2009 Ninth IEEE International Conference on Data Mining

Least Square Incremental Linear Discriminant Analysis

Li-Ping Liu Yuan Jiang Zhi-Hua Zhou
National Key Laboratory for Novel Software Technology
Nanjing University, Nanjing 210093, China
{*liulp, jiangy, zhouzh*}*@lamda.nju.edu.cn*

***Abstract*—Linear discriminant analysis (LDA) is a well-known dimension reduction approach, which projects high-dimensional data into a low-dimensional space with the best separation of different classes. In many tasks, the data accumulates over time, and thus incremental LDA is more desirable than batch LDA. Several incremental LDA algorithms have been developed and achieved success; however, the eigen-problem involved requires a large computation cost, which hampers the efficiency of these algorithms. In this paper, we propose a new incremental LDA algorithm, LS-ILDA, based on the least square solution of LDA. When new samples are received, LS-ILDA incrementally updates the least square solution of LDA. Our analysis discloses that this algorithm produces the exact least square solution of batch LDA, while its computational cost is $O(\min(n, d) \times d)$ for one update on dataset containing n instances in d-dimensional space. Experimental results show that comparing with state-of-the-art incremental LDA algorithms, our proposed LS-ILDA achieves high accuracy with low time cost.**

***Keywords*-Dimension reduction; linear discriminant analysis (LDA); incremental learning; least square**

I. Introduction

In recent years, the amount of data is experiencing an explosive growth. The scale of data becomes larger and larger with the increase of capacity of computers and other data collection devices. The huge amount of data is due to both the high dimension and large number of data items. For example, in the literature of text mining, a passage is usually expressed as a vector with thousands of dimensions, and many corpus have huge number of passages. At the same time, data often come in stream and accumulate as time passes, and thus the amount is often unlimited. To reduce the time and resource consumption, it is generally desired that the data can be reduced to low dimension for processing.

Dimension reduction can reduce the data volume largely and provide much convenience to the later processing by projecting data from high-dimension feature space into low-dimensional feature space. The projection keeps information in the high-dimensional space as much as possible. One widely used supervised dimension reduction approach is LDA (Linear Discriminant Analysis). When projecting data into the low-dimensional space, LDA seeks the best separation of data from different classes by minimizing the within-class distance and maximizing the between-class distance simultaneously [3]. LDA performs well in many applications. Owing to LDA's good properties and the needs in streaming data mining, *incremental* LDA draws more and more interest. In situations where data come in stream, updating the solution to LDA with the incoming data is desired, since it avoids the time-consuming batch-mode re-calculation of LDA solution.

During the past few years, various incremental LDA algorithms have been developed. Most of them provide approximate solutions and suffer from high computational cost. There are some studies on incremental LDA using neural networks [2], [7], yet often suffer from slow convergence and severely undermine the significance of incremental learning. Pang et al. [9] proposed an incremental version of LDA, which provides a method for updating the within-class and between-class scatter matrices, while does not give a solution to the time-consuming step of updating subsequent eigen analysis. In [12] the proposed IDR/QR algorithm applies LDA in a projected subspace, in which the between-class difference is maximized. The deficiency of this algorithm, as shown in [13], is that much information is lost in the first projection. In [6], the concept of sufficient spanning set is used to update the between-class and within-class scatter matrices, where the eigenvectors of both matrices are kept and updated and minor components are removed in every step. In the calculation of the discriminant components, the scatter matrices are projected into a much lower-dimensional space, in which the eigen decomposition is accomplished. The incremental LDA algorithm GSVD-ILDA [13] is the incremental version of the algorithm proposed to solve LDA/GSVD in [11]. The core step of GSVD-ILDA is updating the eigenvectors of the centered data matrix. In updating, minor components are removed and thus the computational cost is reduced. The incremental LDA algorithm in [6] and the GSVD-ILDA algorithm suffer from a common problem, that is, it is difficult to determine to which degree the performance should be traded off for efficiency. If too much minor components are removed, the performance will deteriorate, otherwise the efficiency will be low. Moreover, the performance is sensitive to parameter settings, while tuning the parameters is not easy.

In this paper, we propose the LS-ILDA (Lease Square based Incremental LDA) approach based on LS-LDA [10], which shapes LDA into a multivariate linear regression problem and gives the least square solution to LDA. In our LS-ILDA approach, the least square solution is updated with

1550-4786/09 $26.00 © 2009 IEEE
DOI 10.1109/ICDM.2009.78

exactness when a new sample is inserted. Therefore, LS-ILDA does not need to trade off performance for efficiency. More importantly, LS-ILDA is very fast, since the update only involves some simple matrix manipulations, requiring far less computational cost than eigen analysis.

The rest of this paper is organized as follows. In section 2, we give a brief introduction to the least square solution to LDA. In Section 3 we propose the LS-ILDA approach. Section 4 reports on our experiments. Finally, we conclude the paper in Section 5.

