Incremental Learning of Chunk Data for Online Pattern Classification Systems

Ozawa, Seiichi
Pang, Shaoning
Kasabov, Nikola

(Citation)
IEEE Transactions on Neural Neworks,19(6):1061-1074

(Issue Date)
2008-06

(Resource Type)
journal article

(Version)
Version of Record

(URL)
https://hdl.handle.net/20.500.14094/90001005

Incremental Learning of Chunk Data for Online Pattern Classification Systems

Seiichi Ozawa, *Member, IEEE*, Shaoning Pang, *Senior Member, IEEE*, and Nikola Kasabov, *Senior Member, IEEE*

***Abstract*—This paper presents a pattern classification system in which feature extraction and classifier learning are simultaneously carried out not only online but also in one pass where training samples are presented only once. For this purpose, we have extended incremental principal component analysis (IPCA) and some classifier models were effectively combined with it. However, there was a drawback in this approach that training samples must be learned one by one due to the limitation of IPCA. To overcome this problem, we propose another extension of IPCA called *chunk IPCA* in which a chunk of training samples is processed at a time. In the experiments, we evaluate the classification performance for several large-scale data sets to discuss the scalability of chunk IPCA under one-pass incremental learning environments. The experimental results suggest that chunk IPCA can reduce the training time effectively as compared with IPCA unless the number of input attributes is too large. We study the influence of the size of initial training data and the size of given chunk data on classification accuracy and learning time. We also show that chunk IPCA can obtain major eigenvectors with fairly good approximation.**

***Index Terms*—Feature extraction, incremental learning, online learning, pattern classification, principal component analysis (PCA).**

I. Introduction

In MANY real-world applications such as pattern recognition, data mining, and time-series prediction, we often confront difficult situations where a complete set of training samples is not given when constructing a system. In face recognition, for example, since human faces have large variations due to expressions, lighting conditions, makeup, hairstyles, and so forth, it is hard to consider all variations of face in advance [6], [31], [35]. In many cases, training samples are provided only when a system misclassifies objects; hence the system is learned online to improve the classification performance. On the other hand, in many practical applications of data mining and time-series predictions, the data are generally provided little by little and the property of data source could be changed as time passes. Therefore, the learning of a system must also be conducted sequentially in an online fashion. This type of learning is called *incremental learning* or *continuous learning*, and it has recently received a great attention in many practical applications [26], [30].

In pattern recognition and data mining, input data often have a large set of attributes. Hence, the informative input variables are first extracted before the classification is carried out. This means that when constructing an adaptive classification system, we should consider two types of incremental learning: one is the incremental feature extraction, and the other is incremental learning of classifiers. For this purpose, several incremental algorithms have been independently developed for the feature extraction and the classifier learning. As for the feature extraction, principal component analysis (PCA) and linear discriminant analysis (LDA) have been extended to an incremental version [9], [20], [25], [27], [36]. Hall and Martin have devised a smart method to update eigenvectors and eigenvalues incrementally (i.e., the update of an eigenspace) called incremental principal component analysis (IPCA) [9]. Recently, Ozawa *et al.* have extended this IPCA algorithm such that an eigenaxis is augmented based on the accumulation ratio in order to control the dimensionality of an eigenspace easily [21]. On the other hand, Pang *et al.* [24] have proposed an incremental LDA (ILDA) algorithm in which a between-class scatter matrix and a within-class scatter matrix are incrementally updated, and then the eigenaxes of a feature space are obtained by solving an eigenproblem. Yan *et al.* [39] and Zhao *et al.* [41] have also proposed different ILDA algorithms. As for classifier learning, various incremental learning algorithms have been proposed for neural networks [4], [5], [8], [12], [13], [34], support vector machine [7], [11], [29], decision tree [14], [33], [37], and so forth. Memory-based learning approach is another promising solution [1] where all (or a part) of training samples are accumulated in memory and they are utilized for learning at every learning step. The problem of this approach is that large memory capacity is often needed to store past training samples. To alleviate this problem, several attempts have been made in which representative samples are being selected and kept so that some of them be added to the current training sample for learning [18], [38]. Evolving clustering method (ECM) [15] also belongs to this learning approach where prototype vectors are dynamically created and updated.

Recently, we have proposed a new scheme of incremental learning in which feature extraction and classifier learning are simultaneously carried out online [21], [22], [25]. In our previous work, IPCA or ILDA were adopted as feature extraction methods, and ECM or resource allocating network with long-term memory (RAN-LTM) [18], [22] were adopted as classifiers. A distinctive feature of the proposed scheme is that the learning is conducted incrementally *one pass*; here, one pass

Manuscript received January 23, 2006; revised January 11, 2007, February 26, 2007, and September 14, 2007; accepted October 12, 2007.

S. Ozawa is with the Graduate School of Engineering, Kobe University, Nada-ku, Kobe 657-8501, Japan (e-mail: ozawasei@kobe-u.ac.jp).

S. Pang and N. Kasabov are with the Knowledge Engineering and Discovery Research Institute (KEDRI), Auckland University of Technology, 1061 Auckland, New Zealand (e-mail: shaoning.pang@aut.ac.nz; nkasabov@aut.ac.nz).

Digital Object Identifier 10.1109/TNN.2007.2000059

means that training samples are passed through a system only once for learning purposes [15], [19]. One-pass incremental learning is quite important and effective especially under the environments where a system has to learn from a large-scale data set with limited computational resources [i.e., central processing unit (CPU) performance and memory]. It was verified that the classification accuracy of the aforementioned classification system was improved constantly even if a small set of training samples were provided at a starting point [22], [24]. However, some problems still remain for this approach. The biggest problem is scalability. In our previous approach, a training sample must be learned one by one even if a chunk of training sample is available at a time. This causes inefficiency in computations because the eigenvalue decomposition in IPCA must be applied to each training sample in the chunk.

To solve this problem, we have proposed an extension of IPCA [23] called *chunk IPCA* in which the update of an eigenspace is completed by performing single eigenvalue decomposition. For the same purpose, Hall *et al.* have already proposed a method of merging eigenspace models incrementally [10]. However, they have not clarified how to determine new eigenaxes and their criterion on the axis augmentation is still based on the norm of a residue vector whose proper threshold could largely depend on the data sets used. In this paper, we present a complete version of chunk IPCA [23] in which an effective way to determine a set of new eigenaxes is proposed and an update equation of the accumulation ratio to determine the eigenaxis augmentation is derived. In addition, we combine a simple classifier with chunk IPCA and evaluate the classification performance for several large-scale data sets to discuss the scalability of chunk IPCA under one-pass incremental learning environments.

This paper is organized as follows. Section II gives a quick review on the original IPCA. In Section III, we propose an extension of IPCA called chunk IPCA in which a chunk of training samples is learned at a time and new eigenaxes are efficiently selected based on the accumulation ratio. Then, we present a one-pass incremental learning scheme in which chunk IPCA algorithm is used for eigenspace learning and a naive approach to prototype update is adopted as the learning of a simple nearest neighbor classifier (NNC). In Section IV, the scalability of the proposed incremental learning scheme is studied with seven large-scale data sets. Section V summarizes this paper and discusses directions for further work.

II. Incremental Principal Component Analysis

PCA is one of the most popular and powerful feature extraction techniques. Since the original PCA is not suited for incremental learning purposes, Hall and Martin have proposed IPCA algorithm which can update eigenvectors and eigenvalues incrementally [9]. Let us review IPCA briefly.

Assume that N training samples $\boldsymbol{x}^{(i)} \in \mathcal{R}^n$ $(i = 1, \cdots, N)$ have been presented so far, and an eigenspace model $\Omega = (\bar{\boldsymbol{x}}, \boldsymbol{U}_k, \boldsymbol{\Lambda}_k, N)$ is constructed by calculating the eigenvectors and eigenvalues from the covariance matrix of $\boldsymbol{x}^{(i)}$, where $\bar{\boldsymbol{x}}$ is a mean vector of $\boldsymbol{x}^{(i)}$ $(i = 1, \cdots, N)$, $\boldsymbol{U}_k$ is an $n \times k$ matrix whose column vectors correspond to eigenvectors, and $\boldsymbol{\Lambda}_k$ is a $k \times k$ matrix whose diagonal elements correspond to nonzero eigenvalues. Here, k is the number of eigenaxes spanning the eigenspace (i.e., eigenspace dimensionality).

Now, assume that the $(N+1)$th training sample $\boldsymbol{y} \in \mathcal{R}^n$ is given. The addition of this new sample leads to the changes in both mean vector and covariance matrix; therefore, the eigenvectors and eigenvalues should also be updated. The new mean input vector $\bar{\boldsymbol{x}}'$ is easily obtained as follows:

$$\bar{\boldsymbol{x}}' = \frac{1}{N+1}(N\bar{\boldsymbol{x}} + \boldsymbol{y}) \in \mathcal{R}^n. \tag{1}$$

The problem is how to update the eigenvectors and eigenvalues.

When updating the eigenspace model Ω, we need to check if the eigenspace should be enlarged in term of dimensionality. If the new sample includes almost all energy in the current eigenspace, the dimensionality does not need to be changed. However, if the eigenspace includes certain energy in the complementary eigenspace, the dimensional augmentation is needed, or crucial information on the new sample might be lost. In the original IPCA, the judgment of the eigenspace augmentation is made based on the norm of the following residue vector $\boldsymbol{h} \in \mathcal{R}^n$:

$$\boldsymbol{h} = (\boldsymbol{y} - \bar{\boldsymbol{x}}) - \boldsymbol{U}_k \boldsymbol{g} \tag{2}$$

where

$$\boldsymbol{g} = \boldsymbol{U}_k^T (\boldsymbol{y} - \bar{\boldsymbol{x}}). \tag{3}$$

Here, T means the transposition of vectors and matrices. When the norm of the residue vector $\boldsymbol{h}$ is larger than a threshold value η,[1] the dimensionality of the current eigenspace is increased from k to $k+1$, and a new eigenaxis is added in the direction of $\boldsymbol{h}$. Otherwise, the dimensionality of the eigenspace remains the same. These operations are represented by the following:

$$\hat{\boldsymbol{h}} = \begin{cases} \boldsymbol{h}/\|\boldsymbol{h}\|, & \text{if } \|\boldsymbol{h}\| > \eta \\ \boldsymbol{0}, & \text{otherwise.} \end{cases} \tag{4}$$

Note that too large threshold η would cause serious approximation error for eigenspace models.

It has been shown that the eigenvectors and eigenvalues are updated by solving the following intermediate eigenproblem [9]:

1) if there is a new eigenaxis to be added

$$\left\{ \frac{N}{N+1} \begin{bmatrix} \boldsymbol{\Lambda}_k & \boldsymbol{0} \\ \boldsymbol{0}^T & 0 \end{bmatrix} + \frac{N}{(N+1)^2} \begin{bmatrix} \boldsymbol{g}\boldsymbol{g}^T & \gamma \boldsymbol{g} \\ \gamma \boldsymbol{g}^T & \gamma^2 \end{bmatrix} \right\} \boldsymbol{R} = \boldsymbol{R} \boldsymbol{\Lambda}'_{k+1} \tag{5}$$

2) otherwise

$$\left\{ \frac{N}{N+1} \boldsymbol{\Lambda}_k + \frac{N}{(N+1)^2} \boldsymbol{g}\boldsymbol{g}^T \right\} \boldsymbol{R} = \boldsymbol{R} \boldsymbol{\Lambda}'_k \tag{6}$$

where $\gamma = \hat{\boldsymbol{h}}^T (\boldsymbol{y} - \bar{\boldsymbol{x}})$, $\boldsymbol{R}$ is a matrix whose column vectors correspond to the eigenvectors obtained from the previous intermediate eigenproblem, and $\boldsymbol{0}$ is a k-dimensional zero vector. Here,

[1] In the original IPCA [9], η is set to zero.

