Incremental Linear Discriminant Analysis: A Fast Algorithm and Comparisons

Delin Chu, Li-Zhi Liao, Michael Kwok-Po Ng, and Xiaoyan Wang

***Abstract*— It has always been a challenging task to develop a fast and an efficient incremental linear discriminant analysis (ILDA) algorithm. For this purpose, we conduct a new study for linear discriminant analysis (LDA) in this paper and develop a new ILDA algorithm. We propose a new batch LDA algorithm called LDA/QR. LDA/QR is a simple and fast LDA algorithm, which is obtained by computing the economic QR factorization of the data matrix followed by solving a lower triangular linear system. The relationship between LDA/QR and uncorrelated LDA (ULDA) is also revealed. Based on LDA/QR, we develop a new incremental LDA algorithm called ILDA/QR. The main features of our ILDA/QR include that: 1) it can easily handle the update from one new sample or a chunk of new samples; 2) it has efficient computational complexity and space complexity; and 3) it is very fast and always achieves competitive classification accuracy compared with ULDA algorithm and existing ILDA algorithms. Numerical experiments based on some real-world data sets demonstrate that our ILDA/QR is very efficient and competitive with the state-of-the-art ILDA algorithms in terms of classification accuracy, computational complexity, and space complexity.**

***Index Terms*— Classification accuracy, computational complexity, incremental linear discriminant analysis (ILDA), linear discriminant analysis (LDA).**

I. Introduction

LINEAR discriminant analysis (LDA) is powerful for supervised data dimensionality reduction [8], [9] and has been applied successfully to many applications, including machine learning, data mining, and bioinformatics, which require to deal with high-dimensional data efficiently. LDA has been studied extensively and various extensions have been developed. For example, to overcome the drawback of LDA that it assumes that the class distributions are homoscedastic, subclass discriminant analysis (SDA) is proposed in [29], which attempts to split the classes into subclasses which are linear separable. Experimental results in [29] using a large number of databases and classifiers demonstrate the utility of the proposed approach. It was also shown that SDA can outperform LDA and other LDA variants. Many LDA algorithms are available in the literature. However, almost all existing LDA algorithms are batch algorithms, which require that the data must be available in advance and be given once altogether. However, in many real-world applications, such as customer shopping transaction analysis, mining web logs, and mining DNA sequences, data are incrementally received from various data sources presented as a data stream, and so data grow incrementally. For data stream, the input data to be learned are not available all at once, but rather arrive as a data sequence. Thus, it is common to have a small chunk of data available over a period of time. The batch LDA for dealing with data stream is to collect data whenever new data are presented, and learn all the data from the beginning. However, it is obvious that large memory and high computational complexity are involved in this kind of batch learning, because the system would need to maintain a huge memory to store the data either previously learned or newly presented. For large and high-dimensional data, a huge computational complexity and lack of available storage may be critical issues.

To solve the above problem, it is necessary to develop algorithms that can run in an incremental fashion to accommodate the new data. Learning from new data without forgetting prior knowledge is known as an incremental learning. In this learning scheme, a system must acquire knowledge with the presentation of the new training data and retaining the knowledge acquired in the past without keeping a large number of previous training samples. Compared with the batch learning, incremental learning has the advantages of being more widely applicable and responsive. For example, it can be applied to situations where input data come only in sequence and a timely updating model is crucial for actions. Different from batch learning, an incremental learning deals with theory revision at each time step instead of theory creation from scratch.

Although LDA is very popular for data dimensionality reduction, there is only limited work for incremental LDA (ILDA) algorithms [15]–[18], [26]. The main difficulty is the involvement of the eigenvalue problem of scatter matrices, which is hard to maintain incrementally. It is still a challenging problem to develop a fast and an efficient ILDA algorithm. For this propose, we conduct a new study on LDA in this paper and develop a new and an efficient ILDA algorithm.

Manuscript received October 10, 2013; revised September 16, 2014 and January 9, 2015; accepted January 10, 2015. Date of publication January 29, 2015; date of current version October 16, 2015. The work of D. Chu was supported by the National University of Singapore, Singapore, under Grant R-146-000-187-112. The work of L.-Z. Liao was supported in part by the Faculty Research Grant (FRG), Hong Kong Baptist University (HKBU), Hong Kong, and in part by the General Research Fund (GRF), Hong Kong. The work of M. K.-P. Ng was supported in part by the Research Project under Grant FRG2/13-14/079 and in part by GRF, HKBU, under Grant 201812 and Grant 202013.

D. Chu and X. Wang are with the Department of Mathematics, National University of Singapore, Singapore 119077 (e-mail: matchudl@nus.edu.sg; xiaoyan.nus@gmail.com).

L.-Z. Liao is with the Department of Mathematics, Hong Kong Baptist University, Hong Kong (e-mail: liliao@hkbu.edu.hk).

M. K.-P. Ng is with the Centre for Mathematical Imaging and Vision, Department of Mathematics, Hong Kong Baptist University, Hong Kong (e-mail: mng@math.hkbu.edu.hk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2015.2391201

Our contributions are as follows.

1) We propose a new batch LDA algorithm LDA/QR. It is obtained by computing the economic QR factorization of the data matrix followed by solving a lower triangular linear system. Hence, our LDA/QR is a simple and fast LDA algorithm. The relationship between LDA/QR and uncorrelated LDA (ULDA) is also revealed.
2) We develop a new ILDA algorithm ILDA/QR, which is the exact incremental version of LDA/QR. Our ILDA/QR can easily handle the update from one new sample or a chunk of new samples. It is very fast and it always achieves competitive classification accuracy compared with the state-of-the-art ILDA algorithms.

The rest of this paper is organized as follows. In Section II, some existing ILDA algorithms are reviewed. In Section III, a new fast batch LDA algorithm is proposed. In addition, the relationship between our proposed LDA algorithm and ULDA is also established. As a result, a new ILDA algorithm, which is the exact incremental version of our new batch LDA algorithm, is developed in Section IV. In Section V, the numerical comparisons of our new ILDA algorithm with our new batch LDA algorithm, ULDA, SDA, and four existing ILDA algorithms are reported in terms of computational complexity, space complexity, and classification accuracy. Our numerical experiments with some real-world data sets demonstrate that the performance of our new ILDA algorithm is competitive with the ULDA, the SDA, and the state-of-the-art ILDA algorithms. Finally, the conclusions are drawn in Section VI.

II. Related Work on Incremental Linear Discriminant Analysis

Given a data matrix

$$A = [a_1 \quad \cdots \quad a_n] = [\mathcal{A}_1 \quad \cdots \quad \mathcal{A}_k] \in \mathbf{R}^{m\times n}$$

where each a_i $(1 \le i \le n)$ is a data point in an m-dimensional space and each block matrix $\mathcal{A}_i \in \mathbf{R}^{m\times n_i}$ $(1 \le i \le k)$ is a collection of data items in the ith class, n_i $(1 \le i \le k)$ is the size of the class i, and the total number of data items in data set A is $n = \sum_{i=1}^{k} n_i$. Let $\mathcal{N}_i$ denote the set of column indices that belong to the class i. The global centroid c of A and the local centroid c_i of each class $\mathcal{A}_i$ are given by

$$c = \frac{1}{n}Ae, \quad c_i = \frac{1}{n_i}\mathcal{A}_i e_i, \quad i = 1, \ldots, k$$

respectively, where

$$e = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \in \mathbf{R}^n, \quad e_i = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \in \mathbf{R}^{n_i}, \quad i = 1, \ldots, k.$$

Let

$$\begin{aligned} S_b &= \sum_{i=1}^{k} n_i (c_i - c)(c_i - c)^T \\ S_w &= \sum_{i=1}^{k} \sum_{j\in\mathcal{N}_i} (a_j - c_i)(a_j - c_i)^T \\ S_t &= \sum_{j=1}^{n} (a_j - c)(a_j - c)^T \end{aligned} \tag{1}$$

where S_b, S_w, and S_t are called the between-class scatter matrix, the within-class scatter matrix, and the total scatter matrix, respectively. It is well-known that [14]

$$S_t = S_b + S_w. \tag{2}$$

Denote

$$\begin{aligned} H_b &= \left[\sqrt{n_1}(c_1 - c) \quad \cdots \quad \sqrt{n_k}(c_k - c)\right] \in \mathbf{R}^{m\times k} \\ H_w &= \left[\mathcal{A}_1 - c_1 e_1^T \quad \cdots \quad \mathcal{A}_k - c_k e_k^T\right] \in \mathbf{R}^{m\times n} \\ H_t &= [a_1 - c \quad \cdots \quad a_n - c] = A - ce^T \in \mathbf{R}^{m\times n}. \end{aligned} \tag{3}$$

Then, scatter matrices S_b, S_w, and S_t can be expressed as

$$S_b = H_b H_b^T, \quad S_w = H_w H_w^T, \quad S_t = H_t H_t^T. \tag{4}$$

It follows from the properties of matrix trace that:

$$\begin{aligned} \operatorname{trace}(S_w) &= \sum_{i=1}^{k} \sum_{j\in\mathcal{N}_i} (a_j - c_i)^T (a_j - c_i) \\ &= \sum_{i=1}^{k} \sum_{j\in\mathcal{N}_i} \|a_j - c_i\|_2^2 \\ \operatorname{trace}(S_b) &= \sum_{i=1}^{k} n_i (c_i - c)^T (c_i - c) = \sum_{i=1}^{k} n_i \|c_i - c\|_2^2. \end{aligned}$$

Thus, $\operatorname{trace}(S_b)$ measures the distance between the class local centroids and the global centroid, while $\operatorname{trace}(S_w)$ measures the distance between the data points and their corresponding class local centroid. Note that when data points within each class are tightly located around their local class centroid, the value of $\operatorname{trace}(S_w)$ will be small, while when the local centroids are remote from the global centroid, the value of $\operatorname{trace}(S_b)$ will be large. Therefore, the class quality can be measured by the values of $\operatorname{trace}(S_w)$ and $\operatorname{trace}(S_b)$. When $\operatorname{trace}(S_b)$ is large while $\operatorname{trace}(S_w)$ is small, the different classes will be separated well and the data points within each class will be related tightly. This leads to high class quality.

In the lower dimensional space mapped upon using the linear transformation G, the between-class, within-class, and total scatter matrices are of the forms

$$S_b^L = G^T S_b G, \quad S_w^L = G^T S_w G, \quad S_t^L = G^T S_t G.$$

Ideally, the optimal transformation G should maximize $\operatorname{trace}(S_b^L)$ and minimize $\operatorname{trace}(S_w^L)$ simultaneously, equivalently, maximize $\operatorname{trace}(S_b^L)$ and minimize $\operatorname{trace}(S_t^L)$ simultaneously, which leads to optimization in classical LDA for determining the optimal linear transformation G, namely, the classical Fisher criterion [8], [9]

$$G = \arg\max_{G\in\mathbf{R}^{m\times l}} \operatorname{trace}((G^T S_t G)^{-1}(G^T S_b G)) \tag{5}$$

where $G \in \mathbf{R}^{m\times l}$ projects data in m-dimensional space to l-dimensional subspace.

In the classical LDA (5), $l = k-1$ and the columns of G are the eigenvectors corresponding to the $k-1$ largest eigenvalues of the eigenvalue problem

$$\begin{aligned} S_b x &= \lambda S_t x, \quad \lambda \ne 0 \\ S_t^{-1} S_b x &= \lambda x, \quad \lambda \ne 0. \end{aligned} \tag{6}$$

Classical LDA has a critical drawback, that is, the total scatter matrix S_t must be nonsingular. However, when the data points are from a high-dimensional space and thus usually the number of the data samples is much smaller than the data dimension, i.e., $m > n$, the total scatter matrix S_t is singular. This is known as the undersampled problem [1] and it is also commonly called the small sampled size problem. Thus, we cannot apply the classical LDA to undersampled problems directly.

To make LDA applicable for undersampled problems, various extensions of the classical LDA can be found in the literature. One popular extension is to replace the inverse in classical LDA by pseudoinverse [12], which can avoid the singularity problem. Consequently, the criterion for the LDA becomes

$$G = \arg\max_{G\in\mathbf{R}^{m\times l}} \text{trace}((G^T S_t G)^{(+)}(G^T S_b G)) \tag{7}$$

where $(\cdot)^{(+)}$ denotes the pseudoinverse of $(\cdot)$.

Incremental LDA along with some algorithms have been proposed in [15]–[18] and [26]. In [26], the IDR/QR algorithm is developed by applying regularized LDA in a projected subspace spanned by the class means. As the dimension of the subspace, which is equal to the number of data classes, is low, the IDR/QR algorithm is fast. However, its main limitation is that much information is lost in the first projection as well as some useful components are discarded in the updating process. In addition, an appropriate regularization parameter should be given in advance. In [15] and [16], a different ILDA algorithm, which we call it ILDA/SSS, is presented. ILDA/SSS is based on the concept of sufficient spanning set approximation to update the between-class scatter matrix and the total scatter matrix, where the principal eigenvectors and eigenvalues of these two matrices are kept and updated, and the minor components are removed in each update. The big issue of ILDA/SSS is that three eigenvalue thresholds, respectively, for determining the principal components of the between-class scatter matrix, the total scatter matrix, and the optimal transformation matrix should be given in advance. As shown in [17], ILDA/SSS suffers from a problem that is difficult to determine a balance between the classification performance and the computational efficiency. When there are many small components to be discarded, the performance deteriorates but the efficiency will be increased. Moreover, the performance of IDLA/SSS is quite sensitive to the setting of the approximation parameter which is difficult to tune in practice.

