Boosting Performance of Incremental IDR/QR LDA - from Sequential to Chunk

Yiming Peng

A thesis submitted to Auckland University of Technology
in fulfillment of the requirements
for the degree of Master of Computer and Information Sciences

July, 2011

School of Computing and Mathematical Sciences

Primary Supervisor: Prof. Alvis Fong
Secondary Supervisor: Dr. Shaoning Pang

Declaration

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the qualification of any other degree or diploma of a University or other institution of higher learning, except where due acknowledgment is made in the acknowledgments.

Printed name: Yiming Peng

Signature:

Date:

Abstract

Training data in the real world is often presented in random chunks. Yet existing sequential incremental IDR/QR LDA (sIncLDA) can only process data one instance after another. This thesis proposes a new chunk incremental IDR/QR LDA (cIncLDA) capable of processing multiple data instances at one time. sIncLDA updates the reduced within-class scatter matrix $\boldsymbol{W}$ by a QR decomposition of the centroid matrix for each newly-arrived data instance. It is assumed that the updated $\boldsymbol{Q}' \approx \boldsymbol{Q}$ for any data instance from an existing class and the updated $\boldsymbol{W}' \approx \boldsymbol{W}$ for any data instance from a new class. In practice, the assumption in sIncLDA leads to significant loss of the discriminative information from approximating $\boldsymbol{Q}$ and $\boldsymbol{W}$ when the number of classes is large. By utilizing a new method that accurately updates $\boldsymbol{W}$, the proposed cIncLDA can better pr eserve the discriminative information contained in $\boldsymbol{W}$. The limitation of sIncLDA is hence resolved. Experimental comparisons have been conducted on six facial datasets with diverse class numbers ranging from 40 to 1010. The result indicates that our algorithm achieves an competitive accuracy to batch QR/LDA and is consistently higher than sIncLDA. It is noted in the report that the computational complexity of our algorithm is more expensive than sIncLDA for single data processing (i.e., sequential manner); however, the efficiency of our algorithm surpasses sIncLDA as the chunk size increases for multiple instances processing (i.e., chunk manner).

Acknowledgments

I would like to thank all people who have helped and inspired me during my master study.

I especially want to thank my supervisors, Prof. Alvis Fong and Dr Shaoning Pang, for their guidance during my research and study at Auckland University of Technology. Their perpetual energy and enthusiasm for research have motivated all their students, including me. In addition, Prof. Alvis Fong and Dr. Shaoning Pang were always accessible and willing to help his students with their research. As a result, research life became smooth and rewarding for me. Also, many thanks to the professional editor, Heather Moodie, for correcting my thesis carefully and patiently.

All my lab buddies at the DMLI Lab made it a convivial place to work. In particular, I would like to thank Lei Song and Lei Zhu for their friendship and help during my thesis.

My deepest gratitude goes to my family for their unflagging love and support throughout my life; this thesis would be simply impossible without them. I am indebted to my father, Sige Peng, for his care and love. As a typical father, he worked industriously to support the family and spared no effort to provide the best possible environment for me to grow up and attend school. He has never complained in spite of all the hardships in his life. I cannot ask for more from my mother, Qi Zhang, as she is simply perfect. I have no suitable word that can fully describe her everlasting love for me. I remember her constant support when I encountered difficulties and I remember, most of all, her delicious dishes. Also, many appreciations to my brother Lin Peng and his wife Li Chen, for their encourage, help and love. I would also like to thank my girlfriend Yuanhua Sun for supporting me while I was writing the thesis and for ensuring that I did not lose sight of other aspects in life not related to academia.

Contents

Chapter 1

Introduction

1.1 Background

Incremental learning (IL) addresses real-world situations when data are frequently presented in either sequential (i.e., one sample after another) or chunk manner (i.e., a subset of samples presented at a time) (Giraud-Carrier, 2000; Pang, Ozawa & Kasabov, 2005b). Given an existing learning model $\mathbf{\Omega}$, the incremental learning of $\mathbf{\Omega}$ is an updating process as follows:

$$\mathbf{\Omega}' = \mathcal{F}_{sq}(\mathbf{\Omega}, \widetilde{\boldsymbol{x}}),$$

or

$$\mathbf{\Omega}' = \mathcal{F}_{ck}(\mathbf{\Omega}, \widetilde{\boldsymbol{X}}), \widetilde{\boldsymbol{X}} = \{\widetilde{\boldsymbol{x}}_i\}_{i=1}^{s},$$

where $\widetilde{\boldsymbol{x}}$ denotes a newly presented data sample, and $\{\widetilde{\boldsymbol{x}}_i\}_{i=1}^{s}$ represents a chunk of s data samples. $\mathcal{F}_{sq}$ and $\mathcal{F}_{ck}$ perform respectively the sequential and chunk incremental learning to calculate the updated model $\mathbf{\Omega}'$ over any newly presented data. The $\mathcal{F}_{ck}$ is generally computationally less efficient than $\mathcal{F}_{sq}$, when the chunk size $s = 1$ (Pang et al., 2005b; Cauwenberghs & Poggio, 2000). This thesis studies incremental linear discriminant analysis (IncLDA) with a focus on how an $\mathcal{F}_{ck}$ is able to match and surpass the $\mathcal{F}_{sq}$ on learning efficiency without sacrificing effectiveness.

Linear Discriminant Analysis (LDA) seeks a linear projection of data that best discriminates two or more classes by the Fisher criterion or its equivalents (Fisher, 1936; Fukunaga, 1990; McLachlan, 2004). In principle, the computing of batch LDAs in different disciplines can be formulated as an n-tuple function. The clas-

sic Fisher LDA (Fukunaga, 1990) constructs the discriminant eigenspace based on the within-class scatter matrix $\boldsymbol{S}_w$, between-class scatter matrix $\boldsymbol{S}_b$, and the class label set $\boldsymbol{C}$. Thus a 3-tuple model $\boldsymbol{\Omega} = \{\boldsymbol{S_w}, \boldsymbol{S_b}, \boldsymbol{C}\}$ can be formed (Pang et al., 2005b). The GSVD/LDA (Ye, Janardan, Park & Park, 2004) applies generalized Singular Value Decomposition (SVD) on $\boldsymbol{H}$ to obtain singular vector $\boldsymbol{P}$ and singular value $\boldsymbol{\Gamma}$, then performs again the SVD on a submatrix of $\boldsymbol{P}$ to solve the LDA optimization problem. Here, $\boldsymbol{H} = [\boldsymbol{H}_w\ \boldsymbol{H}_b]^T$ and $\boldsymbol{H} = \boldsymbol{P\Gamma\Pi}^T$. Thus, GSVD/LDA models $\boldsymbol{\Omega} = \{\boldsymbol{H}_w, \boldsymbol{H}_b, \boldsymbol{C}\}$. The Least Square LDA (Ye, 2007) transforms the LDA optimization into Multivariate Linear Regression (MLR), thus rears 3-tuple calculation $\boldsymbol{\Omega} = \{\boldsymbol{M}, \boldsymbol{Y}, \boldsymbol{C}\}$, where $\boldsymbol{M}$ and $\boldsymbol{Y}$ are the input centroid matrix and the output indicator matrix, respectively. The QR/LDA (Ye et al., 2005) employs QR decomposition to implement LDA optimization on size-reduced instead of full scatter matrices. Its eigenspace model is written as $\boldsymbol{\Omega} = \{\boldsymbol{M}, \boldsymbol{W}, \boldsymbol{B}\}$; here $\boldsymbol{W}$ and $\boldsymbol{B}$ represent reduced within-class and between-class scatter matrices respectively, and $\boldsymbol{M}$ is the centroid matrix. Therefore, IncLDA can be explained as an n-tuple model updating process.

Thus, a general IncLDA model can be depicted in the example of existing sequential IDR/QR IncLDA (cIncLDA) (Ye et al., 2005) as

$$\{\boldsymbol{M'}, \boldsymbol{W'}, \boldsymbol{B'}\} = \mathcal{F}_{sq}(\{\boldsymbol{M}, \boldsymbol{W}, \boldsymbol{B}\}, \widetilde{\boldsymbol{x}}).$$

In fact, for a preferable IL method, it should obey four general rules proposed by Polikar, Upda, Upda and Honavar (2001) as,

1. It should be able to learn new information from new data.
2. It should not require access to, and retain in memory, the original data.
3. It should preserve previously acquired information.
4. It should be able to accommodate new classes that may be introduced with new data.

In this thesis, we present two additional criteria for IncLDA, as (5) It should be able to process multiple data samples at one time, and single data sample can be processed as a special case; (6) It should be able to address the Small Sample Size problem.

IncLDA is essentially a model updating process, each component of the model needing to proceed through along a chunk of newly presented samples. As a result, the loss of discriminant information will become severe due to approximate projections at each update round. It is clear that the increase of update rounds is a main source of inefficiencies. To reduce update rounds at each IncLDA cycle, it is necessary to conduct chunk IncLDAs in which multiple samples can be processed at one time, instead of sequential IncLDAs.

Furthermore, most existing LDAs are derived from the Fisher criterion (Fukunaga, 1990) as

$$\boldsymbol{\phi} = \underset{\boldsymbol{\phi}^T\boldsymbol{\phi}=\boldsymbol{I}}{\operatorname{argmax}} \frac{\operatorname{Tr}(\boldsymbol{\phi}^T\boldsymbol{S_b}\boldsymbol{\phi})}{\operatorname{Tr}(\boldsymbol{\phi}^T\boldsymbol{S_w}\boldsymbol{\phi})}, \tag{1.1}$$

and two variations are $\boldsymbol{\phi} = \operatorname{argmax}_{\boldsymbol{\phi}^T\boldsymbol{\phi}=\boldsymbol{I}} \frac{\operatorname{Tr}(\boldsymbol{\phi}^T\boldsymbol{S_t}\boldsymbol{\phi})}{\operatorname{Tr}(\boldsymbol{\phi}^T\boldsymbol{S_w}\boldsymbol{\phi})}$, $\boldsymbol{\phi} = \operatorname{argmax}_{\boldsymbol{\phi}^T\boldsymbol{\phi}=\boldsymbol{I}} \frac{\operatorname{Tr}(\boldsymbol{\phi}^T\boldsymbol{S_b}\boldsymbol{\phi})}{\operatorname{Tr}(\boldsymbol{\phi}^T\boldsymbol{S_t}\boldsymbol{\phi})}$. The criterion requires that either $\boldsymbol{S}_w$ or $\boldsymbol{S}_t$ must be non-singular. However, it is not to be followed in many practical applications as the number of samples may be much smaller than the dimension of the sample space, namely $n \ll d$. This is the so-called 'undersampled' problems, or 'small sample size' (SSS) problem (Fukunaga, 1990). To address the problem, many LDA methods have been developed, such as QR-decomposition based LDA (QR/LDA)(Ye & Li, 2004; Ye et al., 2005), Generalized Singular Value Decomposition based LDA (GSVD/LDA)(Howland, Jeon & Park, 2003; Ye et al., 2004; H. Zhao & Yuen, 2008), Least Square LDA (LS/LDA)(Ye, 2007), Null Space LDA (NS/LDA)(Huang, Liu, Lu & Ma, 2002; Chu & Thye, 2010; Chen, Liao, Ko, Lin & Yu, 2000), PCA+LDA(Belhumeur, Hespanha & Kriegman, 1997), uncorrelated LDA (ULDA)(Jin, Yang, Hu & Lou, 2001; Ye, 2004), orthogonal LDA (OLDA) (Ye, 2005), etc. Accordingly, IncLDAs, which update some of those LDA extensions incrementally, should be able to address the SSS problem.

The existing sIncLDA satisfies all the criteria presented above, except criterion 5. Our work concentrates on developing a method that satisfies criterion 5 to boost its practical usage. To evaluate the proposed cIncLDA, its performance is experimentally compared with sIncLDA on some real-world datasets. Experimental results show that the proposed cIncLDA is more stable and effective on selected datasets than sIncLDA.

For convenience, we summarize most notations used in the thesis in Table 1.1. As a convention, "˜" is used to represent additional information introduced by newly added data samples.

Notation	Descriptions
$\boldsymbol{X}$	data matrix
$\boldsymbol{X_i}$	data matrix of the i-th class
d	dimension of data matrix
n	number of data samples
n_i	number of data samples in the i-th class
$\boldsymbol{C}$	set of class labels
$\boldsymbol{c_i}$	class label of the i-th class
k	number of classes
$\boldsymbol{M}$	centroid matrix of data matrix
$\boldsymbol{m}$	general mean of data matrix
$\boldsymbol{m}_i$	mean of data matrix of the i-th class
$\boldsymbol{W}$	reduced within-class scatter matrix
$\boldsymbol{B}$	reduced between-class scatter matrix

Table 1.1: NOTATIONS

1.2 Research Objectives

This research aims to propose a new chunk incremental IDR/QR LDA to resolve the limitations of the sequential incremental IDR/QR LDA and boost its performance. The proposed method follows all incremental learning criteria and addresses the SSS problem. It is designed to process data presented in random chunks. In addition, it loses less discriminant information than sIncLDA, and the discriminant eigenspace model can be updated more accurately.

1.3 Research Contributions

The contributions of this thesis are:

1. A new evaluation criteria framework for IncLDAs has been proposed. It allows us to assess IncLDAs from a new point of view, not just by comparisons on performance, efficiency and so forth.

2. Two efficient matrix augmentation methods have been proposed. They allow us to easily accommodate information of new classes contained in chunk data through expanding the dimensions of matrices $\boldsymbol{Q}$ and $\boldsymbol{W}$.

3. A new method for incremental updating of $\boldsymbol{W}$, more accurate than the existing approach, has been successfully developed in this thesis. This is achieved by relaxing a key assumption on $\boldsymbol{Q}$ and $\boldsymbol{W}$ that is vital to the existing algorithm but may be significantly violated as the number of classes increases.

4. Unlike Pang's IncLDAs (Pang et al., 2005b) where only the chunk data in one class can be acquired each time, our method is capable of absorbing information obtained from newly added samples in whatever classes.

1.4 Thesis Structure

Chapter 2 investigates the fundamental concepts of LDA first, and reviews several popular LDA methods based on their principles and pseudocodes; After that, incremental extensions of those LDAs methods will be depicted in detail. Finally, based on what has been reviewed, motivations of this research is highlighted.

Chapter 3 recaptures and analyzes the existing batch QR/LDA and sequential incremental IDR/QR LDA from an embedding point of view. The principles of batch QR/LDA and sIncLDA are both recapitulated first. The limitations of the sIncLDA are summarized and discussed in depth at the end of the chapter.

Chapter 4 presents the proposed chunk incremental IDR/QR LDA. To improve computational efficiency, two new matrix augmentation methods are proposed. The proposed cIncLDA has been detailed with mathematical derivations and its pseudocode.

Chapter 5 examine the accuracy and efficiency of the proposed cIncLDA by comparing it with sIncLDA and batch QR/LDA. We have specifically examined its equality to the ground truth eigenspace, its class separability of the embedding eigenspace, and its execution time.

Chapter 6 concludes the work presented by the thesis and gives some directions for future research.

Chapter 2

Literature Review

In this chapter, the fundamental concepts of LDA will be investigated first, and popular LDA methods will be presented based on their principles and pseudocodes; After that, we study incremental extensions of those LDA methods in detail. Finally, based on what has been reviewed, motivations of this research will be highlighted.

2.1 Linear Discriminant Analysis

LDA has been widely demonstrated as an effective technique for dimension reduction and feature extraction (Ye & Li, 2004; Pang et al., 2005b; T.-K. Kim, Stenger, Kittler & Cipolla, 2011). Applications of LDA are growing in number, such as pattern recognition (Bishop, 2006; Duda, Hart & Stork, 2001; Fukunaga, 1990), pedestrian detection (X. Wang, Han & Yan, 2009), information retrieval (Kowalski, 1997; Frakes & Baeza-Yates, 1992), micro-array data analysis (Baldi & Hatfield, 2002; Dudoit, Fridlyand & Speed, 2002), text classification (Jain & Dubes, 1988), original texture analysis (Chan, Kittler & Messer, 2007) as well as face recognition (Jin et al., 2001; Swets & Weng, 1996). The aim of dimension reduction is to either find a linear combination of features, or select a subset of features from the original feature space (Ye et al., 2005). In this way, LDA can be simply defined as: LDA seeks a linear projection of data (in terms of a combination of features) that best discriminates two or more classes by the Fisher criterion or its equivalents (Fisher, 1936; Fukunaga, 1990; McLachlan, 2004).

2.1.1 Classic Fisher LDA

Most existing LDAs follow or extend the classic Fisher criterion. By employing within-class scatter matrix $\boldsymbol{S_w}$, between-class scatter matrix $\boldsymbol{S_b}$, and total scatter matrix $\boldsymbol{S_t}$, the class separability criterion can be formulated as follows (Fisher, 1936; Fukunaga, 1990):

$$\boldsymbol{\phi} = \underset{\boldsymbol{\phi}}{\operatorname{argmax}} \frac{\operatorname{Tr}(\boldsymbol{\phi}^T \boldsymbol{S_b} \boldsymbol{\phi})}{\operatorname{Tr}(\boldsymbol{\phi}^T \boldsymbol{S_w} \boldsymbol{\phi})}, \tag{2.1}$$

and two variations are

$$\boldsymbol{\phi} = \underset{\boldsymbol{\phi}}{\operatorname{argmax}} \frac{\operatorname{Tr}(\boldsymbol{\phi}^T \boldsymbol{S_t} \boldsymbol{\phi})}{\operatorname{Tr}(\boldsymbol{\phi}^T \boldsymbol{S_w} \boldsymbol{\phi})}, \tag{2.2}$$

$$\boldsymbol{\phi} = \underset{\boldsymbol{\phi}}{\operatorname{argmax}} \frac{\operatorname{Tr}(\boldsymbol{\phi}^T \boldsymbol{S_b} \boldsymbol{\phi})}{\operatorname{Tr}(\boldsymbol{\phi}^T \boldsymbol{S_t} \boldsymbol{\phi})}, \tag{2.3}$$

where

$$\boldsymbol{S_b} = \sum_{i=1}^{k} n_i (\boldsymbol{m}_i - \boldsymbol{m})(\boldsymbol{m}_i - \boldsymbol{m})^T \tag{2.4}$$

denotes the scatter of the expected vectors around the general mean,

$$\boldsymbol{S_w} = \sum_{i=1}^{k} \sum_{j=1}^{n_i} (\boldsymbol{x}_{ij} - \boldsymbol{m}_i)(\boldsymbol{x}_{ij} - \boldsymbol{m}_i)^T \tag{2.5}$$

represents the expected vector of the general distribution, and

$$\boldsymbol{S_t} = \boldsymbol{S_w} + \boldsymbol{S_b} \tag{2.6}$$

is the covariance matrix of all samples without consideration of their class assignments. Given $\boldsymbol{X} = [\boldsymbol{X}_1, \ldots, \boldsymbol{X}_k] \in \mathcal{R}^{d \times n}$, in which $\boldsymbol{X}_i = [\boldsymbol{x}_{i1}, \ldots, \boldsymbol{x}_{in_i}] \in \mathcal{R}^{d \times n_i}$ denotes samples belonging to the i-th class, and $i = 1, \ldots, k$. For the i-th class, $\boldsymbol{m}_i$ and n_i denote the mean and the number of samples, respectively. The general mean and the total number of samples are represented by $\boldsymbol{m}$ and n. **Note that**, the equivalence between (2.1) and those two alternative criteria (2.2) and (2.3) is conditional on the non-singular status of $\boldsymbol{S}_w$ or $\boldsymbol{S}_t$. Thus, the Fisher LDA forms a 3-tuple model as $\boldsymbol{\Omega} = \{\boldsymbol{S}_w, \boldsymbol{S}_b, \boldsymbol{C}\}$.

The pseudocode of the above algorithm is given as algorithm 1.

Algorithm 1 Fisher/LDA Algorithm

Input: Newly added training data matrix $\widetilde{\boldsymbol{X}}$, and its class label set $\widetilde{\boldsymbol{C}}$

Output: Optimal transform matrix $\boldsymbol{\phi}$, Fisher-LDA eigenspace model $\boldsymbol{\Omega} = \{\boldsymbol{S}_w, \boldsymbol{S}_b, \boldsymbol{C}\}$.

1: Calculate within-class scatter matrix $\boldsymbol{S}_w$ by (2.5);

2: Calculate between-class scatter matrix $\boldsymbol{S}_b$ by (2.4);

3: Calculate matrix $\boldsymbol{Z} = {\boldsymbol{S}_w}^{-1}\boldsymbol{S}_b$;

4: Perform eigen-decomposition of $\boldsymbol{Z}$: $\boldsymbol{Z} = \boldsymbol{\phi}\boldsymbol{\Lambda}\boldsymbol{\phi}^T$.

2.1.2 Small Sample Size Problem

For many tasks, such as information retrieval and face recognition, LDA has already been showing promising results (Belhumeur et al., 1997; W. Zhao, Chellappa & Phillips, 1999; Chen et al., 2000; Yu & Yang, 2001; C. Liu & Wechsler, 2002; Lu, Plataniotis & Venetsanopoulos, 2003a, 2003b; Ye & Li, 2004). However, LDA suffers from the so-called Small Sample Size (SSS) problem (Raudys & Jain, 1990). The SSS problem is often encountered in high dimensional pattern recognition tasks where the number of training samples (i.e., n) is smaller than the dimensionality of the sample space (i.e., d), namely $n < d$. In such a case, the within-class scatter matrix $\boldsymbol{S}_w$ or the total scatter matrix $\boldsymbol{S}_t$ may not be singular. As a result, the Fisher/LDA may fail.

Simply put, there are three ways to solve the problem:

1. It can be solved by utilizing linear algebra techniques, such pseudo inverse, SVD, or GSVD, on the singular within-class scatter matrix. For example, Tian, Barbero, Gu and Lee (1986) proposed a solution to the problem by using pseudo inverse on the within-class scatter matrix. Some research (Hong & Yang, 1991; W. Zhao et al., 1999) gave a solution of adding a small perturbation to the within-class scatter matrix so that it became nonsingular.

