Incremental Linear Discriminant Analysis for Classification of Data Streams

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

***Abstract*—This paper presents a constructive method for deriving an updated discriminant eigenspace for classification when bursts of data that contains new classes is being added to an initial discriminant eigenspace in the form of random chunks. Basically, we propose an incremental linear discriminant analysis (ILDA) in its two forms: a sequential ILDA and a Chunk ILDA. In experiments, we have tested ILDA using datasets with a small number of classes and small-dimensional features, as well as datasets with a large number of classes and large-dimensional features. We have compared the proposed ILDA against the traditional batch LDA in terms of discriminability, execution time and memory usage with the increasing volume of data addition. The results show that the proposed ILDA can effectively evolve a discriminant eigenspace over a fast and large data stream, and extract features with superior discriminability in classification, when compared with other methods.**

***Index Terms*—Classification, data stream, incremental linear discriminant analysis, incremental principle component analysis, linear discriminant analysis, pattern recognition, principle component analysis.**

I. Introduction

Linear Discriminant Analysis (LDA), also known as Fisher Discriminant Analysis (FDA), seeks directions for efficient discrimination, while another technique known as principle component analysis (PCA) [10], [11] seeks directions efficient for representation. The typical implementation of these two techniques assumes that a complete dataset for training is given in advance, and learning is carried out in one batch. Under the terms, both techniques have been widely used in the research of face recognition and mobile robotics [13]–[16], as well as some data mining and knowledge discovery applications [17], [18].

However, when we conduct LDA/PCA learning over datasets in real-world applications, we often confront difficult situation where a complete set of training samples is not given in advance. Actually in most cases, data are presented as a stream of chunks illustrated in Fig. 1. Due to the fact that we can not know what kind of data will be presented in the future, both the scale of the chunks and the individual samples in each chunk are given randomly.

Manuscript received November 17, 2004. This paper was recommended by Associate Editor Detlef D. Nauck.

S. Pang and N. Kasabov are with the Knowledge Engineering and Discovery Research Institute, Auckland University of Technology, Auckland 1020, New Zealand (e-mail: spang@aut.ac.nz; nkasabov@aut.ac.nz).

S. Ozawa is with the Graduate School of Science and Technology, Kobe University, Kobe 657-8501, Japan (e-mail: ozawsei@kobe-u.ac.jp).

Digital Object Identifier 10.1109/TSMCB.2005.847744

A straightforward approach is that we can collect data whenever new data are presented and then construct a provisional system by a batch learning over the collected data so far. However, it is obvious that such system only works under a condition of a large memory and high computation expenses, because the system would need to maintain a huge memory to store the data either previously learned, or newly presented, possibly without a limit. Moreover, the system has to discard the knowledge acquired in the past, even if the learning of 99.9% data is finished, and repeat the learning from the beginning whenever one additional sample is presented.

Obviously, one-pass incremental learning gives a solution to the above problem. In this learning scheme, a system must acquire knowledge with a single presentation of the training data and retaining the knowledge acquired in the past without keeping a large number of training samples.

To achieve this, several methods [1]–[3] have been proposed to perform incremental learning by updating eigenspace models. Among them, Hall *et al.* [1] proposed Incremental PCA (IPCA) based on the updating of covariance matrix through a residue estimating procedure. However, all these methods have only considered adding exactly one new sample to an eigenspace model at a time. Later, Hall improved his method by proposing a method of merging and splitting eigenspace models [4] that allows a chunk of new samples to be learned in a single step. To speed up the IPCA computation, Weng *et al.* [8] proposed a fast IPCA approximation algorithm computing the principle components of a sequence of samples incrementally without estimating the covariance matrix. Based on Hall's method, we previously developed a one-pass incremental classification algorithm that has effectively solved the problem of online face membership authentication [5], [6]. This pattern recognition algorithm is completely one-pass, because both feature selection and classification are modeled using one-pass incremental learning method. IPCA is used for feature selection. The K-Nearest Neighbor (KNN), with prototypes updated using evolving clustering method (ECM) [7], is used as an incremental classifier.

Motivated by the IPCA, we propose here an incremental linear discriminant analysis (ILDA) for the classification of data streams. A difficulty for incremental LDA modeling, compared with previous IPCA modeling, is that all class data of a complete training dataset may not be presented at every incremental learning stage. The number of classes presented at each learning stage might be very random in real life as shown in Fig. 1. New classes of data may be presented after several learning stages are past, such as the third-class data in Fig. 1, which starts to be presented in the data stream from stage t_3,

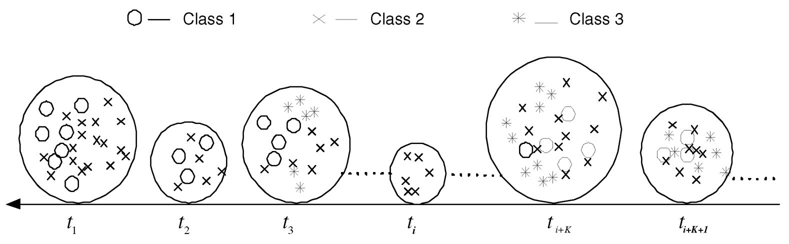

Fig. 1. Data stream for classification.

and was not presented at stages t_1 and t_2. In this paper, we derive a solution of ILDA updating discriminant eigenspace while bursts of new class data are coming in at different times.

The rest of this paper is organized as follows. Section II defines a discriminant eigenspace model at the beginning and later derives two incremental linear discriminant analysis approaches: Sequential ILDA and Chunk ILDA. Section III presents various tests of ILDA on incremental learning over both UCL data and face data. Finally, conclusions are given in Section IV.

II. Incremental LDA

Let us assume that N training samples $\boldsymbol{X} = \{\boldsymbol{x}_i\}$ $(i = 1,\ldots,N)$ in M classes have been presented so far. According to definition [10], LDA seeks a linear transformation $\boldsymbol{U}$ over $\boldsymbol{X}$ in such a way that the ratio of $\boldsymbol{Sb}$ and $\boldsymbol{Sw}$ is maximized, where

$$\boldsymbol{Sw} = \sum_{c=1}^{M} \boldsymbol{\Sigma}_c = \sum_{c=1}^{M} \sum_{\boldsymbol{x}\in\{\boldsymbol{x}_c\}} (\boldsymbol{x}-\bar{\boldsymbol{x}}_c)(\boldsymbol{x}-\bar{\boldsymbol{x}}_c)^T \tag{1}$$

is the within-class scatter matrix, and

$$\boldsymbol{Sb} = \sum_{c=1}^{M} n_c(\bar{\boldsymbol{x}}_c-\bar{\boldsymbol{x}})(\bar{\boldsymbol{x}}_c-\bar{\boldsymbol{x}})^T \tag{2}$$

is the between-class scatter matrix. n_c is the number of samples in class c such that $N = \sum_{c=1}^{M} n_c$. $\bar{\boldsymbol{x}} = (1/N)\sum_{i=1}^{N} \boldsymbol{x}_i$ is the mean vector of X, and $\bar{\boldsymbol{x}}_c$ is the mean vector in class c.

Because the transformation matrix U can be obtained by conducting an eigenvalue decomposition of matrix $\boldsymbol{D} = \boldsymbol{Sw}^{-1}\boldsymbol{Sb}$,

$$\boldsymbol{DU} = \boldsymbol{U\Lambda} \tag{3}$$

a discriminant eigenspace, also called fisherspace, model can be represented by a discriminant eigenspace model as $\Omega = (\boldsymbol{Sw}, \boldsymbol{Sb}, \bar{\boldsymbol{x}}, N)$.

$\boldsymbol{U}$ is an $n \times n$ matrix whose columns correspond to the discriminant eigenvectors. In applications, eigenvectors with small eigenvalue can be discarded to compress a high dimensional data to a low-dimensional feature with an enhanced discrimination.

The traditional LDA works in a batch way assuming that the whole dataset is given in advance and is trained in one batch. We called it Batch LDA in this paper. However, in a streaming environment as in Fig. 1, new samples are being presented continuously, possibly without end. The addition of these new samples will lead to the changes of the original mean vector $\bar{\boldsymbol{x}}$, within class scatter matrix $\boldsymbol{Sw}$, as well as between-class distance matrix $\boldsymbol{Sw}$, therefore the whole discriminant eigenspace model Ω should be updated.

