{x}}_1,\boldsymbol{y}_j),\cdots,k(\hat{\boldsymbol{x}}_m,\boldsymbol{y}_j)\right]^T \equiv \boldsymbol{K}_m(\boldsymbol{y}_j) \quad (14)$$

$$\boldsymbol{\beta}_i \equiv \boldsymbol{H}_{mm}^{-1}\boldsymbol{K}_m(\boldsymbol{y}_i). \quad (15)$$

When the updated accumulation ratio in Eq. (9) is lower than a threshold θ, the useful data should be included in the set of linearly independent data $\boldsymbol{\Phi}_m$. In order to construct a minimum dimensional eigen-feature space, the useful data are selected based on the highest variance $\sigma^2(\phi(\boldsymbol{y}_i))$ from $\boldsymbol{y}$. Then, $\phi(y_i)$ with the highest $\sigma^2(\phi(\boldsymbol{y}_i))$ is redefined as the $(m+1)$th independent data $\phi(\hat{\boldsymbol{x}}_{m+1})$ and added to $\boldsymbol{\Phi}_m$ as shown below.

$$\boldsymbol{\Phi}_{m+1} = \left[\ \boldsymbol{\Phi}_m \quad \phi(\hat{\boldsymbol{x}}_{m+1})\ \right]. \quad (16)$$

Here, $\sigma^2(\phi(\boldsymbol{y}_i))$ is calculated as follows:

$$\sigma^2(\phi(\boldsymbol{y}_i)) \approx \frac{1}{\|\boldsymbol{h}_i\|^2(N+\tilde{l})}\sum_{j=1}^{\tilde{l}}\Big\{k(\boldsymbol{y}_i,\boldsymbol{y}_j) - \frac{N}{N+\tilde{l}}\boldsymbol{\beta}_i^T\tilde{c} - \frac{1}{N+\tilde{l}}\sum_{k=1}^{\tilde{l}}k(\boldsymbol{y}_i,\boldsymbol{y}_k) - \sum_{k=1}^{d}\left(\boldsymbol{K}_m^T(\boldsymbol{y}_i)\boldsymbol{\alpha}_k\right)\boldsymbol{\alpha}_k^T \times\left\{\boldsymbol{K}_m(\boldsymbol{y}_j) - \frac{N\tilde{\boldsymbol{c}}+\tilde{l}\tilde{\boldsymbol{c}}_y}{N+\tilde{l}}\right\}\Big\}^2 \quad (17)$$

and $\tilde{\boldsymbol{c}}$, $\tilde{\boldsymbol{c}}_y$, and $\boldsymbol{K}_m(\boldsymbol{y}_i)$ are updated by

$$\tilde{\boldsymbol{c}} = \boldsymbol{\Phi}_{m+1}^T\boldsymbol{c} = \begin{bmatrix}\boldsymbol{\Phi}_m^T\boldsymbol{c}\\ \beta\left(\hat{\boldsymbol{x}}_{m+1}\right)^T\boldsymbol{\Phi}_m^T\boldsymbol{c}\end{bmatrix} \quad (18)$$

$$\tilde{\boldsymbol{c}}_y = \boldsymbol{\Phi}_{m+1}^T\boldsymbol{c}_y = \begin{bmatrix}\boldsymbol{\Phi}_m^T\boldsymbol{c}_y\\ \frac{1}{\tilde{l}}\sum_{i=1}^{\tilde{l}}k(\hat{\boldsymbol{x}}_{m+1},\boldsymbol{y}_i)\end{bmatrix} \quad (19)$$

$$\boldsymbol{K}_{m+1}(\boldsymbol{y}_i) = \begin{bmatrix}\boldsymbol{K}(\boldsymbol{y}_i)\\ k(\hat{\boldsymbol{x}}_{m+1},\boldsymbol{y}_i)\end{bmatrix}. \quad (20)$$

Besides, the coefficient matrix $\boldsymbol{A}_d$ is updated by

$$\boldsymbol{A}_{d+1} = \begin{bmatrix}\boldsymbol{A}_d & -\frac{1}{\|\boldsymbol{h}_i\|}\sum_{j=1}^{d}\left(\boldsymbol{K}_m^T(\boldsymbol{y}_i)\boldsymbol{\alpha}_j\right)\boldsymbol{\alpha}_j\\ \boldsymbol{0} & \frac{1}{\|\boldsymbol{h}_i\|}\end{bmatrix}. \quad (21)$$

This procedure is repeated until the accumulation ratio becomes higher than a threshold θ. For the notational simplicity, the selected data are denoted as $\tilde{\boldsymbol{Y}} = \{\boldsymbol{y}_i\}_{i=1}^{\tilde{l}}$. The data are arranged into smaller chunks if the size of useful data $\tilde{\boldsymbol{Y}}$ are large. This is to ensure the constant update speed and memory usages. Then, the necessity of the eigen-feature augmentation is checked after the process of data selection.