II. Least Square Solution to LDA

LDA solves a general eigen-problem. Suppose there are $\mathcal{C}$ classes and n number of d-dimensional training samples, and n_c denotes the number of training samples of the class c. Let $\mathbf{1}$ denote all-one vector of proper length. The within-class scatter matrix $\mathbf{S}_w$, the between-class scatter matrix $\mathbf{S}_b$ and the total scatter matrix $\mathbf{S}_t$, are

$$\mathbf{S}_w = \frac{1}{n}\sum_{c=1}^{\mathcal{C}} \left(\hat{\mathbf{X}}_c - \boldsymbol{m}_c\mathbf{1}^{\mathrm{T}}\right)\left(\hat{\mathbf{X}}_c - \boldsymbol{m}_c\mathbf{1}^{\mathrm{T}}\right)^{\mathrm{T}} ,$$

$$\mathbf{S}_b = \frac{1}{n}\sum_{c=1}^{\mathcal{C}} n_c \left(\boldsymbol{m}_c - \boldsymbol{m}\right)\left(\boldsymbol{m}_c - \boldsymbol{m}\right)^{\mathrm{T}} ,$$

$$\mathbf{S}_t = \frac{1}{n}\left(\hat{\mathbf{X}} - \boldsymbol{m}\mathbf{1}^{\mathrm{T}}\right)\left(\hat{\mathbf{X}} - \boldsymbol{m}\mathbf{1}^{\mathrm{T}}\right)^{\mathrm{T}} ,$$

respectively, where $\hat{\mathbf{X}} \in \mathbb{R}^{d\times n}$ is the data matrix in which the columns are training samples, $\hat{\mathbf{X}}_c$ is the data matrix of training samples belonging to the class c, $\boldsymbol{m}$ is the mean vector of all training samples, $\boldsymbol{m}_c$ is the mean vector of training samples belonging to the class c, and T denotes matrix transpose.

LDA computes a linear transformation matrix $\mathbf{W} \in \mathbb{R}^{d\times(\mathcal{C}-1)}$, and usually $d \gg \mathcal{C}$. The transformation matrix projects data from the original high-dimensional space into a low-dimensional space, maximizing the between-class distance while minimizing the within-class distance. Traditional LDA finds the optimal transformation matrix $\mathbf{W}_{LDA}$ by solving the optimization problem

$$\mathbf{W}_{LDA} = \arg\max_{\mathbf{W}} \operatorname{trace}\left(\mathbf{W}^{\mathrm{T}}\mathbf{S}_b\mathbf{W}\left(\mathbf{W}^{\mathrm{T}}\mathbf{S}_t\mathbf{W}\right)^{-1}\right) . \quad (1)$$

According to [4], when the total scatter matrix $\mathbf{S}_t$ is non-singular, the solution $\mathbf{W}_{LDA}$ consists of the top eigenvectors of the matrix $(\mathbf{S}_t^{-1}\mathbf{S}_b)$ corresponding to nonzero eigenvalues. When the total scatter matrix $\mathbf{S}_t$ does not have a full rank, $\mathbf{W}_{LDA}$ consists of the eigenvectors of $(\mathbf{S}_t^{+}\mathbf{S}_b)$ corresponding to the nonzero eigenvalues, where $\mathbf{S}_t^{+}$ denotes the pseudo-inverse of $\mathbf{S}_t$ [5].

In [10], the relationship between LDA and multi-variate linear regression problem is investigated, and LDA is put into the framework of multi-variate linear regression by adding a constraint to the optimization problem of LDA, that is,

$$\max_{\mathbf{Y}} \ \operatorname{trace}\left(\left(\mathbf{W}^{\mathrm{T}}\mathbf{S}_b\mathbf{W}\right)\left(\mathbf{W}^{\mathrm{T}}\mathbf{S}_t\mathbf{W}\right)^{+}\right) \quad (2)$$
$$s.t.\ \mathbf{W} = \left(\mathbf{X}\mathbf{X}^{\mathrm{T}}\right)^{+}\mathbf{X}\mathbf{Y} ,$$

where $\mathbf{X} \in \mathbb{R}^{d\times n}$ is the centered matrix of $\hat{\mathbf{X}}$, and $\mathbf{Y}$ is the indicator matrix to be optimized.

By solving Eq. 2, we can obtain the indicator matrix $\mathbf{Y}'$,

$$\mathbf{Y}'_{ic} = \begin{cases} \sqrt{\frac{n}{n_c}} - \sqrt{\frac{n_c}{n}} & \text{if } X_{\cdot,i} \text{ belongs to class } c \\ -\sqrt{\frac{n_c}{n}} & \text{otherwise} , \end{cases} \quad (3)$$

and the multi-variate linear regression solution to LDA, $\mathbf{W}_{MLR}$, as shown in Eq. 4.

$$\mathbf{W}_{MLR} = \left(\mathbf{X}^{+}\right)^{\mathrm{T}}\mathbf{Y}' \quad (4)$$

Ye [10] proved that one can have Eq. 5 if the mild condition shown in Eq. 6 holds, where $\mathbf{Q}$ is orthogonal, and thus $\mathbf{W}_{MLR}$ is equivalent to $\mathbf{W}_{LDA}$ since they project data into two equivalent spaces.

$$\mathbf{W}_{MLR} = [\mathbf{W}_{LDA}, \mathbf{0}]\,\mathbf{Q}^{\mathrm{T}} \quad (5)$$

$$\operatorname{rank}(\mathbf{S}_b) + \operatorname{rank}(\mathbf{S}_w) - \operatorname{rank}(\mathbf{S}_t) = 0 \quad (6)$$

Actually Eq. 6 often holds in many applications. Even when it does not hold, $\mathbf{W}_{MLR}$ is very similar to $\mathbf{W}_{LDA}$ in the sense of making discrimination in the low-dimensional spaces. In this paper, we will take $\mathbf{W}_{MLR}$ as the solution to LDA and will focus on the design of its incremental version.