Unlike IDR/QR and ILDA/SSS, two recently proposed incremental algorithms, LS-ILDA [17] and ICLDA [18], perform exact incremental updating whenever a new data sample is added. ICLDA is an incremental approach by incrementally implementing CLDA [22] exactly. However, the dimension of the reduced space of ICLDA, which is equal to the rank of the total scatter matrix, is high, the computational complexity, and the memory requirement are very large. Moreover, there are too many items required to be updated when a new sample is added, and therefore, the storage cost is increased. LS-ILDA is based on the idea of LS-LDA [24], which puts LDA into the framework of multivariate regression and gives the least-square solution to LDA. Let the full column rank matrix G_{LDA} be the solution of LDA with its columns being the eigenvectors of $S_T^{(+)}S_b$ corresponding to its all nonzero eigenvalues. Define $Y = (y_{ij}) \in \mathbf{R}^{n\times k}$ by

$$y_{ij} = \begin{cases} \sqrt{\frac{n}{n_j}} - \sqrt{\frac{n_j}{n}}, & \text{if } x_i \text{ belongs to class } j \\ -\sqrt{\frac{n_j}{n}}, & \text{otherwise} \end{cases}$$

denote

$$G_{\text{MLR}-1} = \left(\left(I - \frac{1}{n}ee^T\right)A^T\right)^{(+)} Y. \tag{8}$$

It is shown in [24] that

$$G_{\text{LDA}} = G_{\text{MLR}-1}\mathcal{Q}$$

for some column orthogonal matrix $\mathcal{Q}$, provided $\text{rank}(S_b) + \text{rank}(S_w) = \text{rank}(S_t)$. Hence, $G_{\text{MLR}-1}$ is very similar to G_{LDA}. The assumption $\text{rank}(S_b) + \text{rank}(S_w) = \text{rank}(S_t)$ has been removed in [28] and it is shown that

$$G_{\text{LDA}} = G_{\text{MLR}-2}Z$$

where

$$\begin{aligned} G_{\text{MLR}-2} &= \left[\left(A\left(I - \frac{1}{n}ee^T\right)A^T\right]^{(+)} A\left(I - \frac{1}{n}ee^T\right)E \right. \\ &\quad \left. \times \begin{bmatrix} n_1 & & \\ & \ddots & \\ & & n_k \end{bmatrix}^{-\frac{1}{2}} \right. \\ &= \left[\left(I - \frac{1}{n}ee^T\right)A^T\right]^{(+)} E \begin{bmatrix} n_1 & & \\ & \ddots & \\ & & n_k \end{bmatrix}^{-\frac{1}{2}} \end{aligned} \tag{9}$$

Z is a full column rank matrix, and

$$E = \begin{bmatrix} e_1 & & \\ & \ddots & \\ & & e_k \end{bmatrix} \in \mathbf{R}^{n\times k}. \tag{10}$$

Hence, a general relationship between LDA and ridge regression is uncovered. Clearly, both $G_{\text{MLR}-1}$ and $G_{\text{MLR}-2}$ can be considered as the solutions to LDA. In [17], $G_{\text{MLR}-2}$ is taken as the solution to LDA. The core step of LS-ILDA is to update the pseudoinverse of the centered data matrix $A(I-(1/n)ee^T)$.

It should be pointed out that only ILDA/SSS can handle the case that a chunk of new samples is added. There are no chunk versions of IDR/QR, LS-ILDA, and ICLDA available in the literature.

III. New and Fast Batch LDA Algorithm—LDA/QR

In this section, a new batch LDA algorithm is developed. Let

$$\mathcal{E} = \frac{1}{n}ee^T, \quad \mathcal{E}_i = \frac{1}{n_i}e_i e_i^T, \quad i = 1, \ldots, k.$$

The following results reveal the relationships between the data matrix A and three scatter matrices S_t, S_b, and S_w.

Lemma 3.1 [5]: Let S_t, S_b, and S_w be defined by (1). Then

$$\begin{cases} S_b = A\left(\begin{bmatrix} \mathcal{E}_1 & & \\ & \ddots & \\ & & \mathcal{E}_k \end{bmatrix} - \mathcal{E}\right) A^T \\ S_w = A\left(I - \begin{bmatrix} \mathcal{E}_1 & & \\ & \ddots & \\ & & \mathcal{E}_k \end{bmatrix}\right) A^T \\ S_t = A(I-\mathcal{E})A^T. \end{cases} \tag{11}$$

Using Lemma 3.1, we have that for any G with appropriate dimensions

$$G^T S_t G = G^T A(I-\mathcal{E})A^T G$$
$$G^T S_b G = G^T A\left(\begin{bmatrix} \mathcal{E}_1 & & \\ & \ddots & \\ & & \mathcal{E}_k \end{bmatrix} - \mathcal{E}\right) A^T G.$$

Lemma 3.2 [23]: Let S_t and S_b be defined by (1). Then

$$\max_G \operatorname{trace}\left((G^T S_t G)^{(+)} G^T S_b G\right) = \operatorname{trace}\left(S_t^{(+)} S_b\right).$$

For any $A \in \mathbf{R}^{m\times n}$ with $m \geq n$, $\operatorname{rank}(A) < n$ if and only if the columns of A are linearly dependent. This leads to the following result.

Lemma 3.3: For any two positive integers m and n with $m \geq n$, let $\mathcal{L}$ be the Lebesgue measure on $\mathbf{R}^{m\times n}$. Denote $\mathcal{S}_0 = \{A \in \mathbf{R}^{m\times n},\ \operatorname{rank}(A) < n\}$. Then, $\mathcal{L}(\mathcal{S}_0) = 0$.

Proof: See the Appendix. □

Lemma 3.3 implies that $\operatorname{rank}(A) = n$ holds for almost all matrices $A \in \mathbf{R}^{m\times n}$ with $m \geq n$. So, the assumption $\operatorname{rank}(A) = n$ holds for almost all high-dimensional and undersampled data. Hence, in the following development, we assume $\operatorname{rank}(A) = n$, equivalently, we assume that training samples are linearly independent [23], [24].

Lemma 3.4: Let the set of training samples be linearly independent. Then

$$\operatorname{trace}\left(S_t^{(+)} S_b\right) = k - 1.$$

Proof: See the Appendix. □

Lemmas 3.2 and 3.4 indicate that any G with

$$\max_G \operatorname{trace}\left((G^T S_t G)^{(+)} G^T S_b G\right) = k - 1 \tag{12}$$

is a solution of the optimization problem (7) provided that the training samples are linearly independent; consequently, G is an optimization transformation of LDA. Thus, to compute an optimal transformation G of LDA, we only need to find G such that (12) holds. It is easy to see that (12) is true if

$$G^T S_t G = G^T S_b G, \quad \operatorname{rank}(G^T S_b G) = k - 1. \tag{13}$$

This observation motivates the following result.

Theorem 3.1: Assume training samples are linearly independent. Then, the linear system

$$A^T G = E \tag{14}$$

is solvable, i.e., there exists at least one $G \in \mathbf{R}^{m\times k}$ satisfying the linear system (14). Furthermore, let $G \in \mathbf{R}^{m\times k}$ be a solution of the linear system (14). Then, $\operatorname{rank}(G) = k$ and

$$\operatorname{trace}\left((G^T S_t G)^{(+)} G^T S_b G\right) = k - 1$$

thus, G is an optimal solution of the optimization problem (7).

Proof: See the Appendix. □

ULDA [6], [23], [25] is an important LDA algorithm, which computes an optimal transformation such that the extracted feature vectors are mutually uncorrelated. The optimal transformation of ULDA is a solution of the following optimization problem:

$$\begin{aligned} G &= \arg \max_{G^T S_t G = I} \operatorname{Trace}\left((G^T S_t G)^{\dagger} (G^T S_b G)\right) \\ &= \arg \max_{G^T S_t G = I} \operatorname{Trace}(G^T S_b G). \end{aligned} \tag{15}$$

The following theorem reveals the relationship between the solution G of linear system (14) and ULDA.

Theorem 3.2: Assume training samples are linearly independent. Let $G \in \mathbf{R}^{m\times k}$ be a solution of linear system (14). Then, $G\Psi$ is an optimal transformation of ULDA (15), where

$$\Psi = \begin{bmatrix} \sqrt{n_1} & & \\ & \ddots & \\ & & \sqrt{n_k} \end{bmatrix}^{-1} \mathcal{H}^T \begin{bmatrix} 0 \\ I_{k-1} \end{bmatrix} \in \mathbf{R}^{k\times(k-1)}$$

and $\mathcal{H}$ is the Householder transformation

$$\mathcal{H} = I - \left(\begin{bmatrix} \sqrt{n_1} - \sqrt{n} \\ \sqrt{n_2} \\ \vdots \\ \sqrt{n_k} \end{bmatrix} \Big/ \sqrt{n - \sqrt{n n_1}}\right) \times \left(\begin{bmatrix} \sqrt{n_1} - \sqrt{n} \\ \sqrt{n_2} \\ \vdots \\ \sqrt{n_k} \end{bmatrix} \Big/ \sqrt{n - \sqrt{n n_1}}\right)^T$$

which is orthogonal and satisfies

$$\mathcal{H} = \mathcal{H}^T, \quad \mathcal{H}^T \left(\begin{bmatrix} \sqrt{n_1} \\ \sqrt{n_2} \\ \vdots \\ \sqrt{n_k} \end{bmatrix} \Big/ \sqrt{n}\right) = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

Proof: See the Appendix. □

Under the assumption that the training samples are linearly independent, we have $\operatorname{rank}(A) = n$, and so the economic QR of A is

$$A = QR, \quad Q \in \mathbf{R}^{m\times n}, \quad R \in \mathbf{R}^{n\times n} \tag{16}$$

where Q is the column orthogonal and R is the nonsingular. Then, the solution of the linear system (14) with the minimal 2-norm is

$$G = QR^{-T}E. \tag{17}$$

Algorithm 1 (LDA/QR)

Input: Data matrix $A \in \mathbf{R}^{m\times n}$ of full column rank with class label, size of each class n_i $(i = 1, \ldots, k)$, class number k.
Output: Output: Transformation matrix $G \in \mathbf{R}^{m\times k}$.
Step 1: Compute the economic QR factorization (16) of A. Then solve the lower triangular linear system

$$R^T\mathcal{G} = E$$

where E is defined in (10).
Step 2: Compute

$$G = Q\mathcal{G}.$$

We now study the relationship between the solution G defined in (17) and G_{LDA} defined in Section II.

Theorem 3.3: Assume training samples are linearly independent. Let G be the solution of the linear system (14) with the minimal 2-norm. Then

$$G_{\mathrm{LDA}} = G\mathcal{Z}$$

for some full column rank matrix $\mathcal{Z}$.

Proof: See the Appendix. □

Obviously, Theorems 3.1–3.3 imply that the solution G of the linear system (14) with the minimal 2-norm is equivalent to the solution of ULDA (15) [6], [23], [25] and classical solution G_{LDA} scaled by a full column rank constant matrix.

Theorems 3.1–3.3 yield the following new batch LDA algorithm (Algorithm 1) called LDA/QR.

It should be pointed out that the optimal transformation G in LDA/QR is obtained by computing the economic QR factorization of the data matrix A and then solving a lower triangular linear system. Hence, LDA/QR is simple and fast. Although the optimal transformation G in LDA/QR is very similar to $G_{\mathrm{MLR}-1}$ (8) in [24] and $G_{\mathrm{MLR}-2}$ (9) in [28] theoretically, a significant difference is that the incremental updating of $((I - (1/n)ee^T)A^T)^{(+)}$ in $G_{\mathrm{MLR}-1}$ and $G_{\mathrm{MLR}-2}$ is much more difficult than the updating of the economic QR factorization of A. As shown in the following two sections, the incremental algorithm of our LDA/QR is much faster than LS-ILDA [17], which is the incremental version of $G_{\mathrm{MLR}-1}$ (and $G_{\mathrm{MLR}-2}$).

Remark 3.1: In traditional LDA, the optimal transformation G has $k-1$ columns which are the $k-1$ eigenvectors corresponding to the $k-1$ largest eigenvalues of the eigenvalue problem (2.6). In our LDA/QR Algorithm 1, G has k columns, this implies that the range space of G contains an eigenvector which projects in the null space of S_b. Theoretically, this eigenvector can be discarded from our G. However, our LDA/QR Algorithm 1 computes the G without computing any eigenvalue/eigenvector. Hence, this eigenvector should be kept, otherwise, the eigenvector computation will be involved which is expensive. More importantly, the incremental version of our LDA/QR Algorithm 1 can be established as shown later.

IV. Incremental Implementation of LDA/QR

In this section, the incremental version of our LDA/QR proposed in Section III will be developed. We will adopt the following convention in the rest of this section. For any X, its updated version after the insertion of new samples is denoted by $\tilde{X}$. For example, the data matrix A is changed to $\tilde{A}$ after the insertion of new samples x.

According to LDA/QR, the optimal transformation of LDA/QR after new samples are added is the solution of the linear system

$$\tilde{A}^T\tilde{G} = \tilde{E} \tag{18}$$

with the minimal 2-norm. Therefore, in the following, we only need to compute the solution of the linear system (18) with the minimal 2-norm incrementally.

To present our method clearly, we discuss two cases. The two cases are as follows.

Case 1: Only one new sample is inserted.
Case 2: A chunk of new samples are inserted.
In the following, we first consider Case 1 and then Case 2.
Case 1: There are two subcases for a new inserted sample x as follows.
Subcase 1: x belongs to an existing class ℓ.
Subcase 2: x belongs to a new class.
Subcase 1: Insertion of a new sample belonging to an existing class ℓ.
In this case, we have

$$\tilde{A} = [A \quad x], \quad \tilde{E} = \begin{bmatrix} E \\ z \end{bmatrix}$$

where $z \in \mathbf{R}^k$ is a row vector with the ℓth element 1 and the others 0. Note that the economic QR factorization of A is $A = QR$. We have

$$\tilde{A} = [A \quad x] = [QR \quad x] = [Q \quad x]\begin{bmatrix} R & \\ & 1 \end{bmatrix}.$$

To find out the economic QR factorization of $\tilde{A}$, we seek $q \in \mathbf{R}^m$, $r \in \mathbf{R}^n$, and a scalar α so that

$$[Q \quad x] = [Q \quad q]\begin{bmatrix} I & r \\ & \alpha \end{bmatrix}$$

with $Q^Tq = 0$ and $\|q\|_2 = 1$. By a direct calculation, we obtain

$$r = Q^Tx, \quad \alpha = \|x - Qr\|_2 = \sqrt{x^Tx - r^Tr}, \quad q = \frac{x - Qr}{\alpha}$$

the resulting economic QR factorization of $\tilde{A}$ becomes

$$\tilde{A} = [Q \quad x]\begin{bmatrix} R & \\ & 1 \end{bmatrix} = [Q \quad q]\begin{bmatrix} R & r \\ & \alpha \end{bmatrix} = \tilde{Q}\tilde{R}$$

where

$$\tilde{Q} = [Q \quad q] \in \mathbf{R}^{m\times(n+1)}, \quad \tilde{R} = \begin{bmatrix} R & r \\ & \alpha \end{bmatrix} \in \mathbf{R}^{(n+1)\times(n+1)}.$$

Algorithm 2 (ILDA/QR: Updating an Existing Class)

Input: Column orthogonal matrix Q, optimal transformation G, and new sample x from the ℓ-th class.