$\mathbf{\Lambda}'_{k+1}$ and $\mathbf{\Lambda}'_k$ are the new eigenvalue matrices whose diagonal elements correspond to k and $k+1$ eigenvalues, respectively. Using the solution $\boldsymbol{R}$, we can calculate the new $n \times (k+1)$ eigenvector matrix $\boldsymbol{U}'_{k+1}$ as follows:

1) if there is a new eigenaxis to be added

$$\boldsymbol{U}'_{k+1} = [\boldsymbol{U}_k, \hat{\boldsymbol{h}}]\boldsymbol{R} \tag{7}$$

2) otherwise

$$\boldsymbol{U}'_k = \boldsymbol{U}_k\boldsymbol{R}. \tag{8}$$

From (7) and (8), intuitively we can consider that $\boldsymbol{R}$ gives a rotation from old eigenaxes to new ones; hence, let us call $\boldsymbol{R}$ *rotation matrix* here.

III. Proposed Scheme for Incremental Learning from Chunks of Data

A. Extended IPCA Algorithm for Chunk Data

As stated in Section I, the original IPCA is applied to a single training sample at a time, and the intermediate eigenproblem must be solved repeatedly for every training sample. Therefore, the learning may get stuck in a deadlock if a large chunk of training samples is given to learn in a short term. That is to say, the next chunk of training samples could come before the learning is finished if it takes a long time to update the eigenspace.

To overcome this problem, we have extended the original IPCA so that the eigenspace model Ω can be updated with a chunk of training samples in a single operation [23]. We call this extended algorithm *chunk IPCA* and let us describe the algorithm in the following sections.

1) Update of Eigenspace: Let us assume that N training samples $\boldsymbol{X} = \{\boldsymbol{x}^{(1)}, \cdots, \boldsymbol{x}^{(N)}\} \in \mathcal{R}^{n \times N}$ have been given so far and they were already discarded. Instead of keeping actual training samples, we hold an eigenspace model $\Omega = (\bar{\boldsymbol{x}}, \boldsymbol{U}_k, \mathbf{\Lambda}_k, N)$, where $\bar{\boldsymbol{x}}$, $\boldsymbol{U}_k$, and $\mathbf{\Lambda}_k$ are a mean input vector, an $n \times k$ eigenvector matrix, and a $k \times k$ eigenvalue matrix, respectively. Now, assume that a chunk of L training samples $\boldsymbol{Y} = \{\boldsymbol{y}^{(1)}, \cdots, \boldsymbol{y}^{(L)}\} \in \mathcal{R}^{n \times L}$ is presented.[2] Without the past training samples $\boldsymbol{X}$, we can update the mean vector $\bar{\boldsymbol{x}}'$ as follows:

$$\bar{\boldsymbol{x}}' = \frac{1}{N+L}(N\bar{\boldsymbol{x}} + L\bar{\boldsymbol{y}}). \tag{9}$$

Using (9), the new covariance matrix $\boldsymbol{C}'$ is given by

$$\begin{aligned}\boldsymbol{C}' &= \frac{1}{N+L}\left[\sum_{i=1}^{N}\left(\boldsymbol{x}^{(i)} - \bar{\boldsymbol{x}}'\right)\left(\boldsymbol{x}^{(i)} - \bar{\boldsymbol{x}}'\right)^T + \sum_{i=1}^{L}\left(\boldsymbol{y}^{(i)} - \bar{\boldsymbol{x}}'\right)\left(\boldsymbol{y}^{(i)} - \bar{\boldsymbol{x}}'\right)^T\right] \\ &= \frac{1}{N+L}\left[N\boldsymbol{C} + \frac{NL^2}{(N+L)^2}(\bar{\boldsymbol{y}} - \bar{\boldsymbol{x}})(\bar{\boldsymbol{y}} - \bar{\boldsymbol{x}})^T + \frac{N^2}{(N+L)^2}\sum_{i=1}^{L}\left(\boldsymbol{y}^{(i)} - \bar{\boldsymbol{x}}\right)\left(\boldsymbol{y}^{(i)} - \bar{\boldsymbol{x}}\right)^T + \frac{L(L+2N)}{(N+L)^2}\sum_{i=1}^{L}\left(\boldsymbol{y}^{(i)} - \bar{\boldsymbol{y}}\right)\left(\boldsymbol{y}^{(i)} - \bar{\boldsymbol{y}}\right)^T\right].\end{aligned} \tag{10}$$

Suppose that l eigenaxes are augmented to avoid the serious loss of essential information when a chunk of L training samples $\boldsymbol{Y}$ is provided; that is, the eigenspace dimensions are increased by l. Let us denote the augmented eigenaxes as follows:

$$\boldsymbol{H}_l = [\boldsymbol{h}_1, \cdots, \boldsymbol{h}_l] \in \mathcal{R}^{n \times l}, \qquad 0 \le l \le L. \tag{11}$$

Then, the updated eigenvector matrix $\boldsymbol{U}'_{k+l}$ is represented by using the rotation matrix $\boldsymbol{R}$ and the current eigenvector matrix $\boldsymbol{U}_k$

$$\boldsymbol{U}'_{k+l} = [\boldsymbol{U}_k, \boldsymbol{H}_l]\boldsymbol{R}. \tag{12}$$

Hence, a new eigenproblem to be solved is given by

$$\boldsymbol{C}'\boldsymbol{U}'_{k+l} = \boldsymbol{U}'_{k+l}\mathbf{\Lambda}'_{k+l} \Rightarrow \boldsymbol{C}'[\boldsymbol{U}_k, \boldsymbol{H}_l]\boldsymbol{R} = [\boldsymbol{U}_k, \boldsymbol{H}_l]\boldsymbol{R}\mathbf{\Lambda}'_{k+l}, \tag{13}$$

where $\mathbf{\Lambda}'_{k+l}$ is a new eigenvalue matrix. Substituting (9)–(11) into (13), the following intermediate eigenproblem for Chunk IPCA is obtained.

1) If there is a new eigenaxis to be added (i.e., $l \neq 0$)

$$\left\{\frac{N}{N+L}\begin{bmatrix}\mathbf{\Lambda}_k & \boldsymbol{0}\\ \boldsymbol{0}^T & 0\end{bmatrix} + \frac{NL^2}{(N+L)^3}\begin{bmatrix}\bar{\boldsymbol{g}}\bar{\boldsymbol{g}}^T & \bar{\boldsymbol{g}}\bar{\boldsymbol{\gamma}}^T\\ \bar{\boldsymbol{\gamma}}\bar{\boldsymbol{g}}^T & \bar{\boldsymbol{\gamma}}\bar{\boldsymbol{\gamma}}^T\end{bmatrix} + \frac{N^2}{(N+L)^3}\sum_{i=1}^{L}\begin{bmatrix}\boldsymbol{g}'_i\boldsymbol{g}'^T_i & \boldsymbol{g}'_i\boldsymbol{\gamma}'^T_i\\ \boldsymbol{\gamma}'_i\boldsymbol{g}'^T_i & \boldsymbol{\gamma}'_i\boldsymbol{\gamma}'^T_i\end{bmatrix} + \frac{L(L+2N)}{(N+L)^3}\sum_{i=1}^{L}\begin{bmatrix}\boldsymbol{g}''_i\boldsymbol{g}''^T_i & \boldsymbol{g}''_i\boldsymbol{\gamma}''^T_i\\ \boldsymbol{\gamma}''_i\boldsymbol{g}''^T_i & \boldsymbol{\gamma}''_i\boldsymbol{\gamma}''^T_i\end{bmatrix}\right\}\boldsymbol{R} = \boldsymbol{R}\mathbf{\Lambda}'_{k+l} \tag{14}$$

2) otherwise

$$\left\{\frac{N}{N+L}\mathbf{\Lambda}_k + \frac{NL^2}{(N+L)^3}\bar{\boldsymbol{g}}\bar{\boldsymbol{g}}^T + \frac{N^2}{(N+L)^3}\sum_{i=1}^{L}\boldsymbol{g}'_i\boldsymbol{g}'^T_i + \frac{L(L+2N)}{(N+L)^3}\sum_{i=1}^{L}\boldsymbol{g}''_i\boldsymbol{g}''^T_i\right\}\bar{\boldsymbol{R}} = \bar{\boldsymbol{R}}\mathbf{\Lambda}'_k \tag{15}$$

where

$$\begin{aligned}&\bar{\boldsymbol{g}} = \boldsymbol{U}_k^T(\bar{\boldsymbol{y}} - \bar{\boldsymbol{x}}) \quad \boldsymbol{g}'_i = \boldsymbol{U}_k^T\left(\boldsymbol{y}^{(i)} - \bar{\boldsymbol{x}}\right) \quad \boldsymbol{g}''_i = \boldsymbol{U}_k^T\left(\boldsymbol{y}^{(i)} - \bar{\boldsymbol{y}}\right)\\ &\bar{\boldsymbol{\gamma}} = \boldsymbol{H}_l^T(\bar{\boldsymbol{y}} - \bar{\boldsymbol{x}}) \quad \boldsymbol{\gamma}'_i = \boldsymbol{H}_l^T\left(\boldsymbol{y}^{(i)} - \bar{\boldsymbol{x}}\right) \quad \boldsymbol{\gamma}''_i = \boldsymbol{H}_l^T\left(\boldsymbol{y}^{(i)} - \bar{\boldsymbol{y}}\right).\end{aligned}$$

Solving this intermediate eigenproblem, a new rotation matrix $\bar{\boldsymbol{R}}$ and the eigenvalue matrix $\mathbf{\Lambda}'_{k+l}$ are obtained. Then, the corresponding new eigenvector matrix $\boldsymbol{U}'_{k+l}$ is given by (12).

2) Criterion for Eigenaxis Augmentation: As seen from (4), a new eigenaxis is augmented when the norm of a residue vector

[2]The size of a chunk may often be different at every learning stage and we usually cannot specify it. Although there is no limitation on the chunk size that can be applicable to the proposed chunk IPCA, we can assume the chunk size to be fixed for the simplicity of explanation.

is larger than a threshold value η in the original IPCA. However, this is not a good criterion in practice because a suitable threshold can be varied depending on the magnitude of input values. If the threshold is too small, the dimensionality of a feature space could be excessively large and an efficient feature space with small dimensions is hard to be constructed; this may deteriorate both generalization performance and computational efficiency. On the other hand, if the threshold is too large, essential information on training samples would be lost unexpectedly.

