Fast Online Incremental Learning on Mixture Streaming Data

Yi Wang, Xin Fan, Zhongxuan Luo
School of Software
Dalian University of Technology
& Key Laboratory for Ubiquitous Network
and Service Software of Liaoning Province
Dalian, China
{dlutwangyi, xin.fan, zxluo}@dlut.edu.cn

Tianzhu Wang
No. 254, Deta Leisure Town
Jinzhou New District
Dalian, China
wangtz@126.com

Maomao Min
School of Software
Dalian University of Technology
Dalian, China
neilfvhv@gmail.com

Jiebo Luo
Department of Computer Science
University of Rochester
Rochester, NY 14627, USA
jluo@cs.rochester.edu

Abstract

The explosion of streaming data poses challenges to feature learning methods including linear discriminant analysis (LDA). Many existing LDA algorithms are not efficient enough to incrementally update with samples that sequentially arrive in various manners. First, we propose a new fast batch LDA (FLDA/QR) learning algorithm that uses the cluster centers to solve a lower triangular system that is optimized by the Cholesky-factorization. To take advantage of the intrinsically incremental mechanism of the matrix, we further develop an exact incremental algorithm (IFLDA/QR). The Gram-Schmidt process with reorthogonalization in IFLDA/QR significantly saves the space and time expenses compared with the rank-one QR-updating of most existing methods. IFLDA/QR is able to handle streaming data containing 1) new labeled samples in the existing classes, 2) samples of an entirely new (novel) class, and more significantly, 3) a chunk of examples mixed with those in 1) and 2). Both theoretical analysis and numerical experiments have demonstrated much lower space and time costs ($2 \sim 10$ times faster) than the state of the art, with comparable classification accuracy.

Introduction

Streaming data are explosive with the boom of mobile networks, social media, and video cameras in this decade. To analyze this type of data, an efficient and incremental learning strategy is necessary. Moreover, these data are mixed with known or novel class labels, arriving one by one or chunk by chunk. These characteristics greatly challenge the existing learning methods. We aim at developing an extremely fast incremental learning method for mixture streaming data.

Classical linear discriminant analysis (LDA) (Chen et al. 2000) and its recent variants (Sharma and Paliwal 2015; Kong and Ding 2014; Luo, Ding, and Huang 2011) can effectively extract features for classification. Recently, incremental LDA (ILDA) methods have been emerging to address the above challenges arisen from streaming data. Some of these ILDA methods attempt to modify the objective function, while others use new optimization strategies to accelerate the calculation speed. The method of IDR/QR (Ye et al. 2005) uses the QR-decomposition to update the within-class scatter matrix (S_w) and between-class scatter matrix (S_b). The methods of ICLDA (Lu, Zou, and Wang 2012), IDR/new (Lu, Jian, and Wang 2015) and ILDA/QR (Chu et al. 2015) also follow the idea of using QR-decomposition, and employ various techniques to reduce the matrix size for the decomposition. However, their updating schemes, the key to any incremental algorithms, are not space and time efficient for online learning.

Researchers also resort to the approximation of scattering matrices for efficiency improvements. Liu *et al.* (Liu, Jiang, and Zhou 2009) give the least-square solution of LDA (LS-ILDA), but the incrementally update of the pseudo-inverse of the data matrix is still time consuming. Kim *et al.* (Kim et al. 2011) leverage the sufficient spanning set approximation to S_b and the total scatter matrix S_t by updating the principal eigenvectors and eigenvalues of these two matrices. The time complexity significantly decreases, but there exists an evident gap between the accuracies of incremental and batch versions.

In this paper, we devise an online incremental LDA algorithm that requires much ($2 \sim 10$ times) less computation complexity and smaller space than the state-of-the-art for mixture streaming data. The algorithm is illustrated in Fig. 1. Specifically, our contributions are twofold:

1. We take advantage of the QR-decomposition on a lower triangular matrix (Chu et al. 2015), and propose a new

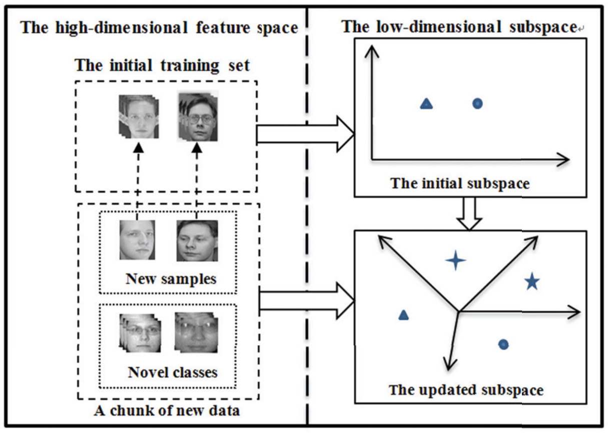

Figure 1: The proposed method incrementally updates the feature space given by new information from a mixture streaming data.

fast batch method, called FLDA/QR. This algorithm takes the centroid of each cluster to constitute the matrix for decomposition, requiring a smaller storage and less computation. Furthermore, we greatly reduce the computation load of the QR-decomposition in FLDA/QR by using the Cholesky-factorization.

2. We develop an *exact* incremental version of FLDA/QR, i.e, IFLDA/QR. It is possible for us to update upon the Gram-Schmidt reorthogonalization process (Daniel et al. 1976) thanks to the well-defined intrinsic incremental mechanism of our FLDA/QR. The updating is simpler and significantly more efficient than the rank-one updating in many other ILDA algorithms based on QR-decomposition.