B. Feature Space Augmentation

In the initial learning phase, the number of eigen-axes are determined by finding minimum d such that the accumulation ratio in Eq. (9) exceeds a threshold. The eigenvectors $\boldsymbol{Z} = \{z_1,\ldots,z_d\} \in \Re^{l\times d}$ are represented as the linear combination of m independent data $\boldsymbol{\Phi}_m$

$$\boldsymbol{z}_i = [\phi(\hat{\boldsymbol{x}}_1),\cdots,\phi(\hat{\boldsymbol{x}}_m)]\begin{bmatrix}\alpha_{1i}\\ \vdots\\ \alpha_{mi}\end{bmatrix} = \boldsymbol{\Phi}_m\boldsymbol{\alpha}_i. \quad (22)$$

Using the selected data $\tilde{\boldsymbol{Y}}$, the accumulation ratio in Eq. (8) is updated based on Eqs. (9)-(15). If the selected data includes almost all energy in the current eigen-feature space, the dimensionality does not need to be expanded. However, if the eigen-feature space includes certain energy in the complementary eigen-feature space, the dimensional augmentation is needed, or essential information would be lost. Therefore, if the accumulation ratio is lower than θ, it means that selected data may contain some linearly independent data. Then, linearly independent data should be searched and added to the set of linearly independent data $\boldsymbol{\Phi}_m$.

C. Selection of Eigen-axis

Assume that the updated accumulation ratio in Eq. (9) is lower than a θ, it means that linearly independent should be obtained from $\tilde{\boldsymbol{Y}}$. The data with the highest variance $\sigma^2(\phi(\boldsymbol{y}_i))$ is found from the selected data. The variance of data projected to the eigenvector is calculated by using Eq. (17) and $\tilde{\boldsymbol{c}}$, $\tilde{\boldsymbol{c}}_y$, and $\boldsymbol{K}_m(\boldsymbol{y}_i)$ are updated by Eqs. (18)-(20). Here, $\phi(\boldsymbol{y}_i)$ is defined as the $m+1$th linearly independent data and added to a set of $\boldsymbol{\Phi}_m$ as shown in Eq. (16). Besides, the coefficient matrix is updated by using Eq. (21). Here, we rewrite $m+1$ as m' and $d+1$ as d' for the sake of the notational simplicity. By adding $\sigma^2(\phi(\boldsymbol{y}_i))$ to the numerator of Eq. (9), the accumulation ratio $C(d)$ is updated. If the updated accumulation ratio $C'(d)$ is still smaller than a threshold θ, another data should be selected. Assume that

we select $(m'+1)$-th data. A candidate axis is obtained as follows:

$$\hat{\boldsymbol{z}}_i' = \frac{\boldsymbol{h}_i'}{||\boldsymbol{h}_i'||} \tag{23}$$

where

$$\begin{aligned} \boldsymbol{h'}_i &= \left[\begin{array}{cc} \boldsymbol{\Phi}_{m'} & \phi(\boldsymbol{y}_i) \end{array} \right] \\ &\times \left[\begin{array}{cc} \boldsymbol{A}_{d'} & -\sum_{j=1}^{d'} \left(\boldsymbol{K}_{m'}^T(\boldsymbol{y}_i)\boldsymbol{\alpha}_j \right) \boldsymbol{\alpha}_j \\ \boldsymbol{0} & 1 \end{array} \right]. \end{aligned} \tag{24}$$

In Eq. (23), $||\boldsymbol{h}_i'||$ can be obtained by calculating its squared norm as follows:

$$\begin{aligned} ||\boldsymbol{h'}_i||^2 &= k(\boldsymbol{y}_i, \boldsymbol{y}_i) - 2\sum_{j=1}^{d'} \left(\boldsymbol{K}_{m'}^T(\boldsymbol{y}_i)\boldsymbol{\alpha}_j \right)^2 \\ &+ \sum_{j=1}^{d'}\sum_{k=1}^{d'} \left(\boldsymbol{K}_{m'}^T(\boldsymbol{y}_i)\boldsymbol{\alpha}_k \right) \left(\boldsymbol{K}_{m'}^T(\boldsymbol{y}_i)\boldsymbol{\alpha}_k \right) \\ &\times \boldsymbol{\alpha}_j^T \boldsymbol{H}_{(m')(m')} \boldsymbol{\alpha}_k. \end{aligned} \tag{25}$$