To reduce the dependency of the threshold on input values, the following accumulation ratio is often used as a criterion:

$$A(\boldsymbol{U}_k) = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{j=1}^{n} \lambda_j} \tag{16}$$

where $\boldsymbol{U}_k = \{\boldsymbol{u}_1, \cdots, \boldsymbol{u}_k\}$ is the eigenvector matrix whose column vectors span the k-dimensional feature space, λ_i $(i = 1, \cdots, k)$ is the eigenvalue of $\boldsymbol{u}_i$, and n is the dimensionality of the input space, respectively. In the proposed chunk IPCA, the number of eigenaxes to be augmented is basically determined by finding the minimum k such that $A(\boldsymbol{U}_k) > \theta$ holds where θ is a positive value between 0 and 1. To introduce this criterion in the incremental learning scheme, we need an update equation of (16) which can be calculated without the past training samples. In the learning algorithm, actually we need two types of accumulation ratios. One is the accumulation ratio for the k-dimensional eigenspace spanned by $\bar{\boldsymbol{U}}'_k = \bar{\boldsymbol{U}}_k \boldsymbol{R}$ and the other is that for the $(k+l)$-dimensional augmented eigenspace spanned by $\boldsymbol{U}'_{k+l} = [\boldsymbol{U}_k, \boldsymbol{H}_l]\boldsymbol{R}$. The former accumulation ratio is used for checking if the current k-dimensional eigenspace should be augmented or not. The latter one is used for checking if further eigenaxes are needed for the $(k+l)$-dimensional augmented eigenspace.

First, let us calculate the updated accumulation ratio $A'(\boldsymbol{U}'_k)$. Considering the fact that the total energy of the training samples in the new k-dimensional eigenspace is not changed only by the rotation of eigenaxes, the calculation of $A'(\boldsymbol{U}'_k)$ is replaced with that of $A'(\boldsymbol{U}_k)$. In addition, using the fact that the total amount of eigenvalues is equivalent to the summation of variances σ_i^2, the numerator of $A'(\boldsymbol{U}'_k)$ in (16) is reduced to

$$\sum_{i=1}^{k} \lambda'_i = \sum_{i=1}^{k} \frac{1}{N+L} \left[\sum_{j=1}^{N} \left\{ \boldsymbol{u}_i^T \left(\boldsymbol{x}^{(j)} - \bar{\boldsymbol{x}}' \right) \right\}^2 + \sum_{j=1}^{L} \left\{ \boldsymbol{u}_i^T \left(\boldsymbol{y}^{(j)} - \bar{\boldsymbol{x}}' \right) \right\}^2 \right]. \tag{17}$$

In (17), the new mean $\boldsymbol{u}_i^T \bar{\boldsymbol{x}}'$ in the feature space is obtained from (9) as follows:

$$\boldsymbol{u}_i^T \bar{\boldsymbol{x}}' = \frac{1}{N+L} \boldsymbol{u}_i^T (N\bar{\boldsymbol{x}} + L\bar{\boldsymbol{y}}). \tag{18}$$

Substituting (18) into (17), the numerator of (16) is given by

$$\sum_{i=1}^{k} \lambda'_i = \frac{N}{N+L} \left[\sum_{i=1}^{k} \lambda_i + \frac{L}{N+L} \left\| \boldsymbol{U}_k^T (\bar{\boldsymbol{x}} - \bar{\boldsymbol{y}}) \right\|^2 + \frac{1}{N} \sum_{j=1}^{L} \left\| \boldsymbol{U}_k^T \left(\boldsymbol{y}^{(j)} - \bar{\boldsymbol{y}} \right) \right\|^2 \right] \tag{19}$$

where $\boldsymbol{U}_k = \{\boldsymbol{u}_1, \cdots, \boldsymbol{u}_k\}$. In the similar manner, the denominator of (16) is given by

$$\sum_{i=1}^{n} \lambda'_i = \frac{N}{N+L} \left[\sum_{i=1}^{n} \lambda_i + \frac{L}{N+L} \|(\bar{\boldsymbol{x}} - \bar{\boldsymbol{y}})\|^2 + \frac{1}{N} \sum_{j=1}^{L} \left\| \left(\boldsymbol{y}^{(j)} - \bar{\boldsymbol{y}} \right) \right\|^2 \right]. \tag{20}$$

Then, the update equation for the accumulation ratio $A'(\boldsymbol{U}'_k)$ is given by

$$A'\left(\boldsymbol{U}'_k\right) = \frac{\sum_{i=1}^{k} \lambda_i + \frac{L}{N+L} \|\bar{\boldsymbol{g}}\|^2 + \frac{1}{N} \sum_{j=1}^{L} \left\|\boldsymbol{g}''_j\right\|^2}{\sum_{i=1}^{n} \lambda_i + \frac{L}{N+L} \|\bar{\boldsymbol{\mu}}\|^2 + \frac{1}{N} \sum_{j=1}^{L} \left\|\boldsymbol{\mu}''_j\right\|^2} \tag{21}$$

where $\bar{\boldsymbol{\mu}} = \bar{\boldsymbol{x}} - \bar{\boldsymbol{y}}$ and $\boldsymbol{\mu}''_j = \boldsymbol{y}^{(j)} - \bar{\boldsymbol{y}}$. To update $A'(\boldsymbol{U}'_k)$, the summation of eigenvalues λ_i $(i = 1, \cdots, n)$ is required to be retained. Hence, the individual eigenvalues λ_i $(i = k+1, \cdots, n)$ in the denominator of $A'(\boldsymbol{U}'_k)$ are not necessary for this update.

Next, let us calculate $A'(\boldsymbol{U}'_{k+l})$, where $\boldsymbol{U}'_{k+l} = [\boldsymbol{U}_k, \boldsymbol{H}_l]\boldsymbol{R}$. Since the updated eigenspace spanned by $\boldsymbol{U}'_{k+l}$ has the same dimensions as that spanned by $\boldsymbol{U}_{k+l}$, the following equation holds: $A'(\boldsymbol{U}'_{k+l}) = A'(\boldsymbol{U}_{k+l})$. Then, the numerator of $A'(\boldsymbol{U}'_{k+l})$ is calculated as follows:

$$\begin{aligned} \sum_{i=1}^{k+l} \lambda'_i &= \sum_{i=1}^{k} \lambda'_i + \sum_{i=1}^{l} \lambda'_{k+i} \\ &= \sum_{i=1}^{k} \lambda'_i + \frac{N}{N+L} \left[\frac{1}{N} \sum_{j=1}^{N} \left\| \boldsymbol{H}_l^T \left(\boldsymbol{x}^{(j)} - \bar{\boldsymbol{x}} \right) \right\|^2 + \frac{L}{N+L} \left\| \boldsymbol{H}_l^T (\bar{\boldsymbol{x}} - \bar{\boldsymbol{y}}) \right\|^2 + \frac{1}{N} \sum_{j=1}^{L} \left\| \boldsymbol{H}_l^T \left(\boldsymbol{y}^{(j)} - \bar{\boldsymbol{y}} \right) \right\|^2 \right]. \end{aligned} \tag{22}$$

As seen from (22), the first term of the brackets [ ] includes the past examples $\boldsymbol{x}^{(j)}$ $(j = 1, \cdots, N)$. Therefore, the exact calculation of the numerator term is impossible under the incremental learning environment where training samples are supposed to be discarded after learning. However, since the subspace spanned by $\boldsymbol{H}_l$ does not have large energy for the past examples, this term could be ignored as long as the threshold θ is not too small.[3] Then, the accumulation ratio $A'(\boldsymbol{U}'_{k+l})$ is approximated to the following:

$$A'\left(\boldsymbol{U}'_{k+l}\right) \simeq \frac{\sum_{i=1}^{k} \lambda_i + \frac{L}{N+L} \left\| \begin{matrix} \bar{\boldsymbol{g}} \\ \bar{\gamma} \end{matrix} \right\|^2 + \frac{1}{N} \sum_{j=1}^{L} \left\| \begin{matrix} \boldsymbol{g}''_j \\ \gamma''_j \end{matrix} \right\|^2}{\sum_{i=1}^{n} \lambda_i + \frac{L}{N+L} \|\bar{\boldsymbol{\mu}}\|^2 + \frac{1}{N} \sum_{j=1}^{L} \left\|\boldsymbol{\mu}''_j\right\|^2}. \tag{23}$$

[3]Even though individual fractions of lost energy are negligible, their accumulation would affect the approximation error of an eigenspace model (see, also, Section IV-E, for further discussion).

As seen from (21) and (23), we need no past sample $\boldsymbol{x}^{(j)}$ and no rotation matrix $\boldsymbol{R}$ to update the accumulation ratio. Therefore, this accumulation ratio is updated with the following information: a chunk of given training samples $\boldsymbol{Y} = \{\boldsymbol{y}^{(1)}, \cdots, \boldsymbol{y}^{(L)}\}$, the eigenspace model $\Omega = (\bar{\boldsymbol{x}}, \boldsymbol{U}_k, \boldsymbol{\Lambda}_k, N)$, and a set of augmented eigenaxes $\boldsymbol{H}_l$ which is obtained by the procedure described in the next section.

3) Selection of Eigenaxes: In IPCA, a new eigenaxis is selected so as to be perpendicular to the existing eigenvectors which are given by the column vectors of $\boldsymbol{U}_k$. A straightforward way to get new eigenaxes is to apply Gram–Schmidt orthogonalization technique to the given chunk of training samples [10]. If the training samples are represented by $\tilde{L}$ linearly independent vectors, the maximum number of eigenaxes to be augmented is $\tilde{L}$. However, the feature space spanned by all of the $\tilde{L}$ eigenaxes is redundant in general; in addition, if the chunk size is large, the computation costs to solve the intermediate eigenproblem in (14) would be considerably expensive. Therefore, we need to find a smallest set of eigenaxes without losing essential information of the given training samples.

The problem of finding an optimal set of eigenaxes is stated as follows.

> Find the smallest set of eigenaxes $\boldsymbol{H}^* = \{\boldsymbol{h}_1, \cdots, \boldsymbol{h}_{l^*}\}$ for the current eigenspace model $\Omega = (\bar{\boldsymbol{x}}, \boldsymbol{U}_k, \boldsymbol{\Lambda}_k, N)$ without keeping the past training samples $\boldsymbol{X}$ such that the accumulation ratio $A'(\boldsymbol{U}'_{k+l^*})$ of the given training samples $[\boldsymbol{X}, \boldsymbol{Y}]$ is larger than a threshold θ.

Assume that we have a candidate set of augmented eigenaxes $\boldsymbol{H}_l = \{\boldsymbol{h}_1, \cdots, \boldsymbol{h}_l\}$. Because the denominator of (23) is constant once the mean vector $\bar{\boldsymbol{y}}$ is calculated at each learning stage, the increment of the accumulation ratio from $A'(\boldsymbol{U}'_k)$ to $A'(\boldsymbol{U}'_{k+l})$ is determined by the numerator terms. Thus, let us define the following difference $\Delta\tilde{A}'(\boldsymbol{U}'_{k+l})$ of the numerator terms between $A'(\boldsymbol{U}'_k)$ and $A'(\boldsymbol{U}'_{k+l})$:

$$\begin{aligned}\Delta\tilde{A}'\left(\boldsymbol{U}'_{k+l}\right) &= \frac{L}{N+L}\left\|\boldsymbol{H}_l^T(\bar{\boldsymbol{x}}-\bar{\boldsymbol{y}})\right\|^2 + \frac{1}{N}\sum_{j=1}^{L}\left\|\boldsymbol{H}_l^T\left(\boldsymbol{y}^{(j)}-\bar{\boldsymbol{y}}\right)\right\|^2 \\ &\stackrel{\text{def}}{=} \sum_{i=1}^{l}\Delta\tilde{A}'_i \end{aligned} \tag{24}$$

where

$$\Delta\tilde{A}'_i = \frac{L}{N+L}\left\{\boldsymbol{h}_i^T(\bar{\boldsymbol{x}}-\bar{\boldsymbol{y}})\right\}^2 + \frac{1}{N}\sum_{j=1}^{L}\left\{\boldsymbol{h}_i^T\left(\boldsymbol{y}^{(j)}-\bar{\boldsymbol{y}}\right)\right\}^2. \tag{25}$$

Equation (24) means that the increments of the accumulation ratio are determined by the linear sum of $\Delta\tilde{A}'_i$. Therefore, to find the smallest set of eigenaxes, first we find $\boldsymbol{h}_i$ with the largest $\Delta\tilde{A}'_i$, and put it into the set of augmented eigenaxes $\boldsymbol{H}_l$ (i.e., $l = 1$ and $\boldsymbol{H}_1 = \boldsymbol{h}_i$). Then, check if the accumulation ratio of $A'(\boldsymbol{U}'_{k+1})$ in (23) becomes larger than the threshold θ. If not, select $\boldsymbol{h}_i$ with the second largest $\Delta\tilde{A}'_i$, and the same procedure is repeated until $A'(\boldsymbol{U}'_{k+l}) > \theta$ is satisfied. This type of greedy algorithm makes the selection of an optimal set of eigenaxes very simple. The algorithm of the eigenaxis selection is summarized in Algorithm 1.