A. Sequential Incremental LDA

Suppose that the new samples are acquired sequentially, $\boldsymbol{x}_{N+1}, \boldsymbol{x}_{N+2}, \ldots$, possibly to infinity.

Let us consider the case that the $(N+1)th$ training sample $\boldsymbol{y}$ is presented with class label k. Now, the question is how to compute the updated discriminant eigenspace model $\Omega' = (\boldsymbol{Sw}', \boldsymbol{Sb}', \bar{\boldsymbol{x}}', N+1)$ for $[X\ \boldsymbol{y}]$ using only Ω and $\boldsymbol{y}$.

The updated mean is

$$\bar{\boldsymbol{x}}' = \frac{(N\bar{\boldsymbol{x}} + \boldsymbol{y})}{(N+1)} \tag{4}$$

For the between-class scatter matrix $\boldsymbol{Sb}$, if $k = M+1$ representing a newly introduced class, then

$$\begin{aligned}\boldsymbol{Sb}' &= \sum_{c=1}^{M} n_c(\bar{\boldsymbol{x}}_c-\bar{\boldsymbol{x}}')(\bar{\boldsymbol{x}}_c-\bar{\boldsymbol{x}}')^T + (\boldsymbol{y}-\bar{\boldsymbol{x}}')(\boldsymbol{y}-\bar{\boldsymbol{x}}')^T\\ &= \sum_{c=1}^{M+1} n_c'(\bar{\boldsymbol{x}}_c-\bar{\boldsymbol{x}}')(\bar{\boldsymbol{x}}_c-\bar{\boldsymbol{x}}')^T\end{aligned} \tag{5}$$

where n_c' is the number of samples in class c after $\boldsymbol{y}$ is presented, $n_c' = n_c$ when $1 \le c \le M$, $n_c' = 1$ when $c = M+1$, and $\bar{\boldsymbol{x}}_c = y$ when $c = M+1$.

If $1 \le c \le M$, then the updated matrix $\boldsymbol{Sb}'$ is

$$\boldsymbol{Sb}' = \sum_{c=1}^{M} n_c'(\bar{\boldsymbol{x}}_c'-\bar{\boldsymbol{x}}')(\bar{\boldsymbol{x}}_c'-\bar{\boldsymbol{x}}')^T \tag{6}$$

where $\bar{\boldsymbol{x}}_c' = (1/(n_c+1))(n_c\bar{\boldsymbol{x}}_c + \boldsymbol{y})$ and $n_c' = n_c+1$, if $\boldsymbol{y}$ belongs to class c; else $\bar{\boldsymbol{x}}_c' = \bar{\boldsymbol{x}}_c$ and $n_c' = n_c$.

For within-class scatter matrix $\boldsymbol{Sw}$, if $\boldsymbol{y}$ is a new class sample, which means k is the $(M+1)$th class, then the updated within-class scatter matrix does not change:

$$\boldsymbol{Sw}' = \sum_{c=1}^{M} \boldsymbol{\Sigma}_c + \boldsymbol{\Sigma}_k = \sum_{c=1}^{M+1} \boldsymbol{\Sigma}_c = \sum_{c=1}^{M} \boldsymbol{\Sigma}_c = \boldsymbol{Sw}. \tag{7}$$

Else, if $1 \le c \le M$, then the updated $\boldsymbol{Sw}$ matrix is (see *Proof A* in the Appendix)

$$\boldsymbol{Sw}' = \sum_{c=1, c\neq k}^{M} \boldsymbol{\Sigma}_c + \boldsymbol{\Sigma}_k' \tag{8}$$

$$\boldsymbol{\Sigma}_k' = \boldsymbol{\Sigma}_k + \frac{n_k}{n_k+1}(\boldsymbol{y}-\bar{\boldsymbol{x}}_k)(\boldsymbol{y}-\bar{\boldsymbol{x}}_k)^T. \tag{9}$$

B. Chunk Incremental LDA

Suppose that in another case, the new samples are acquired in a chunk way $Y_0, Y_1, \ldots$ to infinity. As shown in Fig. 1, at each time point t, a chunk of samples instead of just one sample $Y = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_L\}$ are acquired, where L is the number of samples in one chunk, which is a random positive integer, and $L \geq 1$.

In this sense, we need to derive a solution to the following problem. Let X and Y be two sets observations, where X is the presented observation set, and Y is a set of new observations. Let their discriminant eigenspace models be $\Omega = (\boldsymbol{Sw}, \boldsymbol{Sb}, \bar{\boldsymbol{x}}, N)$ and $\Psi = (\boldsymbol{Sw}_y, \boldsymbol{Sb}_y, \bar{\boldsymbol{y}}, L)$, respectively. As in IPCA [4], the above updating problem here is to compute the new fisherspace model $\Phi = (\boldsymbol{Sw}', \boldsymbol{Sb}', \bar{\boldsymbol{x}}', N + L)$ using fisherspace models Ω and Ψ.

Without loss of generality, we can assume that l_c of L new samples belong to class c, and thus, $n'_c = n_c + l_c$, $N + L = \sum_{c=1}^{M} n'_c = \sum_{c=1}^{M} (n_c + l_c)$ and $\bar{\boldsymbol{x}}'_c = (1/(n_c + l_c))(n_c \bar{\boldsymbol{x}}_c + l_c \bar{\boldsymbol{y}}_c)$, where $\bar{\boldsymbol{y}}_c$ is the mean of new samples in class c.

The updated mean is

$$\bar{\boldsymbol{x}}' = \frac{N\bar{\boldsymbol{x}} + L\bar{\boldsymbol{y}}}{N + L} \tag{10}$$

where $\bar{\boldsymbol{y}} = (1/L)\sum_{j=1}^{L} \boldsymbol{y}_j$

The updated $\boldsymbol{Sb}$ matrix is

$$\boldsymbol{Sb}' = \sum_{c=1}^{M} n'_c (\bar{\boldsymbol{x}}'_c - \bar{\boldsymbol{x}}')(\bar{\boldsymbol{x}}'_c - \bar{\boldsymbol{x}}')^T. \tag{11}$$

The updated $\boldsymbol{Sw}$ matrix is (see *Proof B* in the Appendix)

$$\boldsymbol{Sw}' = \sum_{c=1}^{M} \boldsymbol{\Sigma}'_c \tag{12}$$

$$\begin{aligned}\boldsymbol{\Sigma}'_c = {} & \Sigma_c + \frac{n_c l_c^2}{(n_c + l_c)^2}(\boldsymbol{D}_c) + \frac{n_c^2}{(n_c + l_c)^2}(\boldsymbol{E}_c) \\ & + \frac{l_c(l_c + 2n_c)}{(n_c + l_c)^2}(\boldsymbol{F}_c)\end{aligned} \tag{13}$$

where the second term $\boldsymbol{D_c}$ is the scatter matrix of the new sample class mean vector $\bar{\mathbf{y}}_i$ around the mean vector $\bar{\boldsymbol{x}}_c$ of class c.

$$\boldsymbol{D_c} = (\bar{\boldsymbol{y}}_c - \bar{\boldsymbol{x}}_c)(\bar{\boldsymbol{y}}_c - \bar{\boldsymbol{x}}_c)^T. \tag{14}$$

The third term $\boldsymbol{E_c}$ is the scatter matrix of the new sample mean vector $\bar{\mathbf{y}}_i$ around the mean vector $\bar{\boldsymbol{x}}_c$ of class c

$$\boldsymbol{E_c} = \sum_{j=1}^{l_c} (\boldsymbol{y}_{cj} - \bar{\boldsymbol{x}}_c)(\boldsymbol{y}_{cj} - \bar{\boldsymbol{x}}_c)^T. \tag{15}$$

The fourth term $\boldsymbol{F_c}$ is a within-class scatter matrix of the new samples

$$\boldsymbol{F_c} = \sum_{j=1}^{l_c} (\bar{\boldsymbol{y}}_{cj} - \bar{\boldsymbol{y}}_c)(\bar{\boldsymbol{y}}_{cj} - \bar{\boldsymbol{y}}_c)^T. \tag{16}$$

TABLE I
EVALUATED UCI DATASETS

name	input dim.	class	train. data	test data
Iris	4	3	150	-
Liver-disorder	6	2	345	-
Vehicle	18	4	846	-
Glass	10	7	214	-
Wine	13	3	178	-
Segmentation	19	7	210	2100
Vowel	10	11	528	462
Sonar	60	2	208	-

In addition, we can also assume l_{M+1} of L new samples belong to class $M + 1$ without loss of generality. In this case, the updated between-class matrix (5) can be rewritten as

$$\begin{aligned}\boldsymbol{Sb}' &= \sum_{c=1}^{M} n'_c(\bar{\boldsymbol{x}}_c - \bar{\boldsymbol{x}}')(\bar{\boldsymbol{x}}_c - \bar{\boldsymbol{x}}')^T + l_{M+1}(\bar{\boldsymbol{y}} - \bar{\boldsymbol{x}}')(\bar{\boldsymbol{y}} - \bar{\boldsymbol{x}}')^T \\ &= \sum_{c=1}^{M+1} n'_c(\bar{\boldsymbol{x}}_c - \bar{\boldsymbol{x}}')(\bar{\boldsymbol{x}}_c - \bar{\boldsymbol{x}}')^T\end{aligned} \tag{17}$$

where n'_c is the number of samples in class c after Y is presented. If $c = M + 1$, then $n'_c = l_{M+1}$, else $n'_c = n_c + l_c$.