The variance $\sigma^2(\phi(\boldsymbol{y}_i))$ can be calculated as follows:

$$\begin{aligned} \sigma'^2(\phi(\boldsymbol{y}_i)) &= \frac{1}{||\boldsymbol{h'}_i||^2(N+\tilde{L})} + \sum_{j=1}^{\tilde{L}} \Big\{ k(\boldsymbol{y}_i, \boldsymbol{y}_j) \\ &- \frac{N}{N+\tilde{L}} \boldsymbol{\beta}_i^T \tilde{\boldsymbol{c}} - \frac{1}{N+\tilde{L}} \sum_{k=1}^{\tilde{L}} k(\boldsymbol{y}_i, \boldsymbol{y}_k) \\ &- \sum_{k=1}^{d'} \left(\boldsymbol{K}_{m'}^T(\boldsymbol{y}_i)\boldsymbol{\alpha}_k \right) \boldsymbol{\alpha}_k^T \boldsymbol{K}_{m'}(\boldsymbol{y}_j) \\ &+ \frac{N}{N+\tilde{L}} \sum_{k=1}^{d'} \left(\boldsymbol{K}_{m'}^T(\boldsymbol{y}_i)\boldsymbol{\alpha}_k \right) \boldsymbol{\alpha}_k^T \tilde{\boldsymbol{c}} \\ &+ \frac{\tilde{L}}{N+\tilde{L}} \sum_{k=1}^{d'} \left(\boldsymbol{K}_{m'}^T(\boldsymbol{y}_i)\boldsymbol{\alpha}_k \right) \\ &\times \boldsymbol{\alpha}_k^T \tilde{\boldsymbol{c}}_y \Big\}^2 \end{aligned} \tag{26}$$

where $\tilde{\boldsymbol{c}}$ and $\tilde{\boldsymbol{c}}_y$ can be calculated recursively as follows:

$$\tilde{\boldsymbol{c}} = \left[\begin{array}{c} \tilde{\boldsymbol{c}} \\ \left(\boldsymbol{H}_{mm}^{-1} \boldsymbol{K}_m(\hat{\boldsymbol{y}}) \right)^T \tilde{\boldsymbol{c}} \end{array} \right] \tag{27}$$

$$\tilde{\boldsymbol{c}}_y = \left[\begin{array}{c} \tilde{\boldsymbol{c}}_y \\ \frac{1}{\tilde{L}} \sum_{i=1}^{\tilde{L}} k(\hat{\boldsymbol{x}}_{m'}, \boldsymbol{y}_i) \end{array} \right] \tag{28}$$

$$\boldsymbol{K}_{m'}(y_i) = \left[\begin{array}{c} \boldsymbol{K}_m(y_{(i)}) \\ k(\hat{\boldsymbol{x}}_{m'}, y_{(i)}) \end{array} \right]. \tag{29}$$

Here, $\phi(\boldsymbol{y}_i)$ with the highest $\sigma'^2(\phi(\hat{\boldsymbol{x}}_{m'+1}))$ is selected as $\phi(\hat{\boldsymbol{x}}_{m'+1})$ and added to $\boldsymbol{\Phi}_{m'+1}$ as follows:

$$\boldsymbol{\Phi}_{m'+1} = \left[\begin{array}{cc} \boldsymbol{\Phi}_{m'} & \phi(\hat{\boldsymbol{x}}_{m'+1}) \end{array} \right] \tag{30}$$

where $\phi(\hat{\boldsymbol{x}}_{m'+1})$ is a new linearly independent data. By adding $\sigma'^2(\phi(\hat{\boldsymbol{x}}_{m'+1}))$ to the numerator of $C(d')$, the new accumulation ratio $C(d'+1)$ is obtained. The coefficient matrix is updated as follows:

$$\boldsymbol{A}_{d'+1} = \left[\begin{array}{cc} \boldsymbol{A}_{d'} & -\frac{1}{||\boldsymbol{h}_i||} \sum_{j=1}^{d'} \left(\boldsymbol{K}_{m'}^T(\boldsymbol{y}_i)\boldsymbol{\alpha}_j \right) \boldsymbol{\alpha}_j \\ \boldsymbol{0} & \frac{1}{||\boldsymbol{h}_i||} \end{array} \right]. \tag{31}$$

This procedure is repeated until the accumulation ratio becomes higher than a threshold θ.

D. Update of Feature Space

The update of feature space depends on two cases. The first case is when the new eigenvectors are added and the second case is when the eigenvector is not added into the feature space. When l new eigenvectors are added, the eigenvector matrix is updated by

$$\boldsymbol{Z'}_{d+l} = \left[\begin{array}{cc} \boldsymbol{Z}_d & \hat{\boldsymbol{Z}}_l \end{array} \right] \boldsymbol{R} \tag{32}$$

and $\hat{\boldsymbol{Z}}_l$ can be represented as follows:

$$\hat{\boldsymbol{Z}}_l = \left[\begin{array}{ccc} \boldsymbol{z}_{d+1} & \cdots & \boldsymbol{z}_{d+l} \end{array} \right] \tag{33}$$

where $\boldsymbol{R} \in R^{(d+l)\times(d+l)}$ is a rotation matrix.