Algorithm 1: Selection of Eigenaxes

Input:

- Eigenspace model $\Omega = (\bar{\boldsymbol{x}}, \boldsymbol{U}_k, \boldsymbol{\Lambda}_k, N)$.
- A chunk of L training samples $\boldsymbol{Y} = \{\boldsymbol{y}^{(1)}, \cdots, \boldsymbol{y}^{(L)}\}$.
- Threshold θ of accumulation ratio.

Calculate the mean vector $\bar{\boldsymbol{y}}$ of the given training samples $\boldsymbol{Y}$.

Calculate the accumulation ratio $A'(\boldsymbol{U}'_k)$ based on (21).

if $A'(\boldsymbol{U}'_k) \geq \theta$ **then**

terminate this algorithm.

end if

for $i = 1$ to L **do**

Obtain the following residue vectors $\boldsymbol{h}_i$ using the ith training sample $\boldsymbol{y}^{(i)}$

$$\boldsymbol{h}_i = \frac{\boldsymbol{r}_i}{\|\boldsymbol{r}_i\|}$$

where

$$\boldsymbol{r}_i = \left(\boldsymbol{y}^{(i)} - \bar{\boldsymbol{x}}\right) - \boldsymbol{U}_k\boldsymbol{U}_k^T\left(\boldsymbol{y}^{(i)} - \bar{\boldsymbol{x}}\right). \tag{26}$$

end for

Define an index set $\mathcal{H}$ of $\boldsymbol{h}_i$.

loop

Find the following residue vector $\boldsymbol{h}_{i'}$ which gives the maximum increment $\Delta\tilde{A}'_i$ in (25):

$$\boldsymbol{h}_{i'} = \arg\max_{i\in\mathcal{H}}\Delta\tilde{A}'_i.$$

$\boldsymbol{H} \leftarrow [\boldsymbol{H}, \boldsymbol{h}_{i'}]$, $l \leftarrow l+1$, and $\mathcal{H} \leftarrow \mathcal{H} \setminus i'$.

if $\mathcal{H}$ is empty **then**

terminate this algorithm.

end if

Calculate the updated accumulation ratio $A'(\boldsymbol{U}'_{k+l})$ based on (23).

if $A'(\boldsymbol{U}'_{k+l}) \geq \theta$ **then**

terminate this algorithm.

end if

end loop

Output: Set of augmented eigenaxes $\boldsymbol{H} = \{\boldsymbol{h}_1, \cdots, \boldsymbol{h}_l\}$.

B. Update of Classifier

The update of a feature space by IPCA leads to the rotation of eigenaxes and the dimensional augmentation. This means that the inputs of a classifier change in not only their values

but also the number of input variables. To learn a classification system stably even in one-pass incremental learning environments where no past training sample is kept in memory, we have proposed two classifier training approaches so far: one is the training with a prototype-based classifier (ECM [15] plus k-NNC) [21], [24] and the other is the training with a neural network classifier (RAN-LTM) [22]. In these approaches, the classification accuracy is determined not only by the performance of feature extraction but also by that of classifier. Therefore, to focus on the comparison of feature extraction methods, the classifier should be simple. In this paper, let us adopt an NNC with a fixed number of prototypes which remove randomly from given training samples.

For online classification purpose, the prototypes of NNC must be updated to adapt to the new eigenspace $\Omega' = (\bar{\boldsymbol{x}}', \boldsymbol{U}'_{k+l}, \boldsymbol{\Lambda}'_{k+l}, N+L)$ at every learning stage. This can be done in a straightforward way. Assume that we keep a set of P reference vectors $\mathcal{V} = \{(\boldsymbol{p}_j \in \mathcal{R}^n, z_j) | j = 1, \cdots, P\}$, where z_j is the class label of $\boldsymbol{p}_j$. Then, the prototype vectors $\tilde{\boldsymbol{p}}_j$ of NNC are calculated by projecting $\boldsymbol{p}_j$ to the eigenspace Ω' as follows:

$$\tilde{\boldsymbol{p}}_j = \boldsymbol{U}'^T_{k+l}(\boldsymbol{p}_j - \bar{\boldsymbol{x}}'). \tag{27}$$

The class label z_j is added to $\tilde{\boldsymbol{p}}_j$, and a set of prototypes are defined as $\tilde{\mathcal{V}} = \{(\tilde{\boldsymbol{p}}_j \in \mathcal{R}^k, z_j) | j = 1, \cdots, P\}$. Therefore, the prototypes of NNC must be updated by projecting the reference vectors in $\mathcal{V}$ at every learning stage.

Since NNCs predict a class label by finding the nearest prototype to a query, the training of an NNC is conducted only by updating the prototypes based on (27). The algorithm of classifier update is summarized in Algorithm 2.

Algorithm 2: Update of Classifier

Input:
- Eigenspace model $\Omega = (\bar{\boldsymbol{x}}, \boldsymbol{U}_k, \boldsymbol{\Lambda}_k, N)$.
- Reference vectors $\mathcal{V} = \{(\boldsymbol{p}_j, z_j) | j = 1, \cdots, P\}$.

for $j = 1$ to P **do**

Calculate the prototype vectors $\tilde{\boldsymbol{p}}_j$ based on (27).

end for

Output: Prototype set $\tilde{\mathcal{V}} = \{(\tilde{\boldsymbol{p}}_j, z_j) | j = 1, \cdots, P\}$.

C. Training of Initial Eigenspace

Assume that a set of initial training samples $D_0 = \{(\boldsymbol{x}^{(i)}, z^{(i)}) | i = 1, \cdots, N\}$ is given before the learning gets started. To form an initial eigenspace model $\Omega = (\bar{\boldsymbol{x}}, \boldsymbol{U}_k, \boldsymbol{\Lambda}_k, N, \theta)$, the conventional PCA is applied to D_0. However, before getting Ω, we have to know a threshold θ of the accumulation ratio. In many cases, a proper θ is unknown and often depends on training data sets. For this purpose, the cross-validation technique is often used.

First, we define the number of search points M and the search range $[\theta_1, \theta_M]$. Then, the mth search point θ_m is expressed by

$$\theta_m = \theta_1 + (p-1)\frac{\theta_M - \theta_1}{M-1}. \tag{28}$$

To estimate the appropriateness of θ_m, we evaluate the classification performance based on the leave-one-out cross validation. Namely, $(N-1)$ training examples are picked for training and the remaining one is used for test to evaluate the classification accuracy. All the combinations of training sets are evaluated and the classification accuracies are averaged over N trials. Then, the next search point is moved to θ_{m+1}. The same evaluation is carried out until the final search point θ_M is evaluated. If several θ_m's have the same performance, the smallest one is selected as θ.

After selecting the threshold θ, an initial eigenspace is calculated by applying PCA to all the training samples in D_0. The algorithm of the initial training is shown in Algorithm 3.

Algorithm 3: Training of Initial Eigenspace

Input:
- Initial training set $D_0 = \{(\boldsymbol{x}^{(i)} \in \mathcal{R}^n, z^{(i)}) | i = 1, \ldots, N\}$.
- The number M of search points.
- Search range $[\theta_1, \theta_M]$.
- The number P of prototypes.

for $m = 1$ to M **do**

Set the threshold of the accumulation ratio θ_m using (28).

Call *Calculate Eigenspace* to obtain the eigenspace model of D_0.

for $i = 1$ to N **do**

Set the query $\boldsymbol{y} = (\boldsymbol{x}^{(i)}, z^{(i)})$ and $D'_0 \leftarrow D_0 \setminus \boldsymbol{y}$.

Select $\min(P, N-1)$ samples randomly from D'_0 as reference vectors and put them into $\mathcal{V}$.

Call *Update of Classifier* to obtain the prototypes $\tilde{\mathcal{V}}$.

Call *Classification* to predict the class of $\boldsymbol{x}^{(i)}$.

end for

Evaluate the average classification accuracy.

end for

Find all the θ_m's that give the highest accuracy and put them into a set $\mathcal{P}$.

Set the threshold as follows: $\theta \leftarrow \min_m \theta_m \in \mathcal{P}$.

Call *Calculate Eigenspace* to obtain the eigenspace model of D_0.

Output:
- Threshold of accumulation ratio θ.
- Eigenspace model $\Omega = (\bar{\boldsymbol{x}}, \boldsymbol{U}_k, \boldsymbol{\Lambda}_k, N)$.

Algorithm 4: Calculate Eigenspace

Input:
- Training set $D = \{(\boldsymbol{x}^{(i)} \in \mathcal{R}^n, z^{(i)}) | i = 1, \cdots, N\}$.
- Threshold θ of accumulation ratio.

Apply PCA to D and obtain the eigenvectors $\boldsymbol{U}_n = \{\boldsymbol{u}_1, \cdots, \boldsymbol{u}_n\}$, whose eigenvalues $\boldsymbol{\Lambda}_n = \mathrm{diag}\{\lambda_1, \cdots, \lambda_n\}$ are sorted in decreasing order.

for $k = 1$ to n **do**

Calculate the accumulation ratio $A(\boldsymbol{U}_k)$ based on (16).

if $A(\boldsymbol{U}_k) \geq \theta$ **then**

exit the loop.

end if

end for

Calculate the mean vector $\bar{\boldsymbol{x}}$ of $\boldsymbol{x}^{(i)} \in D$.

Output: Eigenspace model $\Omega = (\bar{\boldsymbol{x}}, \boldsymbol{U}_k, \boldsymbol{\Lambda}_k, N)$.

D. Proposed Incremental Learning Algorithm for Chunk Data

In usual situations, the online process of classification and learning is conducted alternately. After constructing an initial eigenspace model, whenever a set of queries $\boldsymbol{Y} = \{\boldsymbol{y}^{(1)}, \cdots, \boldsymbol{y}^{(q)}\}$ is given to a system, the classification is conducted as shown in Algorithm 5.