In addition, the updated within-class matrix (7) can be rewritten as

$$\boldsymbol{Sw}' = \sum_{c=1}^{M} \boldsymbol{\Sigma}_c + \boldsymbol{\Sigma}_{M+1} = \sum_{c=1}^{M+1} \boldsymbol{\Sigma}_c{}' \tag{18}$$

where $\boldsymbol{\Sigma}_{M+1} = \sum_{y \in \{y_c\}} (\boldsymbol{y} - \bar{\boldsymbol{y}}_c)(\boldsymbol{y} - \bar{\boldsymbol{y}}_c)^T$.

III. RESULTS AND DISCUSSIONS

In this section, we have examined the efficiency and accuracy of our incremental LDA methods compared to incremental PCA (IPCA) [1], [4] and batch LDA method for the classification of datastreams. We conducted a thorough experimental and performance study using datasets with few classes and small-dimensional features as well as datasets with many classes and large-dimensional features. Particularly, we are interested in evaluating 1) the discriminability of ILDA and 2) the execution time and memory usages of ILDA computation with the increasing volumn of data addition. For all experiments, we used Matlab code running on a PC with Intel Pentium 4 2.8-GHZ CPU and 512-Mb RAM.

A. Experimental Setup

For every test, first, we construct an initial feature space (eigenspace) using 10% of the total samples, in which at least two classes data are ensured to be included according to the definition of (2). These training samples are used for calculating eigenvectors and eigenvalues through conventional LDA/PCA. The remaining training data are enumerated into a number of chunks without any consideration about the chunk size and number of classes in each chunk.

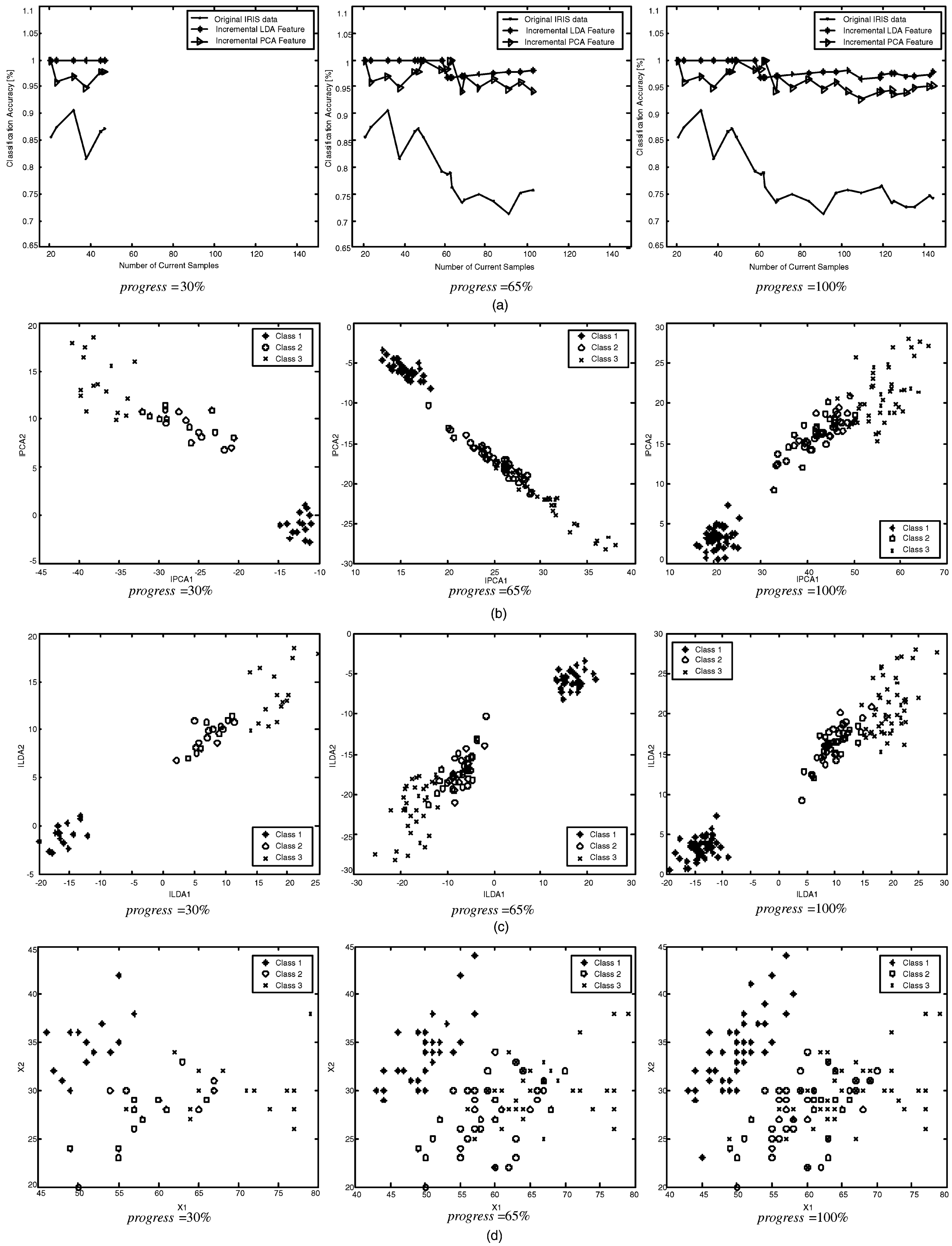

Fig. 2. Comparison of ILDA and IPCA learning on (a) the variation of classification accuracy, (b) IPCA projection of Iris data onto the first two eignevectors, (c) ILDA projection of Iris data onto the first discriminant eigenvectors, and (d) the original distribution of Iris data by the first two dimensions.

When incremental learning is carried out, training samples are randomly drawn from the remaining training dataset in the scale of chunk instead of one by one sample. Then, the eigenspace is updated by an ILDA computation introduced in Section II or IPCA computation of [4] with chunks of data.

Finally, to test the efficiency of ILDA for classification, we encode features by projecting data presented so far to the updated LDA discriminant eigenspace and then classify the feature data using KNN classifier($K = 1$). The classification accuracy is evaluated under a Leave-one-out cross-validation policy. For each dataset, while selecting features of ILDA, we rank the eigenvectors by their energy, and select a set of top energy eignevectors. The number of eigenvectors used in ILDA equals to the number of eigenvectors that Batch LDA uses to get the best classification accuracy on the dataset.

Since the events of data arriving in the above incremental learning may not happen at regular time intervals, we use the term *learning stage* instead of the usual time scale. Here, the number of learning stages is equivalent to the number of samples that have been learned by incremental models. In addition, for the convenience of illustration, we also define the percentage of samples presented so far at current stage to measure the *progress* of incremental learning.

B. UCI Datasets

First presented here are experiments with a focus on evaluating the class-discriminant ability of ILDA. The database we used consists of eight standard datasets selected from UCI Machine Learning Repository [9], where each dataset has its features 100% of continuous/integer values and no missing value. The dataset information is summarized in Table I. As can be seen, every dataset has no more than 12 classes and 60-dimension features.

1) Discriminability: Over the above eight UCI datasets, we carried out the incremental learning described in Section III-A to test the discriminablity of ILDA for classification.