The new IFLDA/QR algorithm has the flexibility to handle streaming data that may contain 1) new labeled samples to the existing classes, 2) samples from an entirely new (novel) class, and more significantly, 3) a *chunk* of samples mixed with those as 1) and 2). Both theoretical analysis and numerical experiments on several benchmark data sets validate the lowest computational costs of FLDA/QR and IFLDA/QR compared with the state-of-the-art batch LDA and incremental LDA algorithms, respectively. Meanwhile, the classification accuracies of FLDA/QR and IFLDA/QR are close or equal to those of the existing algorithms.

Linear Discriminant Analysis

Classical LDA algorithm attempts to project data examples into a feature space, where those from different classes are separated as much as possible. Let linearly independent data matrix $X \in \mathbb{R}^{d\times n}$ be partitioned into k clusters as $X = \{X_1, X_2, \dots, X_k\}$, where $X_i \in \mathbb{R}^{d\times n_i} (i = 1, 2, \dots, k)$ and $n = \sum_{i=1}^{k} n_i$.

Define the between-class scatter S_b, the within-class scatter S_w, and the total scatter S_t, where $S_t = S_b + S_w$. The trace of S_b, $Trace(S_b)$, measures the separation between clusters, where $Trace(S_w)$ gives the closeness within a cluster. Therefore, the objective of LDA is to find an optimal transformation matrix G that maximizes $Trace(S_b)$ while minimizing $Trace(S_w)$ in the lower dimensional space. Equation (1) gives the objective, called the classical Fisher criterion (Chen et al. 2000):

$$G = \arg\max_{G} Trace((G^T S_t G)^{-1}(G^T S_b G)), \qquad (1)$$

The classical LDA fails when S_t is singular $d \gg n > k$. This small sample size (SSS) problem occurs for many applications involving under sampled data such as text and image retrieval (Chen et al. 2000). A typical solution to the SSS problem lies in replacing the inverse in classical LDA by the pseudoinverse as (2), where $A^{(+)}$ denotes the pseudoinverse of A (Sharma and Paliwal 2015; Golub, Van, and Charles 1996).

$$G = \arg\max_{G^T G = I} trace((G^T S_t G)^{(+)}(G^T S_b G)). \qquad (2)$$

Furthermore, any G with

$$trace((G^T S_t G)^{(+)} G^T S_b G) = k - 1, \qquad (3)$$

is a solution to (2), provided that the training samples are linearly independent (Chu et al. 2015).

Fast Batch Linear Discriminant Analysis

In this section, we present a fast batch LDA algorithm, called FLDA/QR, from the perspective of online learning. We aim to lower down the dimensionality of the data matrix $X \in \mathbb{R}^{d\times n}$ for fast computation. Unlike a common treatment using the principal component analysis (PCA) for the reduction (Huang et al. 2002), we take cluster centers as training examples that is handy to pick up and has a definite dimension value in the feature space.

Denote the global center matrix C consisting of k centers as:

$$C = \left[c^{(1)}, \dots, c^{(k)}\right] = XEN_k, \qquad (4)$$

where $E = \begin{bmatrix} e_1 & & \\ & \ddots & \\ & & e_k \end{bmatrix} \in \mathbb{R}^{n\times k} (e_i = [1 \cdots 1] \in \mathbb{R}^{n_i})$,

$N_k = [1/n_1 \quad \cdots \quad 1/n_k]^T$. And we let $M = EN_k (M \in \mathbb{R}^{n\times k})$, then $X = CM^{-1}$ and $M^T E = I$. The cost of constructing the matrix C is as low as $\mathcal{O}(dn)$ in our implementation.

Subsequently, we adopt a lower triangular linear system (5), proposed by (Chu et al. 2015), to solve (2) :

$$X^T G = E, \qquad (5)$$

where $G \in \mathbb{R}^{d\times k}$ and $trace((G^T S_t G)^{(+)} G^T S_b G) = k-1$.

By replacing X with CM^{-1}, we obtain a new expression for the optimal transformation matrix, denoted by G_c, in the following:

$$G_c = (CM^{-1})^{-T} E = C^{-T} M^T E = C^{-T}, \qquad (6)$$

where $G_c \in \mathbb{R}^{d\times k}$.

Since the input training data is the centroid matrix, $S_w = 0$. And it is easy to verify that ${G_c}^T S_t G_c = {G_c}^T S_b G_c$, and

$rank({G_c}^T S_b G_c) = k-1$. Therefore G_c satisfies (3), and is an optimal solution to (2).

We further reduce the computational complexity by calculating C^{-T} with the economic QR-decomposition of $C = Q_c R_c$ (where $Q_c \in \mathbb{R}^{d\times k}$ has orthonormal columns and $R_c \in \mathbb{R}^{k\times k}$ is an upper triangular). It is a crucial observation that the Cholesky-factorization of $C^T C$ (Golub, Van, and Charles 1996) is able to generate the matrix R_c. Hence, we first compute $Q_c = C{R_c}^{-1}$, followed by $G_c = C^{-T} = Q_c {R_c}^{-T}$. This process is much more efficient than directly computing the QR-decomposition of C^{-T}.

FLDA/QR is summarized as Algorithm 1. The computational complexities of the four steps in Algorithm 1 are $\mathcal{O}(dn)$, $\mathcal{O}(dk^2+k^3)$, $\mathcal{O}(dk^2)$ and $\mathcal{O}(dk^2)$, respectively. Refer to Section 5 for detailed complexity comparisons with the existing batch LDA algorithms.