2. It can also be solved by involving a subspace approach. For example, Belhumeur et al. (1997) applied PCA as a preprocessing step on the raw data to remove the null space of $\boldsymbol{S}_w$, and hereby the singularity of $\boldsymbol{S}_w$ was well handled. Another solution used the null space of $\boldsymbol{S}_w$ rather $\boldsymbol{S}_w$ itself (Chen et al., 2000; Chu & Thye, 2010).

3. Regularization is another way to cope with the problem, adding a scaled iden-

tity matrix to diagonal elements of the within-class scatter matrix (Friedman, 1989; Lu, Plataniotis & Venetsanopoulos, 2005).

In summary, the third approach is the most efficient, since it only involves simple matrix addition. To address SSS problem of LDA, we need to extend the Fisher criterion by employing the above approaches. In the following text, three popular LDA extensions, GSVD/LDA (Ye et al., 2004; H. Zhao & Yuen, 2008), LS/LDA(Ye, 2007; L. Liu, Jiang & Zhou, 2009), and QR/LDA (Ye & Li, 2004; Ye et al., 2005), are depicted.

2.1.3 LDA Extensions to the SSS Problem

GSVD/LDA

The concept of GSVD was first introduced by Paige and Saunders (1981). Based on their idea, Ye et al. (2004) proposed a GSVD-based LDA extension. GSVD/LDA follows the Theorem 1 below:

Theorem 1 *Let* $\boldsymbol{e}_i = [1, \ldots, 1]^T \in \mathcal{R}^{n_i \times 1}$,

$$\boldsymbol{H}_w = [\boldsymbol{X}_1 - \boldsymbol{m}_1 \boldsymbol{e}_1^T, \ldots, \boldsymbol{X}_k - \boldsymbol{m}_k \boldsymbol{e}_k^T], \tag{2.7}$$

and

$$\boldsymbol{H}_b = [\sqrt{n_1}(\boldsymbol{m}_1 - \boldsymbol{m}), \ldots, \sqrt{n_k}(\boldsymbol{m}_k - \boldsymbol{m})]. \tag{2.8}$$

Suppose that $\boldsymbol{H} = [\boldsymbol{H}_w \ \boldsymbol{H}_b]^T \in \mathcal{R}^{(k+n)\times d}$, *then there exist orthogonal matrices* $\boldsymbol{U} \in \mathcal{R}^{k\times k}$, $\boldsymbol{V} \in \mathcal{R}^{n\times n}$ *and a nonsingular matrix* $\boldsymbol{\Upsilon} \in \mathcal{R}^{n\times n}$, *such that*

$$\begin{bmatrix} \boldsymbol{U} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{V} \end{bmatrix}^T \boldsymbol{H}\boldsymbol{\Psi} = \begin{bmatrix} \boldsymbol{\Sigma}_b & \boldsymbol{0} \\ \boldsymbol{\Sigma}_w & \boldsymbol{0} \end{bmatrix} \tag{2.9}$$

where

$$\boldsymbol{\Sigma}_b = \begin{bmatrix} \boldsymbol{I}_b & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{D}_b & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0}_b \end{bmatrix} \quad \textit{and} \quad \boldsymbol{\Sigma}_w = \begin{bmatrix} \boldsymbol{0}_w & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{D}_w & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I}_w \end{bmatrix}$$

and $\boldsymbol{\Sigma}_b^T \boldsymbol{\Sigma}_b + \boldsymbol{\Sigma}_w^T \boldsymbol{\Sigma}_w = \boldsymbol{I}_d$.

In the Theorem 1, $\boldsymbol{I}_b \in \mathcal{R}^{t\times t}$ and $\boldsymbol{I}_w \in \mathcal{R}^{(t-s-r)\times(t-s-r)}$ are two identity matrices, in which $t = rank(\boldsymbol{H})$, $r = rank(\boldsymbol{H}) - rank(\boldsymbol{H}_w)$, and $s = rank(\boldsymbol{H}_b) + rank(\boldsymbol{H}_w) - rank(\boldsymbol{H})$. In addition, $\boldsymbol{D}_b = diag(\tau_{t+1}, \ldots, \tau_{t+r}) \in \mathcal{R}^{r\times r}$ and $\boldsymbol{D}_w = diag(\nu_{t+1}, \ldots, \nu_{t+s})$ are diagonal matrices with diagonal elements value positioning between 0 and 1.

From Theorem 1, as $\boldsymbol{S}_w = \boldsymbol{H}_w\boldsymbol{H}_w^T$ and $\boldsymbol{S}_b = \boldsymbol{H}_b\boldsymbol{H}_b^T$, we have

$$\boldsymbol{\Psi}^T\boldsymbol{S}_b\boldsymbol{\Psi} = \begin{bmatrix} \boldsymbol{\Sigma}_b^T\boldsymbol{\Sigma}_b & \\ & \boldsymbol{0} \end{bmatrix} \tag{2.10}$$

$$\boldsymbol{\Psi}^T\boldsymbol{S}_w\boldsymbol{\Psi} = \begin{bmatrix} \boldsymbol{\Sigma}_w^T\boldsymbol{\Sigma}_w & \\ & \boldsymbol{0} \end{bmatrix} \tag{2.11}$$

Accordingly, we can have

$$\nu_i^2\boldsymbol{S}_b\tau_i = \nu_i^2\boldsymbol{S}_w\tau_i, \tag{2.12}$$

where τ_i is the i-th column of the matrix $\boldsymbol{\Upsilon}$, $i = 1, \ldots, d$.

Hence, the leftmost $k-1$ eigenvectors of $\boldsymbol{\Upsilon}$ are obtained as the optimal transform matrix. The algorithm of GSVD/LDA is summarized in algorithm 2.

Algorithm 2 GSVD/LDA Algorithm

Input: Newly added training data matrix $\widetilde{\boldsymbol{X}}$, and its class label set $\widetilde{\boldsymbol{C}}$

Output: Optimal transform matrix $\boldsymbol{\phi}$, GSVD/LDA eigenspace model $\boldsymbol{\Omega} = \{\boldsymbol{H}_w, \boldsymbol{H}_b, \boldsymbol{C}\}$.

1: Calculate matrix $\boldsymbol{H} = [\boldsymbol{H}_w\ \boldsymbol{H}_b]^T$;

2: Calculate the rank of H as $t = rank(\boldsymbol{H})$;

3: Perform SVD on $\boldsymbol{H}$ as $\boldsymbol{H} = \boldsymbol{P}\begin{bmatrix} \boldsymbol{R} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}\boldsymbol{\Pi}^T$;

4: Perform SVD on $\boldsymbol{P}(1:k, 1:t)$ as $\boldsymbol{P}(1:k, 1:t) = \boldsymbol{U}\boldsymbol{\Upsilon}\boldsymbol{\Xi}$;

5: Calculate submatrix $\boldsymbol{\Pi}_k$ as leading principal submatrix of $\boldsymbol{\Pi}$;

6: Calculate the rank of $\boldsymbol{H}_b$ as $\tau = rank(\boldsymbol{H}_b)$;

7: Calculate matrix

$$\boldsymbol{T} = \boldsymbol{\Pi}\begin{bmatrix} \boldsymbol{R}^{-1}\boldsymbol{\Xi} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix};$$

8: Calculate optimal transform matrix $\boldsymbol{\phi} = \boldsymbol{T}(:, 1:\tau)$.

LS/LDA

Ye (2007) investigated the relationship between LDA and the MLR problem and proposed LS/LDA. By adding a constraint to the optimization of the Fisher criterion (2.1), LDA can be integrated into the MLR schema, as

$$\begin{aligned} &\max \mathrm{Tr}((\boldsymbol{\phi}^T \boldsymbol{S}_b \boldsymbol{\phi})(\boldsymbol{\phi}^T \boldsymbol{S}_t \boldsymbol{\phi})^+), \\ &s.t.\ \boldsymbol{\phi} = (\boldsymbol{M}\boldsymbol{M}^T)^+ \boldsymbol{M}\boldsymbol{Y}, \end{aligned} \tag{2.13}$$

where $\boldsymbol{M}$ is the centroid matrix of $\boldsymbol{X}$, and $\boldsymbol{Y}$ is the indicator matrix.

From (2.13), we can compute $\boldsymbol{Y}$, as

$$\boldsymbol{Y}_{ji} = \begin{cases} \sqrt{\frac{n}{n_k} - \frac{n_k}{n}} & \text{if } \boldsymbol{X}_{\boldsymbol{\cdot} i} \text{ belongs to class k} \\ -\sqrt{\frac{n_k}{n}} & \text{otherwise.} \end{cases} \tag{2.14}$$

Thus, the MLR solution $\boldsymbol{\phi}_{MLR}$ can computed from (2.13) as

$$\boldsymbol{\phi}_{MLR} = (\boldsymbol{M}^+)^T \boldsymbol{Y} \tag{2.15}$$

Consequently, we have a 3-tuple model of LS/LDA as $\Omega = \{\boldsymbol{M}, \boldsymbol{Y}, \boldsymbol{C}\}$, and the pseudocode is shown in algorithm 3. In LS/LDA, the SSS problem is solved by transforming the LDA optimization problem into the MLR framework, since the singularity of $\boldsymbol{S}_w$ or $\boldsymbol{S}_t$ can be avoided in the MLR schema.

Algorithm 3 Least Square LDA Algorithm

Input: Newly added training data matrix $\widetilde{\boldsymbol{X}}$, and its class label set $\widetilde{\boldsymbol{C}}$

Output: Optimal transform matrix $\boldsymbol{\phi}$, LS/LDA component model $\Omega = \{\boldsymbol{M}, \boldsymbol{Y}, \boldsymbol{C}\}$.

1: Calculate centroid matrix $\boldsymbol{M}$;

2: Calculate pseudo inverse of centroid matrix $\boldsymbol{M}^+$;

3: Calculate indicator matrix $\boldsymbol{Y}$ by (2.14);

4: Calculate optimal transform matrix $\boldsymbol{\phi}$ by (2.15);

QR/LDA

QR/LDA (Ye & Li, 2004; Ye et al., 2005) used QR-decomposition rather than SVD or GSVD to maximizing the Fisher criterion (2.1) by maximizing the separation

between different classes (i.e., maximize between-class scatter matrix $\boldsymbol{S}_b$), meanwhile minimizing the within-class distance (i.e., maximize within-class scatter matrix $\boldsymbol{S}_w$). Because QR/LDA is the basis of our proposed algorithm, the detailed review on this method is given in section 3.1.

For QR/LDA, to address SSS problem, the regularization method has been employed. Different from previous regularized LDAs, such as Regularized LDA (Lu et al., 2005) and Spectral Regression Discriminant Analysis (SRDA)(Cai, He & Han, 2008), the QR/LDA and sIncLDA applied the regularization step on the reduced within-class scatter matrix $\boldsymbol{W}$ instead of $\boldsymbol{S}_w$. As discussed previously, the regularization method is the most efficient way to address SSS problem.

2.1.4 Summary

In many real-world applications, it is common for training data to be continuously presented over time rather than all being given in one batch in advance. Since the data is often given in random chunks, we cannot know in advance the next sample to process. In view of this, the LDAs discussed above suffer certain limitations. Particularly, the computational cost of batch LDAs, such as Fisher/LDA and GSVD/LDA, is extremely high when the dataset is large. Meanwhile, batch LDAs keep all data samples, incurring a high level of memory usage. This is obviously inefficient for those tasks needing online learning. Because of these limitations, it is necessary to conduct learning in an incremental way, where only information carried by newly presented data is accommodated for the existing model.

2.2 Incremental Linear Discriminant Analysis

This thesis focuses on incremental linear discriminant analysis. In this section, before studying IncLDA in depth, we first revisit the two important characteristics of incremental learning (IL), chunk and one-pass. Most existing IncLDAs will be reviewed and evaluated according to our proposed assessing criteria in Section 1.1.

2.2.1 Chunk Incremental Learning

IL operates numerous updates on a set of components (e.g., within-class scatter matrix $\boldsymbol{S}_w$) to renew the entire space/model. Sequential IL updates the learning

model for every data sample presented, causing many update calculations; As model updating may be computationally intensive, inaccuracies and inefficiencies may build up quickly (Hall, Marshall & Martin, 2000; Ozawa, Pang & Kasabov, 2006). Chunk IL is more likely to improve accuracy and efficiency, because it processes multiple samples at one time.

It is worth noting that chunk IL has recently gained more attention (Hao et al., 2004; Jiang, Song, Wu, Maurizio & Liang, 2006; Hall et al., 2000; Pang, Ozawa & Kasabov, 2004; Pang et al., 2005b; Pang, Ozawa & Kasabov, 2005a; Ozawa et al., 2006; J.-G. Wang, Sung & Yau, 2010). For incremental principle component analysis (IncPCA), Hall et al. (2000) enabled chunk IncPCA computing by merging original and newly created eigenspace; Ozawa et al. (2006) developed another version of chunk IncPCA by implementing both eigenspace rotation and augmentation. For incremental support vector machine (IncSVM), Karasuyama and Takeuchi (2010) extended a sequential (Cauwenberghs & Poggio, 2000) to chunk IncSVM algorithm by updating multiple Lagrange multipliers under the condition of keeping Karush-Kuhn-Tucker (KKT) balance.

2.2.2 One-pass Incremental Learning

When data are presented sequentially or in random chunks, despite incremental learning, a straightforward approach for learning is to collect the data and then conduct batch learning over the data so far collected. Obviously, this requires large memory and high computational cost, because the data presented so far would be stored in the memory until the learning process ends. Moreover, even if the learning is almost finished, adding the last single data sample, still requires repetition of the learning from the beginning (Pang et al., 2004, 2005b).

To cope with the problem, the one-pass concept has been widely applied to the incremental learning (Yen & Meesad, 1999; Loos, 1989; Pang et al., 2005b; Kidera, Ozawa & Abe, 2006; Pang, Ban, Kadobayashi & Kasabov, 2010). According to Loos (1989); Pang et al. (2005b), one-pass can be explained as, in the learning schema, only knowledge from a single presentation of the training data is acquired, learned, and retained, without keeping all data samples presented so far.

In the machine learning field, one-pass incremental learning has become more and more popular. For incremental PCA, Pang et al. (2004) proposed an one-pass PCA, which stores the eigenvectors of the new face data for updating the existing eigenspace

model rather than the raw face data. For incremental clustering, Ataa Allah, Grosky and Aboutajdine (2007) proposed an online single-pass (i.e., one-pass) method, which requires a single, sequential pass over the incoming data it attempts to cluster.

2.2.3 Incremental Linear Discriminant Analysis

A batch LDA can be formed as a component model $\boldsymbol{\Omega}$, and incremental LDA can be described as the update of $\boldsymbol{\Omega}$. In the following text, we will discuss several popular IncLDAs as the update of batch LDAs mentioned in section 2.1.

Pang's IncLDA

Pang et al. (2005b) proposed an incremental LDA based on the Fisher/LDA. As discussed above, Fisher/LDA can be formed as a 3-tuple model $\boldsymbol{\Omega} = \{\boldsymbol{S}_w, \boldsymbol{S}_b, \boldsymbol{C}\}$, and thus can be updated in a sequential manner or chunk manner respectively,

$$\boldsymbol{\Omega}' = \mathcal{F}_{sq}(\boldsymbol{\Omega}, \widetilde{\boldsymbol{x}}) = \{\boldsymbol{S}'_w, \boldsymbol{S}'_b, \boldsymbol{C}'\},$$

or

$$\boldsymbol{\Omega}' = \mathcal{F}_{ck}(\boldsymbol{\Omega}, \widetilde{\boldsymbol{X}}) = \{\boldsymbol{S}'_w, \boldsymbol{S}'_b, \boldsymbol{C}'\}, \widetilde{\boldsymbol{X}} = \{\widetilde{\boldsymbol{x}}_i\}_{i=1}^{s}.$$

For Pang's sequential IncLDA, $\boldsymbol{\Omega}' = \{\boldsymbol{S}'_w, \boldsymbol{S}'_b, \boldsymbol{C}'\}$ can be calculated following the steps below:

1. if $\widetilde{\boldsymbol{x}}$ comes from an existing class, then $n' = n + 1$ and $\boldsymbol{C}' = \boldsymbol{C}$. The updated within-class scatter matrix $\boldsymbol{S}'_w$ is calculated as,

$$\boldsymbol{S}'_w = \sum_{i=1, i\neq\rho}^{k} \boldsymbol{\Sigma}_i + \boldsymbol{\Sigma}'_\rho, \tag{2.16}$$

where

$$\boldsymbol{\Sigma}_i = \sum_{i=1, i\neq\rho}^{k} \sum_{j=1}^{n_i} (\boldsymbol{x}_{ij} - \boldsymbol{m}_i)(\boldsymbol{x}_{ij} - \boldsymbol{m}_i)^T, \tag{2.17}$$

and

$$\boldsymbol{\Sigma}'_\rho = \boldsymbol{\Sigma}_\rho + \frac{n_\rho}{n_\rho + 1}(\widetilde{\boldsymbol{x}} - \boldsymbol{m}_\rho)(\widetilde{\boldsymbol{x}} - \boldsymbol{m}_\rho)^T. \tag{2.18}$$

The updated between-class scatter matrix $\boldsymbol{S}'_b$ can be calculated as,

$$\boldsymbol{S}'_b = \sum_{i=1, i\neq\rho}^{k} n_i(\boldsymbol{m}_i - \boldsymbol{m}')(\boldsymbol{m}_i - \boldsymbol{m}')^T + n'_\rho(\boldsymbol{m}'_\rho - \boldsymbol{m}')(\boldsymbol{m}'_\rho - \boldsymbol{m}')^T, \quad (2.19)$$

where $\boldsymbol{m}'_\rho = \frac{1}{n_\rho+1}(n_\rho \boldsymbol{m}_\rho + \widetilde{\boldsymbol{x}})$ and $n'_\rho = n_\rho + 1$.

2. if $\widetilde{\boldsymbol{x}}$ belongs to a new class, then $n' = n + 1$ and $\boldsymbol{C'} = [\boldsymbol{C}, \widetilde{\boldsymbol{c}}]$. The updated within-class scatter matrix does not change, namely $\boldsymbol{S}'_w = \boldsymbol{S}_w$.

   The updated between-class scatter matrix $\boldsymbol{S}'_b$ is calculated as,

$$\boldsymbol{S}'_b = \sum_{i=1}^{k} n_i(\boldsymbol{m}_i - \boldsymbol{m}')(\boldsymbol{m}_i - \boldsymbol{m}')^T + (\widetilde{\boldsymbol{x}} - \boldsymbol{m}')(\widetilde{\boldsymbol{x}} - \boldsymbol{m}')^T. \quad (2.20)$$

The pseudocode of the above algorithm is given as algorithm 4.

Algorithm 4 Pang's Sequential IncLDA Algorithm

Input: Newly added training data sample $\widetilde{\boldsymbol{x}}$ and its class label $\widetilde{\boldsymbol{c}}$, Fisher/LDA eigenspace model $\boldsymbol{\Omega} = \{\boldsymbol{S}_w, \boldsymbol{S}_b, \boldsymbol{C}\}$.

Output: Optimal transform matrix $\boldsymbol{\phi}'$, Pang's Sequential IncLDA eigenspace model $\boldsymbol{\Omega'} = \{\boldsymbol{S}'_w, \boldsymbol{S}'_b, \boldsymbol{C}'\}$.

1: **if** $\widetilde{\boldsymbol{c}} \in \boldsymbol{C}$ **then**
2: Calculate within-class scatter matrix $\boldsymbol{S'}_w$ by (2.16);
3: Calculate between-class scatter matrix $\boldsymbol{S'}_b$ by (2.19);
4: **else**
5: Calculate $\boldsymbol{C}' = [\boldsymbol{C}, \widetilde{\boldsymbol{c}}]$;
6: Calculate within-class scatter matrix $\boldsymbol{S'}_w = \boldsymbol{S}_w$;
7: Calculate between-class scatter matrix $\boldsymbol{S'}_b$ by (2.20);
8: **end if**
9: Calculate matrix $\boldsymbol{Z}' = {\boldsymbol{S}'_w}^{-1}\boldsymbol{S}'_b$;
10: Perform eigen-decomposition of $\boldsymbol{Z}'$: $\boldsymbol{Z}' = \boldsymbol{\phi}'\boldsymbol{\Lambda}{\boldsymbol{\phi}'}^T$.