Algorithm 5: Classification

Input:
- Query set $\boldsymbol{Y} = \{\boldsymbol{y}^{(1)}, \cdots, \boldsymbol{y}^{(q)}\}$.
- Eigenspace model $\Omega = (\bar{\boldsymbol{x}}, \boldsymbol{U}_k, \boldsymbol{\Lambda}_k, N)$.
- Prototype set $\tilde{\mathcal{V}} = \{(\tilde{\boldsymbol{p}}_j, z_j) | j = 1, \cdots, P\}$.

for $i = 1$ to q **do**

Calculate the projection $\tilde{\boldsymbol{y}}^{(i)}$ of the query $\boldsymbol{y}^{(i)}$ to the eigenspace Ω using (27).

Obtain the class label z_{j^*} as follows:
$z_{j^*} = \arg\min_{j \in \tilde{\mathcal{V}}} \|\tilde{\boldsymbol{y}}^{(i)} - \tilde{\boldsymbol{p}}_j\|$.

Set z_{j^*} to the prediction $z(\boldsymbol{y}^{(i)})$ of $\boldsymbol{y}^{(i)}$.

end for

Output: Prediction $\boldsymbol{Z} = \{z(\boldsymbol{y}^{(1)}), \cdots, z(\boldsymbol{y}^{(q)})\}$.

After the classification is over, all or a part of the queries $\boldsymbol{y}^{(i)}$ are used for training. Assume that L of q queries are selected[4] and the class labels of the L queries are provided by a supervisor. The pairs $(\boldsymbol{y}^{(i)}, z^{(i)})$ $(i = 1, \cdots, L)$ are put into a training set D, and the learning of an eigenspace and a classifier gets started.

The number of chunk training samples L (i.e., the size of D) is generally changed at every learning stage in an online classification process. We should note that the proposed classification system can work even for variable L because there is no limitation on the chunk size to be applied to the proposed chunk IPCA. The main algorithm of learning and classification is summarized in Algorithm 6.

Algorithm 6: Learning and Classification

Input:
- Chunk IPCA algorithm.
- Initial training set $D_0 = \{(\boldsymbol{x}^{(i)}, z^{(i)}) | i = 1, \cdots, N\}$.
- The number P of prototypes.
- The number M of search points for threshold and search range $[\theta_1, \theta_M]$.

Initialization:
1) Call *Training of Initial Eigenspace* to obtain the threshold θ and the initial eigenspace model $\Omega = (\bar{\boldsymbol{x}}, \boldsymbol{U}_k, \boldsymbol{\Lambda}_k, N)$ of D_0.
2) $P' \leftarrow \min(P, N)$.
3) Select P' training samples randomly from D_0 as reference vectors and put them into a set $\mathcal{V}$.

loop // Prediction and Learning

Input: A new chunk of training samples $D = \{(\boldsymbol{y}^{(i)}, z^{(i)}) | i = 1, \cdots, L\}$.

if $P' < P$ **then**

select $\min(P - P', L)$ training samples randomly from D

put them into $\mathcal{V}$

$P' \leftarrow P' + \min(P - P', L)$.

end if

Call *Update of Classifier* to update the prototypes $\tilde{\mathcal{V}}$.

Call *Classification* to predict the class labels $z(\boldsymbol{y}^{(i)})$ of queries $\boldsymbol{y}^{(i)}$ $(i = 1, \cdots, L)$ in D.

Apply chunk IPCA to $\boldsymbol{Y} = \{\boldsymbol{y}^{(1)}, \cdots, \boldsymbol{y}^{(L)}\}$
1) Call *Selection of Eigenaxes* to obtain a matrix $\boldsymbol{H}_l$ of the l augmented eigenaxes.
2) Solve an intermediate eigenproblem in (14) or (15) to obtain a rotation matrix $\boldsymbol{R}$ and an eigenvalue matrix $\boldsymbol{\Lambda}'_{k+l}$.
3) Update the mean vector $\bar{\boldsymbol{x}}'$ and the eigenvector matrix $\boldsymbol{U}'_{k+l}$ based on (9) and (12), respectively.

Update the eigenspace model as follows:
$\Omega = (\bar{\boldsymbol{x}}, \boldsymbol{U}_k, \boldsymbol{\Lambda}_k, N) \leftarrow \Omega' = (\bar{\boldsymbol{x}}', \boldsymbol{U}'_{k+l}, \boldsymbol{\Lambda}'_{k+l}, N + L)$

Output: Prediction $z(\boldsymbol{y}^{(i)})$ $(i = 1, \cdots, L)$.

end loop

[4]In some cases, only the queries that the classifier fails to classify might be selected for training or sometimes all the queries might be used for training.

TABLE I
EVALUATED DATA SETS

Database Name	#Attributes	#Classes	#Train. Data	#Test Data
Adult	14	2	30,162	15,060
Letter Recognition	16	26	10,000	10,000
Spambase	57	2	2,301	2,300
Musk	166	2	3,299	3,299
Isolet Spoken Letter Recog.	617	26	6,238	1,559
Internet Advertisement	1,558	2	1,180	1,179
Microarray	5,000	12	1,110	1,110

Because we assume a continuous learning environment, the learning in the previous algorithm never converges because a chunk of training samples is successively provided to a system forever. However, the convergence at each learning stage is easily proved. The proposed learning algorithm has only one conditional loop in *Selection of Eigenaxes*. This conditional loop is repeated until the accumulation ratio becomes larger than the threshold; however, the maximum repetition of this loop is obviously L which is equal to the number of residue vectors $\boldsymbol{h}_i$ $(i = 1, \cdots, L)$. Therefore, when there is no more residue vector to be added as an eigenaxis, the algorithm is terminated.

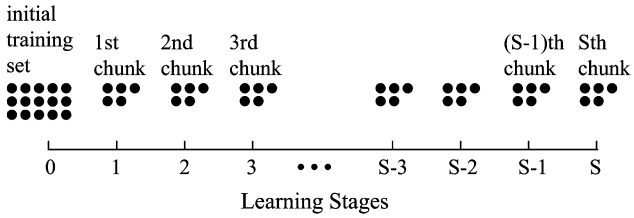

Fig. 1. Presentation of training samples in the assumed incremental learning environments. At stage 0, an initial training set is given for initial training. For the sake of simplicity, it is assumed that the same size of data chunk is given over the following learning stages. A chunk of training samples is randomly selected from a training data set and it has no overlap with other chunks; hence, all the training samples are presented only once.

IV. EXPERIMENTS

A. Experimental Setup

Table I shows the seven data sets for the evaluation of the proposed chunk IPCA. To study on the scalability of the proposed chunk IPCA, we choose large-scale data sets which have a large number of data samples and/or attributes. The first six data sets in Table I are selected from the University of California at Irvine (UCI) Machine Learning Repository [32], and the last one is a synthetic data set with 5000 attributes which is compiled from six different microarray data sets. Each microarray data set is provided as a two-class classification problem to diagnose NS cancer, lymphoma, ovarian, leukemia, lung cancer, and breast cancer. Since these six data sets have only 740 data in total, the number of learning stages is very small when the chunk size is large (e.g., $L = 100$). Thus, we use additional 1480 data for training and test which are generated from the original data by adding white noise. Because "internet advertisement data set," "letter recognition data set," "microarray data set," "musk data set," and "spambase data set" are not divided into training and test samples, we split them randomly into two halves, and the first and the second subsets are used for training and test, respectively. Although musk database includes two data sets in the UCI repository, we select the larger one.

To construct an initial eigenspace, a part of training samples are applied to the conventional PCA. The rate of an initial data set is denoted by p (in percent), which is changed from 0.1% to 20% to investigate the influence of the initial data set size on the performance. The remaining $(100 - p)$ (percent) of training samples are sequentially provided to learn as shown in Fig. 1. As stated in Section III-D, the proposed chunk IPCA algorithm works even though any size of data chunk is given at each learning stage. However, for the sake of simplicity, the chunk size is fixed at a constant L over the entire learning period. In the experiment in Section IV-C, to study the influence of chunk size on the performance, the constant L is changed from 5 to 100; in the other experiments, L is set to 5.

A chunk of training samples is randomly selected and it has no overlap with other chunks; hence, all the training samples are presented only once because we assume one-pass incremental learning environments. Hence, the number of learning stages S is given by $S = \lceil (1 - p)N/L \rceil$, where N is the total number of training samples. Note that the number of training samples in the last chunk can be less than L, and it is given by $N - L\lfloor (1-p)N/L \rfloor$. Since the performance of incremental learning generally depends on the sequence of training samples, 20 trials with different sequences of training samples are carried out to evaluate the average performance for the test data sets in Table I.

In Algorithm 6, we need to specify the following four parameter values: P, M, θ_1, and θ_M. The number of prototypes P is set to $(0.1 \times N)$ where N is the total number of training samples. In the leave-one-out cross validation, the number M of searching points should be as large as possible and the search range $[\theta_1, \theta_M]$ should be as broad as possible to obtain an optimal threshold θ of accumulation ratio. In our experiment, the following values are empirically given so that the computation costs would not be too high: $M = 50$ and $[\theta_1, \theta_M] = [0.8, 0.9999]$. The training samples are allocated to the prototypes of the NNC on an ongoing basis until P reaches $(0.1 \times N)$. To make a fair comparison, we give the same prototypes for the three eigenspace models shown in the next section.

TABLE II
COMPARISON OF AVERAGE CLASSIFICATION ACCURACY (PERCENT) FOR SEVEN DATA SETS WHEN THE PERCENTAGES p OF INITIAL TRAINING DATA ARE 0.5% AND 5%. THE VALUES AFTER $\pm$ MEAN THE STANDARD DEVIATIONS. THE CHUNK SIZE L IS FIXED AT 5 THROUGHOUT THE LEARNING STAGES. THE SINGLE ASTERISK (*) AND THE DOUBLE ASTERISK (**) MEAN THAT THE AVERAGE DIFFERENCE FROM CHUNK IPCA IS STATISTICALLY SIGNIFICANT WITH 5% AND 1% LEVEL, RESPECTIVELY. WHEN THE INITIAL DATA SET IS SMALL, IT IS CLEAR THAT THE UPDATE OF A FEATURE SPACE IS VERY EFFECTIVE. IN MOST CASES, THERE IS NO SIGNIFICANT DIFFERENCE FROM BATCH PCA AND IPCA. THIS RESULTS MEANS THAT THE INCREMENTAL LEARNING OF A FEATURE SPACE IS STABLY CARRIED OUT IN CHUNK IPCA. (A) RATE OF INITIAL DATA ($p = 0.5\%$); (B) RATE OF INITIAL DATA ($p = 5\%$)

(a)

	Adult	Letter	Spam	Musk	Isolet	Advertise	Microarray
Fixed Eigenspace	77.9±0.77**	72.4±2.6**	76.2±3.9**	86.1±1.5**	65.8±3.0**	78.3±9.5**	41.4±8.1**
IPCA	78.6±0.48	75.5±1.2	81.9±1.4	89.6±1.1	77.7±1.5	87.9±2.9	65.1±4.9
Batch PCA	78.5±0.47	75.8±1.1	82.4±1.6	89.8±1.3	79.4±1.3**	88.3±2.5	66.5±4.7
Chunk IPCA	78.6±0.47	75.3±1.2	82.0±1.6	89.6±1.3	77.1±1.5	88.0±2.1	64.8±5.1

(b)

	Adult	Letter	Spam	Musk	Isolet	Advertise	Microarray
Fixed Eigenspace	78.4±0.48	76.5±0.85	83.3±1.6*	89.6±1.4	77.4±2.0	89.2±2.4	54.7±9.7**
IPCA	78.4±0.47	76.5±0.85	82.2±1.7	89.7±1.2	78.3±1.5	88.5±2.1	65.9±5.5
Batch PCA	78.5±0.45	76.6±0.83	82.4±1.6	89.7±1.2	79.3±1.3*	88.9±2.0	66.5±5.0
Chunk IPCA	78.4±0.47	76.5±0.85	82.2±1.7	89.7±1.2	78.3±1.6	88.3±1.7	64.6±4.6

B. Performance Comparison

The proposed chunk IPCA is evaluated through a comparison with the following three eigenspace models.