It is also worth noting that the use of the center matrix to construct a lower triangular linear system forms a well incremental mechanism. We take the mechanism to develop our fast exact incremental LDA algorithm in the next section.

Algorithm 1 FLDA/QR(Cholesky)

```
Input: The data matrix $X \in \mathbb{R}^{d\times n}$.
Output: The optimal transformation matrix $G_c$ .
1: Construct centroid matrix $C$.
2: Compute $R_c$ from the Cholesky Decomposition of $C^T C$.
3: Compute $Q_c = C{R_c}^{-1}$.
4: Compute $G_c = Q_c{R_c}^{-T}$.
```

Incremental Linear Discriminant Analysis

We develop the incremental algorithm IFLDA/QR for three learning tasks to handle mixture streaming data where new samples arrive differently. Otherwise stated, the notation $\tilde{A}$ is the updated version of A.

Task1: Insertion of samples to an existing class

The key issue to insert samples of an existing class for incremental learning is how to update Q_c and R_c after the insertion as $G_c = Q_c{R_c}^{-T}$ in the batch algorithm. A good incremental learning algorithm should learn additional information from new samples, and should leave alone the original examples during updating. We first extract information $c^{(i)}_{new}$ induced by new samples with the class label i, and then convert the computation of the QR-decomposition of $\tilde{C}$ into the production of Gram-Schmidt process with reorthogonalization of an augmented matrix $\begin{bmatrix} Q_c & c^{(i)}_{new}\end{bmatrix}$. This process is simpler and significantly faster than the rank-one updating in most ILDA methods, e.g., IDR/QR (Ye et al. 2005). We detail the algorithm below.

Let h_i be the total number of new samples ${X_{new}}^{(i)} \in \mathbb{R}^{d\times h_i}$ $(i \in 1,\ldots,k)$ to class i, and $\hat{c}^{(i)}_{new}$ be the class center of $X^{(i)}_{new}$. Thus, the updated center of the class i can be given as:

$$\tilde{c}^{(i)} = c^{(i)} + h_i(\hat{c}^{(i)}_{new} - c^{(i)})/(n_i + h_i) = c^{(i)} + c^{(i)}_{new}. \quad (7)$$

We update the new center $\tilde{C}$ of all training examples as:

$$\begin{aligned}\tilde{C} &= \begin{bmatrix} c^{(1)} & \cdots & c^{(i)} + c^{(i)}_{new} & \cdots & c^{(k)}\end{bmatrix} = C + c^{(i)}_{new} g^T \\ &= Q_c R_c + c^{(i)}_{new} g^T = \begin{bmatrix} Q_c R_c & c^{(i)}_{new}\end{bmatrix}\begin{bmatrix} I & g^T\end{bmatrix}^T \\ &= \begin{bmatrix} Q_c & c^{(i)}_{new}\end{bmatrix}\begin{bmatrix} R_c & 0 \\ 0 & 1\end{bmatrix}\begin{bmatrix} I \\ g^T\end{bmatrix}, \end{aligned} \quad (8)$$

where I is the identity matrix, and $g = (0\cdots1\cdots0)^T \in \mathbb{R}^n$ (the element 1 appears at the i^{th} position).

Naturally, we are able to apply the Gram-Schmidt process with reorthogonalization to compute the economic QR-decomposition of $\begin{bmatrix} Q_c & c^{(i)}_{new}\end{bmatrix}$ (Daniel et al. 1976):

$$\begin{bmatrix} Q_c & c^{(i)}_{new}\end{bmatrix} = [Q_c \quad q]\begin{bmatrix} I & r \\ 0 & \alpha\end{bmatrix}, \quad (9)$$

where $q = (c^{(i)}_{new} - Q_c r)/\alpha(\|q\| = 1, {Q_c}^T q = 0), \alpha = \left\| c^{(i)}_{new} - Q_c r\right\|_2 = \sqrt{{c^{(i)}_{new}}^T c^{(i)}_{new} - r^T r}$ and $r = {Q_c}^T c^{(i)}_{new}$.

Accordingly, the QR-decomposition of the updated center matrix $\tilde{C}$ is:

$$\begin{aligned}\tilde{C} &= [Q_c \quad q]\begin{bmatrix} R_c & r \\ 0 & \alpha\end{bmatrix}\begin{bmatrix} I \\ g^T\end{bmatrix} \\ &= [Q_c \quad q]\begin{bmatrix} R_c + rI \\ \alpha g^T\end{bmatrix} = \tilde{Q}_c\tilde{R}_c, \end{aligned} \quad (10)$$

where $\tilde{Q}_c \in \mathbb{R}^{d\times(k+1)}$ and $\tilde{R}_c \in \mathbb{R}^{(k+1)\times(k+1)}$.

Finally, we have the updated transform matrix $\tilde{G}_c$ by adding new information from new samples:

$$\begin{aligned}\tilde{G}_c &= [Q_c \quad q]\begin{bmatrix} R_c & r \\ 0 & \alpha\end{bmatrix}^{-T}\begin{bmatrix} I \\ g^{-1}\end{bmatrix} \\ &= [Q_c \quad q]\begin{bmatrix} {R_c}^{-T} & 0 \\ -r^T{R_c}^{-T}/\alpha & 1/\alpha\end{bmatrix}\begin{bmatrix} I \\ g^{-1}\end{bmatrix} \\ &= Q_c{R_c}^{-T} + (q\times g^{-1} - q{c^{(i)}_{new}}^T Q_c{R_c}^{-T})/\alpha \\ &= G_c + q(g - {G_c}^T c^{(i)}_{new})^T/\alpha, \end{aligned} \quad (11)$$

where $g^{(+)} = g^T$, for $g^{(+)} = (g^T g)^{-1} g^T$.