As an example, Fig. 2 shows a time course of the above incremental learning over Iris data with three learning stages, whose *progress* of incremental learning are at 30%, 65% and 100%, respectively, in which (a) gives the variation of classification accuracy; (d) is distribution of Iris data by the first two dimensions; and (b) and (c) are IPCA projection and ILDA projection of Iris data onto the first two eigenvectors, respectively.

Here, ILDA is compared with IPCA on the variation of classification accuracy and feature distribution with the original data as a reference. As can be seen in Fig. 2(a), ILDA is leading the classification accuracy at most stages of incremental learning, particularly it achieves a better final classification accuracy than IPCA. This superiority on classification can also clearly be reflected from the discrimination differences between corresponding ILDA projections in Fig. 2(d) and IPCA projections in Fig. 2(c).

2) Similarity: To see the similarity of the two discriminant eigenspaces constructed by ILDA and Batch LDA, the inner products between discriminant eigenvectors obtained by ILDA and LDA are evaluated. For ILDA, the discriminant eigenvectors ILDA are being updated over the learning stages. For Batch LDA, the discriminant eigenvectors are calculated from the whole dataset in a batch mode.

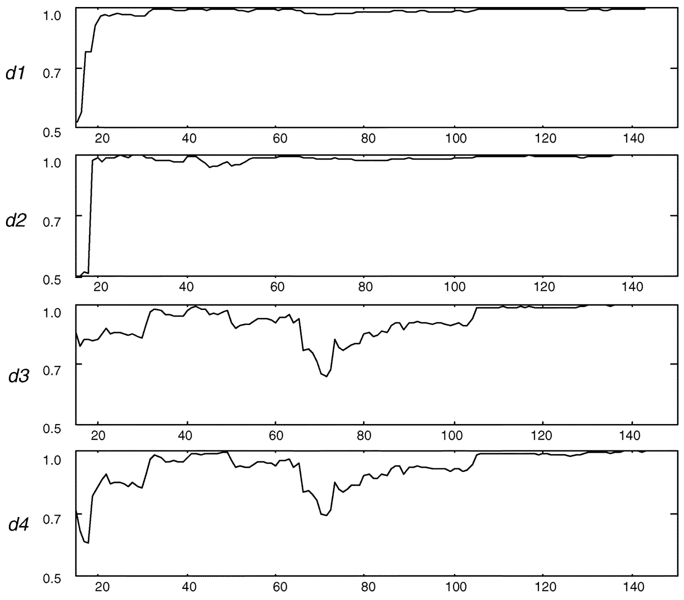

Fig. 3. Typical time courses of inner products $d_1 \sim d_4$ between discriminant eigenvectors obtained by ILDA and batch ILDA as the incremental learning stages proceed.

Fig. 3 shows typical time courses of inner products $d_1 \sim d_4$ between discriminant eigenvectors obtained by ILDA and batch ILDA learning over Iris data. Here, we assume that each eigenvector has an unit length, and the total 4 discriminant eigenvectors of ILDA and LDA are compared correspondingly in Fig. 4.

As seen from Fig. 3, d_1 and d_2 have large values from the beginning of learning stages. Although d_3 and d_4 have some fluctuations during the incremental learning, the values converge to the maximum value at the final stage. Hence, we can observe that ILDA is updating the discriminant eigenspace gradually as the progress of incremental learning grows, and it can finally construct exactly the same discriminant eigenspace as that of Batch LDA.

3) Performances: Table II presents the comparison results of ILDA, Batch LDA, IPCA on the classification at the final incremental learning stage for eight UCL datasets, where the number of Eigenvector (denoted as no. Eig.) specifies the dimension of LDA and PCA eigenfeatures used in classification. As can be seen, when using the same number of eigenvectors, the classification accuracies due to ILDA and LDA are very similar. In most cases, they are exactly the same, but the accuracy from IPCA are obviously lower than that of ILDA. It suggests that discriminant ability of ILDA is equivalent to that of LDA, and is better than IPCA.

Table III presents a comparison between ILDA and IPCA with a focus on the classification stability during the whole course of incremental learning for the above eight datasets. For each dataset, we carried out the incremental learning described in 3.1 for 100 runs. We calculated 1) the course of accuracy (denoted by Course Acc.) by averaging the classification accuracies at every evaluation stage for 100 runs (data are provided and trained in the scale of chunk and not every sample is counted as a stage for evaluation if the chunk-size is greater

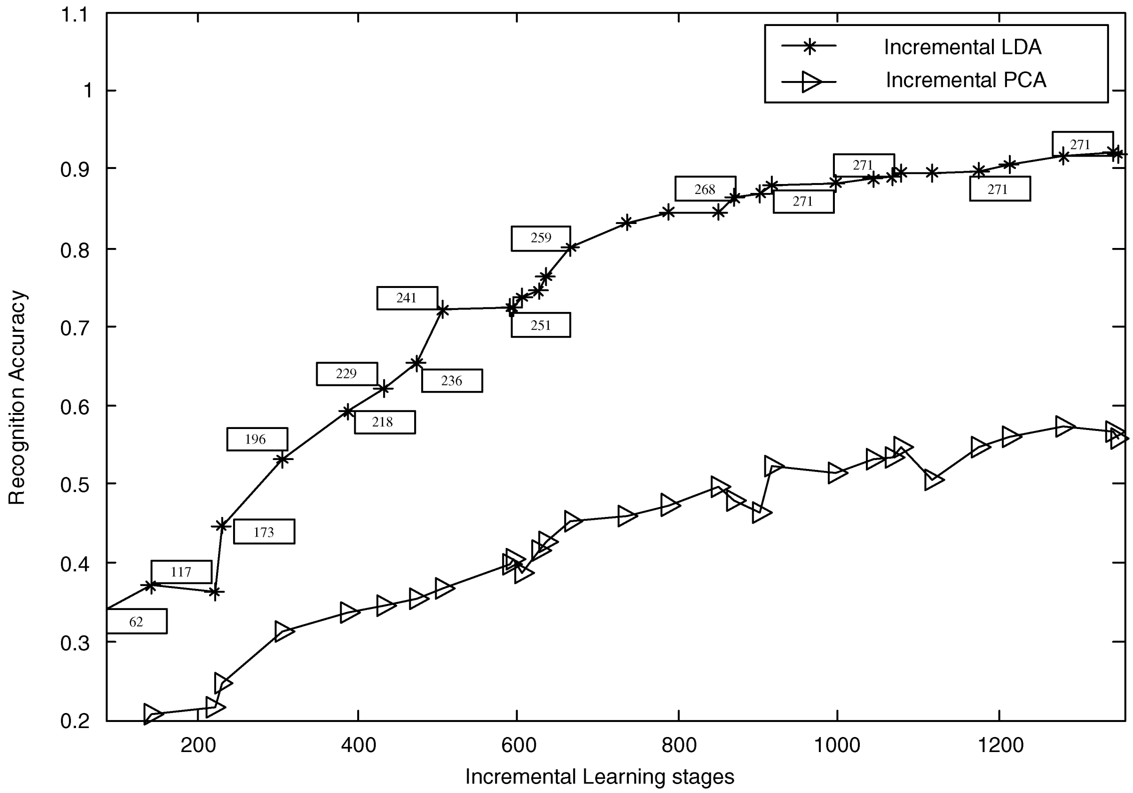

Fig. 4. Variation of ILDA versus IPCA on recognition accuracy as new classes of face images are being added in the form of random chunks. The framed digits around the ILDA curve identify the number of presented classes by the current learning stage.