III. Our LS-LDA Approach

A. Redefining Indicator Matrix

In this paper, the least square solution $\mathbf{W}_{MLR}$ is simplified for convenience of updating. The new least square solution $\mathbf{W}$ is defined in Eq. 7 with a much simpler indicator matrix $\mathbf{Y}$.

$$\mathbf{W} = (\mathbf{X}^{+})^{\mathrm{T}}\mathbf{Y} , \quad (7)$$

where the indicator matrix $\mathbf{Y}$ is

$$Y_{ic} = \begin{cases} \frac{1}{\sqrt{n_c}} & \text{if } X_{\cdot,i} \text{ belongs to class } c , \\ 0 & \text{otherwise} . \end{cases} \quad (8)$$

Proposition III.1. *If $\mathbf{W}_{MLR}$ and $\mathbf{W}$ are defined according to Eqs. 4 and 7, respectively, and the data matrix $\mathbf{X}$ is centered, then*

$$\mathbf{W} = \frac{1}{\sqrt{n}}\mathbf{W}_{MLR} . \quad (9)$$

Proof:

$\mathbf{X1} = \mathbf{0}$ since $\mathbf{X}$ is centered. Therefore,

$$\begin{aligned}(\mathbf{X}^+)^{\mathrm{T}}\mathbf{1} &= (\mathbf{X}^+\mathbf{X}\mathbf{X}^+)^{\mathrm{T}}\mathbf{1} \\ &= (\mathbf{X}^+)^{\mathrm{T}}\mathbf{X}^+(\mathbf{X}\mathbf{1}) \\ &= \mathbf{0}\ .\end{aligned}$$

According to Eqs. 3 and 8, $\mathbf{Y}'$ and $\mathbf{Y}$ have a relation shown in Eq. 10.

$$\mathbf{Y}' = \sqrt{n}\ \mathbf{Y} - \mathbf{1}\left[\sqrt{\frac{n_1}{n}}, \sqrt{\frac{n_2}{n}}, \cdots, \sqrt{\frac{n_{\mathcal{C}}}{n}}\right] \tag{10}$$

Therefore,

$$\mathbf{W}_{MLR} = (\mathbf{X}^+)^{\mathrm{T}}\mathbf{Y}' = (\mathbf{X}^+)^{\mathrm{T}}\sqrt{n}\,\mathbf{Y} = \sqrt{n}\,\mathbf{W}\ . \tag{11}$$

□

From the proposition we can see that $\mathbf{W}$ is a down-scaled solution. If needed, it can be scaled back without much computation. So, in the following we only consider the update of $\mathbf{W}$; and when a variable is updated, a tilde will be added on top of the variable.

B. Updating Indicator Matrix $\mathbf{Y}$

This section will focus on the update of the indicator matrix defined in Eq. 8.

Denote the label of the new sample as p. If p has already been observed before, the number of classes does not change, i.e., $\tilde{\mathcal{C}} = \mathcal{C}$, and the number of samples belonging to the class p increases by 1, i.e., $\tilde{n}_p = n_p + 1$; otherwise, $\tilde{\mathcal{C}} = \mathcal{C} + 1$ and $\tilde{n}_p = 1$. The indicator vector corresponding to the new sample will be appended to the indicator matrix as a new row. The new indicator vector, denoted by $\boldsymbol{y}$, is defined as Eq. 12.

$$y_c = \begin{cases} \frac{1}{\sqrt{n_c+1}} & \text{if } c = p\ , \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

If p has been observed before, i.e., $p \leq \mathcal{C}$, then the matrix $\mathbf{Y}$ is updated according to

$$\tilde{\mathbf{Y}} = \begin{bmatrix} \mathbf{Y} \otimes_p \alpha_p \\ \boldsymbol{y}^{\mathrm{T}} \end{bmatrix}\ , \tag{13}$$

where

$$\alpha_p = \sqrt{\frac{n_p}{n_p + 1}}\ , \tag{14}$$

and the operator $\otimes_p$ means multiplying the p-th column of $\mathbf{Y}$ with α_p.

If p has not been observed before, i.e., $p > \mathcal{C}$, then

$$\tilde{\mathbf{Y}} = \begin{bmatrix} \mathbf{Y} & \mathbf{0} \\ \boldsymbol{y}^{\mathrm{T}} \end{bmatrix}\ , \tag{15}$$

where in this case $\boldsymbol{y}$ is a vector with length of $(\mathcal{C} + 1)$.

Unifying the above two cases into one form, $\mathbf{Y}$ is updated according to

$$\tilde{\mathbf{Y}} = \begin{bmatrix} \mathbf{Y}\bar{\otimes}_p \alpha_p \\ \boldsymbol{y}^{\mathrm{T}} \end{bmatrix}\ , \tag{16}$$

where $\bar{\otimes}_p$ means that if p is smaller than or equal to the width of $\mathbf{Y}$, the p-th column of $\mathbf{Y}$ will be multiplied with α_p; otherwise, a new column with zero elements will be augmented to $\mathbf{Y}$.