Pang's chunk IncLDA requires the newly added samples to belong to one class, namely the class number of incoming data $\widetilde{\boldsymbol{k}} = 1$. Hereby, $\boldsymbol{\Omega}' = \{\boldsymbol{S}'_w, \boldsymbol{S}'_b, \boldsymbol{C}'\}$ is calculated in the following steps:

1. if $\widetilde{\boldsymbol{X_i}}$ comes from an existing class, then $n' = n + s$ and $\boldsymbol{C'} = \boldsymbol{C}$. The updated within-class scatter matrix $\boldsymbol{S}'_w$ is calculated as,

$$\boldsymbol{S}'_w = \sum_{i=1}^{k} \boldsymbol{\Sigma}'_i, \tag{2.21}$$

where

$$\boldsymbol{\Sigma}'_i = \boldsymbol{\Sigma}_i + \frac{n_i s^2}{(n_i+s)^2}(\boldsymbol{D}_i) + \frac{n_i^2}{(n_i+s)^2}(\boldsymbol{E}_i) + \frac{s(s+2n_i)}{(n_i+s)^2}(\boldsymbol{F}_i). \tag{2.22}$$

From (2.22), we have the second term as

$$\boldsymbol{D}_i = (\widetilde{\boldsymbol{m}}_i - \boldsymbol{m}_i)(\widetilde{\boldsymbol{m}}_i - \boldsymbol{m}_i)^T, \tag{2.23}$$

the third term as

$$\boldsymbol{E}_i = \sum_{j=1}^{s} (\widetilde{\boldsymbol{x}}_{ij} - \boldsymbol{m}_i)(\widetilde{\boldsymbol{x}}_{ij} - \boldsymbol{m}_i)^T, \tag{2.24}$$

and the fourth term as

$$\boldsymbol{F}_i = \sum_{j=1}^{s} (\widetilde{\boldsymbol{x}}_{ij} - \widetilde{\boldsymbol{m}}_i)(\widetilde{\boldsymbol{x}}_{ij} - \widetilde{\boldsymbol{m}}_i)^T, \tag{2.25}$$

The updated between-class scatter matrix $\boldsymbol{S}'_b$ can be calculated as

$$\boldsymbol{S}'_b = \sum_{i=1}^{k} n'_i(\boldsymbol{m}'_i - \boldsymbol{m}')(\boldsymbol{m}'_i - \boldsymbol{m}')^T. \tag{2.26}$$

2. if $\widetilde{\boldsymbol{x}}$ belongs to a new class, then $n' = n + 1$ and $\boldsymbol{C'} = \boldsymbol{C} \cup \widetilde{\boldsymbol{C}}$. The updated within-class scatter matrix $\boldsymbol{S}'_w$ is calculated as

$$\boldsymbol{S}'_w = \sum_{i=1}^{k} \boldsymbol{\Sigma}_i + \sum_{j=1}^{s} (\widetilde{\boldsymbol{x}}_j - \widetilde{\boldsymbol{m}})(\widetilde{\boldsymbol{x}}_j - \widetilde{\boldsymbol{m}})^T. \tag{2.27}$$

The updated between-class scatter matrix $\boldsymbol{S}'_b$ is calculated as,

$$\boldsymbol{S}'_b = \sum_{i=1}^{k} n_i(\boldsymbol{m}_i - \boldsymbol{m}')(\boldsymbol{m}_i - \boldsymbol{m}')^T + (\widetilde{\boldsymbol{x}} - \boldsymbol{m}')(\widetilde{\boldsymbol{x}} - \boldsymbol{m}')^T. \tag{2.28}$$

The pseudocode of the above algorithm is given as algorithm 5.

Algorithm 5 Pang's Chunk IncLDA Algorithm

Input: Newly added training data matrix $\widetilde{\boldsymbol{X}}_{\boldsymbol{i}}$, $\widetilde{k} = 1$ and its class label set $\widetilde{\boldsymbol{C}}$, Fisher-LDA eigenspace model $\boldsymbol{\Omega} = \{\boldsymbol{S}_w, \boldsymbol{S}_b, \boldsymbol{C}\}$.

Output: Optimal transform matrix $\boldsymbol{\phi}'$, Pang's Sequential IncLDA eigenspace model $\boldsymbol{\Omega'} = \{\boldsymbol{S}'_w, \boldsymbol{S}'_b, \boldsymbol{C}'\}$.

1: **if** $\widetilde{\boldsymbol{C}} \subset \boldsymbol{C}$ **then**
2: Calculate within-class scatter matrix $\boldsymbol{S'}_w$ by (2.21);
3: Calculate between-class scatter matrix $\boldsymbol{S'}_b$ by (2.26);
4: **else**
5: Calculate $\boldsymbol{C}' = \boldsymbol{C} \cup \widetilde{\boldsymbol{C}}$;
6: Calculate within-class scatter matrix $\boldsymbol{S'}_w$ by (2.27);
7: Calculate between-class scatter matrix $\boldsymbol{S'}_b$ by (2.28);
8: **end if**
9: Calculate matrix $\boldsymbol{Z}' = {\boldsymbol{S}'_w}^{-1}\boldsymbol{S}'_b$;
10: Perform eigen-decomposition of $\boldsymbol{Z}'$: $\boldsymbol{Z}' = \boldsymbol{\phi}'\boldsymbol{\Lambda}{\boldsymbol{\phi}'}^T$.

GSVD IncLDA

GSVD/LDA models $\boldsymbol{\Omega} = \{\boldsymbol{H}_w, \boldsymbol{H}_b, \boldsymbol{C}\}$, according to our previous study, a direct incremental GSVD/LDA can update the $\boldsymbol{\Omega}$ as

$$\boldsymbol{\Omega}' = \mathcal{F}_{ck}(\boldsymbol{\Omega}, \widetilde{\boldsymbol{X}}) = \{\boldsymbol{H}'_w, \boldsymbol{H}'_b, \boldsymbol{C}'\}, \widetilde{\boldsymbol{X}} = \{\widetilde{\boldsymbol{x}}_i\}_{i=1}^{s}.$$

However, the computational cost of twice SVD on the large matrix $\boldsymbol{H}$ is extremely high (H. Zhao & Yuen, 2008). To address the problem, it was necessary to use an efficient and effective SVD updating method, but it was found difficult to conduct SVD-updating on $\boldsymbol{H}$ (H. Zhao & Yuen, 2008). Thus, H. Zhao and Yuen (2008) modified Ye's GSVD/LDA by computing $\boldsymbol{H}_t$ instead of $\boldsymbol{H}$, and derived an incremental GSVD/LDA (GSVD IncLDA) on the modified GSVD/LDA by updating singular vectors $\boldsymbol{Q}$ and $\boldsymbol{P}$, respectively. Thus, the modified GSVD/LDA models $\boldsymbol{\Omega} = \{\boldsymbol{H}_b, \boldsymbol{Q}, \boldsymbol{P}, \boldsymbol{C}\}$.

For $\boldsymbol{Q}$ in $\boldsymbol{\Omega}$, because it is the right singular vector of $\boldsymbol{H}$, and it is also the

eigenvector of the total scatter matrix $\boldsymbol{S}_t$, the method turns to update $\boldsymbol{S}_t$,

$$\begin{aligned}\boldsymbol{S}'_t &= \boldsymbol{H}'_t\boldsymbol{H}'^T_t \\ &= \boldsymbol{H}_t\boldsymbol{H}_t{}^T + (\widetilde{\boldsymbol{X}} - \widetilde{\boldsymbol{m}}\widetilde{\boldsymbol{e}}^T)(\widetilde{\boldsymbol{X}} - \widetilde{\boldsymbol{m}}\widetilde{\boldsymbol{e}}^T)^T \\ &+ \tfrac{n\widetilde{n}}{n+\widetilde{n}}(\boldsymbol{m} - \widetilde{\boldsymbol{m}})(\boldsymbol{m} - \widetilde{\boldsymbol{m}})^T,\end{aligned} \tag{2.29}$$

Let

$$\boldsymbol{G} = \left[\boldsymbol{X} - \boldsymbol{m}\boldsymbol{e}^T, \widetilde{\boldsymbol{X}} - \widetilde{\boldsymbol{m}}\widetilde{\boldsymbol{e}}^T, \sqrt{\frac{n\widetilde{n}}{n+\widetilde{n}}}(\boldsymbol{m} - \widetilde{\boldsymbol{m}})\right] \tag{2.30}$$

as we have

$$\widetilde{\boldsymbol{H}}_t = \left[\boldsymbol{X} - \boldsymbol{m}\boldsymbol{e}^T, \widetilde{\boldsymbol{X}} - \widetilde{\boldsymbol{m}}\widetilde{\boldsymbol{e}}^T\right], \tag{2.31}$$

then the updating of $\boldsymbol{Q}$ can be obtained by calculating the SVD of $\boldsymbol{G}$, since $\boldsymbol{G} = \boldsymbol{H}'_t$, and

$$\boldsymbol{Q}'^T\boldsymbol{S}'_t\boldsymbol{Q} = \boldsymbol{Q}'^T\boldsymbol{H}'^T_t\boldsymbol{H}'_t\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{R}'^2 & 0 \\ 0 & 0 \end{bmatrix} \tag{2.32}$$

When $\widetilde{\boldsymbol{X}}$ is inserted, the first k columns of $\boldsymbol{G}$ are unchanged, thus the SVD of $\boldsymbol{G}$ can take place using the SVD updating technique introduced by Zha and Simon (1997) as:

Given SVD on a $n \times d$ matrix $\boldsymbol{A}$ as $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{\Upsilon}$, the SVD of $[\boldsymbol{A}, \boldsymbol{B}]$ can be calculated as

$$[\boldsymbol{A}, \boldsymbol{B}] = ([\boldsymbol{U}_t, Q]\hat{\boldsymbol{U}})\hat{\boldsymbol{\Lambda}}\left(\begin{bmatrix} \boldsymbol{\Upsilon}_t & 0 \\ 0 & \boldsymbol{I} \end{bmatrix}\hat{\boldsymbol{\Upsilon}}\right)^T \tag{2.33}$$

where $t = rank(\boldsymbol{A})$, $\boldsymbol{U}_t$ and $\boldsymbol{\Upsilon}_t$ are constructed by the first t columns of $\boldsymbol{U}$ and $\boldsymbol{\Upsilon}$, respectively. The $\boldsymbol{Q}$ is from the QR decomposition $\boldsymbol{I} - \boldsymbol{U}_t\boldsymbol{U}_t^T = \boldsymbol{Q}\boldsymbol{L}$; The $\hat{\boldsymbol{U}}$, $\hat{\boldsymbol{\Lambda}}$, and $\hat{\boldsymbol{\Upsilon}}$ comes from the following SVD

$$\begin{bmatrix} \boldsymbol{\Lambda}_t & \boldsymbol{U}_t^T\boldsymbol{B} \\ 0 & \boldsymbol{L} \end{bmatrix} = \hat{\boldsymbol{U}}\hat{\boldsymbol{\Lambda}}\hat{\boldsymbol{\Upsilon}} \tag{2.34}$$

For updating $\boldsymbol{P}$ in the modified GSVD/LDA,

$$\boldsymbol{P}(:, 1:t) = \boldsymbol{H}\boldsymbol{Q}(:, 1:t)\boldsymbol{R}^{-1}, \tag{2.35}$$

Algorithm 6 GSVD IncLDA algorithm

Input: Newly added training data matrix $\widetilde{\boldsymbol{X}}$,and its class label set $\widetilde{\boldsymbol{C}}$, Modified GSVD/LDA model $\boldsymbol{\Omega} = \{\boldsymbol{H}_b, \boldsymbol{Q}, \boldsymbol{P}, \boldsymbol{C}\}$.

Output: Optimal transform matrix $\boldsymbol{\phi}'$, GSVD IncLDA model $\boldsymbol{\Omega}' = \{\boldsymbol{H}_b', \boldsymbol{Q}', \boldsymbol{P}', \boldsymbol{C}'\}$.

1: Construct matrix $\widetilde{\boldsymbol{X}} = \{\widetilde{\boldsymbol{x}}_1, \ldots, \widetilde{\boldsymbol{x}}_{\widetilde{n}}\}$;
2: Calculate updated training data matrix $\boldsymbol{X}' = \left[\boldsymbol{X} \;\; \widetilde{\boldsymbol{X}}\right]$;
3: Calculate $\boldsymbol{H}_b'$ on $\boldsymbol{X}'$;
4: Calculate matrix $\boldsymbol{G}$ by (2.31);
5: Calculate singular vectors $\boldsymbol{Q}'$ and $\boldsymbol{R}'$ by performing SVD updating (i.e., (2.33) and (2.34)) on $\boldsymbol{G}$;
6: Calculate rank of $\boldsymbol{H}_b'$: $\delta \leftarrow rank(\boldsymbol{H}_b')$;
7: Calculate matrix $\boldsymbol{P}'$ by (2.37);
8: Conduct SVD on $\boldsymbol{P}'$: $\boldsymbol{P} = \boldsymbol{U}'\boldsymbol{\Sigma}'\boldsymbol{W}'^T$;
9: Calculate matrix $\boldsymbol{\phi}_o = \boldsymbol{Q}' \begin{bmatrix} \boldsymbol{R}'^{-1}\boldsymbol{W}' & 0 \\ 0 & \boldsymbol{I} \end{bmatrix}$;
10: Calculate transform matrix $\boldsymbol{\phi}' = \boldsymbol{\phi}_o(:, 1:\delta)$.

According to Ye's GSVD/LDA (Ye et al., 2004), we have

$$\begin{aligned} \boldsymbol{P}(1:k, 1:t) &= \boldsymbol{H}(1:k, :)\boldsymbol{Q}(:, 1:t)\boldsymbol{R}^{-1} \\ &= \boldsymbol{H}_b\boldsymbol{Q}(:, 1:t)\boldsymbol{\Xi}(1:t, 1:t)^{-1}, \end{aligned} \tag{2.36}$$

by substituting $\boldsymbol{H} = \begin{bmatrix} \boldsymbol{H}_b^T \\ \boldsymbol{H}_w^T \end{bmatrix}$ to (2.36).

Therefore, $\boldsymbol{P}$ can be updated as

$$\boldsymbol{P}'(:, 1:t) = \boldsymbol{H}_b'\boldsymbol{Q}'(:, 1:t)\boldsymbol{\Xi}'(1:t, 1:t)^{-1} \tag{2.37}$$

In summary, algorithm 6 describes the pseudocode of the GSVD IncLDA,

LS IncLDA

L. Liu et al. (2009) proposed the incremental approach for LS/LDA. Given a 3-tuple LS/LDA model $\boldsymbol{\Omega} = \{\boldsymbol{M}, \boldsymbol{Y}, \boldsymbol{C}\}$, the incremental LS/LDA (LS IncLDA) updates the model as

$$\boldsymbol{\Omega}' = \mathcal{F}_{sq}(\boldsymbol{\Omega}, \widetilde{\boldsymbol{x}}) = \{\boldsymbol{M}', \boldsymbol{Y}', \boldsymbol{C}'\}.$$

Following (2.38), the centroid matrix $\boldsymbol{M}$ is updated as:

$$\boldsymbol{M}' = \left[\begin{array}{cc} \boldsymbol{M} - \frac{1}{n+1}(\widetilde{\boldsymbol{x}} - \boldsymbol{m}')\boldsymbol{e}^T & \frac{n}{n+1}(\widetilde{\boldsymbol{x}} - \boldsymbol{m}') \end{array} \right], \tag{2.38}$$

where $\boldsymbol{m}' = \boldsymbol{m} + \frac{1}{n+1}(\widetilde{\boldsymbol{x}} - \boldsymbol{m})$.

The indicator matrix $\boldsymbol{Y}$ can be updated as: For each new sample $\widetilde{\boldsymbol{x}}$, a new indicator vector $\widetilde{\boldsymbol{y}}$ is appended to $\boldsymbol{Y}$, and $\boldsymbol{Y} = \{y_1, \ldots, y_k\}^T$ is defined as

$$\boldsymbol{Y}_{ji} = \begin{cases} \frac{1}{\sqrt{n_k}} & \text{if } \boldsymbol{X}_{\boldsymbol{\cdot} i} \text{ belongs to class k} \\ 0 & \text{otherwise.} \end{cases} \tag{2.39}$$

1. if $\widetilde{c} \notin C$ then $k' = k + 1$,

$$\boldsymbol{Y}' = \left[\begin{array}{cc} \boldsymbol{Y} & 0 \\ \boldsymbol{y}^T & \end{array} \right] \tag{2.40}$$

2. otherwise $\widetilde{c} \in C$, then

$$\boldsymbol{Y}' = \left[\begin{array}{c} \boldsymbol{Y} \otimes_p \alpha_p \\ \boldsymbol{y}^T \end{array} \right] \tag{2.41}$$

where

$$\alpha_p = \sqrt{\frac{n_p}{n_p + 1}} \tag{2.42}$$

the operator $\otimes_p$ denotes multiplying the p-th column of $\boldsymbol{Y}$ with α_p.

In general, the updating of $\boldsymbol{Y}$ can be an unified form as

$$\boldsymbol{Y}' = \left[\begin{array}{c} \boldsymbol{Y} \bar{\otimes}_p \alpha_p \\ \boldsymbol{y}^T \end{array} \right], \tag{2.43}$$

in which $\bar{\otimes}_p$ is calculated if p is smaller than or equal to the width of $\boldsymbol{Y}$, the p-th column of $\boldsymbol{Y}$ is multiplied with α_p; otherwise, a new column with zero elements is appended to $\boldsymbol{Y}$.

Thus, an LS/LDA can be incremented by updating centroid data matrix $\boldsymbol{M}$ and indicator matrix $\boldsymbol{Y}$, then recalculating the MLR by (2.15).

Towards a computational efficient IncLDA, L. Liu et al. (2009) extended the batch LS/LDA model as $\boldsymbol{\Omega}_{LSLDA} = \{\boldsymbol{M}, \boldsymbol{M}^+, \boldsymbol{Y}, \boldsymbol{C}\}$, and had (2.15) updated on $\widetilde{\boldsymbol{x}}$ with the consideration of whether n is smaller than d or not.

In the case of $n < d$, assuming the rank of $\boldsymbol{M}$ increases by 1 when a new sample

is added. $\boldsymbol{M}^+$ can be updated as

$$\boldsymbol{M}'^{+} = \begin{bmatrix} \boldsymbol{M}^+ - \boldsymbol{M}^+(\widetilde{\boldsymbol{x}} - \boldsymbol{m})\boldsymbol{h}^T - \frac{1}{n}\boldsymbol{e}\boldsymbol{h}^T \\ \boldsymbol{h}^T \end{bmatrix} \tag{2.44}$$

where

$$\boldsymbol{h} = \frac{(\widetilde{\boldsymbol{x}} - \boldsymbol{m}) - \boldsymbol{M}\boldsymbol{M}^+(\widetilde{\boldsymbol{x}} - \boldsymbol{m})}{(\widetilde{\boldsymbol{x}} - \boldsymbol{m})^T(\widetilde{\boldsymbol{x}} - \boldsymbol{m}) - (\widetilde{\boldsymbol{x}} - \boldsymbol{m})^T\boldsymbol{M}\boldsymbol{M}^+(\widetilde{\boldsymbol{x}} - \boldsymbol{m})} \tag{2.45}$$

In this way, (2.15) is updated for LS IncLDA by applying (2.44) and (2.43) to calculate $\boldsymbol{M}'^{+}$ and $\boldsymbol{Y}'$, respectively.

$$\begin{aligned} \boldsymbol{\phi}' &= \boldsymbol{M}'^{+}\boldsymbol{Y}' \\ &= \left((\boldsymbol{M}^+)^T - \boldsymbol{h}(\widetilde{\boldsymbol{x}} - \boldsymbol{m})^T(\boldsymbol{M}^+)^T - \frac{\boldsymbol{h}\boldsymbol{e}^T}{n}\right)(\boldsymbol{Y}\bar{\otimes}_p\alpha_p) + \boldsymbol{h}\boldsymbol{y}^T \\ &= \left(\boldsymbol{\phi} - \boldsymbol{h}(\widetilde{\boldsymbol{x}} - \boldsymbol{m})^T\boldsymbol{\phi} - \frac{\boldsymbol{h}\boldsymbol{e}^T\boldsymbol{Y}}{n}\right)\bar{\otimes}_p\alpha_p + \boldsymbol{h}\boldsymbol{y}^T \end{aligned} \tag{2.46}$$

where $\boldsymbol{h}$ is defined as (2.45), α_p as (2.42) and $\boldsymbol{y}$ as (2.39).

When $n > d$, to avoid the computational cost of LS IncLDA growing with the number of samples, (2.15) can be interpolated as

$$\boldsymbol{\phi} = \boldsymbol{T}^+\boldsymbol{M}\boldsymbol{Y}, \tag{2.47}$$

since $\boldsymbol{T} = \boldsymbol{M}\boldsymbol{M}^T$.

In this sense, an incremental LS/LDA model can be constructed by just updating $\boldsymbol{T}^+$. To do that, $\boldsymbol{T}$ is firstly updated as

$$\begin{aligned} \boldsymbol{T}' &= \boldsymbol{M}'\boldsymbol{M}'^T = \boldsymbol{T} + \tfrac{n}{n+1}(\widetilde{\boldsymbol{x}} - \boldsymbol{m})(\widetilde{\boldsymbol{x}} - \boldsymbol{m})^T, \\ &= \boldsymbol{T} + \boldsymbol{\mu}\boldsymbol{\mu}^T, \end{aligned} \tag{2.48}$$

where $\boldsymbol{\mu} = \sqrt{\frac{n}{n+1}}(\widetilde{\boldsymbol{x}} - \boldsymbol{m})$. Then, $\boldsymbol{T}^+$ can be updated as

$$\boldsymbol{T}^+ = \begin{cases} \boldsymbol{T}^+ - \theta^{-1}\boldsymbol{s}\boldsymbol{s}^T & \text{if } t = 0 \\ \boldsymbol{T}^+ - \frac{\boldsymbol{s}\boldsymbol{r}^T + \boldsymbol{r}\boldsymbol{s}^T}{\boldsymbol{t}^T\boldsymbol{r}} + \frac{\theta\boldsymbol{r}\boldsymbol{r}^T}{(\boldsymbol{r}^T\boldsymbol{r})^2} & \text{otherwise,} \end{cases} \tag{2.49}$$

in which $\boldsymbol{s} = \boldsymbol{T}^+\boldsymbol{\mu}$, $\boldsymbol{r} = (\boldsymbol{I} - \boldsymbol{T}\boldsymbol{T}^+)\boldsymbol{\mu}$, and $\theta = 1 + \boldsymbol{\mu}^T\boldsymbol{T}^+\boldsymbol{\mu}$. The derivation of (2.49) can be found in L. Liu et al. (2009).