In order to obtain the rotation matrix $\boldsymbol{R}$, we solve the following intermediate kernel eigenvalue problem.

$$\begin{aligned} \frac{1}{N+\tilde{L}} \Bigg(N \left[\begin{array}{cc} \boldsymbol{\Lambda} & \boldsymbol{0}_{d\times l} \\ \boldsymbol{0}_{d\times l} & \boldsymbol{0}_{l\times l} \end{array} \right] + \sum_{i=1}^{\tilde{L}} \boldsymbol{f}_i \boldsymbol{f}_i^T \\ + \frac{N\tilde{L}}{N+\tilde{L}} \boldsymbol{p}\boldsymbol{p}^T \Bigg) \boldsymbol{R} = \boldsymbol{R}\boldsymbol{\Lambda}'_{d+l} \end{aligned} \tag{34}$$

where both $\boldsymbol{f}$ and $\boldsymbol{p}$ are given by

$$\boldsymbol{f}_i = \left[\begin{array}{c} \boldsymbol{\alpha}_1^T \\ \vdots \\ \boldsymbol{\alpha}_{d+l}^T \end{array} \right] \left(\boldsymbol{K}_{m+l}(\boldsymbol{y}_i) - \tilde{\boldsymbol{c}}_y^{(m+l)} \right) \tag{35}$$

$$\boldsymbol{p} = \left[\begin{array}{c} \boldsymbol{\alpha}_1^T \\ \vdots \\ \boldsymbol{\alpha}_{d+l}^T \end{array} \right] \left(\tilde{\boldsymbol{c}}^{(m+l)} - \tilde{\boldsymbol{c}}_y^{(m+l)} \right). \tag{36}$$

On the other hand, when no l new eigenvector is added, the updated eigenvectors are given as follows:

$$\boldsymbol{Z'}_d = \boldsymbol{Z}_d \boldsymbol{R} \tag{37}$$

and the intermediate eigenvalue is given by

$$\frac{1}{N+\tilde{L}} \left(N\boldsymbol{\Lambda}_d + \sum_{i=1}^{\tilde{L}} \boldsymbol{f}_i \boldsymbol{f}_i^T + \frac{N\tilde{L}}{N+\tilde{L}} \boldsymbol{p}\boldsymbol{p}^T \right) \boldsymbol{R} = \boldsymbol{R}\boldsymbol{\Lambda'}_d. \tag{38}$$

Algorithm 1 Proposed Method

Require: Initial training data $\boldsymbol{X} = \{x_i\}_{i=1}^{N}$.
Obtain a threshold of accumulation ratio θ based on the cross validation.
Perform KPCA for $\boldsymbol{X}$ and obtain an initial eigen-feature space $\Omega = \{\boldsymbol{\Phi}_m, \boldsymbol{A}_d, \boldsymbol{\Lambda}_d, N\}$.
Obtain the linearly independent data such that the accumulation ratio $C(d)$ is higher than a θ.
loop
  Input: A new chunk of training data $\boldsymbol{Y} = \{\boldsymbol{y}_i\}_{i=1}^{L}$.
  Subdivide $\boldsymbol{Y}$ into smaller chunks $\boldsymbol{y} = \{\boldsymbol{y}_i\}_{i=1}^{\tilde{l}}$.
  Perform the data selection based on the accumulation ratio and obtain the selected data $\tilde{\boldsymbol{Y}} = \{\boldsymbol{y}_i\}_{i=1}^{\tilde{L}}$.
  Update the $C(d)$ by Eq. (9).
  if $C(d) \geq \theta$ **then**
    Obtain $\boldsymbol{R}$ by Eq. (38).
    Update the coefficient matrix:$\boldsymbol{A}'_d = \boldsymbol{A}_d \boldsymbol{R}$.
  else
    while $C(d) < \theta$ **do**
      for i=1: $\tilde{L}$ **do**
        Calculate variance $\sigma^2(\phi(\boldsymbol{y}_i))$ by Eq. (17).
      end for
      Obtain linearly independent data with the $m+1$ largest variances, $\sigma^2(\phi(\boldsymbol{y}_i))$.
      Update the coefficient matrix $\boldsymbol{A}_{d+1}$ by Eq. (21).
      Calculate the accumulation ratio $C(d+1)$ by adding $\sigma^2(\phi(\boldsymbol{y}_i))$ to the numerator of Eq. (9).
    end while
    Obtain $\boldsymbol{R}$ by solving Eq. (34).
    Update the coefficient Matrix:$\boldsymbol{A}'_{d+l} = \boldsymbol{A}_{d+l} \boldsymbol{R}$.
  end if
  Update $\|\boldsymbol{c}\|^2$ and $\tilde{\boldsymbol{c}}'$ by Eqs.(39)-(40).
end loop