TABLE II
COMPARISON RESULTS OF ILDA, BATCH LDA, IPCA ON THE CLASSIFICATION AT THE FINAL INCREMENTAL LEARNING STAGE FOR 8 UCL DATASETS

	No. Eig.	Incremental LDA	Batch LDA	Incremental PCA
Name		Acc. %	Acc. %	Acc. %
Iris	2	98.0	98.0	93.3
Liver-disorder	3	62.6	62.6	58.8
Vehicle	9	75.4	75.4	57.8
Glass	6	67.7	67.3	51.8
Wine	7	96.6	96.6	87.6
Segmentation	6	83.9	80.5	81.4
Vowel	10	60.9	59.8	57.9
Sonar	6	81.2	81.2	73.5

TABLE III
COMPARISON BETWEEN ILDA AND IPCA IN TERMS OF CLASSIFICATION ACCURACY AND VARIANCE IN THE WHOLE COURSE INCREMENTAL LEARNING OVER EIGHT UCL DATASETS

	No. Eig.	Incremental LDA		Incremental PCA	
Name		course Acc. %	Var.	course Acc. %	Var.
Iris	2	96.0	3.9e-04	90.2	1.4e-03
Liver-disorder	3	61.7	1.8e-03	50.5	2.3e-03
Vehicle	9	76.1	5.9e-04	64.8	1.7e-03
Glass	6	64.9	4.8e-03	61.8	4.4e-03
Wine	7	97.8	4.7e-04	80.3	3.5e-03
Segmentation	6	78.9	1.7e-04	74.4	4.9e-02
Vowel	10	53.2	1.0e-02	52.7	1.7e-02
Sonar	6	92.1	5.2e-03	70.2	5.0e-03

than one) and 2) the course of variance (denoted by Course Var.) by conducting variance computing on the classification accuracies at every evaluation stage for 100 runs.

As can be found in Table III, the course of accuracy due to ILDA is better than IPCA on average, and so is the course of variance. It indicates that the proposed ILDA is delivering a set of features optimal for classification during the whole course of incremental learning, and ILDA is superior to IPCA either for classification accuracy or classification stability.

C. Face Database

Next, we will show the performances of ILDA on a database with a large number of classes and high dimension features. Here, we used a benchmark MPEG-7 face database, which consists of 1355 face images of 271 persons (five different face images per person are taken), where each image has the size of 56×46. The images have been selected from AR(Purdue), AT&T, Yale, UMIST, University of Berne, and some face images obtained from MPEG-7 news videos [5], [6], [19].

1) Discriminability: Thus, we conducted the incremental learning described in Section III-A on a database having 271 classes and 2576 (56×46) dimension features, where the first 30 eigenfeatures of ILDA/IPCA are taken to perform K-NN leave-one-out classification. We tested the discriminabilty of ILDA, specifically when bursts of new classes are presented at different times. In addition, we evaluated the execution time and memory costs of ILDA with the increase of new data addition.

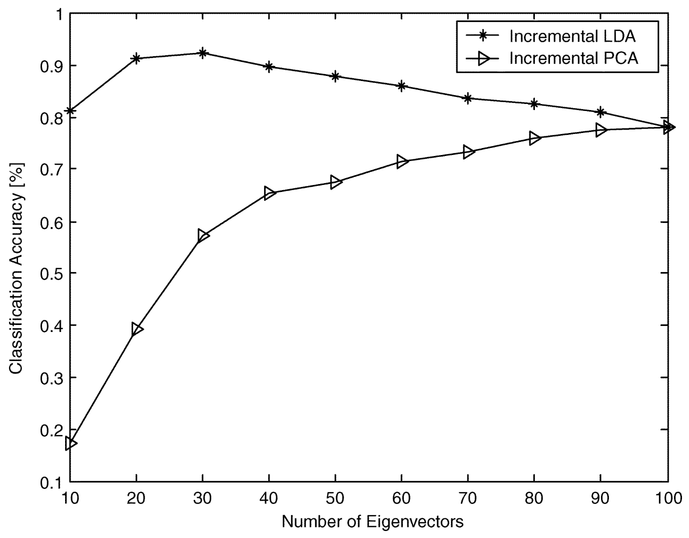

Fig. 5. Variation of ILDA versus IPCA on the final classification accuracy under different number of eigenfeature dimension.

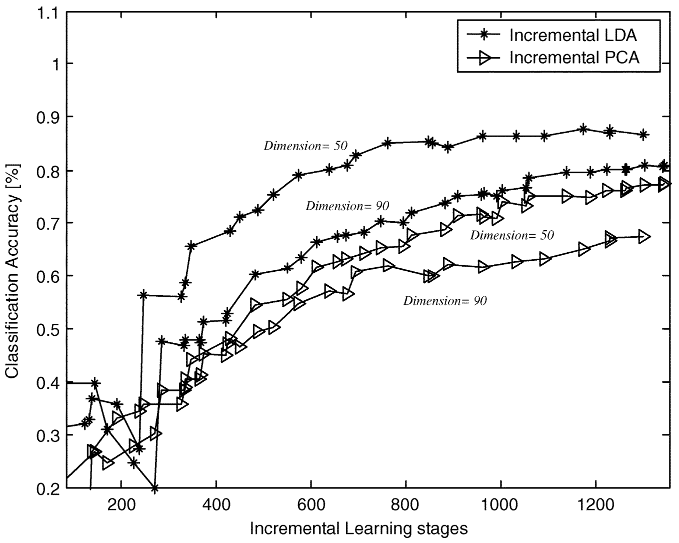

Fig. 6. Same incremental learning of ILDA and IPCA as in Fig. 4, but the used eigenfeature dimensions are 50 and 90.

Fig. 4 illustrates the procedure of ILDA running on the above face database. As the incremental learning proceeds, new data arebeing added to the initial discriminant eigenspace constructed on 10% of total images in a random way. As can be seen in the figure, hundreds of samples are presented in one chunk for some learning stages, but very few samples appear at other stages. As a result of such random chunk-sizes, small and large bursts of new classes data are coming in randomly at different stages. From the numbers around ILDA curve, we can see that the maximum number of new classes appeared in one chunk is 56, and the minimum is 0, which means no new class data appear in the current chunk. In spite of all this unexpectedness of the coming data, the performance of ILDA is still increasing steadily with the increasing of new data addition, and it is also much better than IPCA on the classification accuracy.

As the dimension of eigenfeatures of Batch LDA may strongly affect the classification performance, we have also tested ILDA using a different number of eigenfeatures. Our finding is, when the dimension of eigenfeature is over 30, as shown in Figs. 5 and 6, the discriminability of ILDA in terms of the final classification accuracy decreases, and IPCA outperforming ILDA happens at some initial incremental learning stages. It turns out that the few top energy ILDA discriminant eigenvectors contain almost all the classification information embedded in the original space, and ILDA is not guaranteed to perform better than IPCA in some cases.

2) Execution Time: To evaluate the efficiency of ILDA, we set the chunk-size as a constant value, and measured the time taken by Chunk ILDA to finish an incremental learning on a complete dataset. Note that chunk-size is predetermined by environment (i.e., given task). To discover the relationship between the ILDA execution time and the average chunk-size given in data streams, we also used different chunk-sizes in ILDA.

Over a dataset of 500 face images (including 100 persons, each person presented by five images), the total time taken to

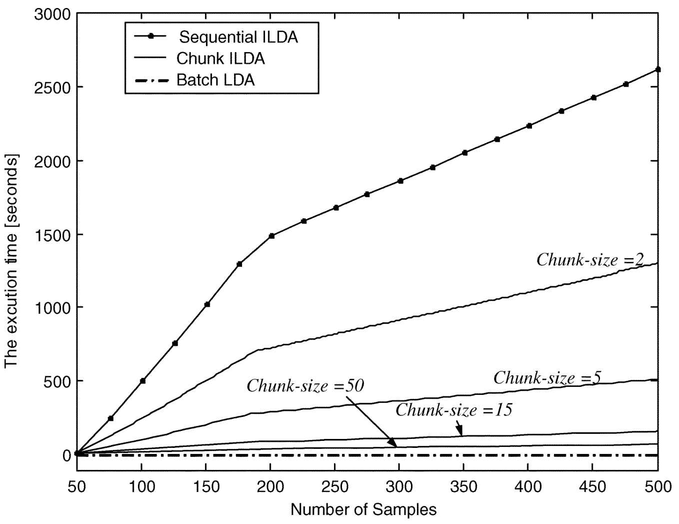

Fig. 7. Variation of time taken by Chunk ILDA and Sequencial ILDA to update the discriminant eigenspace model with the increasing of new data addition.

compute a discriminant eigenspace model using the batch LDA method and our ILDA models is compared in Fig. 7. As can be seen, the time taken by Chunk ILDA with different chunk-sizes is between the sequential ILDA time and the batch LDA time. Actually, the time of ILDA is very much determined by the chunk-size on average in the whole course of incremental learning. As shown in Fig. 8, when the chunk-size is set as 500, which is the size of the whole dataset, then Chunk ILDA uses 3.9 seconds, the same time as batch LDA uses, to finish the learning task. Whereas, if the chunk-size is set as 1, Chunk ILDA takes 2335.8 s, which is the same amount of seconds that sequential ILDA uses to finish the learning task. Chunk ILDA is efficient for dealing with large and fast data streams because whatever a chunk-size of data are in the data stream, Chunk ILDA can process the data in one step. A slightly larger chunk-size results in a big execution time reduction. As seen in Fig. 8, the time taken by Chunk ILDA is being reduced from 2335.8 to 51.6 s, as we increase the chunk-size from 1 to 50.