Output: Updated Q and G.

Step 1 : Compute

$$r := Q^T x, \quad \alpha := \sqrt{x^T x - r^T r}$$
$$Q := \left[Q(x - Qr)/\alpha\right].$$

Step 2 : Compute

$$r := -G^T x, \; r(\ell) := r(\ell) + 1$$
$$r := r/\alpha, \quad G := G + Q(:, n+1)r^T.$$

Consequently, we have

$$\begin{aligned}\tilde{G} &= \tilde{Q}\tilde{R}^{-T}\tilde{E} = \tilde{Q}\begin{bmatrix} R & r \\ & \alpha \end{bmatrix}^{-T}\tilde{E} \\ &= \tilde{Q}\begin{bmatrix} R^{-T} & \\ -\frac{1}{\alpha}r^T R^{-T} & \frac{1}{\alpha} \end{bmatrix}\begin{bmatrix} E \\ z^T \end{bmatrix} \\ &= [Q \quad q]\begin{bmatrix} R^{-T}E \\ \frac{1}{\alpha}(z^T - r^T R^{-T}E) \end{bmatrix} \\ &= QR^{-T}E + \frac{1}{\alpha}(qz^T - qr^T R^{-T}E) \\ &= QR^{-T}E + \frac{1}{\alpha}(qz^T - qx^T QR^{-T}E) \\ &= G + \frac{1}{\alpha}q(z - G^T x)^T.\end{aligned}$$

Subcase 2: Insertion of a new sample belonging to a new class.

In this case, we have

$$\tilde{A} = [A \quad x], \quad \tilde{E} = \begin{bmatrix} E & \\ & 1 \end{bmatrix}$$

and so

$$\begin{aligned}\tilde{G} &= \tilde{Q}\tilde{R}^{-T}\tilde{E} = \tilde{Q}\begin{bmatrix} R^{-T} & \\ -\frac{1}{\alpha}r^T R^{-T} & \frac{1}{\alpha} \end{bmatrix}\begin{bmatrix} E & \\ & 1 \end{bmatrix} \\ &= [Q \quad q]\begin{bmatrix} R^{-T}E & \\ -\frac{1}{\alpha}r^T R^{-T}E & \frac{1}{\alpha} \end{bmatrix} \\ &= \left[QR^{-T}E - \tfrac{1}{\alpha}qr^T R^{-T}E \quad \tfrac{1}{\alpha}q\right] \\ &= \left[QR^{-T}E - \tfrac{1}{\alpha}qx^T QR^{-T}E \quad \tfrac{1}{\alpha}q\right] \\ &= \left[G - \tfrac{1}{\alpha}qx^T G \quad \tfrac{1}{\alpha}q\right].\end{aligned}$$

We are now ready to present the incremental implementation of LDA/QR for one new sample inserted, which involves two steps: 1) the updating of column orthogonal matrix Q and 2) the updating of optimal transformation G. We call this ILDA algorithm as ILDA/QR. ILDA/QR for updating an existing class and adding a new class are shown in the following two separated algorithms, Algorithms 2 and 3, respectively.

Case 2: Now, we consider the case that the new samples $x_1, \ldots, x_\mu$ are acquired in a chunk way. Denote

$$\mathcal{X} = [x_1 \quad \cdots \quad x_\mu]$$

and ℓ_i is the class label of x_i, $i = 1, \ldots, \mu$.

Algorithm 3 (ILDA/QR: Adding a New Class)

Input: Column orthogonal matrix Q, optimal transformation G, and new sample x from a new class.

Output: Updated Q and G.

Step 1 : Compute

$$r := Q^T x, \quad \alpha := \sqrt{x^T x - r^T r}$$
$$Q := [Q \quad (x - Qr)/\alpha].$$

Step 2 : Compute

$$r := Q(:, n+1)/\alpha, \; G := [G - r(x^T G) \quad r].$$

After incorporating $x_1, \ldots, x_\mu$, the new data matrix $\tilde{A} = [A \; \mathcal{X}]$ contains totally $\tilde{k}$ ($\tilde{k} \geq k$) classes of samples and the new indicator matrix has the form

$$\tilde{E} = \begin{bmatrix} E & 0 \\ & Z \end{bmatrix} \in \mathbf{R}^{(n+s)\times\tilde{k}} \tag{19}$$

where $Z = [z_1, \ldots, z_s]^T \in \mathbf{R}^{\mu\times\tilde{k}}$ and $z_i \in \mathbf{R}^{\tilde{k}}$ ($i = 1, \ldots, \mu$) is a unit vector with the ℓ_ith element being 1 and all other elements being 0.

Given the economic QR factorization of $A = QR$, and let the economic QR factorization of $\mathcal{X} - Q(Q^T\mathcal{X})$ be

$$\mathcal{X} - Q(Q^T\mathcal{X}) = \hat{Q}\hat{R}$$

where $\hat{Q} \in \mathbf{R}^{m\times\mu}$ is the column orthogonal and $\hat{R} \in \mathbf{R}^{\mu\times\mu}$ is the upper triangular. Then, the updated QR factorization of $\tilde{A}$ is

$$\begin{aligned}\tilde{A} &= [A \quad \mathcal{X}] = [QR \quad \mathcal{X}] \\ &= \left[Q \quad (I - QQ^T)\mathcal{X}\right]\begin{bmatrix} R & Q^T\mathcal{X} \\ & I \end{bmatrix} \\ &= [Q \quad \hat{Q}]\begin{bmatrix} R & Q^T\mathcal{X} \\ & \hat{R} \end{bmatrix} = \tilde{Q}\tilde{R}\end{aligned}$$

where

$$\tilde{Q} = [Q \quad \hat{Q}], \quad \tilde{R} = \begin{bmatrix} R & Q^T\mathcal{X} \\ & \hat{R} \end{bmatrix}.$$

Consequently, the updated transformation matrix $\tilde{G}$ is

$$\begin{aligned}\tilde{G} &= \tilde{Q}\tilde{R}^{-T}\tilde{E} = \tilde{Q}\begin{bmatrix} R & Q^T\mathcal{X} \\ & \hat{R} \end{bmatrix}^{-T}\tilde{E} \\ &= \tilde{Q}\begin{bmatrix} R^{-T} & \\ -\hat{R}^{-T}\mathcal{X}^T QR^{-T} & \hat{R}^{-T} \end{bmatrix}\begin{bmatrix} E & 0 \\ & Z \end{bmatrix} \\ &= [Q \quad \hat{Q}]\begin{bmatrix} R^{-T}[E \;\; 0] \\ \hat{R}^{-T}Z^T - \hat{R}^{-T}\mathcal{X}^T QR^{-T}[E \;\; 0] \end{bmatrix} \\ &= QR^{-T}[E \;\; 0] + \hat{Q}\hat{R}^{-T}Z - \hat{Q}\hat{R}^{-T}\mathcal{X}^T QR^{-T}[E \;\; 0] \\ &= [G \;\; 0] + \hat{Q}\hat{R}^{-T}Z - \hat{Q}\hat{R}^{-T}\mathcal{X}^T[G \;\; 0] \\ &= \left[G - \hat{Q}(\hat{R}^{-T}(\mathcal{X}^T G)) \; 0\right] + \hat{Q}(\hat{R}^{-T}Z).\end{aligned}$$

Based on the above analysis on the updating of $\tilde{E}$, QR factorization of $\tilde{A}$ and $\tilde{G}$, the incremental implementation of LDA/QR for a chunk of new samples is given in Algorithm 4.

Algorithm 4 (ILDA/QR: Updating With a Chunk of New Samples)

Input: Column orthonormal matrix Q, optimal transformation G, and new sample matrix $\mathcal{X} = [x_1 \cdots x_\mu]$ with class labels ℓ_i, $i = 1, \ldots, \mu$.

Output: Updated Q and G.

Step 1 : Construct

$$Z = [z_1 \quad \cdots \quad z_\mu]^T$$

where z_i $(i = 1, \ldots, \mu)$ is a unit vector with the ℓ_ith element being 1 and all other elements being zero.

Step 2 : Compute the economic QR factorization

$$\mathcal{X} - Q(Q^T\mathcal{X}) = \hat{Q}\hat{R}$$

where $\hat{Q}$ is column orthogonal and $\hat{R}$ is upper triangular.

Step 3 : Update

$$G := \left[G - \hat{Q}(\hat{R}^{-T}(\mathcal{X}^T G)) \; 0\right] + \hat{Q}(\hat{R}^{-T}Z)$$
$$Q := [Q \quad \hat{Q}].$$

Remark 4.1: Under the assumption that all training samples are linearly independent, we have the following.

1) In Algorithms 2 and 3, $\alpha \neq 0$.
2) In Algorithm 4, $\hat{R}$ is nonsingular and thus $\hat{R}^{-T}$ exists.

Remark 4.2: In Algorithm 4, $\mathcal{Y} := (\hat{R}^{-T}(\mathcal{X}^T G))$ and $\mathcal{Z} := \hat{R}^{-T}Z$ can be obtained easily by solving the following two upper triangular linear systems of equations $\hat{R}^T\mathcal{Y} = (\mathcal{X}^T G)$ and $\hat{R}^T\mathcal{Z} = Z$.

With the incremental updating schemes above, the incremental algorithm ILDA/QR works as follows.

1) When a new sample x is inserted, determine whether it is from an existing class or a new class. If it is from an existing class, update the column orthogonal matrix Q and the optimal transformation G by applying Algorithm 2; otherwise, update the column orthogonal matrix Q and the optimal transformation G by applying Algorithm 3.
2) When a chunk of new samples $x_1, \ldots, x_\mu$ are inserted, update the column orthogonal matrix Q and the optimal transformation G by applying Algorithm 4.
3) The above procedure is repeated until all samples are inserted. Then, the optimal transformation is the final updated G.

Obviously, our ILDA/QR algorithm satisfies the general criteria of the incremental learning algorithm [19] as follows.

1) It is able to learn additional information from new samples.
2) It does not need to process the original data during updating.
3) It preserves the previously acquired knowledge.
4) It can accommodate new classes that may be introduced with new data.

V. Comparisons

In this section, we evaluate the efficiency of our new proposed incremental algorithm ILDA/QR by comparing it with batch LDA algorithms LDA/QR, ULDA/QR [6], and SDA [29] and four existing ILDA algorithms IDR/QR [26], ILDA/SSS [15], [16], LS-ILDA [17], and ICLDA [18]. The performance is mainly measured by the computational complexity, the space complexity, and the classification accuracy.

TABLE I
Computational Complexities (Flops) of ULDA/QR [6] and LDA/QR

ULDA/QR [6]:	$\mathbf{O}(4mn^2 + 2mnk + 4n^3)$
LDA/QR:	$\mathbf{O}(4mn^2 + 2mnk - \frac{4}{3}n^3)$

TABLE II
Computational Complexities (Flops) of IDR/QR, ILDA/SSS, LS-ILDA, ICLDA, and ILDA/QR for a Single Insertion

IDR/QR	$\mathbf{O}(2mk^2 + 91k^3/3)$
ILDA/SSS	$\mathbf{O}(2mn^2 + 12n^3)$
LS-ILDA	$\mathbf{O}(14mn + 7mk)$
ICLDA	$\mathbf{O}(2mn^2 + 20n^3)$
ILDA/QR	$\mathbf{O}(4mn + 4mk)$

TABLE III
Computational Complexities (Flops) of ILDA/SSS and ILDA/QR for a Chunk of New Samples $x_1, \ldots, x_\mu$ Added

ILDA/SSS	$\mathbf{O}(2m(n+\mu)^2 + 12(n+\mu)^3)$
ILDA/QR	$\mathbf{O}(4mn\mu + 4m\mu^2 + 4mk\mu + 2m\tilde{k}\mu)$

$\tilde{k}$ is the number of classes after a chunk of new samples are added.

TABLE IV
Space Complexities of IDR/QR, ILDA/SSS, LS-ILDA, ICLDA, and ILDA/QR

IDR/QR	$\mathbf{O}(2mk)$
ILDA/SSS	$\mathbf{O}(mn + 2mk)$
LS-ILDA	$\mathbf{O}(2mn + mk)$
ICLDA	$\mathbf{O}(4mn + 2n^2 + mk)$
ILDA/QR	$\mathbf{O}(mn + mk)$

TABLE V
Space Complexities of ILDA/SSS and ILDA/QR With a Chunk of New Samples $x_1, \ldots, x_\mu$

ILDA/SSS	$\mathbf{O}(mn + 2mk)$
ILDA/QR	$\mathbf{O}(mn + mk)$

A. Comparisons for Computational Complexity and Space Complexity

We summarize the computational complexities of batch LDA algorithms LDA/QR and ULDA/QR [6] and ILDA algorithms IDR/QR, ILDA/SSS, LS-ILDA, ICLDA, and ILDA/QR in Tables I–III,[1] respectively, and the space complexities of ILDA algorithms IDR/QR, ILDA/SSS, LS-ILDA, ICLDA, and ILDA/QR in Tables IV and V.

[1]Tables I does not include the computational complexities (flops) of SDA [18] since the computational complexity of SDA is much higher than ULDA/QR but it is hard to estimate its flops generally.

TABLE VI
DATA STRUCTURES

Type	Data	m	n	k
Text Document	$K1a$	21839	2340	20
	$Tr23$	5832	204	6
	Wap	8460	1560	20
Face Image	$ORL_{32\times32}$	1024	400	40
	$AR_{50\times45}$	2250	1680	120
	$YaleB$	32256	2424	38
	PIE	4096	6661	68
	$Feret$	6400	1000	200
Gene Expression	$Brain$	5597	42	5
	$Prostate$	6033	102	2

Remark 5.1: Since there are no chunk versions of IDR/QR, LS-ILDA, and ICLDA available, we only compare ILDA/SSS and ILDA/QR in Tables III and V.