E. Mean Update

$\left\|\boldsymbol{c}^2\right\|$ are updated by

$$\|\boldsymbol{c}'\|^2 = \frac{1}{(N+\tilde{L})^2}\left(N^2\|\boldsymbol{c}\|^2 + 2N\sum_{i=1}^{L}\boldsymbol{\beta}_i^T\tilde{\boldsymbol{c}} + \sum_{i=1}^{\tilde{L}}\sum_{j=1}^{\tilde{L}}k(\boldsymbol{y}_i,\boldsymbol{y}_j)\right) \quad (39)$$

$$\tilde{\boldsymbol{c}}' = \frac{1}{N+\tilde{L}}(N\tilde{\boldsymbol{c}} + \tilde{L}\tilde{\boldsymbol{c}_y}). \quad (40)$$

When the new selected data are given, $\|\boldsymbol{c}'\|^2$ and $\tilde{\boldsymbol{c}}'$ are replaced with $\|\boldsymbol{c}\|^2$ and $\tilde{\boldsymbol{c}}$, respectively.

The brief learning procedures are shown in Algorithm 1.

IV. Experimental Results

A. Experiment Setup

In order to evaluate the effectiveness of our proposed method, we conduct several experiments based on the five data sets which are selected from the UCI Machine Learning Repository [11]. The evaluated data sets are listed in Table I.

For Poker Hand data, 2000 data are randomly selected from the original 1025010 data as train data and another 1000 data are selected as the test data. On the other hand, for Optical-digit data, 1500 data are selected from 3823 data as the train data and all 1797 test data are used to measure the performance. The initial data consists of 100 training data and remaining data are used for the incremental learning. In our experiments, the results are averaged over 30 trials because incremental learning depends on the sequence of the training data. The kernel function used in the experiments is Gaussian kernel $k(\boldsymbol{x},\boldsymbol{y}) = \exp(-\gamma \mid\mid \boldsymbol{x}-\boldsymbol{y} \mid\mid^2)$. The parameter γ and θ are selected based on the 5-fold cross validation in the initial learning.

The computer used here has the following specifications:

1) Intel(R) Core(TM)2 Duo (3.16 GHz) CPU.
2) 2GB Random acess memory (RAM).
3) Windows 7 (32 bit) OS.

All the programs are develop with Matlab (R2007b).

B. Performance Evaluation of the CIKPCA

The objective of this experiment is to measure the performance of the proposed method. In this experiment, the recognition accuracy, the computational time, and the eigenspace dimension of the traditional KPCA, IKPCA and CIKPCA are evaluated based on the five UCI Machine Learning Repository data sets.

Table II shows the comparison of the recognition accuracy [%] with the standard deviation, the computational time (sec.) with the standard deviation and the dimension of eigenspace with the standard deviation for the traditional KPCA, IKPCA and CIKPCA. The chunk size is set at $L = 100$.

Table II (a) shows the average and the standard deviation of the recognition accuracy that evaluated over 30 trials. The recognition accuracy of the proposed method is compared with the conventional KPCA and Takeuchi et al.'s IKPCA. From the results, we observed that the recognition accuracy based on the five data sets are almost similar for all the models. The recognition accuracy are almost similar with the other two models although the training data in CIKPCA are reduced. Therefore, it suggests that essential information is not lost actually.

Table II (b) shows the computation time for KPCA, IKPCA and CIKPCA. It is observed that the computational time of CIKPCA is obviously faster than KPCA and IKPCA for all the data sets. In KPCA, the kernel matrix of all training data is computed, leading to the large memory and computation time. The learning of IKPCA is not affected by the chunk sizes. Even though the data are received in a chunk, IKPCA would learn the data one by one and the eigenvalue decomposition is solved for every incoming data. On the other hand, the computational time of the kernel matrix decomposition of the selected data is reduced by reducing the amount of data in a chunk. Furthermore, the eigenvalue decomposition is only solved once when a chunk

TABLE I
Evaluated data sets.

Data Sets	# Attrib.	# Classes	# Train	# Test
Ozone	72	2	1000	848
Thyroid	21	3	3772	3428
Poker Hand	10	10	2000	1000
Optical-digit	64	10	1500	1797
Adult	14	2	22611	22611

TABLE II

PERFORMANCE COMPARISON OF (A) RECOGNITION ACCURACY [%], (B) COMPUTATIONAL TIME (SEC.) AND (C) EIGENSPACE DIMENSION.