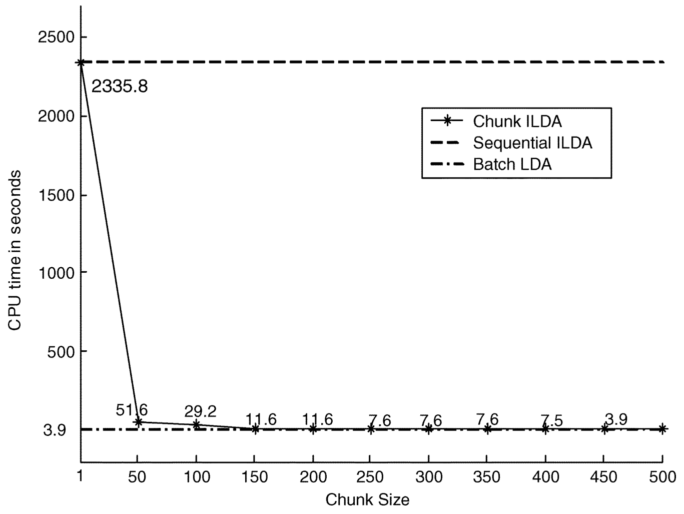

Fig. 8. To compute a complete discriminant eigenspace model for a database of 500 face images, time taken by Chunk ILDA under different chunk-sizes is compared to the time expenses of sequential ILDA and batch LDA.

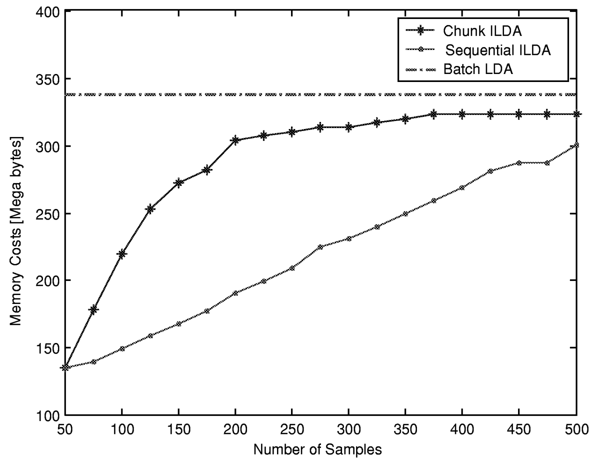

Fig. 9. Variation of Chunk ILDA and sequential ILDA on memory cost with the increase of data addition.

We notice that Fig. 7 shows a distinct change in slope at the stage of about 200 samples. The change of the slope responds to two cases ILDA computing happened during the whole course of incremental learning. At early learning stages, new classes are continuously increasing with chunks of data being presented, and ILDA constructs knowledge basis for each of new class by (17) and (18), thus the execution time of ILDA increases rapidly. After a certain learning stages (in the case of Fig. 7, it is about 200 samples), because very few new classes are presented from now, ILDA is mainly updating the knowledge basis of the existing classes by (11) and (12). The reason why the increasing of ILDA execution time is slow down (which makes the change of the slop in Fig. 7) is due to the computing of (11) and (12) is less time costed than that of (17) and (18).

3) Memory Usage: Along with the above execution time measuring, we also estimated the amount of memory used by ILDA and Batch LDA to complete the eigenspace model learning over the same 500 face images (including 100 classes/persons, each class 5 images). We used Batch LDA's memory usage to compare with the memory usage of Chunk ILDA and Sequential ILDA, as every 50 new samples are presented.

As can be seen in Fig. 9, both Sequential ILDA and Chunk ILDA are producing a memory accumulation with the increase of new data addition. This happens, because the proposed ILDA is based on the updating of within-class scatter matrix $\boldsymbol{S_w}$ and between-class scatter matrix $\boldsymbol{S_b}$. To update $\boldsymbol{S_w}$ according to (8) and (18), it needs to maintain a within-class covariance matrix $\boldsymbol{\Sigma}_c$ in memory for every new class.

It is not surprising that Sequential ILDA has a slower memory usage growth than Chunk ILDA, because Sequential ILDA is always learning on just one sample, while Chunk ILDA is learning data in the scale of chunk. When the incremental learning is finished, the memory usage of Chunk ILDA is converging to a constant value that is about 40 M bytes smaller than the total memory usages of Batch ILDA. For datasets with a small number of classes and large number of samples in each class (e.g., a dataset with five classes and 100 samples per class), the memory usages for storing within-class covariance matrices will be reduced largely. Thus, for data streams of a smaller number of classes, the proposed ILDA would have a much greater gain in memory usage.

IV. Conclusions

Like IPCA [1], [4], the essence of Incremental linear discriminant analysis should be the incremental updating of the eigen-decomposition. As an alternative solution of ILDA, this paper proposed the method of incremental LDA deriving discriminant eigen-space in a streaming environment without updating the eigen-decomposition. The method is in terms of the incremental updating of the between-class and within-class scatter matrices and thus is able to deal with volumes of additional data.

We proposed ILDA with two computational approaches: Sequencial ILDA and Chunk ILDA. Theoretically, Sequential ILDA eventually is a special case of Chunk ILDA. This can be proofed by substituting $L = l_c = 1$ in (10)–(18). They were introduced separately in this paper, because they actually represent two categories of incremental learning: incremental learning by updating old knowledge bases and incremental learning by merging old knowledge bases with a new one, respectively. In addition, they might have different practical applications due to their differences in data streaming formats, computational speed and memory usages.

To test the computational properties of ILDA, we used ILDA in a real-life streaming environment in which data are coming in the form of random chunks. We conducted various tests on datasets with either a small number of classes and small-dimensional features or a large number of classes and large-dimensional features. We can summarize the properties of ILDA as follows: 1) ILDA has an equivalent power to batch LDA in terms of discriminability; 2) ILDA, in particular, chunk ILDA has very high memory and speed efficiency due to its just one-pass computation and chunk data processing, and the efficiency of ILDA is determined by the average chunk-size taken in the whole course of incremental learning; 3) ILDA is effective for handling bursts of new classes coming in at different times; 4) as compared with IPCA, ILDA is usually, but not guaranteed [10], to be superior to IPCA for classification. As shown in the test of Fig. 6, the performance of IPCA outperforming ILDA may happen at some early incremental learning stages, when the LDA knowledge base is not strong due to too few samples in a class presented at those stages.

One limitation of the proposed ILDA is that when the chunk-size is too large, the memory cost of ILDA become expensive, but this problem can be solved by partitioning a big chunk to several smaller chunks and performing ILDA on those smaller chunks data, respectively, or just using the method in [8] to avoid the covariance matrix computation while performing LDA. Nevertheless, it is very distinct that the above optimal properties have determined ILDA as an useful method when we conduct classification on fast and large data streams.