1) *Fixed Eigenspace*: An eigenspace is obtained by applying PCA to an initial training set D_0, and it is fixed over the entire learning stages. This eigenspace model is adopted to see the usefulness of updating a feature space.
2) *IPCA*: An eigenspace model is incrementally updated by the IPCA algorithm in which the criterion of dimensional augmentation is based on the accumulation ratio [21]. The eigenspace model is updated with a single training sample at a time even if the training samples are provided in a chunk. The same threshold θ is used as that of chunk IPCA which is determined through the leave-one-out cross validation (see Algorithm 3 for details). Because the two eigenspaces created by IPCA and chunk IPCA have similar accumulation ratio, it is expected that these two give similar classification accuracy. Therefore, a main purpose of adopting IPCA is to evaluate the computation costs.
3) *Batch PCA*: An eigenfeature space is updated at every learning stage by applying PCA to all the training samples given so far. The computation costs would be more expensive than the incremental learning methods, but an accurate eigenspace is always obtained. The dimensionality of an eigenspace is set to the same value as the proposed chunk IPCA. Therefore, the classification accuracy of this method gives a target value for the proposed chunk IPCA.

Tables II and III, respectively, show the classification accuracy (percent) and the learning time (seconds) of the aforementioned three eigenspace models and the proposed chunk IPCA when the rate p of initial data is 1) $p = 0.5\%$ and 2) $p = 5\%$. The chunk size of training samples is fixed at 5 in the experiments (i.e., $L = 5$). As seen from Table II(a), the classification accuracy of the fixed eigenspace model is inferior to other models when only a small part of training samples ($p = 0.5\%$) are given at a starting point. Moreover, for all of the evaluated data sets, there is statistical significance in the average difference between fixed eigenspace and chunk IPCA with 1% level. This result suggests that the features selected from the initial training samples were inappropriate to classify the remaining training samples correctly; therefore, we can say that it is effective to update the feature space to adapt to the change of data distributions. On the other hand, however, we cannot recognize clear difference between fixed eigenspace and chunk IPCA in Table II(b) except for microarray data set. This result means that 5% of training samples are enough to construct an effective eigenspace for the large-scale data sets. However, we should note that the importance of incremental learning is not lost because we cannot know in advance whether an initial data set includes the whole information on training samples given in future.

On the other hand, there is no significant difference in the classification accuracy between IPCA and chunk IPCA. This is not a surprising result because the same threshold of the accumulation ratio is given to these two methods in our experiments; therefore, the eigenspaces generated by these two methods tend to have similar dimensionality. This means that the eigenspaces hold the same amount of information on training samples. Comparing with batch PCA, the proposed chunk IPCA has a little lower classification accuracy; however, the difference is not significant except for isolet. This means that the proposed method ensures the quasi-optimality in the construction of eigenspaces without keeping past training samples.

The computation costs of the eigenspace update are estimated by measuring the CPU time (seconds) using a Matlab function *cputime*. The computers used in the evaluation have twin Intel Xeon(TM) CPU 3.20 GHz and 2-GB RAM. As seen from Table III(a) and (b), the computation costs of chunk IPCA are

TABLE III
COMPARISON OF AVERAGE LEARNING TIME (SECONDS) FOR SEVEN DATA SETS WHEN THE PERCENTAGES p OF INITIAL TRAINING DATA ARE 0.5% AND 5%. THE CHUNK SIZE L IS FIXED AT 5 THROUGHOUT THE LEARNING STAGES. THE VALUES AFTER $\pm$ MEAN THE STANDARD DEVIATIONS. THE COMPUTATION COSTS OF CHUNK IPCA ARE SIGNIFICANTLY REDUCED AGAINST BOTH IPCA AND BATCH PCA. (A) RATE OF INITIAL DATA ($p = 0.5\%$); (B) RATE OF INITIAL DATA ($p = 5\%$)

(a)

	Adult	Letter	Spam	Musk	Isolet	Advertise	Microarray
IPCA	13.81±2.1	3.81±0.49	6.47±2.3	4.72±1.2	1902±4550	167.4±15.9	36.61±6.9
Batch PCA	340.6±50.5	43.87±8.4	1.95±0.21	74.12±2.2	3096±97.3	4821±254	1953±112
Chunk IPCA	7.04±0.57	1.62±0.29	1.27±0.28	1.02±0.22	94.8±139	31.6±2.0	22.5±2.5

(b)

	Adult	Letter	Spam	Musk	Isolet	Advertise	Microarray
IPCA	9.1±1.5	4.11±0.54	7.59±2.3	7.14±6.1	3805±5221	516.1±593	118.6±107
Batch PCA	281.7±14.6	44.72±5.5	5.65±0.24	72.7±2.8	9358±712	5573±213	2018±73.8
Chunk IPCA	4.99±0.34	1.55±0.18	1.38±0.33	1.42±0.74	348.8±405	92.2±142	51.9±36.7

significantly reduced against both IPCA and batch PCA especially for large-scale data sets.[5] The experimental results here indicate that the proposed method possesses excellent scalability in learning time without sacrificing classification accuracy.

C. Influence of Chunk Size

Fig.2 (a) and (b), respectively, shows the classification accuracy of chunk IPCA at the final learning stage and the learning time when training samples are provided in various chunk size L. Because the actual learning time is quite different depending on the evaluated data sets, we introduce the relative CPU time which is normalized by the CPU time for $L = 5$. As seen from Fig. 2 (a), we simply say that the classification accuracy is not affected by the chunk size.

On the other hand, as seen from Fig. 2 (b), the learning time is largely affected by L. For the data sets with a fairly small number of attributes (adult, letter, spambase, musk, and isolet), the learning time is prone to be shortened as the chunk size becomes large. This reduction in computation time mainly comes from the decrease in the repetition times of solving intermediate eigenproblems. For the data sets with many attributes (advertisement and microarray), however, the relative CPU time grows as the chunk size increases.

To understand this result, let us study on the computational complexity of chunk IPCA. The learning algorithm of chunk IPCA is mainly composed of the following operations: the selection of eigenaxes and the solving of eigenproblems. Table IV shows the computational complexity of the two operations in chunk IPCA when L training data are provided.

In Table IV, l means the number of augmented eigenaxes. In the worst case, l could be close to the chunk size L; then, the computational complexity of the eigenaxis selection is roughly reduced to $O(nL^3)$ because k is also considered to be $O(L)$ in the worst case. On the other hand, the computational complexity of solving eigenproblem is $O(L^3)$. Therefore, in the situation where many eigenaxes are augmented, the operation in chunk IPCA is dominated by eigenaxis selection and its computational costs are serious when the number of attributes n is very large. This is the explanation about the results of advertisement and microarray data sets in Fig. 2 (b).

One of the reasons why many eigenaxes are augmented is that an inappropriate threshold value θ is chosen in the parameter selection (i.e., unnecessarily large θ is selected). However, the performance of the parameter selection generally depends on how good initial training data are selected and this cannot be controlled by a learning system. This property has to be noticed when chunk IPCA is applied to high-dimensional data sets.

The results of advertisement and microarray data sets in Fig. 2 (b) demonstrate a limitation of the proposed chunk IPCA for high-dimensional data sets. However, in real situations, this problem can be alleviated by setting a smaller L (chunk size) in chunk IPCA than the actual chunk size of given data. For example, even if 100 samples are given in a chunk, L can be set to 10 in chunk IPCA by splitting the given chunk of 100 data into ten small chunks; that is, we can perform chunk IPCA with $L = 10$ ten times for the given chunk instead of performing chunk IPCA with $L = 100$. By doing this, the total learning time would significantly be reduced for microarray dataset. As seen from Fig. 2 (b), the training would be at least two times faster than performing chunk IPCA with $L = 100$. Therefore, from the practical point of view, the parameter L should be set to a small value (e.g., 5 or 10) for the high-dimensional data like advertisement and microarray data sets.

D. Influence of Initial Data Size

Fig. 3 (a) and (b), respectively, shows the classification accuracy of chunk IPCA and fixed eigenspace for different rates of initial data. The classification accuracy is evaluated after all the training samples are learned, and the size of chunk data is fixed at 5.

As seen from Fig. 3 (a), the classification accuracy of chunk IPCA is very stable for any size of initial training samples. On the contrary, the results in Fig. 3 (b) demonstrate that the classification accuracy for a small size of initial training samples is seriously deteriorated if the eigenspace is not update over the training period. For most of the data sets, it seems that at least

[5]As will be shown in Section IV-C, if the chunk size is large, sometimes chunk IPCA needs more learning time than IPCA for high-dimensional data sets.

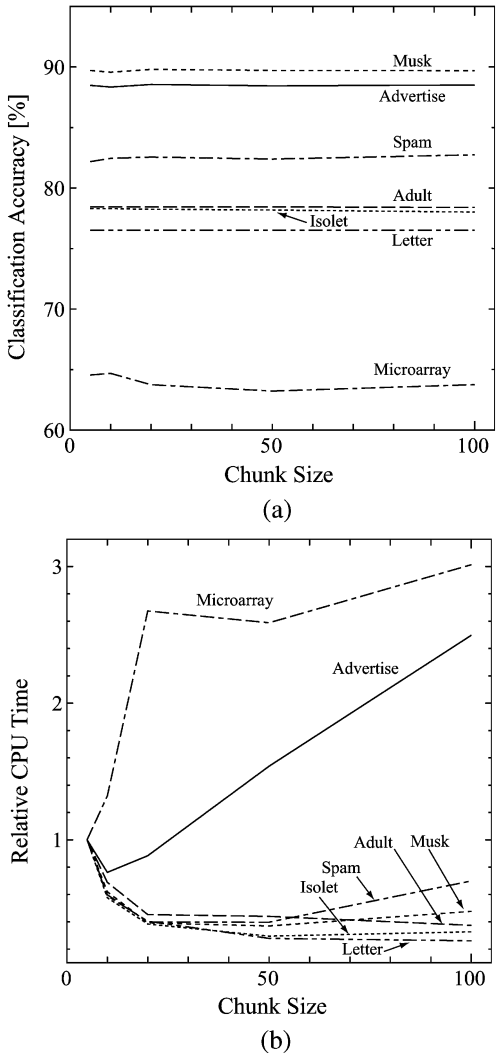

Fig. 2. Comparisons of (a) classification accuracy at the final learning stage and (b) relative CPU time of chunk IPCA for the seven data sets when the chunk size L is set to 5, 10, 20, 50, and 100. The classification accuracy is almost independent of L. On the other hand, the relative CPU time (learning time) is largely affected by L. For the data sets with a fairly small number of attributes (adult, letter, spambase, musk, and isolet), the learning time is prone to be shortened as the chunk size becomes large. For the data sets with many attributes (advertisement and microarray), however, the relative CPU time grows as the chunk size increases.