The pseudo-code of this algorithm is given in the **Algorithm 2**.

Task2: Insertion of a novel cluster

We follow the similar strategy to update the transformation matrix when inserting the samples of a novel cluster. We directly append the center of a novel class c_{new} to C without the need of center re-calculation as **Task1**, and then apply the Gram-Schmidt process with reorthogonalization to compute the economic QR-decomposition of $[Q_c \quad c_{new}]$. The

```
Algorithm 2 IFLDA/QR[1] (the insertion of new samples to an existing class)
Input: The $Q$ and $R$ matrices of the center matrix $C$ of the last step, and labeled new samples $X_{new}^{(i)} \in \mathbb{R}^{d\times h_i}$ $(i \in (1,\dots,k))$.
Output: The updated $\tilde{Q}_c$, $\tilde{R}_c$ and $\tilde{G}_c$.
1: Compute $c_{new}^{(i)}$.
2: Compute r, $\alpha$, q.
3: Update $\tilde{G}_c$ by (11).
4: Update $\tilde{Q}_c$ and $\tilde{R}_c$ by (10).
```

updating performs as follows.

$$\begin{aligned}\tilde{C} &= [C \quad c_{new}] = [Q_cR_c \quad c_{new}] = [Q_c \quad c_{new}]\begin{bmatrix}R_c & 0\\ 0 & 1\end{bmatrix}\\ &= [Q_c \quad q]\begin{bmatrix}I & r\\ 0 & \alpha\end{bmatrix}\begin{bmatrix}R_c & 0\\ 0 & 1\end{bmatrix}\\ &= [Q_c \quad q]\begin{bmatrix}R_c & r\\ 0 & \alpha\end{bmatrix} = \tilde{Q}_c\tilde{R}_c, \qquad (12)\end{aligned}$$

where $r = Q^Tc_{new}$, $\alpha = \sqrt{{c_{new}}^Tc_{new} - r^Tr}$, $q = (c_{new} - Qr)/\alpha$.

The transform matrix is updated as:

$$\begin{aligned}\tilde{G}_c = \tilde{C}^{-T} &= [Q_c \quad q]\begin{bmatrix}R_c & r\\ 0 & \alpha\end{bmatrix}^{-T}\\ &= [Q_c \quad q]\begin{bmatrix}{R_c}^{-T} & 0\\ -r^T{R_c}^{-T}/\alpha & 1/\alpha\end{bmatrix}\\ &= \begin{bmatrix}Q_c{R_c}^{-T} - qr^T{R_c}^{-T}/\alpha & q/\alpha\end{bmatrix}\\ &= \begin{bmatrix}G_c - q{c_{new}}^TG_c/\alpha & q/\alpha\end{bmatrix}. \qquad (13)\end{aligned}$$

Algorithm 3 shows the pseudo-code of inserting samples from a novel cluster.

```
Algorithm 3 IFLDA/QR[2] (the insertion of a novel class)
Input: The $Q$ and $R$ matrices of the center matrix $C$ for the previous step, and a novel cluster $X_{new} \in \mathbb{R}^{d\times n_{new}}$.
Output: The updated $\tilde{Q}_c$, $\tilde{R}_c$ and $\tilde{G}_c$.
1: Compute $c_{new}$.
2: Compute r, $\alpha$, q.
3: Update $\tilde{G}_c$ by (13).
4: Update $\tilde{Q}_c$ and $\tilde{R}_c$ by (12).
```

Task3: Insertion of chunk data

A chunk of new data may contain samples from the existing classes and novel classes. We have to extract the information from these mixed data, while preserving the previously learned ones. It is quite difficult for the existing ILDA methods to deal with such complex situation. In contrast, it is easier for us to separate new information of $\tilde{C}$, for we keep the cluster centers as the input. Accordingly, we transfer the computation of the QR-decomposition of $\tilde{C}$ to the product of two simpler decompositions. Thus, our method is more efficient than the existing ILDA methods, e.g.IDR/new (Lu, Jian, and Wang 2015). Their algorithm repeatedly runs so many rank-one QR-updating for computing the new within-class and between-class scatters.

Suppose the cluster center matrix C_{new} of a chunk containing the centers of some labeled new samples $\left[c_{new}^{(j_1)}\cdots c_{new}^{(j_{f_1})}\right]$ and the centers of a few of novel classes $[c_{new1}\cdots c_{newf_2}]$. Denote f_1 be the number of the existing classes those will be updated by new samples, f_2 be the number of novel classes in the chunk, h be the total number of samples in the chunk, f $(f = f_1 + f_2)$ be the total number of classes in the chunk, and $\tilde{k}$ $(\tilde{k} = k + f_2)$ be the total number of classes in the training set after insertion. Then we have the new data matrix:

$$C_{new} = \left[c_{new}^{(j_1)}\cdots c_{new}^{(j_{f_1})}, c_{new1}\cdots c_{newf_2}\right]. \qquad (14)$$