It is easy to verify that

$$\mathbf{A}(\mathbf{B}\,\bar{\otimes}_p\,\alpha_p) = (\mathbf{A}\mathbf{B})\,\bar{\otimes}_p\,\alpha_p\ . \tag{17}$$

C. Updating LDA Solution $\mathbf{W}$

To improve efficiency, we present two methods for updating $\mathbf{W}$ by considering whether n is smaller than d or not. In addition to updating $\mathbf{W}$, either $\mathbf{X}^+$ or $(\mathbf{X}\mathbf{X}^{\mathrm{T}})^+$ will be updated as intermediate result. In the case of $n < d$, the matrix $\mathbf{X}^+$ is smaller than $(\mathbf{X}\mathbf{X}^{\mathrm{T}})^+$, and therefore we will update $\mathbf{X}^+$; while when $n \geq d$, especially when the number of samples overwhelms the number of dimension, we will update $(\mathbf{X}\mathbf{X}^{\mathrm{T}})^+$ to avoid the computational cost for one update to grow with the number of samples.

1) Updating $\mathbf{W}$ When $n < d$:

In the case $n < d$, the updating formulas will be derived from Eq. 7. In the updating process, the mean of original data matrix $\boldsymbol{m}$, the centered data matrix $\mathbf{X}$, its pseudo-inverse $\mathbf{X}^+$ and the solution $\mathbf{W}$ are kept and updated after every insertion of a new sample. It is assumed that the rank of $\mathbf{X}$ will increase by 1 when a new sample is added.

When a new sample $\boldsymbol{x}$ is inserted, the mean $\boldsymbol{m}$ is first updated according to

$$\tilde{\boldsymbol{m}} = \boldsymbol{m} + \frac{1}{n+1}(\boldsymbol{x} - \boldsymbol{m})\ . \tag{18}$$

Then the centered data matrix is updated. After the insertion of $\boldsymbol{x}$, $\mathbf{X}$ should be re-centered according to

$$\tilde{\mathbf{X}} = \left[\mathbf{X} - \frac{1}{n+1}(\boldsymbol{x} - \boldsymbol{m})\mathbf{1}^{\mathrm{T}} \quad \frac{n}{n+1}(\boldsymbol{x} - \boldsymbol{m})\right]\ , \tag{19}$$

while its pseudo-inverse $\mathbf{X}^+$ changes according to Theorem III.1.

Theorem III.1. *Assume $\mathbf{X}$, $\mathbf{X}^+$, $\boldsymbol{x}$, $\tilde{\mathbf{X}}$, $\boldsymbol{m}$, $\mathbf{1}$, n are defined as above, and rank$(\tilde{\mathbf{X}})$ − rank$(\mathbf{X}) = 1$. The pseudo-inverse $\tilde{\mathbf{X}}^+$ is*

$$\tilde{\mathbf{X}}^+ = \begin{bmatrix} \mathbf{X}^+ - \mathbf{X}^+(\boldsymbol{x} - \boldsymbol{m})\boldsymbol{h}^{\mathrm{T}} - \frac{1}{n}\mathbf{1}\boldsymbol{h}^{\mathrm{T}} \\ \boldsymbol{h}^{\mathrm{T}} \end{bmatrix}\ , \tag{20}$$

where

$$\boldsymbol{h} = \frac{(\boldsymbol{x} - \boldsymbol{m}) - \mathbf{X}\mathbf{X}^+(\boldsymbol{x} - \boldsymbol{m})}{(\boldsymbol{x} - \boldsymbol{m})^{\mathrm{T}}(\boldsymbol{x} - \boldsymbol{m}) - (\boldsymbol{x} - \boldsymbol{m})^{\mathrm{T}}\mathbf{X}\mathbf{X}^+(\boldsymbol{x} - \boldsymbol{m})}\ . \tag{21}$$

Proof:

The fact $\text{rank}(\tilde{\mathbf{X}}) - \text{rank}(\mathbf{X}) = 1$ implies $(\boldsymbol{x} - \boldsymbol{m}) - \mathbf{X}\mathbf{X}^+(\boldsymbol{x} - \boldsymbol{m}) \neq \mathbf{0}$. Denote the denominator of $\boldsymbol{h}$ as k,

$$k = (\boldsymbol{x} - \boldsymbol{m})^{\mathrm{T}}(\boldsymbol{x} - \boldsymbol{m}) - (\boldsymbol{x} - \boldsymbol{m})^{\mathrm{T}}\mathbf{X}\mathbf{X}^+(\boldsymbol{x} - \boldsymbol{m}) \ . \quad (22)$$

Considering that $(\boldsymbol{x}-\boldsymbol{m})-\mathbf{X}\mathbf{X}^+(\boldsymbol{x}-\boldsymbol{m})$ is the component of $\mathbf{X}$ in its null space, we have:

$$\begin{aligned}
&\mathbf{X}^{\mathrm{T}}\boldsymbol{h} = \left(\mathbf{X}^{\mathrm{T}}(\boldsymbol{x} - \boldsymbol{m}) - \mathbf{X}^{\mathrm{T}}\mathbf{X}\mathbf{X}^+(\boldsymbol{x} - \boldsymbol{m})\right)/k = \mathbf{0} \ , \\
&\mathbf{X}^+\boldsymbol{h} = \left(\mathbf{X}^+(\boldsymbol{x} - \boldsymbol{m}) - \mathbf{X}^+\mathbf{X}\mathbf{X}^+(\boldsymbol{x} - \boldsymbol{m})\right)/k = \mathbf{0} \ , \\
&\mathbf{X}\mathbf{1}^{\mathrm{T}} = \mathbf{0} \ , \\
&\mathbf{1}\mathbf{X}^+ = \mathbf{0} \ .
\end{aligned}$$