Applying (2.49) and (2.43) to calculate $\boldsymbol{T}^+$ and $\boldsymbol{Y}'$ respecetively, (2.47) can be

updated as

$$\begin{aligned}\boldsymbol{\phi}' &= \boldsymbol{T}^{+}\boldsymbol{M}'\boldsymbol{Y}', \\ &= \left(\boldsymbol{G} - \frac{\boldsymbol{T}^{+}(\widetilde{\boldsymbol{x}}-\boldsymbol{m})\boldsymbol{e}^{T}\boldsymbol{Y}}{n+1}\right) \bar{\otimes}_p \alpha_p + \frac{{}_n\boldsymbol{T}^{+}(\widetilde{\boldsymbol{x}}-\boldsymbol{m})\boldsymbol{y}}{n+1}\end{aligned} \tag{2.50}$$

where

$$\boldsymbol{G} = \begin{cases} \boldsymbol{\phi} - \theta^{-1} s \boldsymbol{u}^{T}\boldsymbol{\phi} & \text{if } \boldsymbol{r} = 0 \\ \boldsymbol{\phi} - \frac{\boldsymbol{t}\boldsymbol{u}^{T}\boldsymbol{\phi}}{\boldsymbol{r}^{T}\boldsymbol{r}} & \text{otherwise.} \end{cases} \tag{2.51}$$

In a summary, algorithm 7 presents the detailed computational steps for LS IncLDA by updating $\boldsymbol{T}^{+}$. The algorithm integrates two updating algorithms proposed in L. Liu et al. (2009) for both cases of $n < d$ and $n > d$.

Sequential IDR/QR IncLDA

Ye et al. (2005) proposed the incremental version of QR/LDA, which is called sequential Incremental IDR/QR LDA. With the insertion of a new sample $\widetilde{\boldsymbol{x}}$, the centroid matrix $\boldsymbol{M}$, $\boldsymbol{H}_w$, $\boldsymbol{H}_b$ will change accordingly, where $\boldsymbol{H}_w$ and $\boldsymbol{H}_b$ can be obtained by $\boldsymbol{S}_w = \boldsymbol{H}_w\boldsymbol{H}_w^{T}$ and $\boldsymbol{S}_b = \boldsymbol{H}_b\boldsymbol{H}_b^{T}$, respectively. Thus, the reduced within-class scatter matrix $\boldsymbol{W}$ and $\boldsymbol{B}$ are changed as well.

According to Ye et al. (2005), sIncLDA utilizes three stages to accomplish the incremental learning task as follows:

1. Updating centroid matrix $\boldsymbol{M}$ and its QR-decomposition.
2. Updating reduced within-class scatter matrix $\boldsymbol{W}$.
3. Updating reduced between-class scatter matrix $\boldsymbol{B}$.

Consequently, the sIncLDA can be formed as a 3-tuple model updating process from the original bath QR/LDA model $\boldsymbol{\Omega}$ as

$$\boldsymbol{\Omega}' = \mathcal{F}_{sq}(\boldsymbol{\Omega}, \widetilde{\boldsymbol{x}}).$$

The detailed review on this method is given in section 3.2.

2.2.4 Summary

As far as we know, the concept of chunk IncLDA is first discussed in Pang's IncLDA (Pang et al., 2005b), in which both sequential and chunk version of the incremental

Algorithm 7 LS IncLDA Algorithm

Input: Newly added training data sampe $\widetilde{\boldsymbol{x}}$, and its class label $\widetilde{c}$, LS/LDA eigenspace model $\boldsymbol{\Omega} = \{\boldsymbol{M}, \boldsymbol{M}^+, \boldsymbol{Y}, \boldsymbol{C}\}$.

Output: Optimal transform matrix $\boldsymbol{\phi}'$, LS IncLDA model $\boldsymbol{\Omega}' = \{\boldsymbol{M}', \boldsymbol{M}'^+, \boldsymbol{Y}', \boldsymbol{C}'\}$.

if $\widetilde{c} \in \boldsymbol{C}$ **then**
  Calculate $n_i = n_i + 1$;
else
  Set $n_i = 1$;
  Calculate $k = k + 1$;
end if
Calculate indicator matrix $\boldsymbol{Y}'$ by (2.40) and (2.41);
Calculate general mean vector $\boldsymbol{m}' = \boldsymbol{m} + \frac{1}{n+1}(\widetilde{\boldsymbol{x}} - \boldsymbol{m})$;
Calculate n and d as the size of centroid matrix $\boldsymbol{M}$;
if $d \leq n$ **then**
  Calculate matrix $\boldsymbol{T} = \boldsymbol{M}\boldsymbol{M}^T$;
  if $rank(\boldsymbol{T}) == d$ **then**
    Set $\boldsymbol{r} = 0$;
  else
    Calculate $\boldsymbol{r} = (\boldsymbol{I} - \boldsymbol{T}\boldsymbol{T}^+)\boldsymbol{\mu}$;
    Calculate total scatter matrix $\boldsymbol{T}'$ by (2.48);
  end if
  Calculate $\boldsymbol{T}^+$ by (2.49);
  Calculate optimal transform matrix $\boldsymbol{\phi}'$ by (2.50).
else
  Calculate centroid matrix $\boldsymbol{M}'$ by (2.38);
  Calculate pseudo-inverse centroid matrix $\boldsymbol{M}'^+$ by (2.44);
  Calculate optimal transform matrix $\boldsymbol{\phi}'$ by (2.50).
end if

Fisher LDA are developed and compared in terms of computation efficiency, memory usage, and eignespace discriminability. The comparison results indicate that the running efficiency of chunk IncLDA outperforms sequential IncLDA, and improves continuously with the increase in chunk size. However, since this method cannot address the SSS problem (Song, Liu, Zhang & Yang, 2008), thus is not applicable to

any problem that has its number of samples less than dimensionality. To mitigate the problem, H. Zhao and Yuen (2008) proposed an alternative chunk IncLDA, called GSVD IncLDA. This method uses GSVD as its solution to the SSS problem. The disadvantage of the method is, it is not a one-pass incremental learning. In fact, it always retains in memory all data samples newly added at each incremental learning cycle. Apparently, this makes the algorithm memory-inefficient over large size datasets. Considering the drawbacks of GSVD IncLDA, L. Liu et al. (2009) developed another very different IncLDA, called LS IncLDA. This is superior to both IncLDAs above, because it simply updates a multivariate linear regression (Ye, 2007), instead of using eigen-decomposition as all previous methods. However, it is worth noting that the LS IncLDA is sequential-based IL. As a result, inefficiencies may build up over numerous learning cycles.

The sequential IDR/QR IncLDA, proposed by Ye et al. (2005), updates the original batch QR/LDA by using an efficient QR-updating method. The superiority of the sIncLDA as compared with other IncLDAs can be seen as follows:

1. it conducts incremental dimensionality reduction in $d - k$ scale with the least loss of discriminative information.

2. it updates a QR-decomposition of a $d \times k$ matrix and solves an eigen-decomposition of a $k \times k$ matrix. Thus, it is computationally efficient.

3. it is one-pass incremental learning, where IL is executed by a single presentation of new data.

4. SSS problem is well-handled in the algorithm by applying regularization on the reduced within-class scatter matrix $\boldsymbol{W}$.

Table 2.1 compares sIncLDA against all above IncLDAs on four IL algorithm assessing criteria.

2.3 Motivations for the Presented Research

As indicated in Table 2.1, we are aware that the drawback of sIncLDA is that it conducts only sequential IL. Also, we notice that it uses an approximation for updating the reduced within-class scatter matrix $\boldsymbol{W}$, which may be corrupted in practice for data with a large number of classes. In order to process chunk data, we intend

Algorithm	Sequential	Chunk	One-Pass	Solving SSS problems
Pang's IncLDA (Pang et al., 2005b)	YES	YES	YES	NO
sIncLDA (Ye et al., 2005)	YES	NO	YES	YES
GSVD IncLDA (H. Zhao & Yuen, 2008)	YES	YES	NO	YES
LS IncLDA (L. Liu et al., 2009)	YES	NO	YES	YES

Table 2.1: Previous IncLDAs characterized in sequential, chunk, one-pass and SSS problem-solving properties.

to propose in this thesis a new type of chunk IncLDA on the basis of sIncLDA. The proposed cIncLDA not only enjoys the benefits of sIncLDA, but also significantly boosts its performance of the sIncLDA , because of 1) being capable of processing at one time multiple samples, so that its computational efficiency grows with the size of data chunk presented; it follows that the bigger the size of the dataset provided at one time, the faster the speed given by the proposed cIncLDA; and 2) the discriminability of eigenspace upgrades owing to the derivation of a more reliable and accurate update of the reduced within-class scatter matrix $\boldsymbol{W}$.

Chapter 3

Existing Sequential Incremental IDR/QR LDA

This chapter recaptures and analyzes the existing batch QR/LDA and sequential incremental IDR/QR LDA from an embedding point of view. Because our proposed cIncLDA is on the basis of sIncLDA, the technical details of sIncLDA is important and therefore is presented in this chapter. We first recapitulate the principles of batch QR/LDA and sIncLDA. Next, we summarize and discuss in depth the sIncLDA on its limitations.

3.1 Batch QR/LDA

The batch QR/LDA also follows the Fisher criterion (1.1) in two stages:

The first stage maximizes the between-class distance by solving an optimization problem as follows:

$$\boldsymbol{\phi} = \underset{\boldsymbol{\phi}^T\boldsymbol{\phi}=\boldsymbol{I}}{\operatorname{argmax}} \operatorname{Tr}(\boldsymbol{\phi}^T \boldsymbol{S}_b \boldsymbol{\phi}). \tag{3.1}$$

Ye et al. (2005) state that such a solution can be obtained by solving the eigenvalue problem on $\boldsymbol{S_b}$, or by applying QR-decomposition on the centroid matrix $\boldsymbol{M} = [\boldsymbol{m}_1, ..., \boldsymbol{m}_k]$ which is the so-called Orthogonal Centroid Method (OCM)(Park, Jeon & Rosen, 2003). In batch QR/LDA, the latter solution was applied. Hereby, the optimization transformation $\boldsymbol{\phi}$ can be obtained by $\boldsymbol{\phi} = \boldsymbol{QZ}$ for any orthogonal matrix $\boldsymbol{Z}$, where $\boldsymbol{Q} \in \mathcal{R}^{d\times k}$ is orthonormal and $\boldsymbol{R} \in \mathcal{R}^{k\times k}$ is upper triangular. Both are obtained from the reduced QR-decomposition on $\boldsymbol{M}$, namely $\boldsymbol{M} = \boldsymbol{QR}$. The

Algorithm 8 QR/LDA Algorithm

Input: Training data matrix $\boldsymbol{X} \in \mathcal{R}^{d \times n}$ and its class label set $\boldsymbol{C}$.

Output: Optimal transform matrix $\boldsymbol{\phi}$, LDA eigenspace model $\boldsymbol{\Omega} = \{\boldsymbol{M}, \boldsymbol{W}, \boldsymbol{B}, \boldsymbol{Q}, \boldsymbol{R}, \boldsymbol{C}\}$.

1: Calculate centroid matrix $\boldsymbol{M}$, within-class scatter matrix $\boldsymbol{S}_w$, between-class scatter matrix $\boldsymbol{S}_B$;

2: Calculate QR-decomposition of $\boldsymbol{M}$ by $\boldsymbol{M} = \boldsymbol{QR}$;

3: Calculate reduced within-class scatter matrix by $\boldsymbol{W} = \boldsymbol{Q}^T \boldsymbol{S}_w \boldsymbol{Q}$;

4: Calculate reduced between-class scatter matrix by $\boldsymbol{B} = \boldsymbol{Q}^T \boldsymbol{S}_B \boldsymbol{Q}$;

5: Calculate eigen-decomposition on $(\boldsymbol{W} + \mu \boldsymbol{I}_k)^{-1} \boldsymbol{B}$ to obtain $\boldsymbol{Z}$ containing k eigenvectors φ_i with decreasing eigenvalues;

6: Calculate optimal transform matrix by $\boldsymbol{\phi} = \boldsymbol{QZ}$.

detailed proof can be found in Ye et al. (2005). The second stage of batch QR/LDA seeks a transformation matrix $\boldsymbol{\phi}$ such that $\boldsymbol{\phi} = \boldsymbol{QZ}$ for some $\boldsymbol{Z}$, in order to minimize the within-classes distance. Since

$$\boldsymbol{\phi}^T \boldsymbol{S_b} \boldsymbol{\phi} = \boldsymbol{Z}^T (\boldsymbol{Q}^T \boldsymbol{S_b} \boldsymbol{Q}) \boldsymbol{Z}$$
$$\boldsymbol{\phi}^T \boldsymbol{S_w} \boldsymbol{\phi} = \boldsymbol{Z}^T (\boldsymbol{Q}^T \boldsymbol{S_w} \boldsymbol{Q}) \boldsymbol{Z},$$

the original optimization problem on finding $\boldsymbol{\phi}$ is equivalent to finding $\boldsymbol{Z}$. Here, two reduced scatter matrices $\boldsymbol{B} = \boldsymbol{Q}^T \boldsymbol{S_b} \boldsymbol{Q}$ and $\boldsymbol{W} = \boldsymbol{Q}^T \boldsymbol{S_w} \boldsymbol{Q}$ can be also obtained. Note that, $\boldsymbol{W}$ and $\boldsymbol{B}$ are much smaller in size than $\boldsymbol{S}_w$ and $\boldsymbol{S}_b$. This greatly improves computational efficiency of the algorithm. In addition, Ye et al. (2005) give a solution to computing $\boldsymbol{Z}$ by solving a small eigenvalue problem on $(\boldsymbol{W} + \mu \boldsymbol{I})^{-1} \boldsymbol{B}$ for $\mu > 0$. They also indicate that the solution is insensitive to the value of μ, where $\mu = 0.5$ was used in their experiments.

As discussed above, the batch QR/LDA can be written as a 3-tuple discriminant eigenspace model $\boldsymbol{\Omega} = \{\boldsymbol{M}, \boldsymbol{W}, \boldsymbol{B}\}$.

The pseudocode is provided in algorithm 8.

3.2 Sequential Incremental IDR/QR LDA

Given batch QR/LDA eigenspace model on $\boldsymbol{X}$ as $\boldsymbol{\Omega} = \{\boldsymbol{M}, \boldsymbol{W}, \boldsymbol{B}\}$. When a new instance $\widetilde{\boldsymbol{x}}$ is presented in the ith class, the problem of IncLDA can be described as

$$\boldsymbol{\Omega}' = \mathcal{F}_{sq}(\boldsymbol{\Omega}, \widetilde{\boldsymbol{x}}) = \{\boldsymbol{M}', \boldsymbol{W}', \boldsymbol{B}'\}$$

1. if $\widetilde{\boldsymbol{x}}$ belongs to an existing class (i.e., $i \leq k$), then $n'_i = n_i + 1$, $n' = n + 1$. The updating of the centroid matrix is calculated as

$$\boldsymbol{M}' = \boldsymbol{M} + f\boldsymbol{g}^T, \tag{3.2}$$

where $\boldsymbol{f} = \frac{\widetilde{\boldsymbol{x}} - \boldsymbol{m}_i}{n'_i}$ and $\boldsymbol{g} = (0, \ldots, 1, \ldots, 0)$ where the 1 appears at the ith position. By implementing QR-updating in two stages: 1) Rank-One QR-updating; 2) QR-updating as the case of insertion with a new row, we have the updated QR-decomposition so that $\boldsymbol{Q}'\boldsymbol{R}' = \boldsymbol{M}'$.

The updating of the reduced within-class scatter matrix $\boldsymbol{W}$ is

$$\boldsymbol{W}' = \boldsymbol{W} + (\boldsymbol{\alpha} - \boldsymbol{\beta})(\boldsymbol{\alpha} - \boldsymbol{\beta})^T + n_i\boldsymbol{\beta}\boldsymbol{\beta}^T, \tag{3.3}$$

where $\boldsymbol{\alpha} = \boldsymbol{Q}'^T(\widetilde{\boldsymbol{x}} - \boldsymbol{m}_i)$ and $\boldsymbol{\beta} = \boldsymbol{Q}'^T(\boldsymbol{m}'_i - \boldsymbol{m}_i)$.

The updating of the reduced between-class scatter matrix $\boldsymbol{B}$ is

$$\boldsymbol{B}' = (\boldsymbol{R}'\boldsymbol{D} - (\frac{1}{n'}\boldsymbol{R}' \cdot r) \cdot h^T)(\boldsymbol{R}'\boldsymbol{D} - (\frac{1}{n'}\boldsymbol{R}' \cdot r) \cdot h^T)^T, \tag{3.4}$$

where $\boldsymbol{D} = diag(\sqrt{n'_1}, \ldots, \sqrt{n'_k})$, $h = [\sqrt{n'_1}, \ldots, \sqrt{n'_k}]^T$, and $r = (n'_1, \ldots, n'_k)$

2. if $\widetilde{\boldsymbol{x}}$ belongs to a new class (i.e., $i > k$), then for $i = 1, \ldots, k$, $n'_i = n_i$, $n'_{k+1} = 1$, and $n' = n + 1$. The updated centroid matrix is

$$\boldsymbol{M}' = [\boldsymbol{m}_1, \boldsymbol{m}_2, \ldots, \boldsymbol{m}_k, \widetilde{\boldsymbol{x}}] = [\boldsymbol{M}, \widetilde{\boldsymbol{x}}]. \tag{3.5}$$

Then, the updated reduced within-class scatter matrix is

$$\boldsymbol{W}' = \begin{pmatrix} \boldsymbol{W} & 0 \\ 0 & 0 \end{pmatrix} \tag{3.6}$$

The reduced between-class scatter matrix can be updated similarly to (3.4) as

$$\boldsymbol{B}' = (\boldsymbol{R}'\boldsymbol{D} - (\frac{1}{n'}\boldsymbol{R}' \cdot r) \cdot h^T)(\boldsymbol{R}'\boldsymbol{D} - (\frac{1}{n'}\boldsymbol{R}' \cdot r) \cdot h^T)^T, \tag{3.7}$$

with $\boldsymbol{D} = diag(\sqrt{n'_1}, \ldots, \sqrt{n'_{k+1}})$, $h = [\sqrt{n'_1}, \ldots, \sqrt{n'_{k+1}}]^T$, and $r = (n'_1, \ldots, n'_{k+1})$

The steps of sIncLDA above are summarized in algorithm 9.

Algorithm 9 Sequential IDR/QR IncLDA Algorithm

Input: Newly added training data sample $\widetilde{\boldsymbol{x}}$ and its class label $\widetilde{\boldsymbol{c}}$, LDA/sIncLDA eigenspace model $\boldsymbol{\Omega} = \{\boldsymbol{M}, \boldsymbol{W}, \boldsymbol{B}, \boldsymbol{Q}, \boldsymbol{R}, \boldsymbol{C}\}$.

Output: Optimal transform matrix $\boldsymbol{\phi}'$, sIncLDA eigenspace model $\boldsymbol{\Omega'} = \{\boldsymbol{M'}, \boldsymbol{W'}, \boldsymbol{B'}, \boldsymbol{Q'}, \boldsymbol{R'}, \boldsymbol{C'}\}$.

1: **if** $\widetilde{\boldsymbol{c}} \in \boldsymbol{C}$ **then**
2: Calculate centroid matrix $\boldsymbol{M'}$ by (3.2);
3: Calculate reduced between-class matrix $\boldsymbol{B'}$ by (3.4);
4: Calculate reduced within-class matrix $\boldsymbol{W'}$ by (3.3);
5: **else**
6: Calculate centroid matrix $\boldsymbol{M'}$ by(3.5);
7: Calculate reduced between-class matrix $\boldsymbol{B'}$ by (3.7);
8: Calculate reduced within-class matrix $\boldsymbol{W'}$ by (3.6);
9: **end if**
10: Calculate k' eigenvectors $\boldsymbol{\varphi_i}'$ of $(\boldsymbol{W'} + \mu \boldsymbol{I}_{k'})^{-1}\boldsymbol{B'}$ with decreasing eigenvalues;
11: Calculate optimal transform matrix $\boldsymbol{\phi}' = \boldsymbol{Q'}\boldsymbol{Z'}$.

3.3 Discussion

It is worth noting that when data is presented in a chunk manner, the sIncLDA performs learning inefficiently for a single sample at a time. This is because a large number of iterative updates are carried out in sIncLDA for just one sample, and this might be discriminatively redundant (i.e., gives no contribution to the existing discriminant model). Furthermore, the class separability of the sIncLDA deducts for datasets with a large class number. This is because when the $\boldsymbol{W}$ is being updated, discriminative information loss occurs, 1) when the newly presented sample belongs to an existing class, the sIncLDA has the assumption of $\boldsymbol{Q'} \approx \boldsymbol{Q}$ in (3.3); and 2) when the newly presented sample comes from a new class, the sIncLDA has another assumption of $\boldsymbol{W'} \approx \boldsymbol{W}$ in (3.6). However, these two assumptions often lose generality in practice, especially when the class number is large.

Chapter 4

Proposed Chunk Incremental IDR/QR LDA

In this chaper, we propose our new chunk incremental IDR/QR LDA. Given an eigenspace model $\mathbf{\Omega} = \{\boldsymbol{M}, \boldsymbol{W}, \boldsymbol{B}\}$, and a chunk of samples $\widetilde{\boldsymbol{X}}$ in $\widetilde{k}$ classes, regardless of existing or entirely new categories. Simply put, the set of total class labels is updated as $\boldsymbol{C'} = \boldsymbol{C} \cup (\widetilde{\boldsymbol{C}} - \boldsymbol{C} \cap \widetilde{\boldsymbol{C}})$ and $k' = |\boldsymbol{C'}|$. The number of total training samples is renewed as $n' = n + \widetilde{n}$. Then, the problem of the proposed IncLDA can be described as

$$\mathbf{\Omega}' = \mathcal{F}_{ck}(\mathbf{\Omega}, \widetilde{\boldsymbol{X}}) = \{\boldsymbol{M'}, \boldsymbol{W'}, \boldsymbol{B'}\}.$$

$\mathcal{F}_c$ updates $\mathbf{\Omega}$ on $\widetilde{\boldsymbol{X}}$ in three steps: (1) Updating centroid matrix $\boldsymbol{M}$ and its QR decomposition; (2) Updating the reduced within-class scatter matrix $\boldsymbol{W}$; and (3) Updating the reduced between-class scatter matrix $\boldsymbol{B}$. Note that differing from Sequential IDR/QR IncLDA, Chunk IDR/QR IncLDA takes care of newly added data from existing and new classes simultaneously.