(a)

Data Sets	KPCA	IKPCA	CIKPCA
Ozone	95.0±0.01	95.1±0.01	95.1±0.01
Thyroid	91.3±0.02	91.2±0.02	91.4±0.02
Poker Hand	44.1±0.02	44.6±0.02	44.7±0.02
Optical-digit	88.5±0.01	88.5±0.01	88.1±0.01
Adult	73.0±0.26	72.4±0.25	72.6±0.25

(b)

Data Sets	KPCA	IKPCA	CIKPCA
Ozone	117.05±120.60	2.91±7.79	0.53±0.17
Thyroid	792.44±1.88	2.13±1.63	0.86±0.46
Poker Hand	133.30±17.43	1.18±1.00	0.46±0.35
Optical-digit	1293.52±677.33	70.21±241.64	1.53±0.50
Adult	186,418.4±64,463.0	95.6±61.8	38.8±23.8

(c)

Data Sets	KPCA	IKPCA	CIKPCA
Ozone	10.5±12.5	17.9±30.9	8.7±6.4
Thyroid	11.2±2.6	15.0±9.1	16.3±3.8
Poker Hand	8.8±0.9	8.9±2.1	8.5±1.1
Optical-digit	38.4±24.1	111.77±126.5	33.9±10.7
Adult	11.1±4.1	15.9±9.1	13.2±5.3

of selected data is given. Therefore, the computation time for CIKPCA is reduced.

To understand the results of the computational time in Table II (b), let us study the computational complexity of IKPCA and CIKPCA. The highest computation time in IKPCA comes from the calculation of $\|h\|^2$. Its computational complexity is $O(m^2d^2)$ where m and d are the number of linearly independent data and the dimensions of an eigen-feature space, respectively.

On the other hand, the computation in CIKPCA is dominated by the calculation of the denominator of the accumulation ratio shown in Eq.(11) or $\|h\|^2$ in Eq. (25). Its computation complexity is $O(m^3L + nL^2)$ or $O(m^2d^2L)$ where n and L are the number of input space and the size of data chunk, respectively. To calculate the denominator, we need to obtain $\boldsymbol{\beta}^T(\boldsymbol{\Phi}_m^T\boldsymbol{c})$ where $\boldsymbol{\beta}^T = \boldsymbol{H}_{mm}^{-1}\boldsymbol{K}_m(\boldsymbol{x})$ (see Eq. (15)). Here, the computational of $\boldsymbol{H}_{mm}^{-1}$ is $O(m^3)$. In addition, the computation complexity to calculate the third term of the right hand side in Eq.(11) is $O(nL^2)$. Therefore, the computational time is dominated in the calculation of the third term in Eq.(11) when large chunk of L data is received. Otherwise, the computational complexity is dominated in the calculation of $\|h\|^2$ when the moderate size of chunk data is received.

However, in our proposed method, a chunk of data is divided into small chunks and only some training data are selected when a large chunk of data is received. Therefore, the size of data chunk is usually small ($\tilde{l}$ for small chunks and $\tilde{L}$ for selected data in a chunk). In this case, our proposed method would not encounter the problem of large data chunk. The computational complexity of Eq.(11) and $\|h\|^2$ is reduced to $O(m^3 + n)$ and $O(m^2d^2)$, respectively.

In addition, Table II (c) shows the eigen-space dimensionality for the three models. It is observed that the dimension of the eigenspace for CIKPCA are smaller than KPCA for almost all the data sets. Thus, it suggests that although the dimension of the eigen-feature space is small, but it contains almost the same energy as KPCA for the classification. The computational time are reduced with only some useful features.

It is concluded that the computational time of CIKPCA are obviously faster than KPCA and IKPCA while the recognition accuracies are almost similar for all the models. Therefore CIKPCA is more efficient than KPCA and IKPCA.

C. Efficiency of IKPCA with Different Number of Chunk Sizes

As mentioned before, Takaomi et al. [10] found that more time is needed for the large chunk data. Therefore, the objective of this experiment is to measure the influence of the data division and the data selection when a chunk of data is received. The computational complexity of the proposed method is often dominated in the calculation of $\|h\|^2$. Its computational complexity is $O(m^2d^2)$. Thus, in order to measure the important of the data division and the data selection, the computation time of CIKPCA and IKPCA are compared with regards to the different number of chunk sizes. Fig. 1 shows the influence of the chunk sizes on the learning time for the three data sets, Ozone, Thyroid, and Poker Hand data. Since the data in a chunk have been selected, it is expected that the eigen-feature space is constructed with only some useful data. Therefore, the learning time are expected to be shortened. In this experiment, we evaluate the learning time of IKPCA and CIKPCA for the chunks sizes at 100, 300, 500, 800 and 1000.