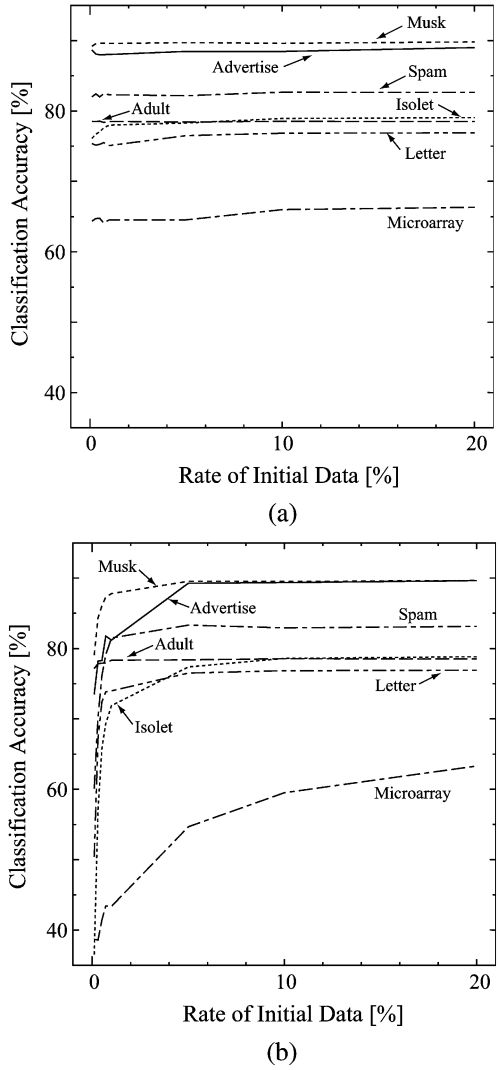

Fig. 3. Classification accuracy of (a) chunk IPCA and (b) fixed eigenspace when different sizes of initial training samples are given at a starting point. Fairly constant accuracy is obtained for chunk IPCA independent of the rate of initial data, while the accuracy of fixed eigenspace is seriously degraded for small initial data sets.

TABLE IV
COMPUTATIONAL COMPLEXITY OF CHUNK IPCA TO LEARN A CHUNK OF L TRAINING SAMPLES. HERE, n, k, AND l ARE THE NUMBER OF INPUT ATTRIBUTES, THE DIMENSION OF THE CURRENT EIGENSPACE, AND THE NUMBER OF AUGMENTED EIGENAXES, RESPECTIVELY

	Solving Eigenproblem	Eigen-axis Selection
Chunk IPCA	$O((k+l)^3)$	$O(n(k+l)lL)$

5% of training samples are necessary for attaining good classification performance even if the eigenspace is fixed. However, for the microarray data set, the classification accuracy of fixed eigenspace is lower than that of chunk IPCA even if 20% of training samples are given at a starting point.

These results demonstrate that chunk IPCA can stably improve the accuracy of a classification system to a satisfied level from any starting point. This property is very effective and important as an incremental learning system.

E. Accuracy of Updated Eigenvectors

To see whether an appropriate feature space is constructed by chunk IPCA, the similarity of eigenvectors obtained by batch PCA and chunk IPCA is examined. The similarity is measured by the following directional cosine d_i:

$$d_i = \frac{1}{M}\sum_{j=1}^{M} \boldsymbol{u}_{ji}^{(b)T}\boldsymbol{u}_{ji}^{(c)} \tag{29}$$

where $\boldsymbol{u}_{ji}^{(b)}$ and $\boldsymbol{u}_{ji}^{(c)}$ are, respectively, the ith eigenvector obtained by batch PCA and chunk IPCA in the jth trial, and M is the number of trials to average the similarity. Note that $\boldsymbol{u}_{ji}^{(b)}$ and $\boldsymbol{u}_{ji}^{(c)}$ are unit vectors. Obviously, if the similarity is one, it means two eigenvectors are identical.

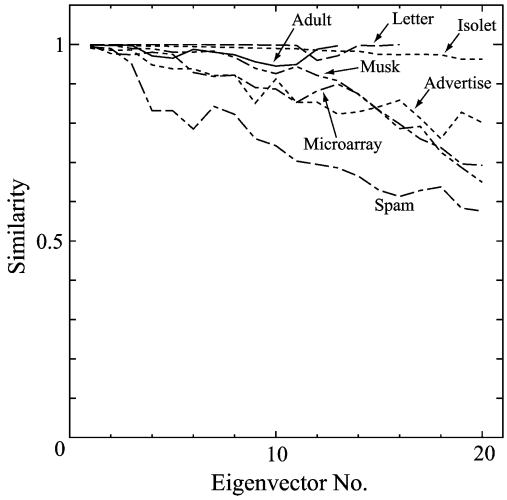

Fig. 4. Average similarity between eigenvectors obtained by batch PCA and chunk IPCA for the seven data sets in Table I when the chunk size is fixed at $L = 5$ and the rate of initial data sets is set to $p = 5\%$. The similarity is maintained at above 0.9 for the first eight major eigenvectors except for spam data set.

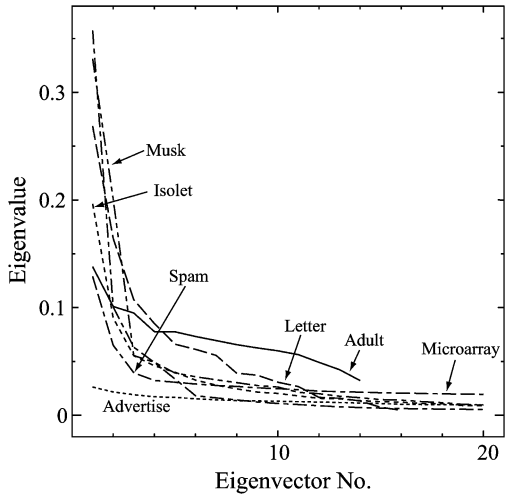

Fig. 5. Average eigenvalues of the major 20 eigenvectors. The ith normalized eigenvalue is defined as the eigenvalue of the ith eigenvector normalized by the sum of all the eigenvalues. The normalized eigenvalues are going down quickly below 0.1 after around the fifth major eigenvector.

Fig. 4 shows the average similarity between the eigenvectors obtained by batch PCA and chunk IPCA for the seven data sets in Table I. The chunk size is fixed at $L = 5$ and the rate of initial data sets is set to $p = 5\%$. The horizontal axis corresponds to the eigenvector number.

From the results in Fig. 4, we can see that the similarity is maintained at above 0.9 for the first eight major eigenvectors except for spam data set. As for adult, isolet, and letter data sets, about 20 major eigenvectors are exactly obtained by chunk IPCA. Fig. 5 shows the distribution of the normalized eigenvalues for the major 20 eigenvectors. Here, the ith normalized eigenvalue is defined as the eigenvalue of the ith eigenvector normalized by the sum of all the eigenvalues. As seen from Fig. 5, the normalized eigenvalues are going down quickly below 0.1 after around the fifth major eigenvector. From the results in Figs. 4 and 5, we can say that the proposed chunk IPCA gives a fairly good approximation to major eigenvectors with large eigenvalues.

On the other hand, the approximation to minor eigenvectors, whose normalized eigenvalues are almost zero in Fig. 5, have large errors. The primary reasons for this is originated from the approximation error introduced in the derivation of the intermediate eigenproblem in (14) and the accumulation ratio in (23). The cumulative effect of this approximation error could be small if the threshold of the accumulation ratio θ was provided properly. However, it is not easy to know a proper value of θ even though the leave-one-out cross validation is introduced into the parameter selection. In the one-pass learning situation assumed here, training samples are discarded immediately after the learning is finished at every stage. Therefore, if the distribution of training samples is largely varied from that in the past, some crucial information might be lost during learning, and the loss would prevent from constructing an effective eigenspace. To overcome this problem, we could introduce an adaptive mechanism for selecting θ. However, currently, the approximation error for minor eigenvectors does not affect the classification accuracy as far as we can see from the results in Table II. Therefore, this problem is left as our future work.

V. Conclusion and Future Work

In our previous works [21], [24], we have proposed an adaptive evolving connectionist model in which IPCA and ECM are effectively combined. This learning scheme gives a new concept for pattern recognition systems; that is, feature extraction and classifier learning are simultaneously carried out online. One drawback of this approach was scalability in terms of the number of samples and the number of their attributes. This drawback comes from the limitation of the previous approach where a training sample must be applied one by one even if a chunk of training sample is given at a time.

To overcome this problem, we also have proposed a new method called chunk IPCA [23], in which a chunk of training samples can be applied at a time to update an eigenspace model incrementally. In addition, we extended the incremental learning of a classifier such that it can also be learned with a chunk of training samples. This paper presents a practical algorithm of eigenaxis selection in chunk IPCA based on the accumulation ratio, and we give a complete form of an online classification system in which the incremental learning of an eigenspace and a simple NNC is simultaneously carried out under one-pass learning environments. This brings us an efficient way to learn a system in terms of computations and memory.

To evaluate the scalability and the learning efficiency of the proposed chunk IPCA, we tested the seven large-scale data sets with a large number of data samples and attributes. The experimental results suggested that the proposed learning scheme worked quite well even if training samples were provided in any size of chunks at a time. That is to say, the chunk size does not affect the final classification accuracy of the system and the learning time can be shortened for large chunk data unless the number of attributes is too large. Furthermore, we investigated

the influence of initial data size, and the results demonstrate that chunk IPCA can stably improve the accuracy of a classification system to a satisfied level from any starting point. Finally, we examined the approximation error of the eigenvectors. As a result, chunk IPCA learned major eigenvectors without serious errors, while minor eigenvectors had large errors. However, as far as we can see from the experimental results, the error for minor eigenvectors does not affect the classification accuracy seriously.

There still remains further work to be done. First, as described in Section IV-E, the approximation error to minor eigenvectors may prevent from constructing an effective eigenspace if the sample distributions were largely varied over time. This can be alleviated by introducing an adaptive mechanism for the threshold θ of accumulation ratio. Second, the system performance can be greatly enhanced by introducing ILDA [25] as feature extraction and by introducing other powerful classifiers such as neural networks [22] and support vector machine. Therefore, pursuing the optimal combination of the feature space learning and the classifier learning is another interesting topic. Third, PCA and LDA transform inputs into linear features, and these features are not always effective for classification purposes. Recently, kernel PCA and kernel LDA were widely noticed as high-performance feature extraction methods [2], [28], [40]; hence, the extension of incremental learning approach to kernel PCA and kernel LDA is very promising. One application of the proposed method is in online learning robotic systems, where the robots reevaluate online the input information and select the most appropriate features for the time [17]. Another application of the proposed method is in computational neurogenetic modeling, where new EEG data is added incrementally and a gene-regulatory network of relevant genes is derived to optimize a matching fitness function between the model and the data [3]. Challenging the aforementioned open questions and pursuing the directions to new application areas are left as our future work.

ACKNOWLEDGMENT

The authors would like to thank Prof. S. Abe for his helpful discussions. They would also like to thank the reviewers for their constructive comments and suggestions that improved this paper.

REFERENCES

[1] C. G. Atkeson, A. W. Moore, and S. Schaal, "Locally weighted learning," *Artif. Intell. Rev.*, vol. 11, pp. 75–113, 1997.

[2] G. Baudat and F. Anouar, "Generalized discriminant analysis using a kernel approach," *Neural Comput.*, vol. 12, pp. 2385–2404, 2000.

[3] L. Benuskova and N. Kasabov, *Computational Neurogenetic Modeling*. NY: Springer-Verlag, 2007.