Then, we derive the updated center matrix $\tilde{C}$ after insertion as the product of two matrices where the new information C_{new} is extracted:

$$\begin{aligned}\tilde{C} &= \left[(c^{(\tilde{1})}\cdots c^{(\tilde{k})}, c_{new1}\cdots c_{newf_2}\right]\\ &= \begin{bmatrix}C & 0\end{bmatrix} + C_{new}\begin{bmatrix}I & {g_1}^T & \cdots & {g_{f_1}}^T & {g_1}^T & \cdots {g_{f_2}}^T\end{bmatrix}^T\\ &= \begin{bmatrix}C & C_{new}\end{bmatrix}\begin{bmatrix}I & 0\\ & Z\end{bmatrix} \qquad (15)\end{aligned}$$

where $Z = [z_1,\dots,z_h] \in \mathbb{R}^{h\times\tilde{k}}$, and $z_i \in \mathbb{R}^{\tilde{k}}$ is a unit vector with the i^{th} element being one and other elements being zeros.

Since

$$\begin{aligned}&\begin{bmatrix}C & C_{new}\end{bmatrix} = \begin{bmatrix}Q_cR_c & C_{new}\end{bmatrix}\\ &= \begin{bmatrix}Q_c & (I - Q_c{Q_c}^T)C_{new}\end{bmatrix}\begin{bmatrix}R_c & {Q_c}^TC_{new}\\ 0 & I\end{bmatrix}. \qquad (16)\end{aligned}$$

We compute the QR-decomposition of $(I - Q_c{Q_c}^T)C_{new}$ as $\hat{Q}_c\hat{R}_c$, and update the center matrix as:

$$\tilde{C} = \begin{bmatrix}Q_c & \hat{Q}_c\end{bmatrix}\begin{bmatrix}R_c & {Q_c}^TC_{new}\\ 0 & \hat{R}_c\end{bmatrix}\begin{bmatrix}I & 0\\ & Z\end{bmatrix}. \qquad (17)$$

Finally, we are able to update the transform matrix $\tilde{G}_c$ by the matrices $\hat{Q}_c$ and $\hat{R}_c$:

$$\begin{aligned}\tilde{G}_c &= \tilde{C}^{-T}\\ &= \begin{bmatrix}Q_c & \hat{Q}_c\end{bmatrix}\begin{bmatrix}{R_c}^{-T} & 0\\ -{\hat{R}_c}^{-T}{C_{new}}^TQ_c{R_c}^{-T} & {\hat{R}_c}^{-T}\end{bmatrix}\begin{bmatrix}I & 0\\ & Z\end{bmatrix}\\ &= Q_c{R_c}^{-T}\begin{bmatrix}I & 0\end{bmatrix} + \hat{Q}_c{\hat{R}_c}^{-T}Z\\ &\quad - \hat{Q}_c{\hat{R}_c}^{-T}{C_{new}}^TQ_c{R_c}^{-T}\begin{bmatrix}I & 0\end{bmatrix}\\ &= \begin{bmatrix}G_c & 0\end{bmatrix} + \hat{Q}_c{\hat{R}_c}^{-T}Z - \hat{Q}_c{\hat{R}_c}^{-T}{C_{new}}^T\begin{bmatrix}G_c & 0\end{bmatrix}\\ &= \begin{bmatrix}G_c - \hat{Q}_c({\hat{R}_c}^{-T}({C_{new}}^T)G_c) & 0\end{bmatrix} + \hat{Q}_c({\hat{R}_c}^{-T}Z). \qquad (18)\end{aligned}$$

The pseudo-code is listed in **Algorithm 4**.

Algorithm 4 $\mathrm{IFLDA/QR}^3$ (the insertion of chunk data)

```
Input: The $Q$ and $R$ matrices of the old center matrix, labeled new samples $X_{new}^{(i)} \in \mathbb{R}^{d\times h_i} (i \in [1,\dots,k])$ and novel classes $X_{newj} \in \mathbb{R}^{d\times h_j} (j \in [1,\dots,f_2])$.
Output: The updated $\tilde{Q}_c$, $\tilde{R}_c$ and $\tilde{G}_c$.
1: Construct $C_{new}$.
2: Compute QR decomposition for $(I - Q_c{Q_c}^T)C_{new} = \hat{Q}_c\hat{R}_c$ and update $\tilde{Q}_c$ and $\tilde{R}_c$.
3: Update $\tilde{G}_c$ by (17).
```

Table 1: Computational Complexities of batch LDA methods for SSS problem ($d \gg n > k$). (n is the total number of training samples, d is the dimension of training samples, and k is the number of training classes.)

Batch Implementations	Computational Complexities
PCA+Null LDA [Huang'02]	$\mathcal{O}(16dn^2 + 4dnk)$
IDR/QR (batch) [Ye'05]	$\mathcal{O}(8dk^2 + 2dnk + dn)$
NLDA/QR [Chu'10]	$\mathcal{O}(4dn^2 + 2dnk)$
CLDA/new [Lu'12]	$\mathcal{O}(4dn^2 + 2dnk)$
FNLDA [Sharma'12]	$\mathcal{O}(dn^2 + 2dnk)$
IDR/new [Lu'15]	$\mathcal{O}(4dk^2 + 2dnk + dn)$
LDA/QR [Chu'15]	$\mathcal{O}(4dn^2 + 2dnk)$
FLDA/QR [Ours]	$\mathcal{O}(3dk^2 + dn)$

Comparisons for Computational Complexity and Space Complexity

We analyze the time and space complexities of FLDA/QR and its incremental version IFLDA/QR compared with recent batch and incremental LDA algorithms.