4.1 Updating the Centroid Matrix and its QR-decomposition

To update $\boldsymbol{M} \in \mathcal{R}^{d \times k}$ to $\boldsymbol{M'} \in \mathcal{R}^{d \times k'}$, the dimensionality of $\boldsymbol{M}$ is augmented from k to k' for accommodating new classes in the centroid matrix. In doing so, we insert zero vectors $\mathbf{0}$ at ith column of $\boldsymbol{M}$ for $i = k+1, \ldots, k'$ and $\widetilde{\boldsymbol{M}}$ for $i = \widetilde{k}+1, \ldots, k'$. Thus, $\boldsymbol{M} \in \mathcal{R}^{d \times k}$ is augmented to $\hat{\boldsymbol{M}} \in \mathcal{R}^{d \times k'}$ upon the presence of $\widetilde{\boldsymbol{X}}$. Here, ‘ˆ’

identifies the result of a matrix augmentation, which will be applied throughout the paper.

Applying matrix augmentation to $\boldsymbol{M} = \boldsymbol{QR}$, we have $\hat{\boldsymbol{M}} = \hat{\boldsymbol{Q}}\hat{\boldsymbol{R}}$. $\hat{\boldsymbol{Q}}\hat{\boldsymbol{R}}$ can be obtained by recomputing a QR decomposition on $\hat{\boldsymbol{M}}$, or by a QR-updating on $\boldsymbol{QR}$. However, both approaches are computationally expensive due to the high complexity of matrix decomposition or multiplication. For efficient $\hat{\boldsymbol{Q}}\hat{\boldsymbol{R}}$ augmentation, we propose the rule of matrix augmentation for QR decomposition as Proposition 1, which involves only manipulations of matrix partition or combination.

Proposition 1 *Given* $\boldsymbol{A} = (\boldsymbol{A}_1, \boldsymbol{A}_2) = \boldsymbol{QR} \in \mathcal{R}^{d\times k}$ *with* $\boldsymbol{A}_1 \in \mathcal{R}^{d\times(i-1)}$, *and* $\boldsymbol{A}_2 \in \mathcal{R}^{d\times(k-i)}$. *If a zero vector* $\mathbf{0}$ *is inserted as* $\hat{\boldsymbol{A}} = (\boldsymbol{A}_1, \mathbf{0}, \boldsymbol{A}_2)$, *then the QR decomposition* $\hat{\boldsymbol{Q}}\hat{\boldsymbol{R}}$ *of* $\hat{\boldsymbol{A}}$ *can be computed as*

$$\hat{\boldsymbol{Q}}\hat{\boldsymbol{R}} = (\boldsymbol{Q}_1, \mathbf{0}, -\boldsymbol{Q}_2)\begin{pmatrix} \boldsymbol{R}_1 & \mathbf{0} & \boldsymbol{R}_2 \\ \mathbf{0}^T & 1 & \mathbf{0}^T \\ \boldsymbol{R}_3 & \mathbf{0} & -\boldsymbol{R}_4, \end{pmatrix}$$

where $\boldsymbol{Q}_1 \in \mathcal{R}^{d\times(i-1)}$, $\boldsymbol{Q}_2 \in \mathcal{R}^{d\times(k-i)}$, $\boldsymbol{R}_1 \in \mathcal{R}^{(i-1)\times(i-1)}$, $\boldsymbol{R}_2 \in \mathcal{R}^{(i-1)\times(k-i)}$, $\boldsymbol{R}_3 \in \mathcal{R}^{(k-i)\times(i-1)}$, *and* $\boldsymbol{R}_4 \in \mathcal{R}^{(k-i)\times(k-i)}$. *Note that,* $\boldsymbol{Q} = (\boldsymbol{Q}_1, \boldsymbol{Q}_2)$, $\boldsymbol{R} = \begin{pmatrix} \boldsymbol{R}_1 & \boldsymbol{R}_2 \\ \boldsymbol{R}_3 & \boldsymbol{R}_4 \end{pmatrix}$, *and* $\boldsymbol{R}_3$ *is a zero matrix. The detailed proof is given in Appendix A.*

According to the definition of centroid matrix, we reformulate $\boldsymbol{M'} = [\frac{\boldsymbol{m}_1 n_1 + \widetilde{\boldsymbol{m}}_1 \widetilde{n}_1}{n'_1}, \ldots, \frac{\boldsymbol{m}_{k'} n_{k'} + \widetilde{\boldsymbol{m}}_{k'} \widetilde{n}_{k'}}{n'_{k'}}]$ as,

$$\boldsymbol{M'} = \hat{\boldsymbol{M}} + \boldsymbol{\Delta}. \tag{4.1}$$

Then, we have $\boldsymbol{\Delta} = [\frac{\widetilde{n}_1}{n'_1}(\widetilde{\boldsymbol{m}}_1 - \boldsymbol{m}_1), \ldots, \frac{\widetilde{n}_{k'}}{n'_{k'}}(\widetilde{\boldsymbol{m}}_{k'} - \boldsymbol{m}_{k'})] \in \mathcal{R}^{d\times k'}$. Note that $\boldsymbol{\Delta}$ here identifies the residue information of newly presented against existing data.

To calculate $\boldsymbol{Q'R'}$, we apply the thin Singular Value Decomposition on $\boldsymbol{\Delta}$, namely $\boldsymbol{\Delta} = \boldsymbol{U\Sigma V^T}$ and obtain a summation of rank one matrices products (Golub & Van Loan, 1996) as,

$$\boldsymbol{\Delta} = \sum_{j=1}^{\eta} \boldsymbol{\Sigma}(j,j)\boldsymbol{U}(1:d,j)\boldsymbol{V^T}(j,1:k') = \sum_{j=1}^{\eta} \boldsymbol{u_j}{\boldsymbol{v_j}}^T \tag{4.2}$$

where $\eta = \text{rank}(\boldsymbol{\Delta})$, $\boldsymbol{u_j} = \boldsymbol{\Sigma}(j,j)\boldsymbol{U}(1:d,j)$ and $\boldsymbol{v_j} = \boldsymbol{V}(j,1:k')$.

Substituting (4.2) into (4.1), we can calculate $\boldsymbol{Q}'\boldsymbol{R}'$ by an iterative QR-updating procedure as,

$$\begin{aligned}
\boldsymbol{Q}'\boldsymbol{R}' &= \hat{\boldsymbol{Q}}^{(\eta+1)}\hat{\boldsymbol{R}}^{(\eta+1)} \\
&= \hat{\boldsymbol{Q}}^{(\eta)}\hat{\boldsymbol{R}}^{(\eta)} + \boldsymbol{u}_\eta\boldsymbol{v}_\eta^T \\
&\cdots \\
&= (\hat{\boldsymbol{Q}}^{(j)}\hat{\boldsymbol{R}}^{(j)} + \boldsymbol{u}_j\boldsymbol{v}_j^T) + \textstyle\sum_{j=j+1}^{\eta}\boldsymbol{u}_j\boldsymbol{v}_j^T \\
&\cdots \\
&= (\hat{\boldsymbol{Q}}^{(2)}\hat{\boldsymbol{R}}^{(2)} + \boldsymbol{u}_2\boldsymbol{v}_2^T) + \textstyle\sum_{j=3}^{\eta}\boldsymbol{u}_j\boldsymbol{v}_j^T \\
&= (\hat{\boldsymbol{Q}}^{(1)}\hat{\boldsymbol{R}}^{(1)} + \boldsymbol{u}_1\boldsymbol{v}_1^T) + \textstyle\sum_{j=2}^{\eta}\boldsymbol{u}_j\boldsymbol{v}_j^T \\
&= \hat{\boldsymbol{Q}}\hat{\boldsymbol{R}} + \textstyle\sum_{j=1}^{\eta}\boldsymbol{u}^j\boldsymbol{v}_j^T.
\end{aligned} \tag{4.3}$$

Here, we define the recursive function $\mathcal{P}$ such that $\hat{\boldsymbol{Q}}^{(j+1)}\hat{\boldsymbol{R}}^{(j+1)} = \mathcal{P}(\hat{\boldsymbol{Q}}^{(j)}\hat{\boldsymbol{R}}^{(j)})$. Thus, for each iteration of (4.3),

$$\begin{aligned}
\mathcal{P}(\hat{\boldsymbol{Q}}^{(j)}\hat{\boldsymbol{R}}^{(j)}) &= \hat{\boldsymbol{Q}}^{(j)}\hat{\boldsymbol{R}}^{(j)} + \boldsymbol{u}_j\boldsymbol{v}_j^T \\
&= \hat{\boldsymbol{Q}}^{(j)}(\hat{\boldsymbol{R}}^{(j)} + \hat{\boldsymbol{Q}}^{(j)T}\boldsymbol{u}_j\boldsymbol{v}_j^T) + (\boldsymbol{I} - \hat{\boldsymbol{Q}}^{(j)}\hat{\boldsymbol{Q}}^{(j)T})\boldsymbol{u}_j\boldsymbol{v}_j^T \\
&= \hat{\boldsymbol{Q}}^{(j)}(\hat{\boldsymbol{R}}^{(j)} + \boldsymbol{w}^{(j)}\boldsymbol{v}_j^T) + \boldsymbol{f}^{(j)}\boldsymbol{v}_j^T,
\end{aligned} \tag{4.4}$$

where $j = [1, \ldots, \eta]$, and $\boldsymbol{w}^{(j)} = \hat{\boldsymbol{Q}}^{(j)T}\boldsymbol{u}_j$ and $\boldsymbol{f}^{(j)} = (\boldsymbol{I} - \hat{\boldsymbol{Q}}^{(j)}\hat{\boldsymbol{Q}}^{(j)T})\boldsymbol{u}_j$. Now, we can calculate $\boldsymbol{Q}'\boldsymbol{R}'$ by applying the two-stage QR-updating method on (4.4) as in Ye et al. (2005).

4.2 Updating the Reduced Within-Class Scatter Matrix

To update the reduced within-class scatter matrix $\boldsymbol{W}$, we augment, similar to the updating of $\boldsymbol{M}$, the k-dimension $\boldsymbol{W}$ to k'-dimension $\hat{\boldsymbol{W}}$. Again to avoid matrix multiplication, we simplify the augmentation of $\boldsymbol{W}$ following the rule of Proposition 2:

Proposition 2 *Given* $\boldsymbol{W} = \boldsymbol{Q}^T\boldsymbol{H}_w\boldsymbol{H}_w^T\boldsymbol{Q} \in \mathcal{R}^{k\times k}$*, if a zero vector* $\mathbf{0}$ *is inserted as the* i*-th column into* $\boldsymbol{Q} \in \mathcal{R}^{d\times k}$*, such that* $\hat{\boldsymbol{Q}} = (\boldsymbol{Q}_1, \mathbf{0}, -\boldsymbol{Q}_2) \in \mathcal{R}^{d\times(k+1)}$ *where*

$\boldsymbol{Q}_1 \in \mathcal{R}^{d\times(i-1)}$ and $\boldsymbol{Q}_2 \in \mathcal{R}^{d\times(k-i)}$, then we have

$$\begin{aligned}\hat{\boldsymbol{W}} &= \hat{\boldsymbol{Q}}^T \boldsymbol{H}_w \boldsymbol{H}_w^T \hat{\boldsymbol{Q}} \\ &= \begin{pmatrix} \boldsymbol{W}_1 & \boldsymbol{0} & -\boldsymbol{W}_2 \\ \boldsymbol{0}^T & 0 & \boldsymbol{0}^T \\ -\boldsymbol{W}_3 & \boldsymbol{0} & \boldsymbol{W}_4, \end{pmatrix},\end{aligned}$$

where $\boldsymbol{W}_1 \in \mathcal{R}^{(i-1)\times(i-1)}$, $\boldsymbol{W}_2 \in \mathcal{R}^{(i-1)\times(k-i)}$, $\boldsymbol{W}_3 \in \mathcal{R}^{(k-i)\times(i-1)}$, $\boldsymbol{W}_4 \in \mathcal{R}^{(k-i)\times(k-i)}$, and $\boldsymbol{W} = \begin{pmatrix} \boldsymbol{W}_1 & \boldsymbol{W}_2 \\ \boldsymbol{W}_3 & \boldsymbol{W}_4, \end{pmatrix}$. The detailed proof is given in Appendix B.

Based on $\hat{\boldsymbol{W}}$, we can compute the reduced within-class scatter matrix $\boldsymbol{W'}$ by the following proposition 3.

Proposition 3 *Given an augmented reduced within-class scatter matrix $\hat{\boldsymbol{W}}$ and a set of new training samples $\widetilde{\boldsymbol{X}}$, the reduced within-class scatter matrix can be updated as*

$$\boldsymbol{W'} = \bar{\boldsymbol{W}} + \widetilde{\boldsymbol{W}} + \boldsymbol{E} \tag{4.5}$$

where $\bar{\boldsymbol{W}} = \boldsymbol{Q'}^T \boldsymbol{H_w} \boldsymbol{H_w}^T \boldsymbol{Q'}$, $\widetilde{\boldsymbol{W}} = \boldsymbol{Q'}^T \widetilde{\boldsymbol{H}}_w \widetilde{\boldsymbol{H}}_w^T \boldsymbol{Q'}$, $\boldsymbol{E} = \sum_{i=1}^{k'} (\widetilde{n}_i(\boldsymbol{\alpha'} - \boldsymbol{\beta'})(\boldsymbol{\alpha'} - \boldsymbol{\beta'})^T + n_i \boldsymbol{\beta'} \boldsymbol{\beta'}^T))$, $\boldsymbol{\alpha'} = \boldsymbol{Q'}^T(\widetilde{\boldsymbol{m}}_{\boldsymbol{i}} - \boldsymbol{m_i})$, and $\boldsymbol{\beta'} = \boldsymbol{Q'}^T(\boldsymbol{m'_i} - \boldsymbol{m_i})$. The detailed proof is given in Appendix C.

Consider $\bar{\boldsymbol{W}} = \boldsymbol{Q'}^T \boldsymbol{H_w} \boldsymbol{H_w}^T \boldsymbol{Q'}$ and $\boldsymbol{Q}^{(j+1)} = \mathcal{P}(\boldsymbol{Q}^{(j)})$, there should exist an iterative function $\mathcal{P}_W$ for $\boldsymbol{W}$

$$\boldsymbol{W}^{(j+1)} = \mathcal{P}_W(\boldsymbol{W}^{(j)}), j = 1, \ldots, \eta \tag{4.6}$$

such that $\bar{\boldsymbol{W}} = \boldsymbol{W}^{(\eta+1)}$ when $j = \eta$.

Following (4.6), we have

$$\begin{aligned}\boldsymbol{W}^{(j+1)} &= \hat{\boldsymbol{Q}}^{(j+1)T} \boldsymbol{H}_w \boldsymbol{H}_w^T \hat{\boldsymbol{Q}}^{(j+1)} \\ &= [\mathcal{P}(\hat{\boldsymbol{Q}}^{(j)T})] \boldsymbol{H}_w \boldsymbol{H}_w^T [\mathcal{P}(\hat{\boldsymbol{Q}}^{(j)})] \\ &\approx \boldsymbol{G}^{(j)T} \cdot \boldsymbol{J}^{(j)T} \cdot \hat{\boldsymbol{Q}}^{(j)T} \boldsymbol{H}_w \boldsymbol{H}_w^T \hat{\boldsymbol{Q}}^{(j)} \cdot \boldsymbol{J}^{(j)} \cdot \boldsymbol{G}^{(j)} \\ &= \boldsymbol{G}^{(j)T} \cdot \boldsymbol{J}^{(j)T} \cdot \hat{\boldsymbol{W}}^{(j)} \cdot \boldsymbol{J}^{(j)} \cdot \boldsymbol{G}^{(j)},\end{aligned} \tag{4.7}$$

where $\boldsymbol{G}^{(j)} = [\boldsymbol{G}^{(j)}_{1,2}, \ldots, \boldsymbol{G}^{(j)}_{k'-1,k'}]$ and $\boldsymbol{J}^{(j)} = [\boldsymbol{J}^{(j)}_{k'-1,k'}, \ldots, \boldsymbol{J}^{(j)}_{1,2}]$ are two sets of Givens rotations obtained from the first QR-updating stage; "$\cdot$" denotes that applying the Givens rotations on $\hat{\boldsymbol{W}}^{(j)}$ one by one. Note that the approximation in (4.7) assumes that $\hat{\boldsymbol{W}}^{(j)}$ is not varied in the second QR-updating stage.

4.3 Updating the Reduced Between-Class Scatter Matrix

We use the same process for updating $\boldsymbol{B}$ as in (Ye et al., 2005), which can be written as

$$\boldsymbol{B'} = \left(\boldsymbol{R'D} - (\tfrac{1}{n'}\boldsymbol{R'r})\boldsymbol{h}^T \right) \left(\boldsymbol{R'D} - (\tfrac{1}{n'}\boldsymbol{R'r})\boldsymbol{h}^T \right)^T, \tag{4.8}$$

where $\boldsymbol{D} = \text{diag}(\sqrt{n'_1}, \ldots, \sqrt{n'_{k'}})$, $\boldsymbol{r} = (n'_1, \ldots, n'_{k'})^T$, and $\boldsymbol{h} = [\sqrt{n'_1}, \ldots, \sqrt{n'_{k'}}]^T$.

4.4 The Pseudocode of the Proposed Algorithm

The pseudocode of the proposed algorithm is presented in algorithm 10.

4.5 Time Complexity Analysis

This section provides the time complexity analysis of the proposed cIncLDA with a comparison to the sIncLDA. The term 'flam', which denotes an addition and a multiplication, is used for presenting operation counts (G.W., 1998). Consider that both IncLDAs calculate an update of the 3-tuple model $\{\boldsymbol{M}, \boldsymbol{W}, \boldsymbol{B}, \}$. We measure the time complexity of each variable update for every single new sample presented for incremental learning. Here, for each variable updating, we only consider the maximum time cost regardless of whatever class the incoming sample belongs to.

For sIncLDA, the maximum time cost occurs when the new sample belongs to an existing class, since this involves more complicated calculations on $\boldsymbol{W}$ and $\boldsymbol{QR}$ than when a new sample comes from a new class. In this case, the updating of $\boldsymbol{M}$ requires dk flam, namely $O(dk)$. According to Daniel, Gragg, Kaufman and Stewart (1976), total operation counts for the two-stage QR-updating are $9dk + \frac{9}{2}k^2$ flam, so its time complexity is written as $O(dk + k^2)$. With regard to the scatter matrices, it requires $dk + 2k^2 + k$ flam for updating $\boldsymbol{W}$ and $\frac{3}{2}k^3 + 2k^2$ flam for $\boldsymbol{B}$. Hence, the

Algorithm 10 Chunk IDR/QR IncLDA Algorithm

Input: Newly added training data matrix $\widetilde{\boldsymbol{X}}$ and its class label set $\widetilde{\boldsymbol{C}}$, LDA/cIncLDA eigenspace model $\boldsymbol{\Omega} = \{\boldsymbol{M}, \boldsymbol{W}, \boldsymbol{B}, \boldsymbol{Q}, \boldsymbol{R}, \boldsymbol{C}\}$.

Output: Optimal transform matrix $\boldsymbol{\phi}'$, cIncLDA eigenspace model $\boldsymbol{\Omega'} = \{\boldsymbol{M'}, \boldsymbol{W'}, \boldsymbol{B'}, \boldsymbol{Q'}, \boldsymbol{R'}, \boldsymbol{C'}\}$.

Calculate current class label set $\boldsymbol{C'} = \boldsymbol{C} \cup (\widetilde{\boldsymbol{C}} - \boldsymbol{C} \cap \widetilde{\boldsymbol{C}})$, $\widetilde{k} = \|\widetilde{\boldsymbol{C}}\|$, and $k' = \|\boldsymbol{C'}\|$;
Calculate centroid matrix $\widetilde{\boldsymbol{M}} = [\widetilde{\boldsymbol{m}}_1, \ldots, \widetilde{\boldsymbol{m}}_{\widetilde{k}}]$, and within-class scatter matrix $\widetilde{\boldsymbol{S}}_{\boldsymbol{w}} = \sum_{i=1}^{\widetilde{k}} \sum_{j=1}^{\widetilde{n}_i} (\widetilde{\boldsymbol{x}}_{ij} - \widetilde{\boldsymbol{m}}_i)(\widetilde{\boldsymbol{x}}_{ij} - \widetilde{\boldsymbol{m}}_i)^T$;
if $k' > \widetilde{k}$ **then**
    Conduct matrix augmentation on $\widetilde{\boldsymbol{M}}$ by inserting $k' - \widetilde{k}$ zero vectors;
end if
if $k' > k$ **then**
    Conduct matrix augmentation on $\boldsymbol{M}$, $\boldsymbol{Q}$, $\boldsymbol{R}$, $\boldsymbol{W}$ by Proposition 1 and Proposition 2;
end if
Calculate centroid matrix $\boldsymbol{M'}$ and residue matrix $\boldsymbol{\Delta}$ by (4.1);
Calculate the rank $\eta = \text{rank}(\boldsymbol{\Delta})$;
Perform SVD of $\boldsymbol{\Delta}$ as, $\boldsymbol{\Delta} = \boldsymbol{U\Sigma V}^T$;
for $j = 1$ to η **do**
    Calculate vector $\boldsymbol{w} = \boldsymbol{Q}^T\boldsymbol{\Sigma}(\boldsymbol{j}, \boldsymbol{j})\boldsymbol{U}(:, \boldsymbol{j})$, and $\boldsymbol{v} = \boldsymbol{Q}^T\boldsymbol{V}(\boldsymbol{j}, :)$;
    Calculate $\boldsymbol{Q'R'}$ by updating QR-decomposition as (4.3) and (4.4);
    Calculate matrix $\bar{\boldsymbol{W}}$ by (4.7);
end for
Calculate $\boldsymbol{E}$ and $\widetilde{\boldsymbol{W}}$ by Proposition 3;
Calculate reduced within-class scatter matrix $\boldsymbol{W'}$ by Proposition 3;
Calculate reduced between-class scatter matrix $\boldsymbol{B'}$ by (4.8);
Calculate k' eigenvectors $\boldsymbol{\varphi}'_i$ of $(\boldsymbol{W'} + \mu\boldsymbol{I}_{k'})^{-1}\boldsymbol{B'}$ with decreasing eigenvalues;
Calculate optimal transform matrix $\boldsymbol{\phi'} = \boldsymbol{Q'Z'}$, where $\boldsymbol{Z'} = [\boldsymbol{\varphi}'_1, \ldots, \boldsymbol{\varphi}'_{k'}]$.

time complexity for updating $\boldsymbol{W}$ and $\boldsymbol{B}$ is simplified as $O(dk + k^2)$ and $O(k^3 + k^2)$, respectively.