Let $\mathbf{P} = \begin{bmatrix} \mathbf{X}^+ - \mathbf{X}^+(\boldsymbol{x} - \boldsymbol{m})\boldsymbol{h}^{\mathrm{T}} - \frac{1}{n}\mathbf{1}\boldsymbol{h}^{\mathrm{T}} \\ \boldsymbol{h}^{\mathrm{T}} \end{bmatrix}$. With the equations above, we have

$$\tilde{\mathbf{X}}\mathbf{P} = \mathbf{X}\mathbf{X}^+ + k\boldsymbol{h}\boldsymbol{h}^{\mathrm{T}} \ , \quad (23)$$

and

$$\mathbf{P}\tilde{\mathbf{X}} = \begin{bmatrix} \mathbf{X}^+\mathbf{X} + \frac{\mathbf{1}\mathbf{1}^{\mathrm{T}}}{n^2+n} & \frac{-\mathbf{1}}{n+1} \\ \frac{-\mathbf{1}^{\mathrm{T}}}{n+1} & \frac{n}{n+1} \end{bmatrix} . \quad (24)$$

According to the property of pseudo-inverse, $\mathbf{X}\mathbf{X}^+ = (\mathbf{X}\mathbf{X}^+)^{\mathrm{T}}$, and therefore $\tilde{\mathbf{X}}\mathbf{P}$ and $\mathbf{P}\tilde{\mathbf{X}}$ are symmetric.

We also have

$$\tilde{\mathbf{X}}\mathbf{P}\tilde{\mathbf{X}} = \begin{bmatrix} \mathbf{X} - \dfrac{(\boldsymbol{x} - \boldsymbol{m})\mathbf{1}^{\mathrm{T}}}{n+1} & \dfrac{n(\boldsymbol{x} - \boldsymbol{m})}{n+1} \end{bmatrix} = \tilde{\mathbf{X}} \quad (25)$$

and

$$\mathbf{P}\tilde{\mathbf{X}}\mathbf{P} = \begin{bmatrix} \mathbf{X}^+ - \mathbf{X}^+(\boldsymbol{x} - \boldsymbol{m})\mathbf{h}^{\mathrm{T}} - \frac{1}{n}\mathbf{1}\boldsymbol{h}^{\mathrm{T}} \\ \boldsymbol{h}^{\mathrm{T}} \end{bmatrix} = \mathbf{P} \ . \quad (26)$$

Therefore $\tilde{\mathbf{X}}^+ = \mathbf{P}$. □

Thus, the least square solution $\mathbf{W}$ to LDA can be updated according to the updated matrix $\tilde{\mathbf{X}}^+$ and $\tilde{\mathbf{Y}}$. Applying Eqs. 16 and 20 to Eq. 7, we have

$$\begin{aligned}
\tilde{\mathbf{W}} &= (\tilde{\mathbf{X}}^+)^{\mathrm{T}}\tilde{\mathbf{Y}} \quad (27)\\
&= \left((\mathbf{X}^+)^{\mathrm{T}} - \boldsymbol{h}(\boldsymbol{x} - \boldsymbol{m})^{\mathrm{T}}(\mathbf{X}^+)^{\mathrm{T}} - \frac{\boldsymbol{h}\mathbf{1}^{\mathrm{T}}}{n}\right)(\mathbf{Y}\bar{\otimes}_p\alpha_p) \\
&\quad + \boldsymbol{h}\boldsymbol{y}^{\mathrm{T}} \\
&= \left(\mathbf{W} - \boldsymbol{h}(\boldsymbol{x} - \boldsymbol{m})^{\mathrm{T}}\mathbf{W} - \frac{\boldsymbol{h}\mathbf{1}^{\mathrm{T}}\mathbf{Y}}{n}\right)\bar{\otimes}_p\,\alpha_p + \boldsymbol{h}\boldsymbol{y}^{\mathrm{T}} \ ,
\end{aligned}$$

where $\boldsymbol{h}$ is defined as Eq. 21 and α_p is defined as Eq. 14.

There is a trick for reducing the computational load. Considering that $\boldsymbol{y}$ is a row vector and only the p-th element is nonzero, the product of $\boldsymbol{h}\boldsymbol{y}^{\mathrm{T}}$ is a matrix with the p-th column as $\boldsymbol{h}y_p$ while all other columns are zero vectors. So, only the nonzero column needs to be added onto the p-th column of the matrix before the plus sign.

The four steps above are the updating method for the least square solution $\mathbf{W}$. In each update, $\boldsymbol{m}$, $\mathbf{X}$, $\mathbf{X}^+$, $\mathbf{W}$ and necessary counters are kept. Every time a new sample is inserted, the four matrices are updated in order. By these updates, Eq. 7 always holds, and thus $\mathbf{W}$ is always the LDA solution for the current dataset.