Table 1 lists the computational complexities for eight representative batch LDA algorithms. Our FLDA/QR reduces the size of the input data matrix, so that the computation cost of the QR-decomposition of $C \in \mathbb{R}^{d\times k}$ by Cholesky factorization is far less than the QR-decomposition of $X \in \mathbb{R}^{d\times n}$ in LDA/QR (Chu et al. 2015). The second stage in FLDA/QR, solving G by the linear system ($G_c = Q_c{R_c}^{-T}$), is simpler and more efficient than the batch versions of IDR/QR (Ye et al. 2005) and IDR/new (Lu, Jian, and Wang 2015). These two existing algorithms apply the regularized LDA to solve an eigenvalue problem on $(Q^T S_w Q + \mu I_c)^{-1} Q^T S_b Q$ (μ is a positive constant), taking great time. Unlike PCA+LDA (Huang et al. 2002), FLDA/QR makes it easier to construct the center matrix for dimensionality reduction, and to determine the dimensionality of the feature space. More significantly, FLDA/QR only runs the QR-decomposition step *once*, while NLDA/QR (Chu and Thye 2010) has to compute the time-consuming QR-decomposition *twice*. The other algorithm FNLDA (Sharma and Paliwal 2012) has to compute the multiplication of $X^T X$ and its eigen decomposition costing $\mathcal{O}(dn^2 + 17n^3)$ in total, much more slowly than FLDA/QR. Besides, the space complexity of FLDA/QR, $\mathcal{O}(dk)$, is also favorable for online learning.

The computational and space complexities of seven incremental LDA methods are demonstrated in Tables 2 and 3, respectively. Those algorithms without the version handling chunk data are labeled with 'NULL' in the columns of 'Chunk' in these two tables. The Gram-Schmidt process with reorthogonalization in our IFLDA/QR runs much faster than the rank-one QR-updating in IDR/QR (Ye et al. 2005) and IDR/new (Lu, Jian, and Wang 2015). Our algorithm is able to insert a novel class with m samples at one time, while ILDA/QR (Chu et al. 2015) only updates one novel sample to form a new class in one step. ILDA/QR inserts the rest $(m - 1)$ samples either by using the algorithm for updating samples of an existing class, or by its chunk version which takes much more time than the single updating as shown in Table 2. Table 2 demonstrates that IFLDA/QR is the fastest, and belongs one of the four algorithms that have the capability to deal with mixture streaming data. Besides, IFLDA/QR consumes less storage space than *all* the others as shown in Table 3.

Table 2: The computational complexities for ILDA methods for SSS problem ($d \gg n > k$). ($\tilde{k}$ is the total number of classes after the insertion; f is the total number of classes in the chunk. h is the total number of samples in a chunk.)

Methods	Single	Chunk
IDR/QR [Ye'05]	$\mathcal{O}(dk + 91k^3/3)$	NULL
IDR/new [Lu'15]	NULL	$\mathcal{O}(d\tilde{k}^2 + d\tilde{k}(n + h) + (n+h)\tilde{k}^2)$
LS-ILDA [Liu'09]	$\mathcal{O}(14dn + 7dk)$	NULL
ILDA/SSS [Kim'11]	$\mathcal{O}(2dn^2 + 12n^3)$	$\mathcal{O}(2d(n+h)^2 + 12(n+h)^3)$
ICLDA [Lu'12]	$\mathcal{O}(2dn^2 + 20n^3)$	NULL
ILDA/QR [Chu'15]	$\mathcal{O}(4dn + 4dk)$	$\mathcal{O}(4dh(n + h + k) + 2d\tilde{k}h)$
IFLDA/QR [Ours]	$\mathcal{O}(dh + 4dk)$	$\mathcal{O}(4df(f + k) + d\tilde{k}f)$

Table 3: The space complexities for ILDA algorithms.($\tilde{k}$ is the total number of classes after the insertion. h is the total number of samples in a chunk.f_1 is the number of the existing classes those will be updated by new samples;f_2 is the number of novel classes in a chunk)

Methods	Single	Chunk
IDR/QR [Ye'05]	$\mathcal{O}(2dk)$	NULL
IDR/new [Lu'15]	NULL	$\mathcal{O}(\tilde{k}^3 + 2dnf_1 + dkf_1)$
LS-ILDA [Liu'09]	$\mathcal{O}(2dn + dk)$	NULL
ILDA/SSS [Kim'11]	$\mathcal{O}(dn + 2dk)$	$\mathcal{O}(dn + 2dk)$
ICLDA [Lu'2]	$\mathcal{O}(4dn + dn^2 + dk)$	NULL
ILDA/QR [Chu'15]	$\mathcal{O}(dn + dk)$	$\mathcal{O}(dn + dk)$
IFLDA/QR [Ours]	$\mathcal{O}(dk)$	$\mathcal{O}(df_2 + 4d{f_2}^2 + 3dkf_2)$

Experiments and Discussions

Settings

In this section, we evaluate the efficiency of FLDA and IFLDA by comparing the computational complexities and the classification accuracies with other methods. The ex-

Table 4: Data structures: d is the dimension of the data, n is the total number of the samples, k is the number of classes, and n_i is the number of samples in each class.

datasets	d	n	k	n_i
ORL	92*112	400	40	10
AWA	4096	1000	20	50
$Tr23$	5832	204	6	$6 \sim 91$

Table 5: Computational time (CT (seconds)) and classification accuracies (CA) for eight batch LDA methods.