In the proposed cIncLDA, a matrix augmentation operation is employed before all updates. We have $\boldsymbol{M} \in \mathcal{R}^{d\times k}$, $\widetilde{\boldsymbol{M}} \in \mathcal{R}^{d\times\widetilde{k}}$, $\boldsymbol{Q} \in \mathcal{R}^{d\times k}$, $\boldsymbol{R} \in \mathcal{R}^{k\times k}$ and $\boldsymbol{W} \in \mathcal{R}^{k\times k}$ augmented from k or $\widetilde{k}$ to k', so the total time complexity for augmentation is $O(dk' + k'^2)$. Updating $\hat{\boldsymbol{M}}$ requires dk' flam whose time complexity is $O(dk')$. For

Main step	**Sequential IDR/QR IncLDA**	**Chunk IDR/QR IncLDA**
Matrices Augmentation	NO	$O(dk' + k'^2)$
Update $\boldsymbol{M}$	$O(dk)$	$O(dk')$
Update $\boldsymbol{Q}$ and $\boldsymbol{R}$	$O(dk + k^2)$	$O(dk'^2 + k'^3 + dk' + k'^2)$
Update $\boldsymbol{W}$	$O(dk + k^2)$	$O(k'^3 + dk' + k'^2)$
Update $\boldsymbol{B}$	$O(k^3 + k^2)$	$O(k'^3 + k'^2)$
General	$O(k^3 + dk)$	$O(dk'^2 + k'^3 + dk')$

Table 4.1: Comparison of Time Complexity for incremental learning on a single sample, sIncLDA vs cIncLDA

updating the $\hat{\boldsymbol{Q}}\hat{\boldsymbol{R}}$, the SVD on $\boldsymbol{\Delta}$ requires $4dk'^2 + 8k'^3$ flam according to G.W. (1998), and the following two-stage QR-updating on $\hat{\boldsymbol{Q}}\hat{\boldsymbol{R}}$ is a η-iteration process, which requires $9\eta dk' + \frac{9}{2}\eta k'^2$ flam. Adding the above two steps, it gives the total time complexity $O(dk'^2+k'^3+dk'+k'^2)$ for updating $\hat{\boldsymbol{Q}}\hat{\boldsymbol{R}}$. As discussed in Section 4.2, the update of the reduced within-class matrix $\hat{\boldsymbol{W}}$ follows Proposition 3, in which the operation counts for computing $\bar{\boldsymbol{W}}$ by (4.6) and (4.7) are $6\eta k'^2+k'^3+dk'+6\eta k'+k'^2$ flam, and those for computing $\widetilde{\boldsymbol{W}}$ and $\boldsymbol{E}$ are $d'k' + k'^2$ and k'^3 flam. Hereby for updating $\hat{\boldsymbol{W}}$, the total operation counts are $2k'^3 + 2dk' + 6\eta k'^2 + 2k'^2 + 6\eta k'$ flam, namely $O(k'^3 + dk' + k'^2)$. For updating the reduced between-class scatter matrix $\boldsymbol{B}$, we can calculate the time complexity similarly to sIncLDA as $O(k'^3 + k'^2)$ by (4.8).

The time complexity discussed above is summarized in Table 4.1. As seen in Table 4.1, for incremental learning on a single sample, the general complexity of cIncLDA $O(dk'^2 + k'^3 + dk')$ is higher than the sIncLDA $O(k^3 + dk)$ since $d \gg k$ and $k' \geq k$.

However, for incremental learning on a chunk of s samples, sIncLDA can only process the data iteratively in s cycles. Thus, it gives $O(s(k^3 + dk))$. In contrast, the proposed cIncLDA processes whole chunk s samples at one time, and the actual time cost by Proposition 3 is determined by η, the rank of $\boldsymbol{\Delta}$. It follows that s samples are compressed informatively into η samples in the proposed cIncLDA. Thus, the time complexity for cIncLDA is $O(\eta(dk'^2 + k'^3 + dk'))$.

Matching the running efficiency of the cIncLDA with sIncLDA, $s(k^3 + dk) = \eta(dk'^2 + k'^3 + dk')$, since $k \leq k'$, we have the minimal chunk size s as,

$$s = \frac{\eta(dk'^2 + k'^3 + dk')}{k^3 + dk} \geq \frac{\eta k'^2 d}{k^3 + dk} + \eta. \tag{4.9}$$

Here, s is greater than η as proved below, and it follows that the cIncLDA matches

the sIncLDA efficiency when the chunk size $s > \eta$. Because s often grows much faster than the η for cIncLDA, the general efficiency of cIncLDA (in terms of the total time cost to complete incremental learning of a whole dataset) is expected to surpass the sIncLDA with the increase of the chunk size s.

Proof: Consider $k \ll d$, without loss of generality, we enlarge the denominator of (4.9) by replacing the k^3 item with dk^2, thus we have

$$s \geq \frac{\eta k'^2 d}{dk^2 + dk} + \eta = \frac{\eta k'^2}{k(k+1)} + \eta > \eta \tag{4.10}$$

Chapter 5

Experiment Evaluation

In this chapter, we examine the accuracy and efficiency of the proposed cIncLDA by comparing it with sIncLDA and batch QR/LDA. We have specifically examined its equality to the ground truth eigenspace, its class separability of the embedding eigenspace, and its execution time. All the experiments were conducted on a Intel 2.26GHZ Core I5 PC with 4GB Ram.

5.1 Data Description

As we have discussed previously, the performance of sIncLDA may suffer when data with a large number of classes are presented. Thus, we need to choose datasets with diverse class numbers to examine the performance of both IncLDAs. In addition, to our best, we can find these five benchmarked datasets, which are widely used in other related work (T.-K. Kim, Kim, Hwang & Kittler, 2005; T.-K. Kim et al., 2011; Pang, Kim & Bang, 2003; M.-S. Kim, Kim & Lee, 2003; Pang et al., 2004). Therefore, the following five benchmarked facial recognition datasets and one combined face dataset have been selected for the following performance and efficiency evaluation experiments. Those have different class numbers from 40 to 1010.

1. The AT&T (ORL) face dataset [1] consists of 400 face images from 40 persons (10 images per person). Each image has the size of 92×112. The images were taken at the same background but at different times, with varied facial expressions (e.g., open/closed eyes and smiling/non-smiling) or facial details (e.g., glasses/no-glasses).

[1] http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html.

2. The Altkom dataset [2] (T.-K. Kim et al., 2005, 2011) is composed of 1200 face images from 80 persons (15 images per person) with different poses. The size of image is normalized to 46×56.

3. The MPEG-7 dataset has 1355 face images from 271 persons (five images per person) (Pang et al., 2003; M.-S. Kim et al., 2003; Pang et al., 2004). Each image has the size of 23×28 pixels. The dataset is collected from AR(Purdue), AT&T, Yale, UMIST, University of Berne, and some of them were obtained from MPEG-7 news Videos.

4. The XM2VTS dataset [3] (T.-K. Kim et al., 2005, 2011) contains 2950 face images from five recordings of 295 persons in which each recording has two head shots taken at two different time (i.e., 10 images per person). Each image has the same size of 46×56.

5. The extended version 1 MPEG dataset [4] (T.-K. Kim et al., 2005, 2011) is a combination of several public face sets (e.g., AR and ORL). It collects 3175 images from 635 persons (five images per person) showing illumination and view variations. The size of image is the same, 46×56.

6. The combined face dataset includes 5050 images from 1010 persons (five images per person), in which we selected 400 images from the Altkom dataset (80 persons), 1475 images from XM2VTS dataset (295 persons), and 3175 images from the version 1 MPEG dataset (635 persons). The size of image is kept the same, 46×56.

5.2 Experimental setup

To conduct incremental learning on each dataset, we first construct an initial discriminant eigenspace using 10% of the face images, in which at least two classes of data are guaranteed to be included according to the definition of classic LDA. The remaining 90% training instances are divided equally into nine chunks/subsets and presented to the learner sequentially. As a reference, we also apply batch QR/LDA

[2] http://www.iis.ee.ic.ac.uk/∼tkkim/code.htm.
[3] http://www.iis.ee.ic.ac.uk/∼tkkim/code.htm.
[4] http://www.iis.ee.ic.ac.uk/∼tkkim/code.htm.

to perform the same incremental learning task. As the incremental learning proceeds, the learner is provided with chunks of new data in a sequential manner until all subsets are exhausted.

5.3 Results Evaluation

5.3.1 Similarity to Ground Truth Eigenspace

To examine the similarity between the discriminant eigenspace from the cIncLDA and that from the ground truth QR/LDA, we set the chunk size s as 1, and use the cIncLDA for sequential incremental learning. We experiment cIncLDA, sIncLDA and the batch LDA on the UCI Iris dataset (Frank & Asuncion, 2010) which has three classes and 150 samples in a 4-dimension space. We calculate the inner products between the discriminant eigenvectors from the cIncLDA and the counterpart of the batch QR/LDA at every learning stage, and record the converging procedure of the cIncLDA and sIncLDA, respectively.

As a result, Fig. 5.1 reveals the difference between two IncLDAs on their converging procedure to batch QR/LDA. As seen, both IncLDAs converge to the Batch QR/LDA as the incremental learning proceeds, demonstrating the cIncLDA equivalent to the ground truth on Iris data. The proposed cIncLDA has not only fewer but also far smaller fluctuations than the sIncLDA on all three axes. Specifically on the $d1$ axis, the cIncLDA is found converging to the ground truth over 100 stages earlier than the sIncLDA; and on the other two axes performing similarly to the sIncLDA. This suggests that with reference to the ground truth, the proposed cIncLDA has deterministically less discriminative information lost for incremental QR/LDA learning, and thus is expected to have better stage learning performance (in terms of stage class separability evaluation) than the sIncLDA even if it is being used just in sequential learning mode.

5.3.2 Class Separability

To evaluate the class separability of the proposed cIncLDA, we project the data presented so far to the current LDA eigenspace and then classify the data using a k-Nearest Neighbor (kNN) classifier. The dimensions of the embedding spaces of both reported methods are set to the number of classes, and the classification accuracy is

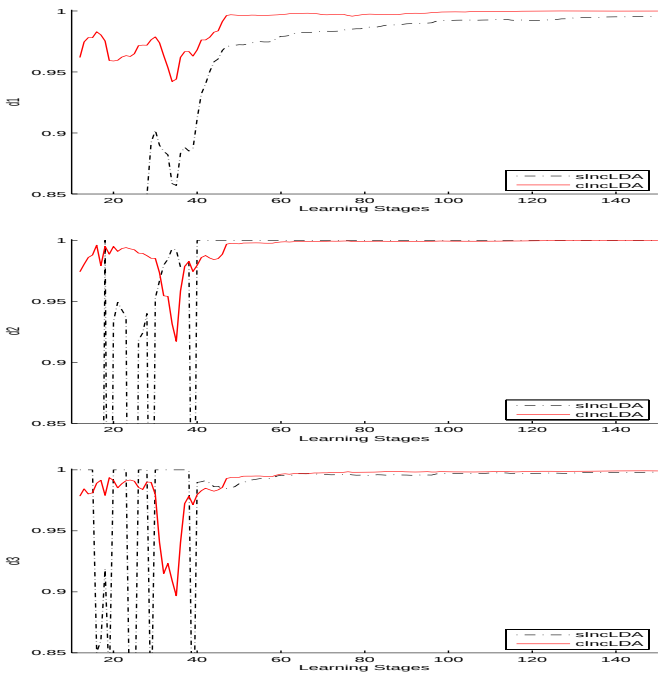
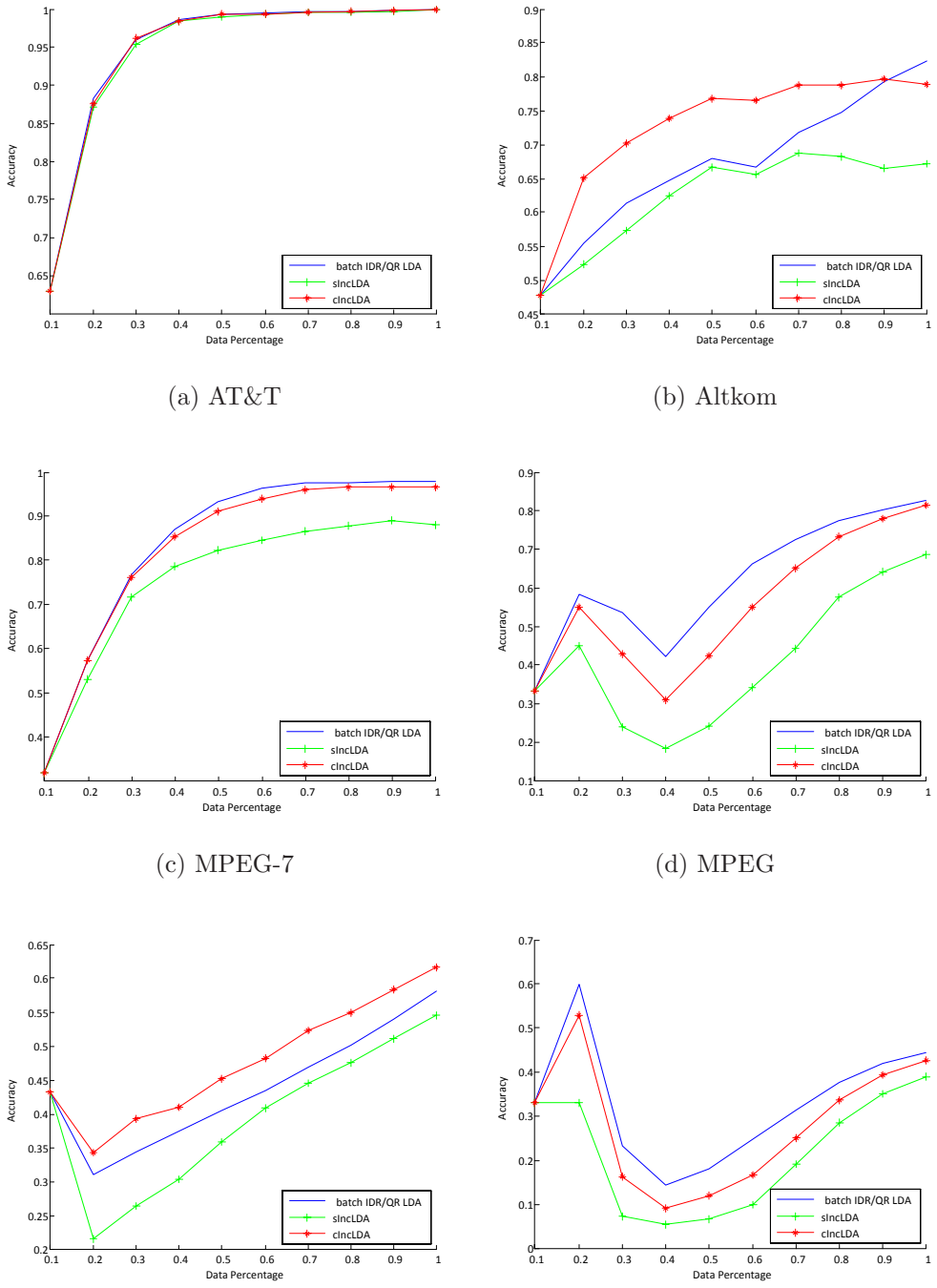


Figure 5.1: inner products of eigenvectors, cIncLDA vs. sIncLDA on their converging procedure to batch QR/LDA

measured by leave-one-out (LOO) cross-validation.

Fig 5.2 shows the experimental results, where the proposed cIncLDA is compared to sIncLDA and batch QR/LDA, at every incremental learning stage, which is identified as the percentage of data presented so far. As seen in the figure, the cIncLDA resembles the trend of sIncLDA for all six datasets; however the performance of the cIncLDA apparently approaches the ground truth batch QR/LDA better than the sIncLDA. Such superiority arises at the beginning stage and amplifies quickly in a few cycles of learning. For dataset Altkom and XM2V., it grows to even outperform the batch QR/LDA. Most importantly, it turns to be stable at later stages to the end of incremental learning.

We further measure the statistical difference from the proposed cIncLDA to the exisiting sIncLDA on class separability by calculating the average stage classification accuracy difference (i.e., stage Acc. diff.) and the final stage classification accuracy difference (i.e., final Acc. diff.). Table 5.1 gives the comparison for all six case studies, in which the quantitative properties of each dataset are detailed, and the class separability differences are calculated as the sIncLDA minus cIncLDA on classification rate.



(a) AT&T (b) Altkom

(c) MPEG-7 (d) MPEG

(e) XM2V (f) Combined

Figure 5.2: The comparison of class separability between the proposed cIncLDA and sIncLDA, with reference to the batch QR/LDA

As seen from the table, both type differences are positive for all six datasets. This indicates that the proposed cIncLDA retains more discriminant information than the sIncLDAs at every incremental learning stage. The stage difference is seen to be smaller than the final difference for most cases except the At&T dataset. This suggests that the superiority of the cIncLDA to sIncLDA grows at every incremental learning stage, ensuring a reliable and significant superiority on the learning of the whole data.

For the AT&T dataset with a small number of classes (k=40), only subtle differences are exhibited between two IncLDAs, whereas for datasets with larger number of classes, such as the MPEG and XM2VTS data, the differences are more evident. However, the superiority is not determined by merely the class number, but also the number of samples and the dimensionality, despite the class number playing an important role.

Dataset	# of samples	# of dimensions	# of classes	stage Acc. diff.	final Acc. diff.
AT&T(ORL)	400	92×112 = 10304	40	+0.0018	+0.0005
Altkom	1220	46×56 = 2576	80	+0.1037	+0.1172
MPEG-7	1355	23×28 = 644	271	+0.1851	+0.2590
XM2VTS	2950	46×56 = 2576	295	+0.0825	+0.0707
MPEG	3175	46×56 = 2576	635	+0.1439	+0.1272
Combined	5050	46×56 = 2576	1010	+0.0636	+0.0382

Table 5.1: The stage accuracy differences and final accuracy differences of the incremental learning for six datasets, sIncLDA versus cIncLDA

5.3.3 Computational Efficiency

In this experiment, cIncLDA is compared with sIncLDA on CPU time cost. To observe the relationship between the execution time and chunk size (size of the added subsets), we use different chunk sizes ranging from 5 to 1220 with a step of five in the experiment, and record the differences between cIncLDA and sIncLDA at every incremental learning stage.

Fig. 5.3 (a) gives a comparison of the time cost at each stage between cIncLDA and sIncLDA for chunk sizes at 100, 250, and 350, respectively. It is seen that when the chunk size is 100, the stage learning time cost of cIncLDA is less than that of sIncLDA, and the difference is as large as 20 seconds on average. However at 250, the efficiency of cIncLDA is competitive with sIncLDA, and its efficiency surpasses the sIncLDA entirely, when the chunk size is just 100 bigger.

We further measure the total time cost by cIncLDA to compute a discriminant model for all 1,355 samples under different chunk sizes. To better clarify the comparison between cIncLDA and sIncLDA, we show the sIncLDA time minus the cIncLDA time in Fig. 5.3 (b). As is seen, the efficiency of cIncLDA grows nicely with the increase of the chunk size. When the chunk size reaches 240, cIncLDA matches sIncLDA in general. The superiority of cIncLDA occurs and remarks when the incremental sizes are larger than 290. The difference scales up to over 100 seconds, when the chunk size increases to 1220.

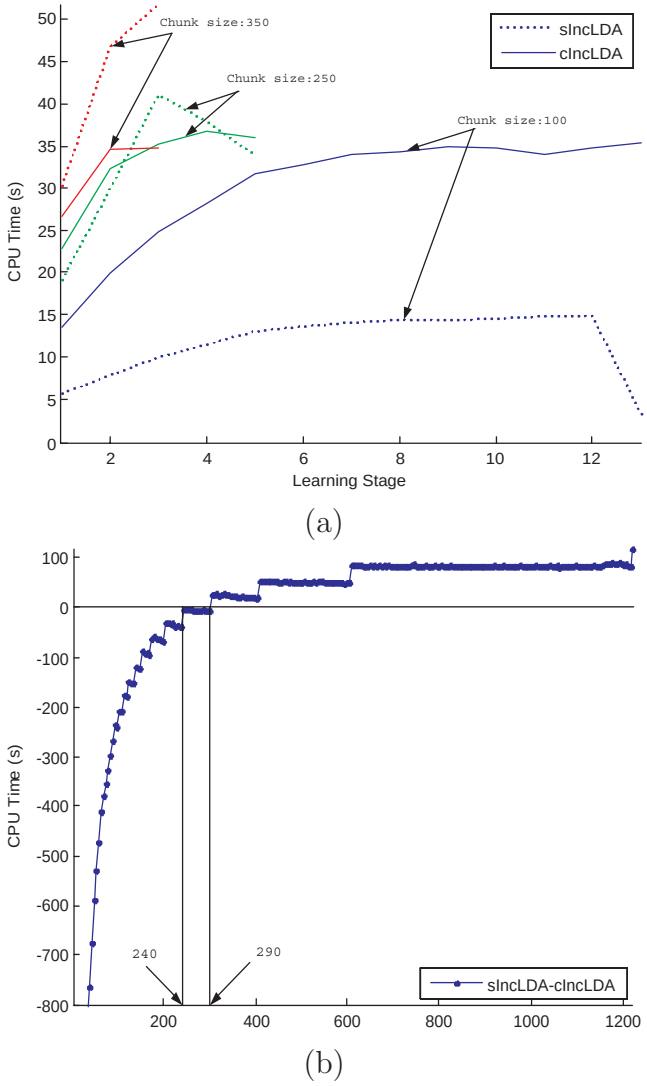


Figure 5.3: Comparison on time cost with different incremental sizes, cIncLDA vs. sIncLDA.