Tables I–V lead to the following observations.

1) The computational complexity of ILDA/QR is much lower than those of LDA/QR, ULDA/QR [6], and SDA [29].
2) The complexity of LDA/QR is lower than those of ULDA/QR [6] and SDA [29], in particular, when the sample size n is large.
3) Among the five ILDA algorithms IDR/QR, ILDA/SSS, LS-ILDA, ICLDA, and ILDA/QR, algorithms ILDA/QR and IDR/QR have the lowest and pleasant computational complexities. In addition, when the class number k is very small compared with the sample size n, IDR/QR should be faster than ILDA/QR, otherwise, ILDA/QR shall be faster.
4) LS-ILDA has an acceptable computational complexity, but, much higher than those of IDA/QR and ILDA/QR.
5) The computational complexities of ILDA/SSS and ICLDA are very high.
6) Among the five ILDA algorithms IDR/QR, ILDA/SSS, LS-ILDA, ICLDA, and ILDA/QR, ILDA/QR, IDR/QR, and ILDA/SSS have pleasant space complexities.

B. Experimental Setup and Data Sets

Before reporting the experimental results, we discuss the data sets and the experimental setting.

1) Experimental Platforms: All experiments were conducted on a Sun v40z computer with 2.4-GHz Opteron 850 CPUs and 32-GB RAM in Center for Computational Science and Engineering, National University of Singapore.

2) Experimental Data Sets: Our experiments are performed on the following ten real-world data sets from three different sources, including text document, face image, and gene expression. The structures of these data are summarized in Table VI, where m is the dimension of the data, n is the total sample size, and k is the number of classes; a more detailed description of these data is presented as follows:

1) text document:
   a) Data set $Tr23$ is derived from TREC collection `http://trec.nist.gov/`. It is also available at `http://shi-zhong.com/software/docdata.zip`. The categories correspond to the documents relevant to particular queries.
   b) Data sets $K1a$ and Wap are from the WebACE project [2], [3], [13], where each document corresponds to a web page listed in the subject hierarchy of Yahoo (http://www.yahoo.com).
2) face image:
   a) ORL data set contains a set of ten different images, each of which has 40 distinct subjects. For some subjects, the images were taken at different times, varying the lighting, the facial expressions, and the facial details. The database can be retrieved from `http://www.cl.cam.ac.uk/Research/DTG/attarchive:pub/data/att_faces.tar.Z`. In our experiment, the size of each image is resized to 32×32 pixels for $\text{ORL}_{32\times32}$.
   b) AR data set consists of 4000 color images corresponding to 126 people's faces (70 men and 56 women). Images feature frontal view faces with different facial expressions, illumination conditions, and occlusions, second sessions repeated same conditions. In our experiment, we selected pictures of 120 individuals (65 men and 55 women) in two sessions, and 28 face images, 14 for each session, were used for each individual. The face portion of each image was cropped to 50×45 pixels for $\text{AR}_{50\times45}$. AR face data set was available at `http://cobweb.ecn.purdue.edu/~aleix/aleix_face_DB.html`.
   c) $YaleB$ data set includes both of the original Yale Face Database B [10] with 10 subjects and the extended Yale Face Database B [10] with 28 subjects. All image data were manually aligned, cropped, and resized to 168×192 images. The database of $YaleB$ is available at `http://vision.ucsd.edu/~leekc/ExtYaleDatabase/Yale%20Face%20Database.htm`.
   d) The Carnegie Mellon University PIE data set[2] contains 41 368 images of 68 people, each person under 13 different poses, 43 different illumination conditions, and with four different expressions. We choose two near-frontal poses (C05 and C27) and use all the images under different illuminations and expressions. In total, there are 6661 images, where each individual has about 98 images. All the face images are manually aligned and cropped to 64×64. The database used in this paper is downloaded from `http://www.cad.zju.edu.cn/home/dengcai/Data/FaceData.html`
   e) $Feret$ database contains 1564 sets of images for a total of 14 126 images that includes 1199 individuals and 365 duplicate sets of images. The database of $Feret$ is available at `http://www.itl.nist.gov/iad/humanid/feret/feret_-master.html`. A subset of

[2] `http://www.ri.cmu.edu/projects/project_418.html`

Feret database was used in our experiment. This subset includes 1000 sets of images of 200 individuals each of which has five images. It consists of the images marked with two lowercase character string: 1) *ba*; 2) *bj*; 3) *bk*; 4) *be*; and 5) *bf*. The facial portion of each original image was cropped and resized to 80×80 pixels.

3) gene expression:
   a) Data set *Brain* presented in [20] contains 42 microarray gene expression profiles from five different tumors of the central nervous system, that is, ten medulloblastomas, ten malignant gliomas, ten atypical teratoid/ rhabdoid tumors, eight primitive neuroectodermal tumors, and four human cerebella. The raw data were originated using the Affymetrix technology and are publicly available at `http://www-genome.wi.mit.edu/cancer`. The processed data set is available at ftp://stat.ethz.ch/Manuscripts/dettling/brain.rda.
   b) Data set *Prostate* are available at `cancerghttp://www-genome.wi.mit.edu/cancer` and comprise the expression of 52 prostate tumors and 50 nontumor prostate samples, obtained using the Affymetrix technology. It was processed in [7] and the data set is available at `ftp://stat.ethz.ch/Manuscripts/dettling/prostate.rda`.

K-nearest neighbor (K-NN) method [8] is a popular method for classification that gives the maximum likelihood estimation of the class posterior probabilities by finding the closest K training points according to some metric, e.g., Euclidean, Manhattan, and so on. In our experiments, K-NN with $K = 1$, which predicts the same class as the nearest instance in the training set, measured by Euclidean distance is used as classification algorithm. CPU time in this experiment is recorded by MATLAB command *tic toc*, which can record the small number of the execution time.

C. Comparison for the Case of One New Sample Insertion

For all data sets used in this section, we perform our study by repeated random splitting into two groups using the following algorithm. Within each class, we randomly reorder the data and then for each class with size n_i, the first $\lceil 0.5n_i \rceil$ data are sorted into Group I and the others are sorted into Group II, whereby $\lceil \cdot \rceil$ is the ceiling function. Initially, we select the first $\lceil 0.5k \rceil$ classes of samples from Group I for training, while the others are inserted into the training set one-by-one incrementally. Incremental learning is completed until all samples in Group I are added into the training set. The classification accuracy of the final updated transformation matrix is then computed using Group II as the test data. The computational time is the CPU time of updating the transformation matrix for one single insertion. For each algorithm, to reduce the variability, this process is repeated for ten times, and the average results are recorded.

TABLE VII
COMPARISON OF ULDA/QR [6], LDA/QR, AND ILDA/QR

Data	Method	Accuracy	Standard Deviation
$K1a$	ULDA/QR	80.34	0.63
	SDA	80.34	0.67
	LDA/QR	**82.37**	0.84
	ILDA/QR	**82.37**	0.84
$Tr23$	ULDA/QR	74.40	3.04
	SDA	74.50	3.10
	LDA/QR	**80.90**	2.34
	ILDA/QR	**80.90**	2.34
Wap	ULDA/QR	76.40	1.03
	SDA	76.40	1.08
	LDA/QR	**80.05**	1.45
	ILDA/QR	**80.05**	1.45
$ORL_{32\times32}$	ULDA/QR	91.25	1.63
	SDA	91.25	1.72
	LDA/QR	**91.35**	1.55
	ILDA/QR	**91.35**	1.55
$AR_{50\times45}$	ULDA/QR	79.15	0.78
	SDA	79.15	0.82
	LDA/QR	**79.36**	0.70
	ILDA/QR	**79.36**	0.70
$YaleB$	ULDA/QR	93.29	1.28
	SDA	93.29	1.35
	LDA/QR	**94.23**	1.31
	ILDA/QR	**94.23**	1.31
PIE	ULDA/QR	93.38	0.22
	SDA	**94.65**	0.36
	LDA/QR	93.35	0.26
	ILDA/QR	93.35	0.26
$Feret$	ULDA/QR	69.88	1.63
	SDA	69.88	2.70
	LDA/QR	**70.28**	1.46
	ILDA/QR	**70.28**	1.46
$Brain$	ULDA/QR	80.00	5.55
	SDA	**83.33**	6.04
	LDA/QR	81.90	5.13
	ILDA/QR	81.90	5.13
$Prostate$	ULDA/QR	91.57	2.78
	SDA	**92.50**	2.41
	LDA/QR	91.57	2.78
	ILDA/QR	91.57	2.78

1) Comparison With Batch LDA Algorithms LDA/QR, ULDA/QR [6], and SDA [29]: We compare the performance of ILDA/QR with batch LDA algorithms LDA/QR, ULDA/QR [6], and SDA [29] first.

For batch algorithms LDA/QR, ULDA/QR, and SDA, when a new data sample is inserted, the construction of optimal transformation G is achieved from scratch. That is, LDA/QR, ULDA/QR [6], and SDA [29] will repeat the learning from the beginning whenever one additional sample is presented, and the knowledge acquired in the past is discarded. The CPU time for LDA/QR and ULDA/QR [6] is the total time of computing the optimal transformation.[3] While for incremental algorithm, ILDA/QR, the optimal transformation is updated from the previously obtained information when a new sample is added. Thus, the CPU time for ILDA/QR to construct the optimal transformation is just the updating time.

The results of mean classification accuracies of the final optimal transformation and ten times standard deviation are shown in Table VII. To give a concrete idea of the benefit

[3]Since SDA [29] is much more time consuming than ULDA/QR [6], we do not plot the CPU times of SDA in Fig. 1.

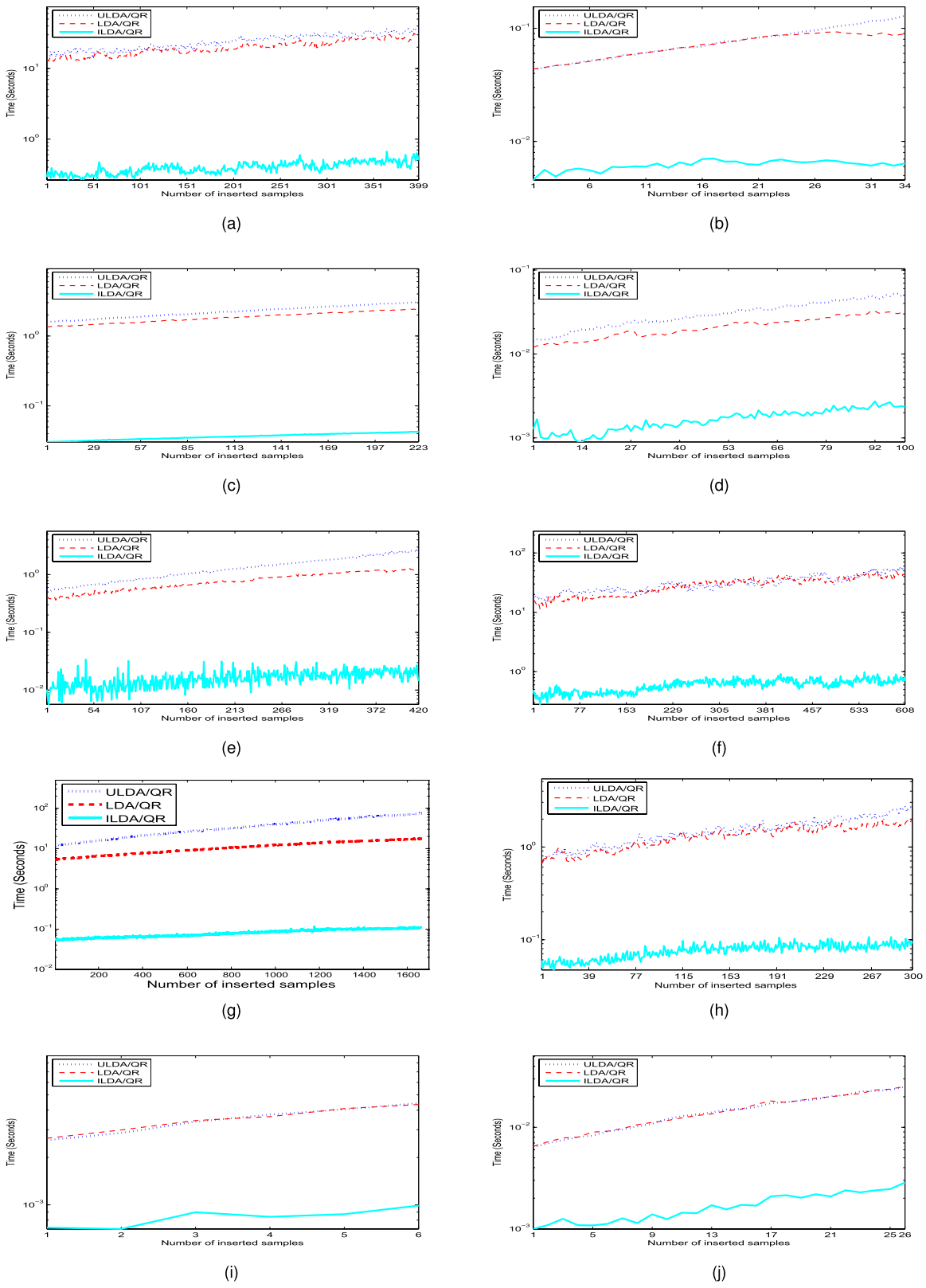


Fig. 1. Comparison of the CPU time for LDA Algorithms ULDA/QR [6], LDA/QR, and ILDA/QR (measured in seconds and plotted using MATLAB code semilogy). (a) CPU time for $K1a$. (b) CPU time for $Tr23$. (c) CPU time for Wap. (d) CPU time for $ORL_{32\times32}$. (e) CPU time for $AR_{50\times45}$. (f) CPU time for $YaleB$. (g) CPU time for PIE. (h) CPU time for $Feret$. (i) CPU time for $Brain$. (j) CPU time for $Prostate$.

of using incremental method from the perspective of the computational efficiency, we show a comparison on the execution time of ILDA/QR with LDA/QR and ULDA/QR for each single updating in Fig. 1. In the figures, the horizontal axis shows the number of new inserted data items, and the vertical axis indicates the CPU time (seconds) of computing

the transformation matrix.