Methods	ORL		AWA		$Tr23$	
	CT	CA	CT	CA	CT	CA
PCA+NLDA	0.22	88.50	1.02	88.00	0.058	74.80
IDR/QR-b	0.05	95.32	0.04	85.50	0.009	76.55
IDR/new-b	0.03	95.06	0.02	85.50	0.005	76.34
NLDA/QR	0.12	95.00	0.38	88.22	0.025	67.00
CLDA/new	0.21	95.00	0.69	85.33	0.058	69.66
FNLDA	0.04	95.00	0.19	88.00	0.014	67.00
ILDA/QR-b	0.10	90.44	0.15	87.12	0.018	87.00
FLDA/QR	0.01	93.35	0.01	86.56	0.003	86.00

periments are conducted on a face image dataset: ORL (Samaria and Harter 1994), an animal feature dataset: AWA (Lampert, Nickisch, and Harmeling 2013) and a text document feature set: $Tr23$ (derived from TREC collection (http://trec.nist.gov/). Each class of $Tr23$ is of unequal length). For AWA, we selected a subset of VGG19 feature (fc7 layer of very deep 19-layer CNN, pretrained on ILSVRC2014) as input. The statistics of these datasets are summarized in Table 4.

The experiments are performed on a 2.5GHZ Intel Core I7 Apple MacBook Pro with 16GB RAM in MATLAB 2014b environment. And the matlab codes of the proposed FLDA/QR and IFLDA/QR algorithms can be downloaded from website : https://github.com/dlut-dimt/FLDA-QR-and-IFLDA-QR.

Comparison with Batch LDA Algorithms

We randomly selected eighty percent of each dataset for training and the rest for testing. K-nearest neighbor (K-NN) method (with $K = 1$) has been used for classification purpose, and fivefold cross-validation was conducted. The average computational time and classification accuracies are shown in Table 5. The results show our FLDA/QR has the lowest computational complexities. FLDA/QR is about $4 \sim 8$ times faster than IDR/QR (batch), $2 \sim 4$ times faster than IDR/new (batch), and $10 \sim 100$ times faster than other methods. Besides, our classification performances are as good as or not far less than the others.

Comparisons of Incremental LDA Algorithms

From the analysis for computational complexity in Table 2, it is obvious that LS-ILDA, ILDA/SSS and ICLDA are not comparable with other ILDA methods, therefore, we only compare the rest ones in this experiments.

For all datasets, we first randomly split each class into two parts by a fixed ratio ($Part1 : Part2 = 4 : 1$) in terms of sample number, and take $Part1$ as the training set and

Table 6: The settings of datasets.(n_i is the number of samples in each class and n_q is the total number of classes in each group.)

Methods	P1:ITS		P1:NSS		P1:NCS		P2:TS	
	n_i	n_q	n_i	n_q	n_i	n_q	n_i	n_q
ORL	30	6	30	2	10	2	40	2
AWA	15	30	15	10	5	10	20	10
$Tr23$	4	$5 \sim 57$	4	4	2	4	6	$2 \sim 30$

Table 7: Computational time (CT(seconds)) and classification accuracies (CA) for three algorithms.

Methods	ORL		AWA		$Tr23$	
	CT	CA	CT	CA	CT	CA
IDR/QR	1.781	92.50	0.047	87.33	0.096	61.67
ILDA/QR	6.102	95.00	0.214	89.33	0.024	75.77
IFLDA/QR	1.104	92.25	0.024	88.00	0.005	74.80

$Part2$ as the testings set (TS). We then divide $Part1$ into three groups: the initial training set (ITS), the new sample set (labeled) (NSS) and the novel class set (NCS). Fivefold cross-validation of partition was evaluated. The settings of datasets are listed in Table 6.

Insertion of new sample streams to the existing classes This simulation is performed by randomly choosing and inserting new samples from the new sample set(NSS). We record the total execution time of every updating and the final classification accuracies for three methods in Table 7. It can be seen that IFLDA/QR are the fastest. IFLDA/QR is about 2 times faster than IDR/QR and $5 \sim 9$ times faster than ILDA/QR. And the accuracies of IFLDA/QR are close to the best one on three datasets.

Insertion of one novel sample This simulation is performed by randomly choosing one instance from each novel class, then inserting them into the training set. We record the total execution time of every updating and the final classification accuracies for three methods in Table 8. It is evident, IFLDA/QR are the fastest, and IFLDA/QR is more accurate than IDR/QR, and close to ILDA/QR on three datasets.

Insertion of a chunk of data This simulation is performed by randomly choosing and then inserting new samples, chunk by chunk. We set five chunks for ORL and AWA. For ORL, each chunk contains two novel classes and twelve labeled samples. For AWA, each chunk contains one novel class and thirty labeled samples. $Tr23$ only has three chunks. Each of first two chunks contains eight labeled samples, and the third contains four labeled samples and two novel classes.

We record the execution time of each updating for the three algorithms in Fig.2. And the total computational time with the classification accuracy of the final transformation, with their batch algorithms, are listed in Table 9.

From Fig.2 and Table 9, we can come to the following conclusions: (1) chunk IFLDA/QR achieves the same accuracy as that of batch FLDA/QR. (2) IFLDA/QR is faster than ILDA/QR and IDR/new both in the chunk mode and batch mode. (3) IFLDA/QR achieves comparative classification

Table 8: Computational time (CT(seconds)) and classification accuracies (CA) for three algorithms.