To discover how the running efficiency of the proposed cIncLDA is maximized by a large chunk size, we conduct incremental learning additionally in two settings: the first uses half, and the second uses all the remaining dataset as one chunk data input to IndLDAs. We measure the time costs of each incremental learning for all the above six datasets, respectively. Table 5.2 presents all the time costs from the proposed cIncLDA with a comparison to those of sIncLDA, where the time saving is calculated in both CPU seconds and the percentage to the corresponding sIncLDA cost.

As seen in Table 5.2, the time cost of the sIncLDA learning varies little on chunk size choices. In contrast, when the chunk size is doubled, the proposed cIncLDA reduces the time cost to half for all 6 datasets regardless of their differences on sample size (i.e., n), dimensionality (i.e., d), and the number of classes (i.e., k). Moreover, the proposed cIncLDA saves time against sIncLDA consistently in all cases. The minimum saving is on the MPEG-7 dataset, where the saved 84.46 CPU seconds is 54% of the sIncLDA time cost. The maximum time saving (in terms of percentage to sIncLDA) is on the Altkom dataset, where 473.16 seconds are equivalent to 92.67% of the sIncLDA time cost.

Dataset (#sample/#dimension/#class)	**Chunk Size**	**CPU Time(s)**		**Time saving(s)**
		sIncLDA	**cIncLDA**	
AT&T(ORL)	180	2279.30	473.40	1805.9 / 79.23%
(400/10304/40)	360	2278.20	242.29	2035.9 / 89.37%
Altkom	540	509.50	70.14	439.36 / 86.23%
(1200/2576/80)	1080	510.58	37.42	473.16 / 92.67%
MPEG-7	610	153.68	69.22	84.46 / 54.96%
(1355/644/271)	1220	152.74	35.76	116.99 / 76.59%
XM2VTS	1327	2893.30	462.09	2431.2 / 84.03%
(2950/2576/295)	2655	2893.20	235.16	2658.1 / 91.87%
MPEG	1428	6194.80	1807.10	4387.7 / 70.83%
(3175/2576/635)	2857	6195.60	913.40	5282.2 / 85.26%
Combined	2272	17419.00	4500.80	12918.00 / 74.16%
(5050/2576/1010)	4545	17417.20	2291.70	15125.00 / 86.84%

Table 5.2: The execution time of incremental learning on six datasets, sIncLDA versus cIncLDA, with two different chunk sizes.

5.4 Summary

The experimental result indicates that our algorithm achieves an accuracy level that is competitive to batch QR/LDA and is consistently higher than sIncLDA. The proposed algorithm also produces the optimal transform matrix that is very similar to those obtained by batch QR/LDA. The efficiency of our algorithm can match and surpass sIncLDA as the chunk size increases for multiple instances processing. In view of this, we believe that our proposed cIncLDA is much better than sIncLDA, and will be applied to more real-world tasks.

Chapter 6

Conclusions and Directions for Future Research

This chapter provides a brief retrospect of the content presented in the thesis. We summarize the work reported in each chapter and identify the contributions. In addition, several directions for future research will be highlighted.

6.1 Conclusions

In this thesis, we first reviewed major incremental learning methods including LDAs and IncLDAs. Based on the review, we proposed a new framework for assessing IncLDAs. Further, we intensively analyzed current popular IncLDAs in terms of a component model, and evaluated those IncLDAs based on the framework. In particular, we analyzed existing sequential IDR/QR IncLDA from an embedding point of view, and indicated its limitations, such as inefficiency of processing chunk data, etc.

To resolve the limitations of sIncLDA, we proposed a new Chunk IDR/QR IncLDA algorithm that is capable of processing a whole chunk of data for every incremental learning cycle. Following this, we explicitly compared the time complexity of cIncLDA and sIncLDA,. Our analyisis showed that the computational efficiency of cIncLDA is capable of matching and surpassing that of sIncLDA as the chunk increases.

Experimental results showed that performance (i.e., accuracy) is improved in our solution, especially when processing data with large number of classes. In addition,

compared with sIncLDA, the proposed cIncLDA also produces the transform matrix much similar to the one obtained by batch QR/LDA. Moreover, the experimental study on computational efficiencies agrees well with theoretical time complexity analysis. As evidenced in the thesis, the running efficiency of the proposed cIncLDA can be maximized by large chunks.

The main contributions of this paper are summarized as below:

1. A new evaluation criteria framework for IncLDAs has been proposed. It allows us to assess IncLDAs from a new point of view, not just by comparisons on performance, efficiency and so forth.

2. Two efficient matrix augmentation methods have been proposed. They allow us to easily accommodate information of new classes contained in chunk data through expanding the dimensions of matrices $\boldsymbol{Q}$ and $\boldsymbol{W}$.

3. A new method for incremental updating of $\boldsymbol{W}$, more accurate than the existing approach, has been successfully developed in this thesis. This is achieved by relaxing a key assumption on $\boldsymbol{Q}$ and $\boldsymbol{W}$ that is vital to the existing algorithm but may be significantly violated as the number of classes increases.

4. Unlike Pang's IncLDAs (Pang et al., 2005b) where only the chunk data in one class can be acquired each time, our method is capable of absorbing information obtained from newly added samples in whatever classes.

6.2 Directions for Future Research

As we noted in the thesis, the reduced between-class scatter matrix $\boldsymbol{B}$ in the existing sIncLDA and the proposed cIncLDA are not incrementally updated but completely recomputed. As a matter of fact, whenever new data are presented for the incremental learning procedure, there should exist an incremental learning rule such as $\boldsymbol{B'} = \boldsymbol{B} + \boldsymbol{\Sigma}$. One important future challenge would be to develop such an incremental mechanism for calculating $\boldsymbol{\Sigma}$ and subsequently updating $\boldsymbol{B}$. Once realized, we believe the efficiency of cIncLDA could be improved further. As analyzed in section 4.5, the time complexity for recomputing $\boldsymbol{B}$ is $O(k^3)$. It is straightforward to see that if the reduced between-class scatter matrix can be updated through $\boldsymbol{B'} = \boldsymbol{B} + \boldsymbol{\Sigma}$, which only requires $O(k^2)$, significant reduction of computation cost can be achieved.

Our time complexity analysis also showed that when $s > \eta$, the running efficiency of cIncLDA can match and even exceed that of sIncLDA. This has been mathematically proved. However, it remains as a challenge to find a general formula for computing the exact match point and surpass point.

Finally, in the thesis, only face recognition application has been touched. It is also interesting to investigate other potential applications of our proposed cIncLDA, and design experiments on other real-world datasets to see the generability and stability of our work.

Appendix A

Proof of Proposition 1

Given $\boldsymbol{A} = (\boldsymbol{A}_1, \boldsymbol{A}_2) = \boldsymbol{Q}(\boldsymbol{R}_1, \boldsymbol{R}_2) = \boldsymbol{QR} \in \mathcal{R}^{d\times k}$ with $\boldsymbol{A}_1 \in \mathcal{R}^{d\times(i-1)}$, and $\boldsymbol{A}_2 \in \mathcal{R}^{d\times(k-i)}$. According to Daniel et al. (1976), if a zero vector $\mathbf{0}$ is inserted at the i-th column of $\boldsymbol{A}$, then we have the augmented matrix $\hat{\boldsymbol{A}}$ as,

$$\hat{\boldsymbol{A}} = (\boldsymbol{A}_1, \mathbf{0}, \boldsymbol{A}_2) = (\boldsymbol{Q}, \mathbf{0}) \begin{pmatrix} \boldsymbol{R}_1 & \mathbf{0} & \boldsymbol{R}_2 \\ \mathbf{0}^T & 1 & \mathbf{0}^T \end{pmatrix} = \boldsymbol{\Psi\Xi},$$

where $\boldsymbol{\Psi} = (\boldsymbol{Q}, \mathbf{0})$ and $\boldsymbol{\Xi} = \begin{pmatrix} \boldsymbol{R}_1 & \mathbf{0} & \boldsymbol{R}_2 \\ \mathbf{0}^T & 1 & \mathbf{0}^T \end{pmatrix}$.

For QR-updating, the re-orthogonalization step stated by Daniel et al. (1976) can be avoided, because $\mathbf{0}$ is orthogonal to any vectors. We choose directly the Givens rotations $\boldsymbol{G} = [\boldsymbol{G}_{i,i+1}, \ldots, \boldsymbol{G}_{k-1,k}, \boldsymbol{G}_{k,k+1}]$ so that

$$\boldsymbol{G}^T \cdot \begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ 1 \\ \mathbf{0} \end{pmatrix} \in \mathcal{R}^{k\times 1} \tag{A.1}$$

where 1 positions at the i-th row, and "$\cdot$" denotes the application of individual Givens rotation $\boldsymbol{G}_{i,i+1}$ on $\begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix}$ one by one. As a result, we obtain $\boldsymbol{G}_{i,i+1} = \cdots = \boldsymbol{G}_{k-1,k} = \boldsymbol{G}_{k,k+1} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$.

Consequently, we can apply $\boldsymbol{G}^T$ on $\boldsymbol{\Xi}$ to compute $\hat{\boldsymbol{R}}$ as,

$$\begin{aligned}
\boldsymbol{G}^T \cdot \boldsymbol{\Xi} &= \boldsymbol{G}^T_{i,i+1} \dots \boldsymbol{G}^T_{k,k+1} \begin{pmatrix} \boldsymbol{R}_1 & \boldsymbol{0} & \boldsymbol{R}_2 \\ \boldsymbol{0}^T & 1 & \boldsymbol{0}^T \end{pmatrix} \\
&= \boldsymbol{G}^T_{i,i+1} \dots \boldsymbol{G}^T_{k-1,k} \begin{pmatrix} \boldsymbol{R}^{(1)}_1 & \boldsymbol{0} & \boldsymbol{R}^{(1)}_2 \\ \boldsymbol{0}^T & 1 & \boldsymbol{0}^T \\ \boldsymbol{R}^{(1)}_3 & 0 & -\boldsymbol{R}^{(1)}_4 \end{pmatrix} \\
&\dots \\
&= \boldsymbol{G}^T_{i,i+1} \begin{pmatrix} \boldsymbol{R}^{(k-i)}_1 & \boldsymbol{0} & \boldsymbol{R}^{(k-i)}_2 \\ \boldsymbol{0}^T & 1 & \boldsymbol{0}^T \\ \boldsymbol{R}^{(k-i)}_3 & \boldsymbol{0} & -\boldsymbol{R}^{(k-i)}_4 \end{pmatrix} \\
&= \begin{pmatrix} \boldsymbol{R}^{(k+1-i)}_1 & \boldsymbol{0} & \boldsymbol{R}^{(k+1-i)}_2 \\ \boldsymbol{0}^T & 1 & \boldsymbol{0}^T \\ \boldsymbol{R}^{(k+1-i)}_3 & \boldsymbol{0} & -\boldsymbol{R}^{(k+1-i)}_4 \end{pmatrix} \\
&= \hat{\boldsymbol{R}}.
\end{aligned} \tag{A.2}$$

Here $\boldsymbol{R}^{(k+1-i)}_1 \in \mathcal{R}^{(i-1)\times(i-1)}$, $\boldsymbol{R}^{(k+1-i)}_2 \in \mathcal{R}^{(i-1)\times(k-i)}$, $\boldsymbol{R}^{(k+1-i)}_3 \in \mathcal{R}^{(k-i)\times(i-1)}$, $\boldsymbol{R}^{(k+1-i)}_4 \in \mathcal{R}^{(k-i)\times(k-i)}$, and $\begin{pmatrix} \boldsymbol{R}^{(k+1-i)}_1 & \boldsymbol{R}^{(k+1-i)}_2 \\ \boldsymbol{R}^{(k+1-i)}_3 & \boldsymbol{R}^{(k+1-i)}_4 \end{pmatrix} = \boldsymbol{R}$ with its submatrix $\boldsymbol{R}^{(k+1-i)}_3$ being a zero matrix. Note that (A.2) performs an iterative calculation, because an individual Givens rotation $\boldsymbol{G}_{i,i+1}$ effects only two rows of $\boldsymbol{\Xi}$ (Golub & Van Loan, 1996).

Similarly, we apply $\boldsymbol{G}$ on $\boldsymbol{\Psi}$ to compute $\hat{\boldsymbol{Q}}$ as,

$$\begin{aligned}
\boldsymbol{\Psi} \cdot \boldsymbol{G} &= (\boldsymbol{Q}, \boldsymbol{0}) \boldsymbol{G}_{k,k+1} \dots \boldsymbol{G}_{i,i+1} \\
&= (\boldsymbol{Q}^{(1)}_1, \boldsymbol{0}, -\boldsymbol{Q}^{(1)}_2) \boldsymbol{G}_{k-1,k} \dots \boldsymbol{G}_{i,i+1} \\
&\dots \\
&= (\boldsymbol{Q}^{(k-i)}_1, \boldsymbol{0}, -\boldsymbol{Q}^{(k-i)}_2) \boldsymbol{G}_{i,i+1} \\
&= (\boldsymbol{Q}^{(k+1-i)}_1, \boldsymbol{0}, -\boldsymbol{Q}^{(k+1-i)}_2) \\
&= \hat{\boldsymbol{Q}}.
\end{aligned} \tag{A.3}$$

Here $\boldsymbol{Q}^{(k+1-i)}_1 \in \mathcal{R}^{d\times(i-1)}$,$\boldsymbol{Q}^{(k+1-i)}_2 \in \mathcal{R}^{d\times(k-i)}$, and $(\boldsymbol{Q}^{(k+1-i)}_1, \boldsymbol{Q}^{(k+1-i)}_2) = \boldsymbol{Q}$. Note that, only two columns of $\boldsymbol{\Psi}$ are effected by an individual Givens rotation $\boldsymbol{G}_{i,i+1}$ (Golub & Van Loan, 1996).

Hereby, we have $\hat{\boldsymbol{A}} = \hat{\boldsymbol{Q}}\hat{\boldsymbol{R}}$ as required.

Appendix B

Proof of Proposition 2

Given $\boldsymbol{W} = \boldsymbol{Q}^T \boldsymbol{H}_w \boldsymbol{H}_w^T \boldsymbol{Q} \in \mathcal{R}^{k \times k}$, if a zero vector $\mathbf{0}$ is inserted as the i-th column into $\boldsymbol{Q} \in \mathcal{R}^{d \times k}$, so that $\hat{\boldsymbol{Q}} = (\boldsymbol{Q}_1, \mathbf{0}, -\boldsymbol{Q}_2) \in \mathcal{R}^{(k+1)\times(k+1)}$ where $\boldsymbol{Q}_1 \in \mathcal{R}^{d\times(i-1)}$ and $\boldsymbol{Q}_2 \in \mathcal{R}^{d\times(k-i)}$. According to (A.3), we derive the augmented reduced within-class matrix as follows,

$$
\begin{aligned}
\hat{\boldsymbol{W}} &= \hat{\boldsymbol{Q}}^T \boldsymbol{H}_w \boldsymbol{H}_w^T \hat{\boldsymbol{Q}} \\
&= (\boldsymbol{G}^T \cdot \boldsymbol{\Psi}^T) \boldsymbol{H}_w \boldsymbol{H}_w^T (\boldsymbol{\Psi} \cdot \boldsymbol{G}) \\
&= \boldsymbol{G}^T \cdot (\boldsymbol{Q}, \mathbf{0})^T \boldsymbol{H}_w \boldsymbol{H}_w^T (\boldsymbol{Q}, \mathbf{0}) \cdot \boldsymbol{G} \\
&= \boldsymbol{G}^T \cdot \begin{pmatrix} \boldsymbol{W} & \mathbf{0} \\ \mathbf{0} & 0 \end{pmatrix} \cdot \boldsymbol{G} \\
&= \boldsymbol{G}_{i,i+1}^T \ldots \boldsymbol{G}_{k,k+1}^T \begin{pmatrix} \boldsymbol{W} & \mathbf{0} \\ \mathbf{0} & 0 \end{pmatrix} \boldsymbol{G}_{k,k+1} \ldots \boldsymbol{G}_{i,i+1} \\
&= \boldsymbol{G}_{i,i+1}^T \ldots \boldsymbol{G}_{k-1,k}^T \begin{pmatrix} \boldsymbol{W}_1^{(1,0)} & \mathbf{0} & \mathbf{0} \\ \mathbf{0}^T & 0 & 0 \\ -\boldsymbol{W}_3^{(1,0)} & -\boldsymbol{W}_4^{(1,0)} & \mathbf{0} \end{pmatrix} \boldsymbol{G}_{k,k+1} \ldots \boldsymbol{G}_{i,i+1} \\
&\ldots \\
&= \begin{pmatrix} \boldsymbol{W}_1^{(k+1-i,0)} & \mathbf{0} & \mathbf{0} \\ \mathbf{0}^T & \mathbf{0}^T & 0 \\ -\boldsymbol{W}_3^{(k+1-i,0)} & -\boldsymbol{W}_4^{(k+1-i,0)} & \mathbf{0} \end{pmatrix} \boldsymbol{G}_{k,k+1} \ldots \boldsymbol{G}_{i,i+1} \\
&\ldots \\
&= \begin{pmatrix} \boldsymbol{W}_1^{(k+1-i,k+1-i)} & \mathbf{0} & -\boldsymbol{W}_2^{(k+1-i,k+1-i)} \\ \mathbf{0}^T & 0 & \mathbf{0}^T \\ -\boldsymbol{W}_3^{(k+1-i,k+1-i)} & \mathbf{0} & \boldsymbol{W}_4^{(k+1-i,k+1-i)} \end{pmatrix},
\end{aligned}
$$

where $\boldsymbol{W}_1^{(k+1-i,k+1-i)} \in \mathcal{R}^{(i-1)\times(i-1)}$, $\boldsymbol{W}_2^{(k+1-i,k+1-i)} \in \mathcal{R}^{(i-1)\times(k-i)}$, $\boldsymbol{W}_3^{(k+1-i,k+1-i)} \in \mathcal{R}^{(k-i)\times(i-1)}$, $\boldsymbol{W}_4^{(k+1-i,k+1-i)} \in \mathcal{R}^{(k-i)\times(k-i)}$, and $\begin{pmatrix} \boldsymbol{W}_1^{(k+1-i,k+1-i)} & \boldsymbol{W}_2^{(k+1-i,k+1-i)} \\ \boldsymbol{W}_3^{(k+1-i,k+1-i)} & \boldsymbol{W}_4^{(k+1-i,k+1-i)} \end{pmatrix} = \boldsymbol{W}$.