[4] G. A. Carpenter and S. Grossberg, "The ART of adaptive pattern recognition by a self-organizing neural network," *IEEE Computer*, vol. 21, no. 3, pp. 77–88 , Mar. 1988.

[5] D. Chakraborty and N. R. Pal, "A novel training scheme for multilayered perceptrons to realize proper generalization and incremental learning," *IEEE Trans. Neural Netw.*, vol. 14, no. 1, pp. 1–14, Jan. 2003.

[6] O. Déniz, M. Castrillón, J. Lorenzo, and M. Hernández, "An incremental learning algorithm for face recognition," in *Biometric Authentication*, M. Tistarelli, J. Bigun, and A. K. Jain, Eds. New York: Springer-Verlag, 2002, pp. 1–9.

[7] C. P. Diehl and G. Cauwenberghs, "SVM incremental learning, adaptation and optimization," in *Proc. Int. Joint Conf. Neural Netw.*, 2003, vol. 4, pp. 2685–2690.

[8] H.-C. Fu, Y.-P. Lee, C.-C. Chiang, and H.-T. Pao, "Divide-and-conquer learning and modular perceptron networks," *IEEE Trans. Neural Netw.*, vol. 12, no. 2, pp. 250–263, Mar. 2001.

[9] P. Hall and R. Martin, "Incremental eigenanalysis for classification," in *Proc. British Mach. Vis. Conf.*, 1998, vol. 1, pp. 286–295.

[10] P. Hall, D. Marshall, and R. Martin, "Merging and splitting eigenspace models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 9, pp. 1042–1049, Sep. 2000.

[11] J.-H. Hong and S.-B. Cho, "Incremental support vector machine for unlabeled data classification," in *Proc. 9th Int. Conf. Neural Inf. Process.*, 2002, vol. 3, pp. 1403–1407.

[12] G.-B. Huang, P. Saratchandran, and N. Sundararajan, "An efficient sequential learning algorithm for growing and pruning RBF (GAP-RBF) networks," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 34, no. 6, pp. 2284–2292, Dec. 2004.

[13] G.-B. Huang, L. Chen, and C.-K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Trans. Neural Netw.*, vol. 17, no. 4, pp. 879–892, Jul. 2006.

[14] D. Kalles and T. Morris, "Efficient incremental induction of decision trees," *Mach. Learn.*, vol. 24, no. 3, pp. 231–242, 1996.

[15] N. Kasabov, *Evolving Connectionist Systems: Methods and Applications in Bioinformatics, Brain Study and Intelligent Machines*. New York: Springer-Verlag, 2002.

[16] N. Kasabov and Q. Song, "DENFIS: Dynamic evolving neuro-fuzzy inference system and its application for time-series prediction," *IEEE Trans. Fuzzy Syst.*, vol. 10, no. 2, pp. 144–154, Apr. 2002.

[17] N. Kasabov, *Evolving Connectionist Systems: The Knowledge Engineering Approach*. London, U.K.: Springer-Verlag, 2007.

[18] M. Kobayashi, A. Zamani, S. Ozawa, and S. Abe, "Reducing computations in incremental learning for feedforward neural network with long-term memory," in *Proc. Int. Joint Conf. Neural Netw.*, 2001, vol. 3, pp. 1989–1994.

[19] H. G. Loos, "Parity madeline: A neural net with complete boolean repertoire capable of one-pass learning," in *Proc. Int. Joint Conf. Neural Netw.*, 1989, vol. 2, pp. 111–118.

[20] E. Oja and J. Karhunen, "On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix," *J. Math. Anal. Appl.*, vol. 106, pp. 69–84, 1985.

[21] S. Ozawa, S. Pang, and N. Kasabov, "A modified incremental principal component analysis for on-line learning of feature space and classifier," in *PRICAI 2004: Trends in Artificial Intelligence LNAI*, C. Zhang, H. W. Guesgen, and W. K. Yeap, Eds. New York: Springer-Verlag, 2004, pp. 231–240.

[22] S. Ozawa, S. L. Toh, S. Abe, S. Pang, and N. Kasabov, "Incremental learning of feature space and classifier for face recognition," *Neural Netw.*, vol. 18, no. 5–6, pp. 575–584, 2005.

[23] S. Ozawa, S. Pang, and N. Kasabov, "An incremental principal component analysis for chunk data," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, 2006, pp. 2278–2285.

[24] S. Pang, S. Ozawa, and N. Kasabov, "One-pass incremental membership authentication by face classification," in *Biometric Authentication*, D. Zhang and A. K. Jain, Eds. New York: Springer-Verlag, 2004, Lecture Notes in Computer Science, pp. 155–161.

[25] S. Pang, S. Ozawa, and N. Kasabov, "Incremental linear discriminant analysis for classification of data streams," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 35, no. 5, pp. 905–914, Oct. 2005.

[26] R. Polikar, L. Upda, S. S. Upda, and V. Honavar, "Learn++: An incremental learning algorithm for supervised neural networks," *IEEE Trans. Syst. Man Cybern. C, Human Syst.*, vol. 31, no. 4, pp. 497–508, Aug. 2001.

[27] T. D. Sanger, "Optimal unsupervised learning in a single-layer linear feedforward neural network," *Neural Netw.*, vol. 2, no. 6, pp. 459–473, 1989.

[28] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," MPI Tech. Rep. 44, 1996.

[29] A. Shilton, M. Palaniswami, D. Ralph, and A. C. Tsoi, "Incremental training of support vector machines," *IEEE Trans. Neural Netw.*, vol. 16, no. 1, pp. 114–131, Jan. 2005.

[30] S. Thrun and L. Pratt, *Learning to Learn*. Norwell, MA: Kluwer, 1998.

[31] S. L. Toh and S. Ozawa, "A face recognition system using neural networks with incremental learning ability," in *Proc. 8th Australian/New Zealand Conf. Intell. Inf. Syst.*, 2003, pp. 389–394.

[32] University of California at Irvine, Machine Learning Repository, Irvine, CA [Online]. Available: http://www.ics.uci.edu/~mlearn/ML-Repository.html

[33] P. E. Utgoff, "Incremental induction of decision trees," *Mach. Learn.*, vol. 4, no. 2, pp. 161–186, 1989.
[34] S. Wan and L. E. Banta, "Parameter incremental learning algorithm for neural networks," *IEEE Trans. Neural Netw.*, vol. 17, no. 6, pp. 1424–1438, Nov. 2006.
[35] J. Weng, C. H. Evans, and W.-S. Hwang, "An incremental learning method for face recognition under continuous video stream," in *Proc. 4th IEEE Int. Conf. Autom. Face Gesture Recognit.*, 2000, pp. 251–256.
[36] J. Weng, Y. Zhang, and W.-S. Hwang, "Candid covariance-free incremental principal component analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 8, pp. 1034–1040, Aug. 2003.
[37] J. Weng and W.-S. Hwang, "Incremental hierarchical discriminant regression," *IEEE Trans. Neural Netw.*, vol. 18, no. 2, pp. 397–415, Mar. 2007.
[38] K. Yamauchi, N. Yamaguchi, and N. Ishii, "Incremental learning methods with retrieving of interfered patterns," *IEEE Trans. Neural Netw.*, vol. 10, no. 6, pp. 1351–1365, Nov. 1999.
[39] J. Yan, Q.-S. Cheng, Q. Yang, and B. Zhang, "An incremental subspace learning algorithm to categorize large scale text data," in *Prof. 7th Asia-Pacific Web Conf.*, 2005, pp. 52–63.
[40] M. Yang, "Kernel eigenfaces vs. kernel fisherfaces: Face recognition using kernel methods," in *Proc. IEEE Int. Conf. Autom. Face Gesture Recognit.*, 2002, pp. 215–220.
[41] H. Zhao, P. C. Yuen, and J. T. Kwok, "A novel incremental principal component analysis and its application for face recognition," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 36, no. 4, pp. 873–886, Aug. 2006.

Seiichi Ozawa (M'02) received the B.E. and M.E. degrees in instrumentation engineering and the Ph.D. degree in computer science from Kobe University, Kobe, Japan, in 1987, 1989, and 1998, respectively.

Currently, he is an Associate Professor at the Graduate School of Engineering, Kobe University. From 2005 to 2006, he was a Visiting Researcher at Arizona State University, Tempe. His primary research interests include neural networks, machine learning, intelligent data processing, and pattern recognition.

Dr. Ozawa is a member of the International Neural Network Society (INNS), the Institute of Electrical Engineers of Japan (IEEJ), the Institute of Electronics, Information and Communication Engineers (IEICE), the Society of Instrument and Control Engineers (SICE), and the Institute of Systems, Control and Information Engineers (ISCIE).

Shaoning Pang (M'04–SM'05) received the B.Sc. degree in physics and the M.Sc. degree in electronic engineering from Xinjiang University, China, in 1994 and 1997, respectively, and the Ph.D. degree in computer science from Shanghai Jiao Tong University, China, in 2000.

From 2001 to 2003, he was a Research Associate at the Pohang University of Science and Technology (POSTECH), South Korea. Currently, he is a Permanent Senior Research Fellow at Knowledge Engineering and Discovery Research Institute (KEDRI), Auckland University of Technology, Auckland, New Zealand. His research interests include support vector machine (SVM) aggregating intelligence, incremental machine learning, bioinformatics, and neural computing for industrial applications.

Dr. Pang has been serving as a Program Member and Session Chair for several international conferences including International Symposium on Neural Networks (ISNN), International Conference on Neural Information Processing (ICONIP), and International Conference on Neural Networks and Signal Processing (ICNNSP). He was a best paper winner of the 2003 IEEE ICNNSP and an invited speaker of BrainIT 2007. He is currently acting as a regular paper reviewer for a number of refereed international journals and conferences. He is a Member of the Institute of Electronics, Information, and Communication Engineers (IEICE) and Association for Computing Machinery (ACM).

Nikola Kasabov (SM'97) received the M.Sc. degree in computer science and engineering, the M.Sc. degree in applied mathematics, and the Ph.D. degree in mathematical sciences from the Technical University of Sofia, Sofia, Bulgaria, in 1971, 1972, and 1975, respectively.

Currently, he is the Founding Director and the Chief Scientist of the Knowledge Engineering and Discovery Research Institute (KEDRI) and the Chair of Knowledge Engineering at the School of Computing and Mathematical Sciences at Auckland University of Technology, Auckland, New Zealand. He has worked at the University of Otago, New Zealand; University of Essex, U.K.; University of Trento, Italy; Technical University of Sofia, Bulgaria; University of California at Berkeley, CA; RIKEN Brain Science Institute, Tokyo, Japan; and TU Kaiserslautern Germany. He has published more than 400 publications, which include 12 books and 28 patents. His main research interests are in the areas of: evolving connectionist systems, neurocomputing, soft computing, bioinformatics, brain study, speech and image processing, and novel methods for data mining and knowledge discovery.

Dr. Kasabov is a Fellow of the Royal Society of New Zealand and the New Zealand Computer Society. He is a Vice President of the International Neural Network Society (INNS), and the President of the Asia Pacific Neural Network Assembly (APNNA). He is an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS, the IEEE TRANSACTIONS ON FUZZY SYSTEMS, *Neural Networks*, and other journals. He has chaired a series of artificial neural networks, expert systems, and evolving intelligence conferences in New Zealand since 1993.