Methods	ORL		AWA		$Tr23$	
	CT	CA	CT	CA	CT	CA
IDR/QR	0.077	61.55	0.050	65.00	0.002	63.63
ILDA/QR	0.171	63.75	0.362	74.00	0.068	70.00
IFLDA/QR	0.039	62.75	0.039	72.00	0.001	67.63

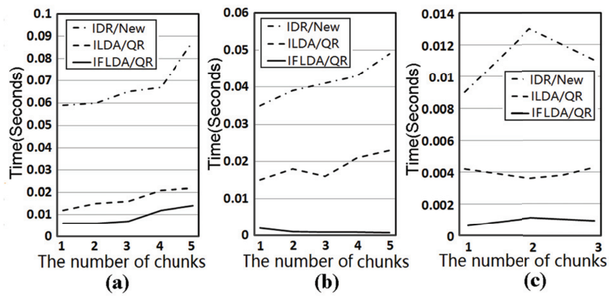

Figure 2: Comparison of computational time on real datasets: (a) ORL; (b) AWA; (c) $Tr23$.

accuracies on three real datasets. In addition, IFLDA/QR is more suitable for data with small within-class discrimination such as datasets AWA and $Tr23$.

Conclusion

In this paper, we first propose a new batch algorithm, called FLDA/QR, which is by far the fastest LDA implementation. Next, we present IFLDA/QR with three versions for inserting the sample stream into the existing clusters, inserting novel clusters, and inserting a chunk of instances at a time. Theoretical analysis and numerical experiments have shown that our FLDA/QR and IFLDA/QR are fast, efficient, and competitive with the state-of-the-art in terms of classification accuracy, computational complexity, and space requirement. FLDA/QR and IFLDA/QR can be applied to many online incremental learning scenarios, especially for those datasets with small within-class discrimination.

Acknowledgments

This research is sponsored in part by the National Natural Science Foundation of China (No.61402072, No.61272371, No. 61572096, and No. 61432003).

References

Chen, L. F.; Liao, H. Y. M.; Ko, M. T.; Lin, J. C.; and 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., and Thye, G. S. 2010. A new and fast implementation for null space based linear discriminant analysis. *Pattern Recognition* 43(4):1373–1379.

Chu, D.; Liao, L. Z.; Ng, M. K. P.; and Wang, X. 2015. Incremental linear discriminant analysis: a fast algorithm and comparisons. *IEEE Transactions on Neural Networks and Learning Systems* 26(11):2716–2735.

Table 9: Computational time (CT(seconds)) and classification accuracies (CA) for three algorithms in chunk(-c) and batch(-b) mode.

Methods	ORL		AWA		$Tr23$	
	CT	CA	CT	CA	CT	CA
IDR/new-b	0.061	78.5	0.049	86.0	0.014	58.3
IDR/new-c	0.353	78.5	0.064	83.0	0.015	60.0
ILDA/QR-b	0.075	75.0	0.116	89.5	0.015	50.0
ILDA/QR-c	0.273	75.0	0.023	89.5	0.005	45.7
FLDA/QR	0.017	78.3	0.007	86.0	0.004	63.3
IFLDA/QR-c	0.052	78.3	0.002	86.0	0.001	63.3

Daniel, J. W.; Gragg, W. B.; Kaufman, L.; and Stewart, G. 1976. Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization. *Mathematics of Computation* 30(136):772–795.

Golub, G. H.; Van, L.; and Charles, F. 1996. *Matrix Computations*. Baltimore: Johns Hopkins, third edition.

Huang, R.; Liu, Q.; Lu, H.; and Ma, S. 2002. Solving the small sample size problem of LDA. In *Proceedings of International Conference on Pattern Recognition*, volume 3, 29–32.

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

Kong, D., and Ding, C. H. 2014. Pairwise-Covariance Linear Discriminant Analysis. In *Proceedings of the Association for the Advance of Artificial Intelligence*, 1925–1931.

Lampert, C. H.; Nickisch, H.; and Harmeling, S. 2013. Attribute-Based Classification for Zero-Shot Visual Object Categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36(3):453–465.

Liu, L.-P.; Jiang, Y.; and Zhou, Z.-H. 2009. Least square incremental linear discriminant analysis. In *Proceedings of IEEE International Conference on Data Mining*, 298–306.

Lu, G.-F.; Jian, Z.; and Wang, Y. 2015. Incremental learning from chunk data for IDR/QR. *Image and Vision Computing* 36:1–8.

Lu, G.-F.; Zou, J.; and Wang, Y. 2012. Incremental complete LDA for face recognition. *Pattern Recognition* 45(7):2510–2521.

Luo, D.; Ding, C. H.; and Huang, H. 2011. Linear Discriminant Analysis: New Formulations and Overfit Analysis. In *Proceedings of the Association for the Advance of Artificial Intelligence*, 417–422.

Samaria, F. S., and Harter, A. C. 1994. Parameterisation of a stochastic model for human face identification. In *Applications of Computer Vision*, 138–142.

Sharma, A., and Paliwal, K. K. 2012. A new perspective to null linear discriminant analysis method and its fast implementation using random matrix multiplication with scatter matrices. *Pattern Recognition* 45(6):2205–2213.

Sharma, A., and Paliwal, K. K. 2015. Linear discriminant analysis for the small sample size problem: an overview. *International Journal of Machine Learning and Cybernetics* 6(3):443–454.

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