APPENDIX

Proof A

Given a new sample $\boldsymbol{y}$ in the kth class, where $k \in [1, M]$, we then have the equation shown at the bottom of the page. Since $\sum_{\boldsymbol{x}\in\{\boldsymbol{x}_k\}} (\boldsymbol{x} - \bar{\boldsymbol{x}}_k) = 0$, the updated covariance matrix thus equals

$$\begin{aligned}\boldsymbol{\Sigma}'_k &= \boldsymbol{\Sigma}_k + \frac{n_k^2 + n_k}{(n_k+1)^2}(\boldsymbol{y} - \bar{\boldsymbol{x}}_k)(\boldsymbol{y} - \bar{\boldsymbol{x}}_k)^T \\ &= \boldsymbol{\Sigma}_k + \frac{n_k}{n_k+1}(\boldsymbol{y} - \bar{\boldsymbol{x}}_k)(\boldsymbol{y} - \bar{\boldsymbol{x}}_k)^T.\end{aligned}$$

Proof B

Given l_c new samples $\boldsymbol{y}$ belong to class c, according to definition, the updated covariance matrix is

$$\boldsymbol{\Sigma}'_c = \sum_{i=1} n_c(\boldsymbol{x}_i - \bar{\boldsymbol{x}}')(\boldsymbol{x}_i - \bar{\boldsymbol{x}}')^T + \sum_{i=1} l_c(\boldsymbol{y}_i - \bar{\boldsymbol{x}}')(\boldsymbol{y}_i - \bar{\boldsymbol{x}}')^T. \tag{19}$$

There is two terms in above equation. For the first term

$$\begin{aligned}&\sum_{i=1} n_c(\boldsymbol{x}_i - \bar{\boldsymbol{x}}')(\boldsymbol{x}_i - \bar{\boldsymbol{x}}')^T \\ &= \sum_{i=1}^{n_c}\left[(\boldsymbol{x}_i - \bar{\boldsymbol{x}}) - \frac{l_c}{n_c+l_c}(\bar{\boldsymbol{y}} - \bar{\boldsymbol{x}})\right]\left[(\boldsymbol{x}_i - \bar{\boldsymbol{x}}) - \frac{l_c}{n_c+l_c}(\bar{\boldsymbol{y}} - \bar{\boldsymbol{x}})\right]^T \\ &= \sum_{i=1}^{n_c}(\boldsymbol{x}_i - \bar{\boldsymbol{x}})(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^T - \frac{l_c}{n_c+l_c}\sum_{i=1}^{n_c}(\boldsymbol{x}_i - \bar{\boldsymbol{x}})(\bar{\boldsymbol{y}} - \bar{\boldsymbol{x}})^T \\ &\quad - \frac{l_c}{n_c+l_c}\sum_{i=1}^{n_c}(\bar{\boldsymbol{y}} - \bar{\boldsymbol{x}})(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^T + \frac{l_c^2}{(n_c+l_c)^2}\sum_{i=1}^{n_c}(\bar{\boldsymbol{y}} - \bar{\boldsymbol{x}})(\bar{\boldsymbol{y}} - \bar{\boldsymbol{x}})^T \\ &= \sum_{i=1}^{n_c}(\boldsymbol{x}_i - \bar{\boldsymbol{x}})(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^T + \frac{l_c^2}{(n_c+l_c)^2}\sum_{i=1}^{n_c}(\bar{\boldsymbol{y}} - \bar{\boldsymbol{x}})(\bar{\boldsymbol{y}} - \bar{\boldsymbol{x}})^T.\end{aligned}$$

For the second term

$$\begin{aligned}&\sum_{i=1} l_c(\boldsymbol{y}_i - \bar{\boldsymbol{x}}')(\boldsymbol{y}_i - \bar{\boldsymbol{x}}')^T = \frac{1}{(n_c+l_c)^2} \\ &\quad \cdot \sum_{i=1} l_c[n_c(\boldsymbol{y}_i - \bar{\boldsymbol{x}}) + l_c(\boldsymbol{y}_i - \bar{\boldsymbol{y}})]\,[n_c(\boldsymbol{y}_i - \bar{\boldsymbol{x}}) + l_c(\boldsymbol{y}_i - \bar{\boldsymbol{y}})]^T \\ &= \frac{1}{(n_c+l_c)^2}\Bigg[n_c^2 \sum_{i=1} l_c(\boldsymbol{y}_i - \bar{\boldsymbol{x}})(\boldsymbol{y}_i - \bar{\boldsymbol{x}})^T \\ &\quad + n_c l_c \sum_{i=1}^{l_c}(\boldsymbol{y}_i - \bar{\boldsymbol{x}})(\boldsymbol{y}_i - \bar{\boldsymbol{y}})^T \\ &\quad + n_c l_c \sum_{i=1}^{l_c}(\boldsymbol{y}_i - \bar{\boldsymbol{y}})(\boldsymbol{y}_i - \bar{\boldsymbol{x}})^T + l_c^2 \sum_{i=1}^{l_c}(\boldsymbol{y}_i - \bar{\boldsymbol{y}})(\boldsymbol{y}_i - \bar{\boldsymbol{y}})^T\Bigg] \\ &= \frac{1}{(n_c+l_c)^2}\Bigg[n_c^2 \sum_{i=1} l_c(\boldsymbol{y}_i - \bar{\boldsymbol{x}})(\boldsymbol{y}_i - \bar{\boldsymbol{x}})^T \\ &\quad + n_c l_c \sum_{i=1}^{l_c}(\boldsymbol{y}_i\boldsymbol{y}_i^T - \boldsymbol{y}_i\bar{\boldsymbol{y}}^T - \bar{\boldsymbol{x}}\boldsymbol{y}_i^T + \bar{\boldsymbol{x}}\bar{\boldsymbol{y}}^T) \\ &\quad + n_c l_c \sum_{i=1}^{l_c}(\boldsymbol{y}_i\boldsymbol{y}_i^T - \boldsymbol{y}_i\bar{\boldsymbol{x}}^T - \bar{\boldsymbol{y}}\boldsymbol{y}_i + \bar{\boldsymbol{y}}\bar{\boldsymbol{x}}^T) \\ &\quad + l_c^2 \sum_{i=1}^{l_c}(\boldsymbol{y}_i\boldsymbol{y}_i^T - \boldsymbol{y}_i\bar{\boldsymbol{y}}^T - \bar{\boldsymbol{y}}\boldsymbol{y}_i^T + \bar{\boldsymbol{y}}\bar{\boldsymbol{y}}^T)\Bigg].\end{aligned}$$

Since, in the above equation

$$\begin{aligned}\sum_{i=0}^{l_c}(\bar{\boldsymbol{x}}\bar{\boldsymbol{y}}^T - \bar{\boldsymbol{x}}\boldsymbol{y}_i^T) &= \sum_{i=0}^{l_c}(\bar{\boldsymbol{y}}\bar{\boldsymbol{x}}^T - \bar{\boldsymbol{y}}\boldsymbol{x}_i^T) \\ &= \sum_{i=0}^{l_c}(\bar{\boldsymbol{y}}\bar{\boldsymbol{y}}^T - \boldsymbol{y}_i\bar{\boldsymbol{y}}^T) = 0.\end{aligned}$$

Thus, the second term equation can be simplified as

$$\begin{aligned}&\sum_{i=1} l_c(\boldsymbol{y}_i - \bar{\boldsymbol{x}}')(\boldsymbol{y}_i - \bar{\boldsymbol{x}}')^T \\ &= \frac{1}{(n_c+l_c)^2}\Bigg[n_c^2 \sum_{i=1} l_c(\boldsymbol{y}_i - \bar{\boldsymbol{x}})(\boldsymbol{y}_i - \bar{\boldsymbol{x}})^T \\ &\quad + (2n_c l_c + l_c^2)\sum_{i=1}^{l_c}(\boldsymbol{y}_i\boldsymbol{y}_i^T - \boldsymbol{y}_i\bar{\boldsymbol{y}}^T)\Bigg] \\ &= \frac{1}{(n_c+l_c)^2}\Bigg[n_c^2 \sum_{i=1} l_c(\boldsymbol{y}_i - \bar{\boldsymbol{x}})(\boldsymbol{y}_i - \bar{\boldsymbol{x}})^T \\ &\quad + l_c(l_c + 2n_c)\sum_{i=1}^{l_c}(\boldsymbol{y}_i - \bar{\boldsymbol{y}})(\boldsymbol{y}_i - \bar{\boldsymbol{y}})^T\Bigg].\end{aligned}$$