2) Update $\mathbf{W}$ When $n \geq d$:

When the number of samples is not smaller than the number of dimensions, it is more appropriate to store the scatter matrix $\mathbf{T}$[1], $\mathbf{T} = \mathbf{X}\mathbf{X}^{\mathrm{T}}$ and its pseudo-inverse $\mathbf{T}^+$, thus the time and space complexity will not increase with n. Since $\mathbf{X}^+ = (\mathbf{X}\mathbf{X})^+\mathbf{X}$, $\mathbf{W}$ can be rewritten as Eq. 28, from which the following updating method is derived.

$$\mathbf{W} = \mathbf{T}^+\mathbf{X}\mathbf{Y} \quad (28)$$

The mean of data matrix $\boldsymbol{m}$ is updated first according to Eq. 18. Then, $\mathbf{T}$ is updated according to

$$\begin{aligned}
\tilde{\mathbf{T}} &= \tilde{\mathbf{X}}\tilde{\mathbf{X}}^{\mathrm{T}} \quad (29)\\
&= \mathbf{T} + \frac{n}{n+1}(\boldsymbol{x} - \boldsymbol{m})(\boldsymbol{x} - \boldsymbol{m})^{\mathrm{T}} \\
&= \mathbf{T} + \boldsymbol{u}\boldsymbol{u}^{\mathrm{T}} \ ,
\end{aligned}$$

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

The update of $\mathbf{T}^+$ depends on whether the new sample has components out of the space spanned by $\mathbf{T}$. Theorem III.2 shows how to update $\mathbf{T}^+$ according to the update of $\mathbf{T}$.

Theorem III.2. *Assume $\mathbf{T}$, $\boldsymbol{x}$, $\boldsymbol{m}$ are defined as above, and $\tilde{\mathbf{T}}$ is defined by Eq. 29. Denote $\boldsymbol{s} = \mathbf{T}^+\boldsymbol{u}$, $\boldsymbol{t} = (\mathbf{I} - \mathbf{T}\mathbf{T}^+)\boldsymbol{u}$, $\theta = 1 + \boldsymbol{u}^{\mathrm{T}}\mathbf{T}^+\boldsymbol{u}$. Then,*

$$\tilde{\mathbf{T}}^+ = \begin{cases} \mathbf{T}^+ - \theta^{-1}\boldsymbol{s}\boldsymbol{s}^{\mathrm{T}} & \text{if } \boldsymbol{t} = 0 \\ \mathbf{T}^+ - \frac{\boldsymbol{s}\boldsymbol{t}^{\mathrm{T}} + \boldsymbol{t}\boldsymbol{s}^{\mathrm{T}}}{\boldsymbol{t}^{\mathrm{T}}\boldsymbol{t}} + \frac{\theta\boldsymbol{t}\boldsymbol{t}^{\mathrm{T}}}{(\boldsymbol{t}^{\mathrm{T}}\boldsymbol{t})^2} & otherwise \ . \end{cases} \quad (30)$$

Proof:

According to Theorem 2.1 in [1], when a matrix $\mathbf{M}$ is modified from $\mathbf{A}$ as

$$\mathbf{M} = \mathbf{A} + \boldsymbol{b}\boldsymbol{c}^{\mathrm{T}} \ , \quad (31)$$

the pseudo-inverse of $\mathbf{M}$ can be calculated as

$$\mathbf{M}^+ = \begin{cases} \mathbf{A}^+ - \lambda^{-1}\boldsymbol{d}\boldsymbol{e}^{\mathrm{T}} & \text{if } \boldsymbol{f} = \mathbf{0}, \boldsymbol{g} = \mathbf{0}, \lambda \neq 0 \\ \mathbf{A}^+ - \frac{\boldsymbol{d}\boldsymbol{f}^{\mathrm{T}}}{\phi} - \frac{\boldsymbol{g}\boldsymbol{e}^{\mathrm{T}}}{\psi} + \frac{\lambda\boldsymbol{g}\boldsymbol{f}^{\mathrm{T}}}{\phi\psi} & \text{if } \boldsymbol{f} \neq \mathbf{0}, \boldsymbol{g} \neq \mathbf{0} \ , \end{cases}$$

where $\boldsymbol{d} = \mathbf{A}^+\boldsymbol{b}$, $\boldsymbol{e} = (\mathbf{A}^+)^{\mathrm{T}}\boldsymbol{c}$, $\boldsymbol{f} = \left(\mathbf{I} - \mathbf{A}\mathbf{A}^+\right)\boldsymbol{b}$, $\boldsymbol{g} = (\mathbf{I} - \mathbf{A}^+\mathbf{A})\,\boldsymbol{c}$, $\phi = \boldsymbol{f}^{\mathrm{T}}\boldsymbol{f}$, $\psi = \boldsymbol{g}^{\mathrm{T}}\boldsymbol{g}$ and $\lambda = 1 + \boldsymbol{c}^{\mathrm{T}}\mathbf{A}^+\boldsymbol{b}$. Only two relevant cases are listed here.

We need to prove that the updating of $\mathbf{T}^+$ can fit into the two cases. Let $\mathbf{A} = \mathbf{T}$, $\boldsymbol{b} = \boldsymbol{c} = \boldsymbol{u}$ and $\mathbf{M} = \tilde{\mathbf{T}}$. Since $\mathbf{T}$

[1]It is also called scatter matrix, which is up-scaled from $\mathbf{S}_t$ by n.

is scatter matrix, which is positive semi-definite, and $\mathbf{T}^+$ is also positive semi-definite, we have

$$\lambda = 1 + \boldsymbol{u}^{\mathrm{T}}\mathbf{T}^{+}\boldsymbol{u} > 0 \ .$$