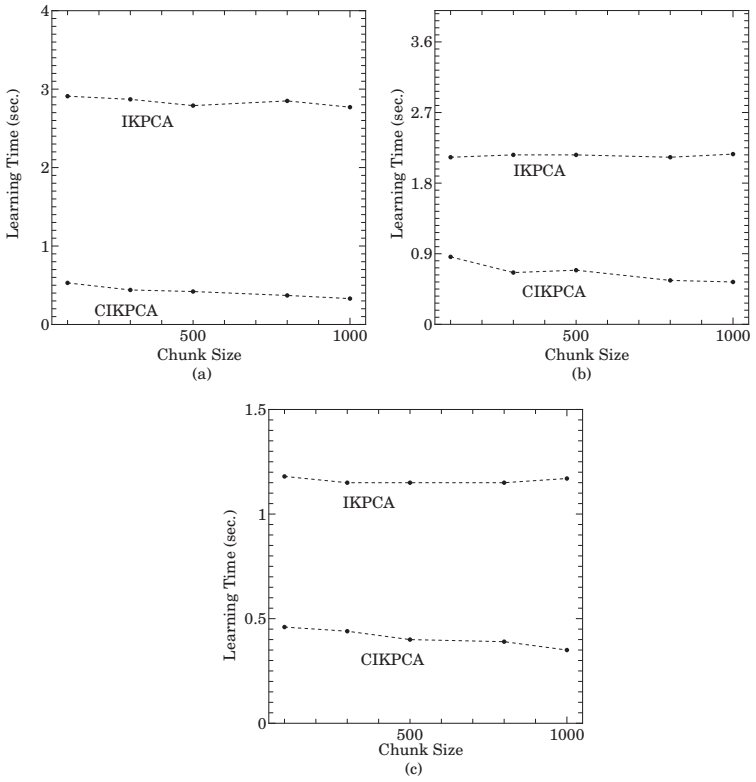

Fig. 1. Learning time (sec.) vs. different number of chunk sizes for (a) Ozone, (b) Thyroid and (c) Poker Hand data.

As seen in Fig. 1, we observed that the computation time for CIKPCA are significantly shorter than IKPCA with regards to the different number of chunk sizes for the three data sets. In CIKPCA, the data are first selected when a chunk of data is given; therefore, the training data in a chunk are reduced. The kernel matrix is computed based on the smaller size of data leading to the significant reduction in the computation time. Furthermore, the computation time are decreased as the chunk size becomes large. This is due to the reduction of the total number of eigenvalue decomposition when the chunk size becomes large. For IKPCA, the training data is learned one by one although the training data are given in a chunk. Hence, the learning time is not affected by the chunk sizes.

Since the computation time are reduced significantly in the proposed method, we further evaluate the efficiency of CIKPCA by evaluating the influence of chunk sizes on the recognition accuracy.

Fig. 2 shows the recognition accuracy of IKPCA and CIKPCA with regards to the different number of chunk sizes for the three data sets. As seen in Fig. 2, the recognition accuracies are almost similar between IKPCA and CIKPCA for the three data sets. The learning accuracies of the proposed method are almost the same with IKPCA with regards to the different number of chunk sizes although the amount of training data are reduced. It suggests that the essential information is not lost and the selected data contains some useful information for the eigen-feature augmentation.

Therefore, it is concluded that the proposed method is more efficient than Takeuchi et al.'s IKPCA where the computation time are reduced significantly without reducing the recognition accuracy.

V. Conclusions

As a conclusion, fast CIKPCA is proposed. To reduce the computational time of the proposed method, the data given in a chunk are first divided into small chunks and data are selected from the smaller chunks based on the accumulation ratio. The reason of data selection is to retain only some useful data for the eigen-feature space learning. In addition, the computation time are reduced since the computation time of kernel matrix decomposition are reduced.

Then, compact feature space is constructed by the linear combination of the linearly independent data that are selected from the reduced data and the eigenvalue problem is solved only once for the selected data.

In this paper, CIKPCA is compared with KPCA and Takeuchi et al.'s IKPCA. Experimental results demonstrated that CIKPCA requires shorter learning time than IKPCA and KPCA with similar recognition accuracy for all the models. To further evaluate the effectiveness of CIKPCA, the computational time of CIKPCA is compared with IKPCA with regards to the different number of chunk sizes. The results of all the data sets revealed that CIKPCA requires