Now, substitute the above two simplified equation into the first definition equation, and combine terms with the same

$$\begin{aligned}\boldsymbol{\Sigma}'_k &= \sum_{\boldsymbol{x}\in\{\boldsymbol{x}_k, y\}} (\boldsymbol{x} - \bar{\boldsymbol{x}}'_k)(\boldsymbol{x} - \bar{\boldsymbol{x}}'_k)^T = \sum_{\boldsymbol{x}\in\{\boldsymbol{x}_k\}} (\boldsymbol{x} - \bar{\boldsymbol{x}}'_k)(\boldsymbol{x} - \bar{\boldsymbol{x}}'_k)^T + (\boldsymbol{y} - \bar{\boldsymbol{x}}'_k)(\boldsymbol{y} - \bar{\boldsymbol{x}}'_k)^T \\ &= \sum_{\boldsymbol{x}\in\{\boldsymbol{x}_k\}} \left(\boldsymbol{x} - \bar{\boldsymbol{x}}_k - \frac{\boldsymbol{y} - \bar{\boldsymbol{x}}_k}{n_k+1}\right)\left(\boldsymbol{x} - \bar{\boldsymbol{x}}_k - \frac{\boldsymbol{y} - \bar{\boldsymbol{x}}_k}{n_k+1}\right)^T + \left(\boldsymbol{y} - \bar{\boldsymbol{x}}_k - \frac{\boldsymbol{y} - \bar{\boldsymbol{x}}_k}{n_k+1}\right)\left(\boldsymbol{y} - \bar{\boldsymbol{x}}_k - \frac{\boldsymbol{y} - \bar{\boldsymbol{x}}_k}{n_k+1}\right)^T \\ &= \boldsymbol{\Sigma}_k + \sum_{\boldsymbol{x}\in\{\boldsymbol{x}_k\}}\left[\frac{(\boldsymbol{y} - \bar{\boldsymbol{x}}_k)(\boldsymbol{y} - \bar{\boldsymbol{x}}_k)^T}{(n_k+1)^2} - (\boldsymbol{x} - \bar{\boldsymbol{x}}_k)\frac{(\boldsymbol{y} - \bar{\boldsymbol{x}}_k)^T}{n_k+1} - \frac{(\boldsymbol{y} - \bar{\boldsymbol{x}}_k)}{n_k+1}(\boldsymbol{x} - \bar{\boldsymbol{x}}_k)^T\right] + \frac{n_k^2}{(n_k+1)^2}(\boldsymbol{y} - \bar{\boldsymbol{x}}_k)(\boldsymbol{y} - \bar{\boldsymbol{x}}_k)^T.\end{aligned}$$

scatter matrix; then, we can obtain the updated new covariance matrix in the form of four terms as

$$\boldsymbol{\Sigma}_c' = \frac{1}{n_c + l_c}\left[\sum_{i=1}^{n_c}(\boldsymbol{x}_i - \bar{\boldsymbol{x}})(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^T + \frac{n_c l_c^2}{(n_c+l_c)^2}(\bar{\boldsymbol{y}} - \bar{\boldsymbol{x}})(\bar{\boldsymbol{y}} - \bar{\boldsymbol{x}})^T + \frac{n_c^2}{(n_c+l_c)^2}\sum_{i=1}^{l_c}(\boldsymbol{y}_i - \bar{\boldsymbol{x}})(\boldsymbol{y}_i - \bar{\boldsymbol{x}})^T + \frac{l_c(l_c+2n_c)}{(n_c+l_c)^2}\sum_{i=1}^{l_c}(\boldsymbol{y}_i - \bar{\boldsymbol{y}})(\boldsymbol{y}_i - \bar{\boldsymbol{y}})^T\right].$$

ACKNOWLEDGMENT

The authors express their special thanks to Professor S. Abe of Kobe University for his encouragement and supports on this collaboration research.

REFERENCES

[1] P. Gall and R. Martin, "Incremental eigenanalysis for classification," in *Proc. Brit. Machine Vision Conf.*, vol. 1, pp. 286–295.

[2] S. Chandrasekaran, B. S. Manjunath, Y. F. Wang, J. Winkler, and H. Zhang, "An eigenspace update algorithm for image analysis," *Graphical Models Image Process.*, vol. 59, no. 5, pp. 321–332, Sep. 1997.

[3] R. D. DeGroat and R. Roberts, "Efficient, numericlly stablized rank-one eigenstructure updating," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 38, no. 2, pp. 301–316, Feb. 1990.

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

[5] S. Pang, S. Ozawa, and N. Kasabov, "One-pass incremental membership authentication by face classification," in *Proc. ICBA*: Springer, 2004, vol. 3072, Lecture Notes Comput. Sci., pp. 155–161.

[6] S. Pang, D. Kim, and S. Y. Bang, "Membership authentication in the dynamic group by face classification using SVM ensemble," *Pattern Recognition Lett.*, vol. 24, pp. 215–225, 2003.

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

[8] J. Weng, Y. Zhang, and W.-S. Hwang, "Candid covariance-free incremental principal component analysis," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 25, no. 8, pp. 1034–1040, Aug. 2003.

[9] [Online]. Available: http://www.ics.uci.edu/mlearn/MLRepository.html

[10] A. M. Martinez and A. C. Kak, "PCA versus LDA," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 23, no. 2, pp. 228–233, Feb. 2001.

[11] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, "Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection," *IEEE Pattern Anal. Machine Intell.*, vol. 19, no. 7, pp. 711–720, Jul. 1997.

[12] J. Ye *et al.*, "A new optimization criterion for generalized discriminant analysis on undersampled problems," in *Proc. Third IEEE Int. Conf. Data Mining*, Nov. 2003, pp. 419–426.

[13] J. Lu, K. N. Plataniotis, and A. N. Venetsanopoulos, "Face recognition using LDA-based algorithms," *IEEE Trans. Neural Networks*, vol. 14, no. 1, pp. 195–200, Jan. 2003.

[14] H.-C. Kim, D. Kim, and S. Y. Bang, "Face recogntion using LDA mixture model," in *Proc. 16th Int. Conf. Pattern Recognition*, vol. 2, Aug. 2002, pp. 486–489.

[15] L. Chen and H. Man, "Discriminant analysis of stochastic models and its application to face recognition," in *Proc. IEEE Int. Workshop Anal. Modeling Faces Gesture*, Oct. 2003, pp. 5–10.

[16] D. L. Swets and J. J. Weng, "Using discriminant eigenfeatures for image retrieval," *IEEE Pattern Anal. Machine Intell.*, vol. 18, no. 8, pp. 831–836, Aug. 1996.

[17] Y. Koren and L. Carmel, "Visulization of labeled data using linear transformation," in *Proc. IEEE Symp. Inf. Visualization*, Oct. 2003, pp. 121–128.

[18] C. C. Aggarwal *et al.*, "On demand classification of data streams," in *Proc. ACM SIGKDD Int. Conf. Knowledge Discovery Data Mining*, Aug. 2004, pp. 503–508.

[19] M. Kim, D. Kim, S. Bang, and S. Lee, "Face Recognition Descriptor using the Embedded HMM with the 2nd-Order Block-Specific Eigenvectors,", Jeju, Korea, ISO/IEC JTC1/SC21/WG11/M7997, 2002.

Shaoning Pang (M'00) received the B.S. degree in physics, the M.S. degree in electronic engineering, and the Ph.D. degree in computer science.

From 2001 to 2003, he was a research associate with Pohang University of Science and Technology (POSTECH), Pohang, Gyungbuk, Korea. Currently, he is a senior research fellow with the Knowledge Engineering and Discovery Research institute, Auckland University of Technology, Auckland, New Zealand. His research interests include support vector machines, brain-like computing, incremental learning, and bioinformatics.

Dr. Pang is a member of IEICE and ACM.

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

He is currently an Associative Professor with Graduate School of Science and Technology, Kobe University. His primary research interests include neural networks, machine learning, intelligent data processing, and pattern recognition.

Nikola Kasabov (SM'01) received the M.Sc. and Ph.D. degrees from the Technical University of Sofia, Sofia, Bulgaria.

He is the Founding Director and the Chief Scientist of the Knowledge Engineering and Discovery Research Institute (KEDRI), Auckland, New Zealand. At present, he holds a Chair of Knowledge Engineering at the School of Computer and Information Sciences, Auckland University of Technology. His main research interests are in the areas of intelligent information systems, soft computing, neuro-computing, bioinformatics, brain study, speech and image processing, data mining, and knowledge discovery, of which he has published 10 books, 380 refereed papers, 25 patents, and other work. He has an extensive academic experience, holding positions at the Technical University of Sofia; University of Essex, Essex, U.K.; University of Otago, Dunedin, New Zealand; and University of Trento, Trento, Italy.

Dr. Kasabov is a Fellow of the Royal Society of New Zealand and a Fellow of the New Zealand Computer Society. He is the chair of the Adaptive Systems Task Force of the Neural Network Technical Committee of the IEEE Computational Intelligence Society. He is a member of the Board of Governors of the INNS and a board member of the Asia Pacific Neural Network Assembly (APNNA). He is on the editorial boards of eight international journals and has been on the planning committees of 50 international conferences in the last five years. He chaired the series of ANNES conferences from 1993 to 2001.