$\mathbf{T}$ is symmetric, and $\mathbf{T}^+$ is also symmetric, so

$$\boldsymbol{f} = \left(\mathbf{I} - \mathbf{T}\mathbf{T}^{+}\right)\boldsymbol{u} = \left(\mathbf{I} - \mathbf{T}^{+}\mathbf{T}\right)\boldsymbol{u} = \boldsymbol{g} \ .$$

If $(\boldsymbol{f} = \boldsymbol{g}) = 0$, then it fits into the first case; if $(\boldsymbol{f} = \boldsymbol{g}) \neq 0$, then it fits into the second case. Putting $\mathbf{M} = \tilde{\mathbf{T}}$, $\mathbf{A} = \mathbf{T}$ and $\boldsymbol{a} = \boldsymbol{b} = \boldsymbol{u}$ into the formula of $\mathbf{M}^+$, we can get the conclusion. □

When $\mathbf{T}$ has full rank, the update always takes the formula of the first case, in which it requires far less computation and does not need to keep the matrix $\mathbf{T}$. After accumulation of enough samples, $\mathbf{T}$ always has full rank, and therefore $\boldsymbol{t} = (\mathbf{I} - \mathbf{T}\mathbf{T}^{+})\boldsymbol{u}$ is always $\mathbf{0}$ and $\mathbf{T}^+$ is updated as the first case shown in Eq. 30. In implementation, we use a counter r to record the rank of $\mathbf{T}$. The counter r is initialized with rank $(\mathbf{T})$. r is updated according to [2]

$$\tilde{r} = \begin{cases} r+1 & \text{if } \boldsymbol{t} \neq 0 \\ r & \text{otherwise} \ . \end{cases} \tag{32}$$

Once $\mathbf{T}$ has full rank (i.e. $r = d$), $\mathbf{T}$ and r can be discarded and only $\mathbf{T}^+$ needs to be updated in the first case of Eq. 30. This is because that $\boldsymbol{t}$ will always be $\mathbf{0}$, and r and $\mathbf{T}$ become unnecessary for updating.

Finally we can update $\mathbf{W}$ with $\tilde{\mathbf{T}}^+$, $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{Y}}$. Applying Eqs. 16, 19 and 30 to Eq. 28, we have

$$\begin{aligned} \tilde{\mathbf{W}} &= \tilde{\mathbf{T}}^{+}\tilde{\mathbf{X}}\tilde{\mathbf{Y}} \qquad (33) \\ &= \tilde{\mathbf{T}}^{+}\left[\mathbf{X} - \frac{(\boldsymbol{x}-\boldsymbol{m})\,\mathbf{1}^{\mathrm{T}}}{n+1} \ \frac{n(\boldsymbol{x}-\boldsymbol{m})}{n+1}\right]\begin{bmatrix} \mathbf{Y}\bar{\otimes}_p\,\alpha_p \\ \boldsymbol{y}^{\mathrm{T}} \end{bmatrix} \\ &= \left(\tilde{\mathbf{T}}^{+}\mathbf{X}\mathbf{Y} - \frac{\tilde{\mathbf{T}}^{+}(\boldsymbol{x}-\boldsymbol{m})\,\mathbf{1}^{\mathrm{T}}\mathbf{Y}}{n+1}\right)\bar{\otimes}_p\,\alpha_p \\ &\quad + \frac{n\tilde{\mathbf{T}}^{+}(\boldsymbol{x}-\boldsymbol{m})\,\boldsymbol{y}^{\mathrm{T}}}{n+1} \ . \end{aligned}$$

Let $\mathbf{G} = \tilde{\mathbf{T}}^{+}\mathbf{X}\mathbf{Y}$ and expand it, then

$$\mathbf{G} = \begin{cases} \mathbf{W} - \theta^{-1}\boldsymbol{s}\boldsymbol{u}^{\mathrm{T}}\mathbf{W} & \text{if } \boldsymbol{t} = 0 \ , \\ \mathbf{W} - \frac{\boldsymbol{t}\boldsymbol{u}^{\mathrm{T}}\mathbf{W}}{\boldsymbol{t}^{\mathrm{T}}\boldsymbol{t}} & \text{otherwise} \ , \end{cases} \tag{34}$$

where Eq. 35 is used in the expansion.

$$\begin{aligned} \boldsymbol{t}^{\mathrm{T}}\mathbf{X} &= \boldsymbol{u}^{\mathrm{T}}(\mathbf{I} - \mathbf{T}\mathbf{T}^{+})^{\mathrm{T}}\mathbf{X} \qquad (35) \\ &= \boldsymbol{u}^{\mathrm{T}}(\mathbf{I} - \mathbf{X}\mathbf{X}^{+})\mathbf{X} \\ &= \mathbf{0} \end{aligned}$$

[2] $\boldsymbol{t} \neq 0$ indicates that the new sample $\boldsymbol{x}$ has some component lies in the null space of $\mathbf{T}$, so the insertion of $\boldsymbol{x}$ will increase the rank of $\mathbf{T}$ and $\mathbf{T}^+$ by 1. See [1] for a detailed proof.