Main observations from Table VII and Fig. 1 are as follows.

1) ILDA/QR achieves the same accuracy as that of LDA/QR, which coincides with our theoretical analysis that our new proposed incremental algorithm ILDA/QR is an exact incremental scheme of LDA/QR.
2) ILDA/QR and LDA/QR are comparative with ULDA/QR and SDA in terms of classification accuracy.
3) ILDA/QR is much faster than LDA/QR and ULDA/QR [6]. As more new samples are inserted, that is, the sample size n increases, the speedup of incremental algorithm ILDA/QR over batch algorithms LDA/QR and ULDA/QR keeps increasing.

2) Comparison With Existing Incremental LDA Algorithms: In the following, we compare the performance of our new proposed incremental algorithm ILDA/QR with four existing ILDA algorithms IDR/QR [26], ILDA/SSS [15], [16], LS-ILDA [17], and ICLDA [18].

For ILDA/SSS [16], we used the MATLAB code written by one of the authors on the website `http://www.iis.ee.ic.ac.uk/~tkkim/code.htm`. While for IDR/QR [26], LS-ILDA [17], and ICLDA [18], we wrote the MATLAB codes that follow their algorithms presented in [17], [18], and [26]. ILDA/SSS has three parameters: 1) the threshold for significant components of the total scatter matrix; 2) the threshold for significant components of the between-class scatter matrix; and 3) the threshold for the discriminative components. The authors of ILDA/SSS did not give the best parameter setting either in this paper or on the website `http://www.iis.ee.ic.ac.uk/~tkkim/code.htm`, we used the same threshold 0.1 for these three parameters as selected in [17], which enables the algorithm to achieve its best performance. The regularization parameter μ in IDR/QR was set to be 0.5, which produced good overall results in [26].

The execution time for incremental algorithms in our experiments is the total CPU time of updating the optimal transformation matrix in a single insertion, and the classification accuracy is of the final transformation when incremental updating is completed. The results of mean classification accuracies of ten times different splitting of data and the corresponding standard deviation are shown in Table VIII, the mean execution time of each updating is shown in Fig. 2, in which the horizontal axis shows the number of inserted samples while the vertical axis indicates the execution time (in log scale) of different tested methods.[4]

Note that the earlier versions of MATLAB counted the flops. With the incorporation of LAPACK in MATLAB 6 (and newer versions) this is no longer practical since many optimization techniques are adopted in LAPACK, and so the algorithm flop analysis in Table II and the running time may not be consistent.

The following observations can be made from Table VIII and Fig. 2.

[4]Since ILDA/SSS and ICLDA are too time consuming, the results of ILDA/SSS and ICLDA on PIE data set below are ones of the first splitting of the data.

TABLE VIII
COMPARISON OF IDR/QR, ILDA/SSS, LS-ILDA, ICLDA, AND ILDA/QR

Data	Method	Accuracy	Standard Deviation
$K1a$	IDR/QR	80.71	0.37
	ILDA/SSS	80.34	0.63
	LS-ILDA	80.34	0.63
	ICLDA	76.62	9.35
	ILDA/QR	**82.37**	0.84
$Tr23$	IDR/QR	72.80	5.31
	ILDA/SSS	74.40	3.04
	LS-ILDA	74.40	3.04
	ICLDA	74.10	2.77
	ILDA/QR	**80.90**	2.34
Wap	IDR/QR	79.06	1.37
	ILDA/SSS	76.40	1.03
	LS-ILDA	76.40	1.03
	ICLDA	75.57	2.60
	ILDA/QR	**80.05**	1.45
$ORL_{32\times32}$	IDR/QR	92.80	1.12
	ILDA/SSS	91.25	1.63
	LS-ILDA	91.25	1.63
	ICLDA	**97.00**	1.02
	ILDA/QR	91.35	1.55
$AR_{50\times45}$	IDR/QR	71.33	1.09
	ILDA/SSS	79.15	0.78
	LS-ILDA	79.15	0.78
	ICLDA	**87.71**	1.11
	ILDA/QR	79.36	0.70
$YaleB$	IDR/QR	75.40	1.52
	ILDA/SSS	84.40	1.68
	LS-ILDA	93.29	1.28
	ICLDA	92.72	1.15
	ILDA/QR	**94.23**	1.31
PIE	IDR/QR	**95.48**	0.5
	ILDA/SSS	91.56	
	LS-ILDA	93.38	0.22
	ICLDA	93.15	
	ILDA/QR	93.35	0.26
$Feret$	IDR/QR	51.08	2.51
	ILDA/SSS	69.88	1.63
	LS-ILDA	69.88	1.63
	ICLDA	**85.60**	1.30
	ILDA/QR	70.28	1.46
$Brain$	IDR/QR	81.43	6.19
	ILDA/SSS	80.00	5.55
	LS-ILDA	80.00	5.55
	ICLDA	79.52	5.65
	ILDA/QR	**81.90**	5.13
$Prostate$	IDR/QR	74.12	7.00
	ILDA/SSS	**91.57**	2.78
	LS-ILDA	**91.57**	2.78
	ICLDA	**91.57**	2.78
	ILDA/QR	**91.57**	2.78

1) ILDA/QR always achieves a comparative classification accuracy compared with the four incremental algorithms IDR/QR, ILDA/SSS, LS-ILDA, and ICLDA.
   a) For data sets $AR_{50\times45}$, $YaleB$, $Feret$, and $Prostate$, IDR/QR produces relatively lower accuracies compared with the other four algorithms. This is mainly caused by its discarding much useful information during the updating process.
   b) It is interesting to note that LS-ILDA achieves almost the same accuracies as ILDA/SSS. However, the performance of ILDA/SSS is largely

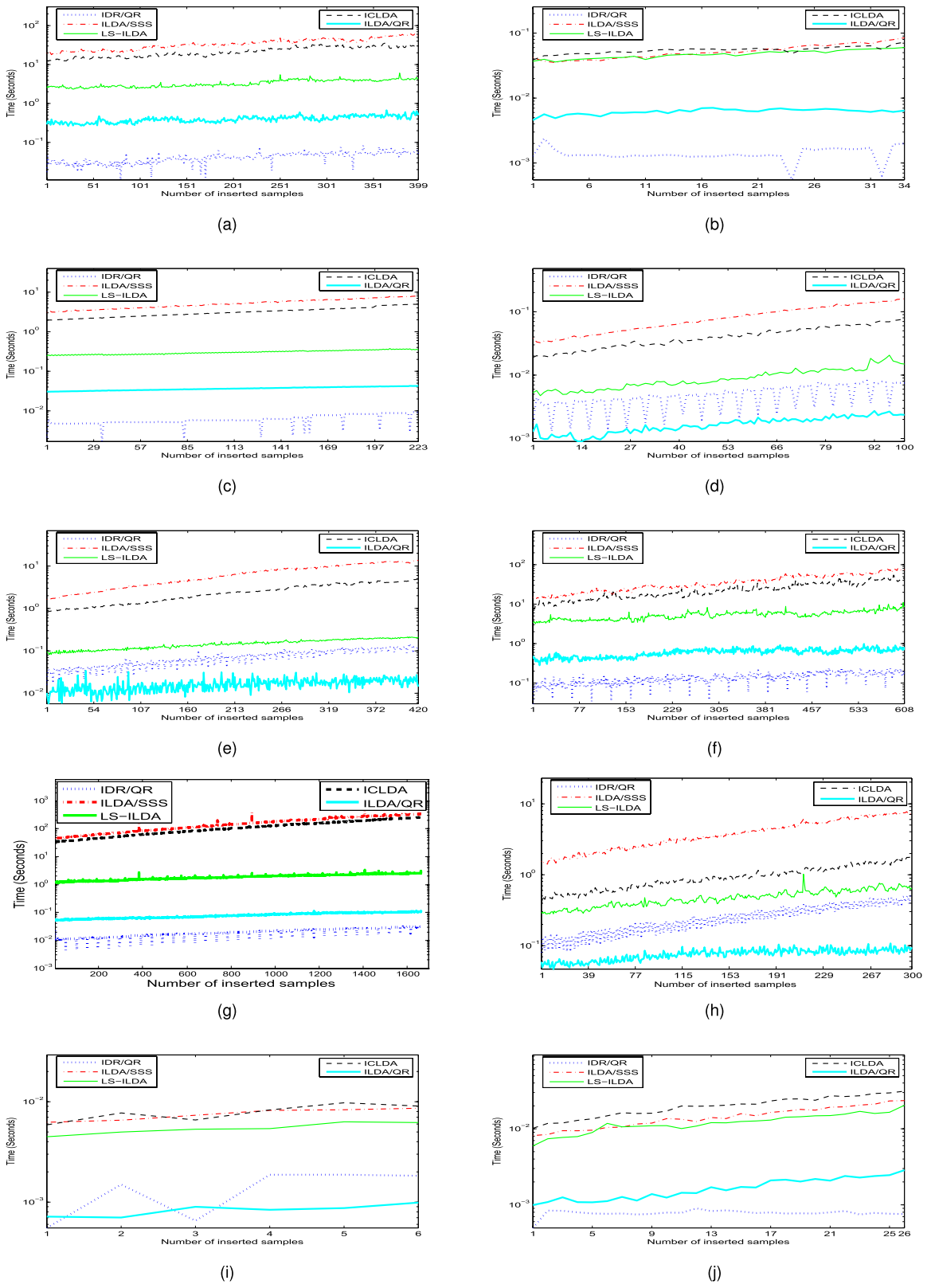


Fig. 2. Comparison of the CPU time of IDR/QR, ILDA/SSS, LS-ILDA, ICLDA, and ILDA/QR (measured in seconds and plotted using MATLAB code semilogy). (a) CPU time for $K1a$. (b) CPU time for $Tr23$. (c) CPU time for Wap. (d) CPU time for $ORL_{32\times32}$. (e) CPU time for $AR_{50\times45}$. (f) CPU time for $YaleB$. (g) CPU time for PIE. (h) CPU time for $Feret$. (i) CPU time for $Brain$. (j) CPU time for $Prostate$.

determined by the threshold. As shown in Table IX, the accuracy deviation between ILDA/SSS with thresholds 0.1 and 1 exceeds 10%.

2) The execution times in a single updating by IDR/QR and ILDA/QR are significantly smaller than those by ILDA/SSS, LS-ILDA, and ICLDA.

TABLE IX
COMPARISON OF CLASSIFICATION ACCURACIES OF ILDA/SSS WITH DIFFERENT THRESHOLDS: 0.1 AND 1

Data	$ORL_{32\times32}$	$Brain$
threshold 0.1	91.25	80.00
threshold 1	80.10	57.61

a) ILDA/QR and IDR/QR are very fast, much faster than ILDA/SSS, LS-ILDA, and ICLDA. ILDA/QR is faster than IDR/QR on face imagine data sets $ORL_{32\times32}$, $AR_{32\times32}$, and $Feret$, while IDR/QR is faster than ILDA/QR on text document data sets. This is understandable, since when $n \gg k$, IDR/QR costs less than ILDA/QR, otherwise, ILDA/QR performs faster.
b) ILDA/SSS and ICLDA are the two slowest incremental algorithms.
c) LS-ILDA is faster than ICLDA and ILDA/SSS but much slower than IDR/QR and ILDA/QR.

Remark 5.2: From Table II, LS-ILDA has an acceptable computational complexity (flops) and it should cost about four times more than ILDA/QR, but Fig. 2 shows that it is in general approximately eight times slower than ILDA/QR. Our MATLAB code for LS-ILDA is the direct MATLAB implementation of the pseudocode given in [17]. The difference in running time between LS-ILDA and ILDA/QR is given by implementation or running details rather than algorithm complexity (flops).

D. Comparison for the Case of One Chunk of New Samples Insertion

Again, for all data sets used in this section, we perform our study by repeated random splitting into two groups using the following algorithm. Within each class, we randomly reorder the data and then for each class with size n_i, the first $\lceil 0.5n_i \rceil$ data are sorted into Group I and the others are sorted into Group II, whereby $\lceil \cdot \rceil$ is the ceiling function. Initially, we select the first $\lceil 0.5k \rceil$ classes of samples from Group I for training, while the others are separated into certain chunks and then are inserted into the training set chunk-by-chunk incrementally. The chunk structures of these chunks are as follows. Incremental learning is completed until all chunks of samples in Group I are added into the training set. The classification accuracy of the final updated transformation matrix is then computed using Group II as test data. The computational cost is the CPU time of updating the transformation matrix for the insertion of a chunk data. For each algorithm, to reduce the variability, this process is repeated for ten times, and the average results are recorded.

1) There are ten chunks for $K1a$, each of the first nine chunks contains 40 samples, but the last one contains 39 sample.
2) There are 12 chunks for $Tr23$, each of the first 11 chunks contains three samples, but the last one contains only one sample.
3) There are ten chunks for Wap, each of the first nine chunks contains 22 samples, but the last one contains 23 samples.