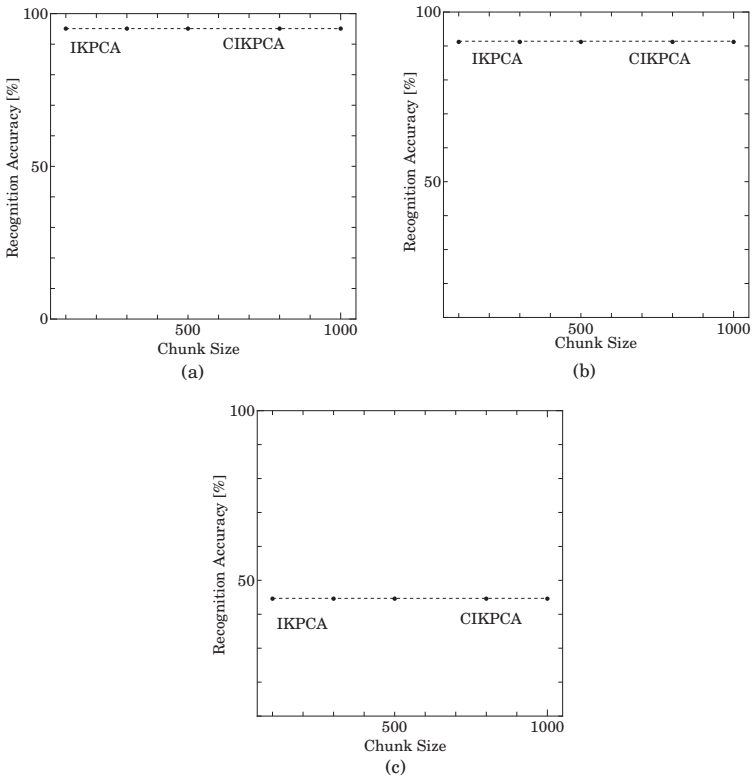

Fig. 2. Recognition Accuracy (%.) vs. different number of chunk sizes for (a) Ozone, (b) Thyroid and (c) Poker Hand data.

shorter time compared to IKPCA for all the chunk sizes. The learning time are reduced when the chunk sizes becomes large. In addition, the total number of eigenvalue decomposition are reduced when the chunk size becomes large. In general, the proposed method could accelerate the learning time without reducing the recognition accuracy. Therefore, we conclude that division of data into smaller chunks and selection of training data is important in order to accelerate the learning time. In the future, we will compare the proposed method with Chunk Incremental Principal Component Analysis (CIPCA).

REFERENCES

[1] I. Jolliffe, *Principal component analysis,* Springer, New York, 1986.

[2] F. Han, and H. Liu, "High Dimensional Semiparametric Scale-invariant Principal Component Analysis," *Advances in Neural Information Processing Systems (NIPS)*, pp. 171-179, 2012.

[3] P. Honeine, "Online Kernel Principal Component Analysis: A reduced-Order Model," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 34, no. 9, pp. 1814-1826, 2012.

[4] H. Zhao, P. Chi, and J. T. Kwok, "A Novel Incremental Principal Component Analysis and Its Application for Face Recognition," *IEEE Trans. on Systems, Man and Cybernetics, Part B*, vol. 36, no. 4, pp. 873-886, 2006.

[5] T.-J. Chin, and D. Suter, "Incremental Kernel Principal Component Analysis," *IEEE Trans. on Image Processing*, vol. 16, no. 6, pp. 1662-1674, 2007.

[6] S. Ozawa, S. Pang, and N. Kasabov, "Incremental Learning of Chunk Data for On-line Pattern Classification Systems," *IEEE Trans. on Neural Networks*, vol. 19, no. 6, pp. 1061-1074, 2008.

[7] S. Ozawa, S. L. Toh, S. Abe, S. Pang, and N. Kasabov, "Incremental Learning of Feature Space and Classifier for Face Recognition," *Neural Networks*, vol. 18, nos. 5-6, pp. 575-578, 2005.

[8] B. Schölkopf, A. Smola, K. Muller, "Kernel Principal Component Analysis," In:*W. Gerstner, M. Hasler, A. Germond, J.-D. Nicoud (eds.) ICANN 1997. LNCS, Springer-Heideberg*, no. 1327, pp. 583-588, 1997.

[9] Y. Takeuchi, S. Ozawa, S. Abe, "An Efficient Incremental Kernel Principal Component Analysis for Online Feature Extraction," *Proc. of Int. Joint Conf. Neural Networks 2007*, pp. 2346-2351, 2007.

[10] T. Tokumoto and S. Ozawa, "Property of Learning Chunk Data Using Incremental Kernel Principal Component Analysis," *Proc. IEEE Workshop on Evolving and Adaptive Intelligent Systems (EAIS2012, Madrid)*, pp. 7-10, 2012.

[11] S. Asunction, D. J. Newman, "UCI Machine Learning Repository," *UC. Irvine, School of Info. and Comp. Sci.*, 2007.

[12] S. Abe, *Support Vector Machines for Pattern Classification,* Second Edition, Springer-Verlag, London, 2010.