Appendix C

Proof of Proposition 3

As $\boldsymbol{W} = \boldsymbol{Q^T H_w H_w^T Q}$ (Ye et al., 2005), its update for incremental learning is calculated as,

$$\begin{aligned} \boldsymbol{W'} &= \boldsymbol{Q'^T H'_w H'^T_w Q'} \\ &= \boldsymbol{Q'}^T \textstyle\sum_{i=1}^{k'} (\boldsymbol{H'}_{w_i} \boldsymbol{H'}^T_{w_i}) \boldsymbol{Q'} \ . \end{aligned} \tag{C.1}$$

As we know, $\boldsymbol{H'_w} = [\boldsymbol{X'}_1 - \boldsymbol{m'}_1 \boldsymbol{e'}_1^T, ..., \boldsymbol{X'}_{k'} - \boldsymbol{m'}_{k'} \boldsymbol{e'}_{k'}^T] \in \mathcal{R}^{d \times k'}$ and $\boldsymbol{e'}_i = \begin{bmatrix} \boldsymbol{e}_i \\ \widetilde{\boldsymbol{e}}_i \end{bmatrix} = (1, \ldots, 1)^T \in \mathcal{R}^{n'_i}$, $\boldsymbol{H'}_{w_i}$ in (C.1) gives,

$$\begin{aligned} \boldsymbol{H'}_{w_i} &= \boldsymbol{X'_i} - \boldsymbol{m'}_i \boldsymbol{e'}_i^T \\ &= [\boldsymbol{X_i}, \widetilde{\boldsymbol{X}_i}] - \boldsymbol{m'}_i \boldsymbol{e'}_i^T \\ &= [\boldsymbol{X_i} - \boldsymbol{m_i e_i^T}, \widetilde{\boldsymbol{X}_i} - \widetilde{\boldsymbol{m}_i} \widetilde{\boldsymbol{e}_i^T}] - \boldsymbol{m'}_i \boldsymbol{e'}_i^T + \boldsymbol{m}_i \boldsymbol{e'}_i^T \\ &= [\boldsymbol{H}_{w_i}, \widetilde{\boldsymbol{X}_i} - \widetilde{\boldsymbol{m}_i} \widetilde{\boldsymbol{e}_i^T} + \widetilde{\boldsymbol{m}_i} \widetilde{\boldsymbol{e}_i^T} - \boldsymbol{m_i} \widetilde{\boldsymbol{e}_i^T}] - (\boldsymbol{m'_i} - \boldsymbol{m_i}) \boldsymbol{e'}_i^T \\ &= [\boldsymbol{H}_{w_i}, \widetilde{\boldsymbol{H}}_{w_i} + (\widetilde{\boldsymbol{m}_i} - \boldsymbol{m}_i) \widetilde{\boldsymbol{e}}_i^T] - (\boldsymbol{m'_i} - \boldsymbol{m_i}) \boldsymbol{e'}_i^T \\ &= [\boldsymbol{H}_{w_i}, \widetilde{\boldsymbol{H}}_{w_i} + \boldsymbol{\alpha} \widetilde{\boldsymbol{e}}_i^T] - \boldsymbol{\beta} \boldsymbol{e'}_i^T, \end{aligned} \tag{C.2}$$

where $\boldsymbol{\alpha} = \widetilde{\boldsymbol{m}_i} - \boldsymbol{m_i}$ and $\boldsymbol{\beta} = \boldsymbol{m'_i} - \boldsymbol{m_i}$. Hereby, $\boldsymbol{H'_{w_i}} \boldsymbol{H'_{w_i}}^T$ in (C.1) is computed as

$$
\begin{aligned}
\boldsymbol{H'_{w_i}}\boldsymbol{H'^{T}_{w_i}} &= ([\boldsymbol{H}_{w_i}, \widetilde{\boldsymbol{H}}_{w_i} + \boldsymbol{\alpha}\widetilde{\boldsymbol{e}}_i^T] - \boldsymbol{\beta}\boldsymbol{e'}_i^T)([\boldsymbol{H}_{w_i}, \widetilde{\boldsymbol{H}}_{w_i} + \boldsymbol{\alpha}\widetilde{\boldsymbol{e}}_i^T] - \boldsymbol{\beta}\boldsymbol{e'}_i^T)^T \\
&= ([\boldsymbol{H}_{w_i}, \widetilde{\boldsymbol{H}}_{w_i} + \boldsymbol{\alpha}\widetilde{\boldsymbol{e}}_i^T] - \boldsymbol{\beta}\boldsymbol{e'}_i^T)(\begin{bmatrix} \boldsymbol{H}_{w_i}^T \\ (\widetilde{\boldsymbol{H}}_{w_i} + \boldsymbol{\alpha}\widetilde{\boldsymbol{e}}_i^T)^T \end{bmatrix} - \boldsymbol{e'}_i\boldsymbol{\beta}^T) \\
&= [\boldsymbol{H}_{w_i}, \widetilde{\boldsymbol{H}}_{w_i} + \boldsymbol{\alpha}\widetilde{\boldsymbol{e}}_i^T] \begin{bmatrix} \boldsymbol{H}_{w_i}^T \\ (\widetilde{\boldsymbol{H}}_{w_i} + \boldsymbol{\alpha}\widetilde{\boldsymbol{e}}_i^T)^T \end{bmatrix} - \boldsymbol{\beta}\boldsymbol{e'}_i^T \begin{bmatrix} \boldsymbol{H}_{w_i}^T \\ (\widetilde{\boldsymbol{H}}_{w_i} + \boldsymbol{\alpha}\widetilde{\boldsymbol{e}}_i^T)^T \end{bmatrix} \\
&-[\boldsymbol{H}_{w_i}, \widetilde{\boldsymbol{H}}_{w_i} + \boldsymbol{\alpha}\widetilde{\boldsymbol{e}}_i^T]\boldsymbol{e'}_i\boldsymbol{\beta}^T + \boldsymbol{\beta}\boldsymbol{e'}_i^T\boldsymbol{e'}_i\boldsymbol{\beta}^T \\
&= \boldsymbol{H}_{w_i}\boldsymbol{H}_{w_i}^T + (\widetilde{\boldsymbol{H}}_{w_i} + \boldsymbol{\alpha}\widetilde{\boldsymbol{e}}_i^T)(\widetilde{\boldsymbol{H}}_{w_i} + \boldsymbol{\alpha}\widetilde{\boldsymbol{e}}_i^T)^T - \boldsymbol{\beta}[\boldsymbol{e}_i^T, \widetilde{\boldsymbol{e}}_i^T] \begin{bmatrix} \boldsymbol{H}_{w_i}^T \\ (\widetilde{\boldsymbol{H}}_{w_i} + \boldsymbol{\alpha}^T\widetilde{\boldsymbol{e}}_i)^T \end{bmatrix} \\
&-[\boldsymbol{H}_{w_i}, \widetilde{\boldsymbol{H}}_{w_i} + \boldsymbol{\alpha}\widetilde{\boldsymbol{e}}_i^T] \begin{bmatrix} \boldsymbol{e}_i \\ \widetilde{\boldsymbol{e}}_i \end{bmatrix} \boldsymbol{\beta}^T + n'_i\boldsymbol{\beta}\boldsymbol{\beta}^T \\
&= \boldsymbol{H}_{w_i}\boldsymbol{H}_{w_i}^T + (\widetilde{\boldsymbol{H}}_{w_i} + \boldsymbol{\alpha}\widetilde{\boldsymbol{e}}_i^T)(\widetilde{\boldsymbol{H}}_{w_i}^T + \widetilde{\boldsymbol{e}}_i\boldsymbol{\alpha}^T) - \boldsymbol{\beta}(\boldsymbol{e}_i^T\boldsymbol{H}_{w_i}^T + \widetilde{\boldsymbol{e}}_i^T(\widetilde{\boldsymbol{H}}_{w_i} + \boldsymbol{\alpha}\widetilde{\boldsymbol{e}}_i^T)^T) \\
&-(\boldsymbol{H}_{w_i}\boldsymbol{e}_i + (\widetilde{\boldsymbol{H}}_{w_i} + \boldsymbol{\alpha}\widetilde{\boldsymbol{e}}_i^T)\widetilde{\boldsymbol{e}}_i)\boldsymbol{\beta}^T + n'_i\boldsymbol{\beta}\boldsymbol{\beta}^T \\
&= \boldsymbol{H}_{w_i}\boldsymbol{H}_{w_i}^T + \widetilde{\boldsymbol{H}}_{w_i}\widetilde{\boldsymbol{H}}_{w_i}^T + \boldsymbol{\alpha}\widetilde{\boldsymbol{e}}_i\widetilde{\boldsymbol{H}}_{w_i}^T + \widetilde{n}_i\boldsymbol{\alpha}\boldsymbol{\alpha}^T + \widetilde{\boldsymbol{H}}_{w_i}\widetilde{\boldsymbol{e}}_i\boldsymbol{\alpha}^T \\
&-\boldsymbol{\beta}(\boldsymbol{e}_i^T\boldsymbol{H}_{w_i}^T + \widetilde{\boldsymbol{e}}_i^T\widetilde{\boldsymbol{H}}_{w_i}^T + \widetilde{\boldsymbol{e}}_i^T\widetilde{\boldsymbol{e}}_i\boldsymbol{\alpha}^T) - (\boldsymbol{H}_{w_i}\boldsymbol{e}_i + \widetilde{\boldsymbol{H}}_{w_i}\widetilde{\boldsymbol{e}}_i + \boldsymbol{\alpha}\widetilde{\boldsymbol{e}}_i^T\widetilde{\boldsymbol{e}}_i)\boldsymbol{\beta}^T + n'_i\boldsymbol{\beta}\boldsymbol{\beta}^T \\
&= \boldsymbol{H}_{w_i}\boldsymbol{H}_{w_i}^T + \widetilde{\boldsymbol{H}}_{w_i}\widetilde{\boldsymbol{H}}_{w_i}^T + \widetilde{n}_i(\boldsymbol{\alpha} - \boldsymbol{\beta})(\boldsymbol{\alpha} - \boldsymbol{\beta})^T + n_i\boldsymbol{\beta}\boldsymbol{\beta}^T,
\end{aligned} \tag{C.3}
$$

since $\boldsymbol{H}_{w_i}\boldsymbol{e}_i = \widetilde{\boldsymbol{H}}_{w_i}\widetilde{\boldsymbol{e}}_i = 0$, $\widetilde{\boldsymbol{e}}_i^T\widetilde{\boldsymbol{e}}_i = \widetilde{n}_i$ and $\boldsymbol{e'}_i^T\boldsymbol{e'}_i = n'_i$. Substituting $\boldsymbol{H'_{w_i}}\boldsymbol{H'_{w_i}}^T$ in $\boldsymbol{H'_w}\boldsymbol{H'_w}^T = \sum_{i=1}^{k'}\boldsymbol{H'_{w_i}}\boldsymbol{H'_{w_i}}^T$, we have

$$
\boldsymbol{H'_w}\boldsymbol{H'_w}^T = \boldsymbol{H_w}\boldsymbol{H_w}^T + \widetilde{\boldsymbol{H}}_{\boldsymbol{w}}\widetilde{\boldsymbol{H}}_{\boldsymbol{w}}^T + \sum_{i=1}^{k'}(\widetilde{n}_i(\boldsymbol{\alpha} - \boldsymbol{\beta})(\boldsymbol{\alpha} - \boldsymbol{\beta})^T + n_i\boldsymbol{\beta}\boldsymbol{\beta}^T). \tag{C.4}
$$

Further substituting (C.4) into (C.1), we obtain the update of reduced within-class scatter matrix as,

$$
\begin{aligned}
\boldsymbol{W'} &= \boldsymbol{Q'}^T\boldsymbol{H'}_w\boldsymbol{H'}_w^T\boldsymbol{Q'} \\
&= \boldsymbol{Q'}^T\boldsymbol{H_w}\boldsymbol{H_w}^T\boldsymbol{Q'} + \boldsymbol{Q'}^T\widetilde{\boldsymbol{H}}_w\widetilde{\boldsymbol{H}}_w^T\boldsymbol{Q'} \\
&+\boldsymbol{Q'}^T(\textstyle\sum_{i=1}^{k'}(\widetilde{n}_i(\boldsymbol{\alpha} - \boldsymbol{\beta})(\boldsymbol{\alpha} - \boldsymbol{\beta})^T + n_i\boldsymbol{\beta}\boldsymbol{\beta}^T))\boldsymbol{Q'} \\
&= \bar{\boldsymbol{W}} + \widetilde{\boldsymbol{W}} + (\textstyle\sum_{i=1}^{k'}(\widetilde{n}_i(\boldsymbol{\alpha'} - \boldsymbol{\beta'})(\boldsymbol{\alpha'} - \boldsymbol{\beta'})^T + n_i\boldsymbol{\beta'}\boldsymbol{\beta'}^T)) \\
&= \bar{\boldsymbol{W}} + \widetilde{\boldsymbol{W}} + \boldsymbol{E},
\end{aligned} \tag{C.5}
$$

where $\bar{\boldsymbol{W}} = \boldsymbol{Q'}^T\boldsymbol{H_w}\boldsymbol{H_w}^T\boldsymbol{Q'}$, $\widetilde{\boldsymbol{W}} = \boldsymbol{Q'}^T\widetilde{\boldsymbol{H}}_w\widetilde{\boldsymbol{H}}_w^T\boldsymbol{Q'}$, $\boldsymbol{E} = \sum_{i=1}^{k'}(\widetilde{n}_i(\boldsymbol{\alpha'} - \boldsymbol{\beta'})(\boldsymbol{\alpha'} - \boldsymbol{\beta'})^T + n_i\boldsymbol{\beta'}\boldsymbol{\beta'}^T)$, $\boldsymbol{\alpha'} = \boldsymbol{Q'}^T\boldsymbol{\alpha}$ and $\boldsymbol{\beta'} = \boldsymbol{Q'}^T\boldsymbol{\beta}$.

References

Ataa Allah, F., Grosky, W. & Aboutajdine, D. (2007). On-line single-pass clustering based on diffusion maps. In Z. Kedad, N. Lammari, E. Mtais, F. Meziane & Y. Rezgui (Eds.), *Natural language processing and information systems* (Vol. 4592, p. 107-118). Springer Berlin / Heidelberg.

Baldi, P. & Hatfield, G. (2002). *Dna microarrays and gene expression: from experiments to data analysis and modeling*. Cambridge University Press.

Belhumeur, P. N., Hespanha, J. P. & Kriegman, D. J. (1997). Eigenfaces vs. fisherfaces: recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *19*(7), 711-720.

Bishop, C. (2006). *Pattern recognition and machine learning*. Springer.

Cai, D., He, X. & Han, J. (2008). Srda: An efficient algorithm for large-scale discriminant analysis. *IEEE Transactions on Knowledge and Data Engineering*, *20*, 1-12.

Cauwenberghs, G. & Poggio, T. (2000). *Incremental and decremental support vector machine learning*.

Chan, C.-H., Kittler, J. & Messer, K. (2007). Multi-scale local binary pattern histograms for face recognition. In S.-W. Lee & S. Li (Eds.), *Advances in biometrics* (Vol. 4642, p. 809-818). Springer Berlin Heidelberg.

Chen, L.-F., Liao, H.-Y. M., Ko, M.-T., Lin, J.-C. & Yu, G.-J. (2000). A new lda-based face recognition system which can solve the small sample size problem. *Pattern Recognition*, *33*(10), 1713-1726.

Chu, D. & Thye, G. S. (2010). A new and fast implementation for null space based linear discriminant analysis. *Pattern Recognition*, *43*(4), 1373-1379.

Daniel, J. W., Gragg, W. B., Kaufman, L. & Stewart, G. W. (1976). Reorthogonalization and stable algorithms for updating the gram-schmidt qr factorization. *Mathematics of Computation*, *30*(136), 772-795.

Duda, R., Hart, P. & Stork, D. (2001). *Pattern classification*. Wiley.

Dudoit, S., Fridlyand, J. & Speed, T. P. (2002). Comparison of discrimination methods for the classification of tumors using gene expression data. *Journal of the American Statistical Association*, *97*(457), 77-87.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Human Genetics*, *7*(2), 179-188.

Frakes, W. & Baeza-Yates, R. (1992). *Information retrieval: data structures &*

algorithms. Prentice Hall.

Frank, A. & Asuncion, A. (2010). *UCI machine learning repository.* Available from `http://archive.ics.uci.edu/ml`

Friedman, J. H. (1989). Regularized discriminant analysis. *Journal of the American Statistical Association*, *84*(405), 165-175.

Fukunaga, K. (1990). *Introduction to statistical pattern recognition*. Academic Press.

Giraud-Carrier, C. (2000). A note on the utility of incremental learning. *AI Communications*, *13*(4), 215-223.

Golub, G. H. & Van Loan, C. F. (1996). *Matrix computations* (third ed.). Baltimore, M.D.: Johns Hopkins University Press.

G.W., S. (1998). Baisc decompositions. In *Matrix algorithms* (Vol. I). SIAM.

Hall, P., Marshall, D. & Martin, R. (2000). Merging and splitting eigenspace models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*(9), 1042-1049.

Hao, Z., Yu, S., Yang, X., Zhao, F., Hu, R. & Liang, Y. (2004). Online ls-svm learning for classification problems based on incremental chunk. In F.-L. Yin, J. Wang & C. Guo (Eds.), *Advances in neural networks c isnn 2004* (Vol. 3173, p. 558-564). Springer Berlin / Heidelberg.

Hong, Z.-Q. & Yang, J.-Y. (1991). Optimal discriminant plane for a small number of samples and design method of classifier on the plane. *Pattern Recognition*, *24*(4), 317-324.

Howland, P., Jeon, M. & Park, H. (2003). Structure preserving dimension reduction for clustered text data based on the generalized singular value decomposition. *SIAM Journal on Matrix Analysis and Applications*, *25*(1), 165-179.

Huang, R., Liu, Q., Lu, H. & Ma, S. (2002). *Solving the small sample size problem of lda* (Vol. 3).

Jain, A. K. & Dubes, R. C. (1988). *Algorithms for clustering data*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc.

Jiang, J., Song, C., Wu, C., Maurizio, M. & Liang, Y. (2006). Support vector machine regression algorithm based on chunking incremental learning. In V. Alexandrov, G. van Albada, P. Sloot & J. Dongarra (Eds.), *Computational science c iccs 2006* (Vol. 3991, p. 547-554). Springer Berlin / Heidelberg.

Jin, Z., Yang, J., Hu, Z. & Lou, Z. (2001). Face recognition based on the uncorrelated discriminant transformation. *Pattern Recognition*, *34*(7), 1405-1416.

Karasuyama, M. & Takeuchi, I. (2010). Multiple incremental decremental learning of support vector machines. *Trans. Neur. Netw.*, *21*(7), 1048-1059.

Kidera, T., Ozawa, S. & Abe, S. (2006). *An incremental learning algorithm of ensemble classifier systems.*

Kim, M.-S., Kim, D. & Lee, S.-Y. (2003). Face recognition using the embedded hmm with second-order block-specific observations. *Pattern Recognition*, *36*(11), 2723-2735.

Kim, T.-K., Kim, H., Hwang, W. & Kittler, J. (2005). Component-based lda face description for image retrieval and mpeg-7 standardisation. *Image Vision Comput.*, *23*(7), 631-642.

Kim, T.-K., Stenger, B., Kittler, J. & Cipolla, R. (2011). Incremental linear discriminant analysis using sufficient spanning sets and its applications. *International Journal of Computer Vision*, *91*(2), 216-232.

Kowalski, G. (1997). *Information retrieval systems: theory and implementation.* Kluwer Academic Publishers.

Liu, C. & Wechsler, H. (2002). Gabor feature based classification using the enhanced fisher linear discriminant model for face recognition. *IEEE Transactions on Image Processing*, *11*(4), 467-476.

Liu, L., Jiang, Y. & Zhou, Z. (2009). *Least square incremental linear discriminant analysis* (Vol. 0).

Loos, H. G. (1989, 18-22 Jun 1989). *Parity madeline: a neural net with complete boolean repertoire capable of one-pass learning.*

Lu, J., Plataniotis, K. N. & Venetsanopoulos, A. N. (2003a). Face recognition using kernel direct discriminant analysis algorithms. *IEEE Transactions on Neural Networks*, *14*(1), 117-126.

Lu, J., Plataniotis, K. N. & Venetsanopoulos, A. N. (2003b). Face recognition using lda-based algorithms. *IEEE Transactions on Neural Networks*, *14*(1), 195-200.

Lu, J., Plataniotis, K. N. & Venetsanopoulos, A. N. (2005). Regularization studies of linear discriminant analysis in small sample size scenarios with application to face recognition. *Pattern Recognition Letters*, *26*(2), 181-191.

McLachlan, G. (2004). *Discriminant analysis and statistical pattern recognition.* Wiley-Interscience.

Ozawa, S., Pang, S. & Kasabov, N. (2006). *An incremental principal component analysis for chunk data.*

Paige, C. C. & Saunders, M. A. (1981). Towards a generalized singular value decomposition. *SIAM Journal on Numerical Analysis*, *18*(3), 398-405.

Pang, S., Ban, T., Kadobayashi, Y. & Kasabov, N. (2010, 18-23 July 2010). *Incremental and decremental lda learning with applications.*

Pang, S., Kim, D. & Bang, S. Y. (2003). Membership authentication in the dynamic group by face classification using svm ensemble. *Pattern Recognition Letters*, *24*(1-3), 215-225.

Pang, S., Ozawa, S. & Kasabov, N. (2004). One-pass incremental membership authentication by face classification. In D. Zhang & A. Jain (Eds.), *Biometric authentication* (Vol. 3072, p. 1-44). Springer Berlin / Heidelberg.

Pang, S., Ozawa, S. & Kasabov, N. (2005a). Chunk incremental lda computing on data streams. In J. Wang, X.-F. Liao & Z. Yi (Eds.), *Advances in neural networks* (Vol. 3497, p. 829-831). Springer Berlin / Heidelberg.

Pang, S., Ozawa, S. & Kasabov, N. (2005b). Incremental linear discriminant analysis for classification of data streams. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, *35*(5), 905-914.

Park, H., Jeon, M. & Rosen, J. B. (2003). Lower dimensional representation of text data based on centroids and least squares. *BIT Numerical Mathematics*, *43*(2), 427-448.

Polikar, R., Upda, L., Upda, S. S. & Honavar, V. (2001). Learn++: an incremental learning algorithm for supervised neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, *31*(4), 497-508.

Raudys, S. J. & Jain, A. K. (1990). *Small sample size effects in statistical pattern recognition: recommendations for practitioners and open problems* (Vol. i).

Song, F., Liu, H., Zhang, D. & Yang, J. (2008). A highly scalable incremental facial feature extraction method. *Neurocomputing*, *71*(10-12), 1883-1888.

Swets, D. L. & Weng, J. J. (1996). Using discriminant eigenfeatures for image retrieval. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *18*(8), 831-836.

Tian, Q., Barbero, M., Gu, Z. & Lee, S. (1986). *Image classification by the foley-sammon transform.*

Wang, J.-G., Sung, E. & Yau, W.-Y. (2010). Incremental two-dimensional linear discriminant analysis with applications to face recognition. *Journal of Network*

and Computer Applications, *33*(3), 314-322.

Wang, X., Han, T. X. & Yan, S. (2009). *An hog-lbp human detector with partial occlusion handling.*

Ye, J. (2004). Using uncorrelated discriminant analysis for tissue classification with gene expression data. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, *1*, 181-190.

Ye, J. (2005). Characterization of a family of algorithms for generalized discriminant analysis on undersampled problems. *J. Mach. Learn. Res.*, *6*, 483-502.

Ye, J. (2007). *Least squares linear discriminant analysis.* ACM.

Ye, J., Janardan, R., Park, C. H. & Park, H. (2004). An optimization criterion for generalized discriminant analysis on undersampled problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *26*(8), 982-994.

Ye, J. & Li, Q. (2004). Lda/qr: an efficient and effective dimension reduction algorithm and its theoretical foundation. *Pattern Recognition*, *37*(4), 851-854.

Ye, J., Li, Q., Xiong, H., Park, H., Janardan, R. & Kumar, V. (2005). Idr/qr: An incremental dimension reduction algorithm via qr decomposition. *IEEE Transactions on Knowledge and Data Engineering*, *17*(9), 1208-1222.

Yen, O. & Meesad, P. (1999). *Pattern classification by an incremental learning fuzzy neural network* (Vol. 5).

Yu, H. & Yang, H. (2001). *A direct lda algorithm for high-dimensional data - with application to face recogniion.*

Zha, H. & Simon, H. (1997). *On updating problems in latent semantic indexing* (Tech. Rep.). ((United States) [Lawrence Berkeley National Lab., CA] [Pennsylvania State Univ., University Park, PA . Dept. of Computer Science and Engineering] Other Information: PBD: Nov 1997)

Zhao, H. & Yuen, P. C. (2008). Incremental linear discriminant analysis for face recognition. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, *38*(1), 210-221.

Zhao, W., Chellappa, R. & Phillips, P. (1999). *Subspace linear discriminant analysis for face recognition.* Computer Vision Laboratory, Center for Automation Research, University of Maryland. ((College Park, Md.))