TABLE X
COMPARISON OF LDA/QR, ILDA/QR, AND ILDA/QR (CHUNK)

Data	Method	Accuracy	Standard Deviation
$K1a$	LDA/QR	82.37	0.84
	ILDA/QR	82.37	0.84
	ILDA/QR(Chunk)	82.37	0.84
$Tr23$	LDA/QR	80.90	2.34
	ILDA/QR	80.90	2.34
	ILDA/QR(Chunk)	80.90	2.34
Wap	LDA/QR	80.05	1.45
	ILDA/QR	80.05	1.45
	ILDA/QR(Chunk)	80.05	1.45
$ORL_{32\times32}$	LDA/QR	91.35	1.55
	ILDA/QR	91.35	1.55
	ILDA/QR(Chunk)	91.35	1.55
$AR_{50\times45}$	LDA/QR	79.36	0.70
	ILDA/QR	79.36	0.70
	ILDA/QR(Chunk)	79.36	0.70
$YaleB$	LDA/QR	94.23	1.31
	ILDA/QR	94.23	1.31
	ILDA/QR(Chunk)	94.23	1.31
PIE	ULDA/QR	93.35	0.26
	LDA/QR	93.35	0.26
	ILDA/QR(Chunk)	93.35	0.26
$Feret$	LDA/QR	70.28	1.46
	ILDA/QR	70.28	1.46
	ILDA/QR(Chunk)	70.28	1.46
$Brain$	LDA/QR	81.90	5.13
	ILDA/QR	81.90	5.13
	ILDA/QR(Chunk)	81.90	5.13
$Prostate$	LDA/QR	91.57	2.78
	ILDA/QR	91.57	2.78
	ILDA/QR(Chunk)	91.57	2.78

4) There are ten chunks for $ORL_{32\times32}$, each chunk contains ten samples.
5) There are ten chunks for $AR_{50\times45}$, each chunk contains 42 samples.
6) There are ten chunks for $YaleB$, each of the first nine chunks contains 61 samples, but the last one contains 59 samples.
7) There are ten chunks for PIE, each of the first nine chunks contains 167 samples, but the last one contains 158 samples.
8) There are ten chunks for $Feret$, each chunk contains 30 samples.
9) There are three chunks for $Brain$, each chunk contains two samples.
10) There are six chunks for $Prostate$, each of the first five chunks contains five samples, but the last one contains only one sample.

1) Comparison With LDA/QR and Sequential ILDA/QR: We compare the performance of the chunk version of ILDA/QR with LDA/QR and the sequential implementation of ILDA/QR here.

To demonstrate the importance of the ability of our ILDA/QR for a chunk of new samples added, in this comparison, we also consider a chunk of new samples $x_1, \ldots, x_\mu$ as μ new samples inserted one-by-one incrementally, then we apply our incremental ILDA/QR μ times to complete the insertion of these μ new samples.

The results of mean classification accuracies of the final optimal transformation and ten times standard deviation are shown in Table X. Their execution times for updating each chunk of new samples are shown in Fig. 3.

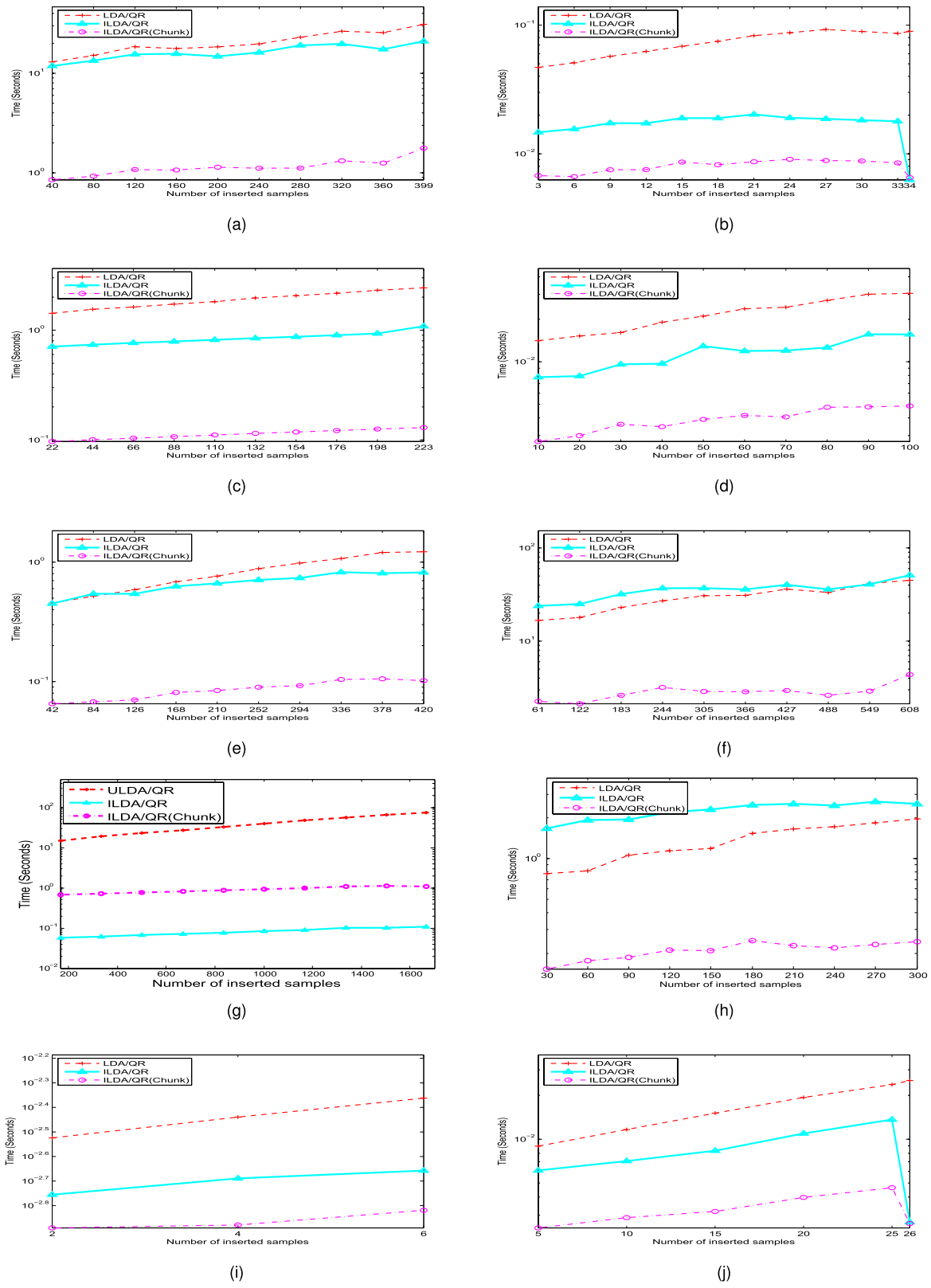


Fig. 3. Comparison of the CPU time of LDA/QR, ILDA/QR, and chunk ILDA/QR (measured in seconds and plotted using MATLAB code semilogy). (a) CPU time for $K1a$. (b) CPU time for $Tr23$. (c) CPU time for Wap. (d) CPU time for $ORL_{32\times32}$. (e) CPU time for $AR_{50\times45}$. (f) CPU time for $YaleB$. (g) CPU time for PIE. (h) CPU time for $Feret$. (i) CPU time for $Brain$. (j) CPU time for $Prostate$.

Table X and Fig. 3 indicate the following aspects.

1) The chunk version of ILDA/QR achieves the same accuracy as that of LDA/QR and the sequential implementation of ILDA/QR.
2) The chunk version of ILDA/QR is much faster than its sequential implementation and LDA/QR [6]. Clearly, it is very important that an efficient ILDA algorithm must be able to efficiently handle the case that a

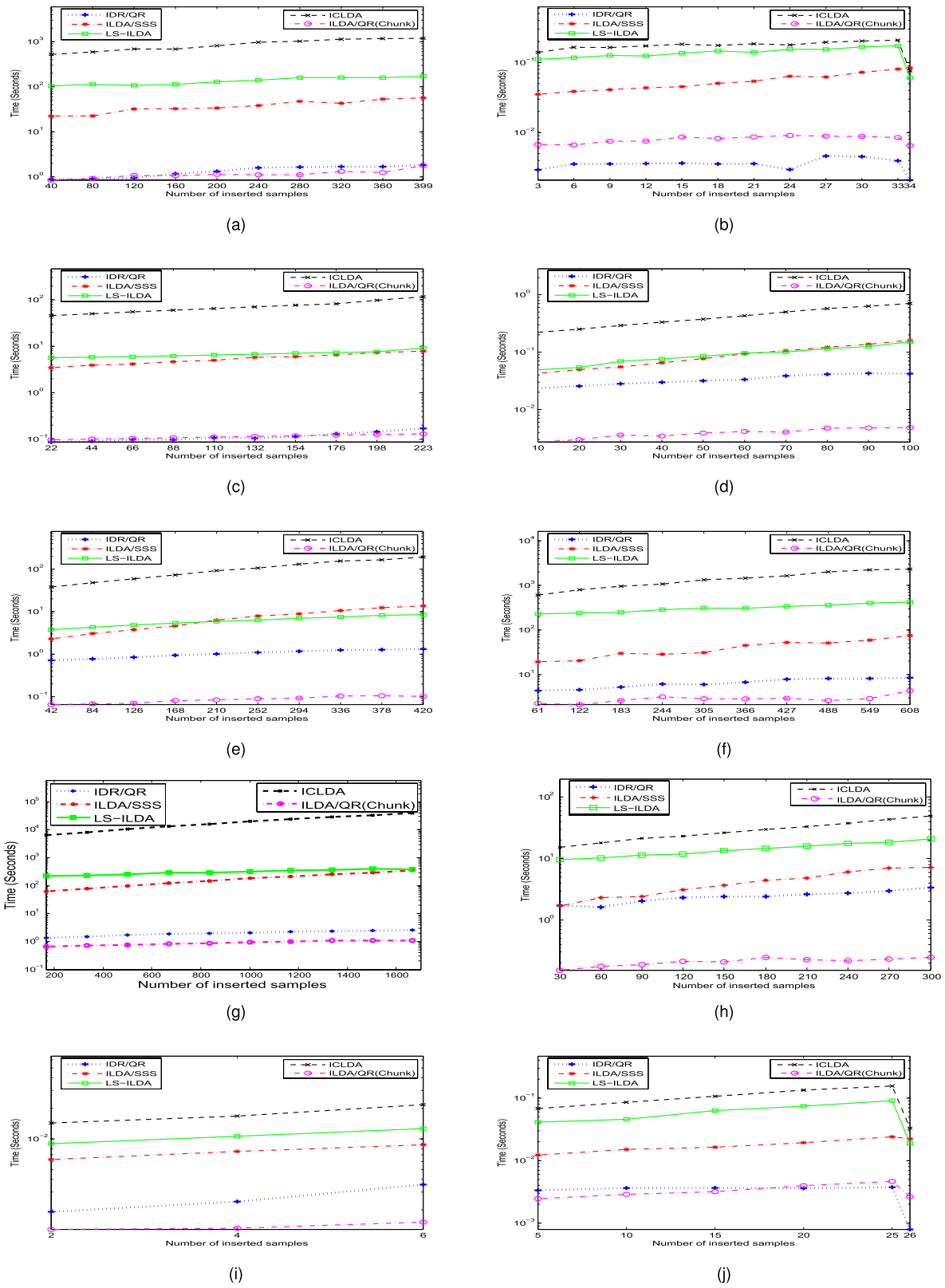


Fig. 4. Comparison of the CPU time of IDR/QR, ILDA/SSS, LS-ILDA, ICLDA, and chunk ILDA/QR (measured in seconds and plotted using MATLAB code semilogy). (a) CPU time for $K1a$. (b) CPU time for $Tr23$. (c) CPU time for Wap. (d) CPU time for $ORL_{32\times32}$. (e) CPU time for $AR_{50\times45}$. (f) CPU time for $YaleB$. (g) CPU time for PIE. (h) CPU time for $Feret$. (i) CPU time for $Brain$. (j) CPU time for $Prostate$.

chunk of new samples are inserted. For the case that a chunk of new samples are added, it is necessary to apply the chunk version, not the sequential implementation.

2) Comparison With Existing Incremental LDA Algorithms: Since only ILDA/QR and ILDA/SSS have chunk versions, but, IDR/QR, LS-ILDA, and ICLDA have no chunk versions available, in the following comparisons, for a chunk of new

TABLE XI
COMPARISON OF IDR/QR, ILDA/SSS, LS-ILDA, ICLDA, AND CHUNK ILDA/QR

Data	Method	Accuracy	Standard Deviation
$K1a$	IDR/QR	80.71	0.37
	ILDA/SSS	80.34	0.63
	LS-ILDA	80.34	0.63
	ICLDA	76.62	9.35
	ILDA/QR(Chunk)	**82.37**	0.84
$Tr23$	IDR/QR	72.80	5.31
	ILDA/SSS	74.40	3.04
	LS-ILDA	74.40	3.04
	ICLDA	74.10	2.77
	ILDA/QR(Chunk)	**80.90**	2.34
Wap	IDR/QR	79.06	1.37
	ILDA/SSS	76.40	1.03
	LS-ILDA	76.40	1.03
	ICLDA	75.17	3.74
	ILDA/QR(Chunk)	**80.05**	1.45
$ORL_{32\times32}$	IDR/QR	92.80	1.12
	ILDA/SSS	91.25	1.63
	LS-ILDA	91.25	1.63
	ICLDA	**97.00**	1.02
	ILDA/QR(Chunk)	91.35	1.55
$AR_{50\times45}$	IDR/QR	71.33	1.09
	ILDA/SSS	79.15	0.78
	LS-ILDA	79.15	0.78
	ICLDA	**87.71**	1.11
	ILDA/QR(Chunk)	79.36	0.70
$YaleB$	IDR/QR	75.40	1.52
	ILDA/SSS	84.40	1.68
	LS-ILDA	93.29	1.28
	ICLDA	92.72	1.15
	ILDA/QR(Chunk)	**94.23**	1.31
PIE	IDR/QR	**95.48**	0.5
	ILDA/SSS	91.56	
	LS-ILDA	93.38	0.22
	ICLDA	93.15	
	ILDA/QR(Chunk)	93.35	0.26
$Feret$	IDR/QR	51.08	2.51
	ILDA/SSS	69.88	1.63
	LS-ILDA	69.88	1.63
	ICLDA	**85.60**	1.30
	ILDA/QR(Chunk)	70.28	1.46
$Brain$	IDR/QR	81.43	6.19
	ILDA/SSS	80.00	5.55
	LS-ILDA	80.00	5.55
	ICLDA	79.52	5.65
	ILDA/QR(Chunk)	**81.90**	5.13
$Prostate$	IDR/QR	74.12	7.00
	ILDA/SSS	**91.57**	2.78
	LS-ILDA	**91.57**	2.78
	ICLDA	**91.57**	2.78
	ILDA/QR(Chunk)	**91.57**	2.78

samples $x_1, \ldots, x_\mu$, we consider them as μ new samples inserted one-by-one incrementally, then we apply IDR/QR, LS-ILDA, and ICLDA μ times to complete the insertion of these μ new samples.

The results of mean classification accuracies of the final optimal transformation and ten times standard deviation are shown in Table XI. Their execution times for updating each chunk of new samples are shown in Fig. 4.

We can observe the following from Table XI and Fig. 4.

1) ILDA/QR (chunk) always achieves a comparative classification accuracy compared with the four incremental algorithms IDR/QR, ILDA/SSS, LS-ILDA, and ICLDA.
2) For data sets $AR_{50\times45}$, $YaleB$, $Feret$, and $Prostate$, IDR/QR produces relatively lower accuracies compared with the other four algorithms.
3) IDR/QR and ILDA/QR (chunk) are much faster than ILDA/SSS, LS-ILDA, and ICLDA.
4) LS-ILDA has no chunk version; it is not clear how an efficient chunk version for LS-ILDA can be developed. However, it is very important that an efficient incremental algorithm must be able to efficiently handle the case that a chunk of new samples are inserted. When a chunk of new samples are added, it is necessary to apply the chunk version, not the sequential manner. The importance of this issue is obvious from Fig. 4. When only one new sample is inserted, ILDA/QR is about eight times faster than LS-ILDA. However, when a chunk of new samples are inserted, ILDA/QR is significantly faster than LS-ILDA. For example, ILDA/QR is $\sim$100 times faster for $K1a$ and $YaleB$ and 60 times faster for Wap and $AR_{50\times45}$.

In conclusion, ILDA/QR is the best choice among the five compared incremental algorithms in terms of classification accuracy, computational complexity, and space complexity. It provides an efficient incremental dimensionality reduction for large-scale streaming data sets.

VI. CONCLUSION

In this paper, we first propose a new batch LDA algorithm LDA/QR which is closely related to ULDA [6]. Our LDA/QR is obtained by computing the economic QR factorization of the data matrix and then solving a lower triangular linear system. Hence, LDA/QR is a simple and fast LDA algorithm. Its efficiency has been demonstrated by some real-world data. Based on this LDA/QR, we develop an ILDA algorithm ILDA/QR which is the exact incremental version of LDA/QR. Our ILDA algorithm ILDA/QR has the following features.

1) It can easily handle not only the case that only one new sample is inserted, but also the case that a chunk of new samples are added.
2) It has pleasant computational complexity and space requirement.
3) It is very fast and always achieves comparative classification accuracy compared with ULDA algorithm and the existing ILDA algorithms.

Numerical experiments show that our ILDA/QR is fast, efficient, and competitive with the state-of-the-art ILDA algorithms in terms of classification accuracy, computational complexity, and space requirement.

All main results in this paper are based on a mild assumption that all samples in the training set are linearly independent. This assumption holds for almost all high-dimensional and undersampled data. Similar singularity problems also occur for LS-ILDA [17] and IDR/QR [26]. For example, for undersampled problems, LS-ILDA [17] is developed under the assumption that $\text{rank}(A) - \text{rank}(A(I - (1/n)ee^T)) = 1$. To overcome such singularity problems, regularization approaches have been adopted in [23], [24], [26], and [28]. To extend our LDA/QR and ILDA/QR to the general case, it is important to study how the regularization approach can be efficiently incorporated into LDA/QR and ILDA/QR so that they can efficiently handle the general cases, including oversampled problems. This is an interesting topic for our future research.

Recently, an approach is proposed in [21] to feature extraction that is a generalization of LDA on the basis of deep neural network. This generalized discriminant analysis uses nonlinear transformations that are learnt by deep neural networks in a semisupervised fashion. It has been demonstrated that the feature extraction based on this approach displays excellent performance on real-world recognition and detection tasks. Furthermore, it has also been shown that this approach can preprocess truly high-dimensional input data to low-dimensional representations that facilitate accurate predictions even if simple linear predictors or measures of similarity are used. Mixed SDA (MSDA) is investigated and a theoretical link between MSDA and a restricted Gaussian model is presented in [11]. This link allows the derivation of a new discriminant analysis method under the expectation maximization framework. In addition, fractional step MSDA and kernel MSDA are also proposed to alleviate the subclass separation problem of MSDA and to handle cases where MSDA subclasses are not linearly separable, respectively. MSDA (and other subclass variants of LDA) can resolve the problem of nonlinearly separable classes as long as a subclass division that results in linearly separable subclasses is identified. If this is not possible, a subclass-based approach that can deal with nonlinearly separable subclasses is desirable, often using an appropriate kernel to map the nonlinearly separable subclass divisions into a new space where they are linearly separable [11]. Kernel mapping is one of the most used approaches to intrinsically derive nonlinear classifiers. A major problem in the design of kernel methods is to find the kernel parameters that make the problem linear in the mapped representation. In [27], a criterion is derived that specifically aims to find a kernel representation where the Bayes classifier becomes linear. This result is successfully applied in several kernel discriminant analysis algorithms. Experimental results using a large number of databases and classifiers demonstrate the utility of the proposed approach. It was also shown (theoretically and experimentally) that a kernel version of SDA yields the highest recognition rates.

The generalized discriminant analysis methods above can outperform LDA and other LDA variants as extensive experimentation shows in [11], [21], and [27]. There are no any incremental versions of these powerful methods available yet. Therefore, it is an important research topic in incremental learning that should be studied in the future.

APPENDIX

Proof of Lemma 3.3: For $i = 1, \ldots, n$, define

$$\mathcal{S}_i = \left\{ A = \left[v_1 \quad \cdots \quad v_{i-1} \quad \sum_{j=1}^{i-1} \beta_j v_j + \sum_{j=i+1}^{n} \beta_j v_j \quad v_{i+1} \quad \cdots \quad v_n \right] : v_j \in \mathbf{R}^m, \ \beta_j \in \mathbf{R}, \ j = 1, \ldots, i-1, i+1, \ldots, n \right\}.$$

Then, for each $i = 1, \ldots, n$, $\mathcal{S}_i$ is a subset of $\mathbf{R}^{m \times n}$ and its dimension is

$$m(n-1) + (n-1) = mn - (m+1-n) < mn.$$

Since the dimension of $\mathbf{R}^{m \times n}$ is mn, so, the Lebesgue measure of $\mathcal{S}_i$ on $\mathbf{R}^{m \times n}$ is zero [4], i.e., $\mathcal{L}(\mathcal{S}_i) = 0$, $i = 1, \ldots, n$. Note that $\mathcal{S}_0 = \mathcal{S}_1 \cup \cdots \cup \mathcal{S}_n$, thus, we have that

$$0 \leq \mathcal{L}(\mathcal{S}_0) \leq \mathcal{L}(\mathcal{S}_1) + \cdots + \mathcal{L}(\mathcal{S}_n) = 0 + \cdots + 0 = 0$$

which gives that $\mathcal{L}(\mathcal{S}_0) = 0$.

Proof of Theorem 3.1: Since training samples are linearly independent, we have $\operatorname{rank}(A) = n$ which implies that A^T is of full row rank, thus, the linear system (14) is solvable.

For any $G \in \mathbf{R}^{m \times k}$ satisfying (14), we have

$$k \geq \operatorname{rank}(G) \geq \operatorname{rank}(A^T G) = \operatorname{rank}(E) = k$$

thus, $\operatorname{rank}(G) = k$. We get from (11) that

$$\begin{aligned} G^T S_w G &= G^T A \left(I - \begin{bmatrix} \mathcal{E}_1 & & \\ & \ddots & \\ & & \mathcal{E}_k \end{bmatrix} \right) A^T G \\ &= E^T \left(I - \begin{bmatrix} \mathcal{E}_1 & & \\ & \ddots & \\ & & \mathcal{E}_k \end{bmatrix} \right) E \\ &= E^T E - E^T \begin{bmatrix} \frac{1}{\sqrt{n_1}} e_1 & & \\ & \ddots & \\ & & \frac{1}{\sqrt{n_k}} e_k \end{bmatrix} \\ &\quad \times \begin{bmatrix} \frac{1}{\sqrt{n_1}} e_1 & & \\ & \ddots & \\ & & \frac{1}{\sqrt{n_k}} e_k \end{bmatrix}^T E \\ &= \begin{bmatrix} n_1 & & \\ & \ddots & \\ & & n_k \end{bmatrix} - \begin{bmatrix} \sqrt{n_1} & & \\ & \ddots & \\ & & \sqrt{n_k} \end{bmatrix} \\ &\quad \times \begin{bmatrix} \sqrt{n_1} & & \\ & \ddots & \\ & & \sqrt{n_k} \end{bmatrix}^T \\ &= 0. \end{aligned}$$

Therefore, equality (2) implies that

$$G^T S_t G = G^T S_b G. \tag{20}$$

We have using (11) again that

$$\begin{aligned} G^T S_b G &= G^T A \left(\begin{bmatrix} \mathcal{E}_1 & & \\ & \ddots & \\ & & \mathcal{E}_k \end{bmatrix} - \mathcal{E} \right) A^T G \\ &= E^T \left(\begin{bmatrix} \mathcal{E}_1 & & \\ & \ddots & \\ & & \mathcal{E}_k \end{bmatrix} - \mathcal{E} \right) E \\ &= \begin{bmatrix} n_1 & & \\ & \ddots & \\ & & n_k \end{bmatrix} - \frac{1}{n} \begin{bmatrix} n_1 \\ \vdots \\ n_k \end{bmatrix} [n_1 \quad \cdots \quad n_k]. \end{aligned} \tag{21}$$

So, $\operatorname{rank}(G^T S_b G) = k-1$, which together with (20) yields

$$\operatorname{trace}\left((G^T S_t G)^{(+)} G^T S_b G\right) = k-1.$$

Proof of Theorem 3.2: Since G is a solution of linear system (14), we have using (21) that

$$\begin{aligned} G^T S_t G &= G^T S_b G \\ &= \begin{bmatrix} n_1 & & \\ & \ddots & \\ & & n_k \end{bmatrix} - \frac{1}{n}\begin{bmatrix} n_1 \\ \vdots \\ n_k \end{bmatrix}[n_1 \quad \cdots \quad n_k] \\ &= \begin{bmatrix} \sqrt{n_1} & & \\ & \ddots & \\ & & \sqrt{n_k} \end{bmatrix}\left(I - \frac{1}{n}\begin{bmatrix} \sqrt{n_1} \\ \vdots \\ \sqrt{n_k} \end{bmatrix}\begin{bmatrix} \sqrt{n_1} \\ \vdots \\ \sqrt{n_k} \end{bmatrix}^T\right) \\ &\quad \times \begin{bmatrix} \sqrt{n_1} & & \\ & \ddots & \\ & & \sqrt{n_k} \end{bmatrix} \end{aligned}$$

thus

$$\begin{aligned} &\Psi^T G^T S_t G \Psi \\ &\quad = \Psi^T G^T S_b G \Psi \\ &\quad = [0 \quad I_{k-1}]\mathcal{H}\left(I - \frac{1}{n}\begin{bmatrix} \sqrt{n_1} \\ \vdots \\ \sqrt{n_k} \end{bmatrix}\begin{bmatrix} \sqrt{n_1} \\ \vdots \\ \sqrt{n_k} \end{bmatrix}^T\right)\mathcal{H}^T\begin{bmatrix} 0 \\ I_{k-1} \end{bmatrix} \\ &\quad = [0 \quad I_{k-1}]\left(I - \left(\mathcal{H}\begin{bmatrix} \sqrt{n_1} \\ \vdots \\ \sqrt{n_k} \end{bmatrix} \Big/ \sqrt{n}\right)\right. \\ &\qquad \left. \times \left(\mathcal{H}\begin{bmatrix} \sqrt{n_1} \\ \vdots \\ \sqrt{n_k} \end{bmatrix} \Big/ \sqrt{n}\right)^T\right)\begin{bmatrix} 0 \\ I_{k-1} \end{bmatrix} \\ &\quad = [0 \quad I_{k-1}]\left(I - \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}^T\right)\begin{bmatrix} 0 \\ I_{k-1} \end{bmatrix} \\ &\quad = I_{k-1}. \end{aligned}$$

Therefore, $G\Psi$ is an optimal solution of ULDA (15).

Proof of Theorem 3.3: Define Householder transformations

$$\begin{aligned} \mathcal{H}_i = I - &\left(\begin{bmatrix} 1-\sqrt{n_i} \\ 1 \\ \vdots \\ 1 \end{bmatrix} \Big/ \sqrt{n_i - \sqrt{n_i}}\right) \\ &\times \left(\begin{bmatrix} 1-\sqrt{n_i} \\ 1 \\ \vdots \\ 1 \end{bmatrix} \Big/ \sqrt{n_i - \sqrt{n_i}}\right)^T \end{aligned}$$

for $i = 1, \ldots, k$. Notice that the matrices $\mathcal{H}_i$ $(i = 1, \ldots, k)$ are orthogonal and satisfy

$$\mathcal{H}_i = \mathcal{H}_i^T, \quad \mathcal{H}_i^T(e_i/\sqrt{n_i}) = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad i = 1, \ldots, k.$$

Let $\mathcal{P}$ be the permutation matrix obtained by exchanging the $(\sum_{j=1}^{i-1} n_j + 1)$th column and the ith column (for $i = 2, \ldots, k$) of the $n \times n$ identity matrix, but otherwise leaving the order of the remaining columns unchanged. Denote

$$\begin{bmatrix} R_1 & R_2 & R_3 \end{bmatrix} = R\begin{bmatrix} \mathcal{H}_1 & & \\ & \ddots & \\ & & \mathcal{H}_k \end{bmatrix}\mathcal{P}\begin{bmatrix} \mathcal{H} & \\ & I \end{bmatrix} \tag{22}$$

where $R_1 \in \mathbf{R}^{n\times 1}$, $R_2 \in \mathbf{R}^{n\times(k-1)}$, and $R_3 \in \mathbf{R}^{n\times(n-k)}$. Let the QR factorization of $[R_2 \ R_3]$ be

$$[R_2 \quad R_3] = \mathcal{Q}\begin{bmatrix} R_{12} & R_{13} \\ 0 & R_{23} \\ 0 & 0 \end{bmatrix}$$

where $\mathcal{Q} \in \mathbf{R}^{n\times n}$ is orthogonal, $R_{12} \in \mathbf{R}^{(k-1)\times(k-1)}$, and $R_{23} \in \mathbf{R}^{(n-k)\times(n-k)}$. Since R is nonsingular, we have

$$\operatorname{rank}(R_{12}) = k-1, \ \operatorname{rank}(R_{23}) = n-k.$$

Set $\mathcal{Q}^T R_1 = \begin{bmatrix} R_{11} \\ R_{21} \\ R_{31} \end{bmatrix}$, where $R_{11} \in \mathbf{R}^{(k-1)\times 1}$, $R_{21} \in \mathbf{R}^{(n-k)\times 1}$, and $R_{31} \in \mathbf{R}^{1\times 1}$. Then

$$\begin{aligned} S_b &= \left(\mathcal{Q}\mathcal{Q}\begin{bmatrix} R_{12} \\ 0 \\ 0 \end{bmatrix}\right)\left(\mathcal{Q}\mathcal{Q}\begin{bmatrix} R_{12} \\ 0 \\ 0 \end{bmatrix}\right)^T \\ S_t &= \left(\mathcal{Q}\mathcal{Q}\begin{bmatrix} R_{12} & R_{13} \\ 0 & R_{23} \\ 0 & 0 \end{bmatrix}\right)\left(\mathcal{Q}\mathcal{Q}\begin{bmatrix} R_{12} & R_{13} \\ 0 & R_{23} \\ 0 & 0 \end{bmatrix}\right)^T \end{aligned}$$

so

$$S_t^{(+)} S_b = \mathcal{Q}\mathcal{Q}\begin{bmatrix} I_{k-1} & 0 & 0 \\ -R_{23}^{-T} R_{13}^T & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}(\mathcal{Q}\mathcal{Q})^T \tag{23}$$

that is

$$S_t^{(+)} S_b \mathcal{Q}\mathcal{Q}\begin{bmatrix} I_{k-1} \\ -R_{23}^{-T} R_{13}^T \\ 0 \end{bmatrix} = \mathcal{Q}\mathcal{Q}\begin{bmatrix} I_{k-1} \\ -R_{23}^{-T} R_{13}^T \\ 0 \end{bmatrix}. \tag{24}$$

Hence, the columns of $\mathcal{Q}\mathcal{Q}\begin{bmatrix} I_{k-1} \\ -R_{23}^{-T} R_{13}^T \\ 0 \end{bmatrix}$ span the eigenspace of $S^{(+)} S_b$ corresponding to its all nonzero eigenvalues.

On the other hand

$$\begin{aligned} R &= [R_1 \quad R_2 \quad R_3]\begin{bmatrix} \mathcal{H} & \\ & I \end{bmatrix}^T \mathcal{P}^T \begin{bmatrix} \mathcal{H}_1 & & \\ & \ddots & \\ & & \mathcal{H}_k \end{bmatrix}^T \\ &= \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & 0 & R_{23} \\ R_{31} & 0 & 0 \end{bmatrix}\begin{bmatrix} \mathcal{H} & \\ & I \end{bmatrix}^T \mathcal{P}^T \begin{bmatrix} \mathcal{H}_1 & & \\ & \ddots & \\ & & \mathcal{H}_k \end{bmatrix}^T \end{aligned}$$

and so, the solution G of the linear system (14) with the minimal 2-norm, which is given in (17), satisfies

$$\begin{aligned} G &= QR^{-T}E \\ &= \mathcal{Q}\begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & 0 & R_{23} \\ R_{31} & 0 & 0 \end{bmatrix}^{-T} \begin{bmatrix} \mathcal{H}^T \begin{bmatrix} \sqrt{n_1} & & \\ & \ddots & \\ & & \sqrt{n_k} \end{bmatrix} \\ 0 \end{bmatrix} \\ &= \mathcal{Q}\begin{bmatrix} 0 & I \\ 0 & -R_{23}^{-T}R_{13}^T \\ R_{31}^{-T} & 0 \end{bmatrix} \\ &\quad \times \begin{bmatrix} I & \left(R_{21}^T R_{23}^{-T} R_{13}^T - R_{11}^T\right)R_{12}^{-T} \\ 0 & R_{12}^{-T} \end{bmatrix} \mathcal{H}^T \\ &\quad \times \begin{bmatrix} \sqrt{n_1} & & \\ & \ddots & \\ & & \sqrt{n_k} \end{bmatrix}. \end{aligned} \tag{25}$$

Therefore, Theorem 3.3 follows directly from (24) and (25).

Proof of Lemma 3.4: In (23) above, $\mathcal{Q} \in \mathbf{R}^{n\times n}$ is the orthogonal, Q is the column orthogonal. Let $[Q\ \tilde{Q}] \in \mathbf{R}^{m\times m}$ be orthogonal. Then, $[Q\ \tilde{Q}]\begin{bmatrix} \mathcal{Q} & \\ & I \end{bmatrix}$ is the orthogonal and (23) gives that

$$\begin{aligned} S_t^{(+)}S_b &= \mathcal{Q}\mathcal{Q}\begin{bmatrix} I_{k-1} & 0 & 0 \\ -R_{23}^{-T}R_{13}^T & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}(\mathcal{Q}\mathcal{Q})^T \\ &= [Q\ \ \tilde{Q}]\begin{bmatrix} \mathcal{Q} & \\ & I_{m-n} \end{bmatrix}\begin{bmatrix} I_{k-1} & 0 & 0 & 0 \\ -R_{23}^{-T}R_{13}^T & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \\ &\quad \times \left([Q\ \ \tilde{Q}]\begin{bmatrix} \mathcal{Q} & \\ & I_{m-n} \end{bmatrix}\right)^T. \end{aligned}$$

Hence, we obtain that

$$\begin{aligned} \operatorname{trace}(S_t^{(+)}S_b) &= \operatorname{trace}\left(\begin{bmatrix} I_{k-1} & 0 & 0 & 0 \\ -R_{23}^{-T}R_{13}^T & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}\right) \\ &= \operatorname{trace}(I_{k-1}) \\ &= k-1. \end{aligned}$$

REFERENCES

[1] A. A. Alizadeh *et al.*, "Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling," *Nature*, vol. 403, no. 6769, pp. 503–511, 2000.

[2] D. Boley *et al.*, "Document categorization and query generation on the World Wide Web using WebACE," *J. Artif. Intell. Rev.*, vol. 13, nos. 5–6, pp. 365–391, 1999.

[3] D. Boley *et al.*, "Partitioning-based clustering for web document categorization," *J. Decision Support Syst.*, vol. 27, no. 3, pp. 329–341, Dec. 1999.

[4] F. Burk, *Lebesgue Measure and Integration: An Introduction*. New York, NY, USA: Wiley, 1998.

[5] D. Chu and G. S. Thye, "A new and fast implementation for null space based linear discriminant analysis," *Pattern Recognit.*, vol. 43, no. 4, pp. 1373–1379, Apr. 2010.

[6] D. Chu, G. S. Thye, and Y. S. Hung, "Characterization of all solutions for undersampled uncorrelated linear discriminant analysis problems," *SIAM J. Matrix Anal. Appl.*, vol. 32, no. 3, pp. 820–844, 2011.

[7] M. Dettling and P. Bühlmann, "Supervised clustering of genes," *Genome Biol.*, vol. 12, no. 3, pp. 1–15, 2002.

[8] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. New York, NY, USA: Wiley, 2000.

[9] K. Fukunaga, *Introduction to Statistical Pattern Recognition*. New York, NY, USA: Academic, 1990.

[10] A. S. Georghiades, P. N. Belhumeur, and D. Kriegman, "From few to many: Illumination cone models for face recognition under variable lighting and pose," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 6, pp. 643–660, Jun. 2001.

[11] N. Gkalelis, V. Mezaris, I. Kompatsiaris, and T. Stathaki, "Mixture subclass discriminant analysis link to restricted Gaussian model and other generalizations," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 1, pp. 8–21, Jan. 2013.

[12] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed. Baltimore, MD, USA: The Johns Hopkins Univ. Press, 1996.

[13] E.-H. Han *et al.*, "WebACE: A web agent for document categorization and exploration," in *Proc. 2nd Int. Conf. Auto. Agents*, Minneapolis, MN, USA, May 1997, pp. 408–415.

[14] P. Howland, M. Jeon, and H. Park, "Structure preserving dimension reduction for clustered text data based on the generalized singular value decomposition," *SIAM J. Matrix Anal. Appl.*, vol. 25, no. 1, pp. 165–179, 2003.

[15] T.-K. Kim, B. Stenger, J. Kittler, and R. Cipolla, "Incremental linear discriminant analysis using sufficient spanning sets and its applications," *Int. J. Comput. Vis.*, vol. 91, no. 2, pp. 216–232, 2011.

[16] T.-K. Kim, K.-Y. K. Wong, B. Stenger, J. Kittler, and R. Cipolla, "Incremental linear discriminant analysis using sufficient spanning set approximations," in *Proc. Comput. Vis. Pattern Recognit. Conf.*, Minneapolis, MN, USA, Jun. 2007, pp. 1–8.

[17] L.-P. Liu, Y. Jiang, and Z.-H. Zhou, "Least square incremental linear discriminant analysis," in *Proc. 9th IEEE Int. Conf. Data Mining*, Miami, FL, USA, Dec. 2009, pp. 298–306.

[18] G.-F. Lu, J. Zou, and Y. Wang, "Incremental learning of complete linear discriminant analysis for face recognition," *Knowl. Based Syst.*, vol. 31, pp. 19–27, Jul. 2012.

[19] 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, Appl. Rev.*, vol. 31, no. 4, pp. 497–508, Nov. 2001.

[20] S. L. Pomeroy, P. Tamayo, M. Gaasenbeek, and L. M. Sturla, "Prediction of central nervous system embryonal tumour outcome based on gene expression," *Nature*, vol. 415, pp. 436–442, Jan. 2002.

[21] A. Stuhlsatz, J. Lippel, and T. Zielke, "Feature extraction with deep neural networks by a generalized discriminant analysis," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 4, pp. 596–608, Apr. 2012.

[22] J. Yang and J.-Y. Yang, "Why can LDA be performed in PCA transformed space?" *Pattern Recognit.*, vol. 36, no. 2, pp. 563–566, Feb. 2003.

[23] J. Ye, "Characterization of a family of algorithms for generalized discriminant analysis on undersampled problems," *J. Mach. Learn. Res.*, vol. 6, pp. 483–502, Apr. 2005.

[24] J. Ye, "Least squares linear discriminant analysis," in *Proc. 24th Int. Conf. Mach. Learn.*, Corvallis, OR, USA, Jun. 2007, pp. 1087–1094.

[25] J. Ye, R. Janardan, Q. Li, and H. Park, "Feature reduction via generalized uncorrelated linear discriminant analysis," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 10, pp. 1312–1322, Oct. 2006.

[26] J. Ye, Q. Li, H. Xiong, H. Park, R. Janardan, and V. Kumar, "IDR/QR: An incremental dimension reduction algorithm via QR decomposition," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 9, pp. 1208–1222, Sep. 2005.

[27] D. You, O. C. Hamsici, and A. M. Martinez, "Kernel optimization in discriminant analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 3, pp. 631–638, Mar. 2011.

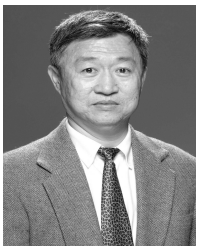
[28] Z. Zhang, G. Dai, C. Xu, and M. I. Jordan, "Regularized discriminant analysis, ridge regression and beyond," *J. Mach. Learn. Res.*, vol. 11, pp. 2199–2228, Aug. 2010.

[29] M. Zhu and A. M. Martinez, "Subclass discriminant analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 8, pp. 1274–1286, Aug. 2006.

Delin Chu received the Ph.D. degree from the Department of Applied Mathematics, Tsinghua University, Beijing, China, in 1991.

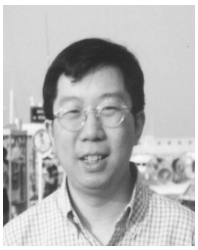
He is currently with the Department of Mathematics, National University of Singapore, Singapore. His current research interests include data mining, numerical linear algebra, scientific computing, numerical analysis, and matrix theory and computations.

Dr. Chu is also on the Editorial Boards of *Automatica* and the *Journal of Computational and Applied Mathematics*.



Li-Zhi Liao received the B.S. degree in applied mathematics from Tsinghua University, Beijing, China, in 1984, and the M.S. and Ph.D. degrees from the School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY, USA, in 1988 and 1990, respectively.

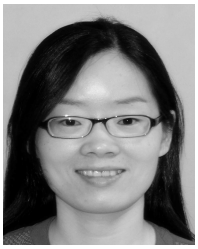
He is currently a Professor with the Department of Mathematics, Hong Kong Baptist University, Hong Kong. His current research interests include optimization, optimal control, variational inequality, and scientific computing.



Michael Kwok-Po Ng received the B.Sc. and M.Phil. degrees from the University of Hong Kong, Hong Kong, in 1990 and 1992, respectively, and the Ph.D. degree from the Chinese University of Hong Kong, Hong Kong, in 1995.

He was a Research Fellow with the Computer Sciences Laboratory, Australian National University, Canberra, ACT, Australia, from 1995 to 1997, and an Assistant/Associate Professor with the University of Hong Kong from 1997 to 2005. He is currently a Professor with the Department of Mathematics, Hong Kong Baptist University, Hong Kong. His current research interests include bioinformatics, data mining, image processing, scientific computing, and data mining.

Prof. Ng serves on the Editorial Boards of international journals.



Xiaoyan Wang received the B.S. degree from the Department of Mathematics, Ningbo University, Zhejiang, China, in 2005, the M.S. degree from the Department of Mathematics, East China Normal University, Shanghai, China, in 2008, and the Ph.D. degree in mathematics from the National University of Singapore, Singapore, in 2012.

Her current research interests include numerical analysis, matrix computations, and data mining.