Table I
PSEUDO CODE OF THE FIRST UPDATING METHOD

Input: Initial centered data matrix $\mathbf{X}$, its pseudo-inverse $\mathbf{X}^+$, mean matrix $\boldsymbol{m}$, indicator matrix $\mathbf{Y}$, transform matrix $\mathbf{W}$, new sample $\boldsymbol{x}$ and its label p

Output: Updated centered matrix $\tilde{\mathbf{X}}$, its pseudo-inverse $\tilde{\mathbf{X}}^+$, new mean matrix $\tilde{\boldsymbol{m}}$, new indicator matrix $\tilde{\mathbf{Y}}$, new transform matrix $\tilde{\mathbf{W}}$

Process:
```
begin:
  [d, n] = size(X)
  if d ≤ n
    error('Please use the second updating method')
    return
  end
  if p ≤ C
    n_p = n_p + 1
  else
    n_p = 1
    C̃ = C + 1
  end
  define y according to Eq. 12
  update indicator matrix Ỹ according to Eq. 16
  m̃ = m + 1/(n+1) (x − m)
  X̃ = [X − 1/(n+1)(x − m)1^T   n/(n+1)(x − m)]
  calculate X̃^+ according to Eq. 20
  update transform matrix W̃ according to Eq. 27
end
```

Thus, the final updating formula of $\mathbf{W}$ becomes

$$\begin{aligned} \tilde{\mathbf{W}} = &\left(\mathbf{G} - \frac{\tilde{\mathbf{T}}^{+}(\boldsymbol{x}-\boldsymbol{m})\mathbf{1}^{\mathrm{T}}\mathbf{Y}}{n+1}\right)\bar{\otimes}_p\,\alpha_p \\ &+ \frac{n\tilde{\mathbf{T}}^{+}(\boldsymbol{x}-\boldsymbol{m})\boldsymbol{y}^{\mathrm{T}}}{n+1} \ , \end{aligned} \tag{36}$$

where $\mathbf{G}$ and α_p are defined in Eqs. 34 and 14, respectively.

This is the final update formula for the least square solution $\mathbf{W}$. The last term has only one nonzero column, i.e., the p-th column $\frac{n}{n+1}\mathbf{T}^{+}(\boldsymbol{x}-\boldsymbol{m})y_p$, and only this column needs to be added to the p-th column of the matrix before the plus sign.

Pseudo codes of the first and the second updating method are shown in Tables I and II, respectively.

D. Complexity Analysis

The LS-ILDA algorithm has a computational complexity of $O(min(n, d) \times d)$. In both our two updating methods of LS-ILDA, the update of $\mathbf{W}$ and other intermediate matrices can be decomposed into limited number of 'simple operations' if appropriate order for calculation is taken. These simple operations include multiplication between a matrix and a vector, addition between matrices, multiplication and addition between vectors, and multiplication between a vector and a scalar. Note that, there are no multiplication between matrices. When two matrices are added together, the number of basic additions equals the number of elements

Table II
PSEUDO CODE OF THE SECOND UPDATING METHOD

Input: Total scatter matrix $\mathbf{T}$, its pseudo-inverse $\mathbf{T}^+$, its rank r, mean matrix $\boldsymbol{m}$, indicator matrix $\mathbf{Y}$, transform matrix $\mathbf{W}$, new sample $\boldsymbol{x}$ and its label p, dimension of data matrix $[d, n]$

Output: Updated total scatter matrix $\tilde{\mathbf{T}}$, its pseudo-inverse $\tilde{\mathbf{T}}^+$, new rank $\tilde{r}$, new mean matrix $\tilde{\boldsymbol{m}}$, new indicator matrix $\tilde{\mathbf{Y}}$, new transform matrix $\tilde{\mathbf{W}}$, dimension of data matrix $[d, \tilde{n}]$

Process:
begin:
if $d > n$
error('Please use the first updating method')
return
end
if $p \leq \mathcal{C}$
$n_p = n_p + 1$
else
$n_p = 1$
$\tilde{\mathcal{C}} = \mathcal{C} + 1$
end
define $\boldsymbol{y}$ according to Eq. 12
update indicator matrix $\tilde{\mathbf{Y}}$ according to Eq. 16
$\tilde{\boldsymbol{m}} = \boldsymbol{m} + \frac{1}{n+1}(\boldsymbol{x} - \boldsymbol{m})$
if $r == d$
$\boldsymbol{t} = 0$
else
$\boldsymbol{t} = \boldsymbol{u} - \mathbf{T}(\mathbf{T}^+\boldsymbol{u})$
$\tilde{\mathbf{T}} = \mathbf{T} + \frac{n}{n+1}(\boldsymbol{x} - \boldsymbol{m})(\boldsymbol{x} - \boldsymbol{m})^{\mathrm{T}}$
end
calculate $\tilde{\mathbf{T}}^+$ according to Eq. 30
update $\tilde{r}$ according to Eq. 32
update transform matrix $\tilde{\mathbf{W}}$ according to Eq. 36
$\tilde{n} = n + 1$
end

of one matrix. When a matrix multiplies a vector, the number of basic multiplications and the number of basic additions equal the number of elements of the matrix. When a vertical vector multiplies a horizontal vector, the number of basic multiplications also equals the number of elements of the resulting matrix. Therefore, the complexity of these operations is at most $O(n \times d)$ for the first method, and $O(d \times d)$ for the second method. For each update, the number of such operations is constant. Since LS-ILDA always picks up the method with lower complexity, its complexity of one update is $O(min(n, d) \times d)$. The space complexity is also $O(min(n, d) \times d)$, since the matrix $\mathbf{X}$ and $\mathbf{X}^+$ or $\mathbf{T}$ and $\mathbf{T}^+$ have to be kept.

Indeed the first updating method can be substituted by the second one. When $n < d$, the solution $\mathbf{W}$, the scatter matrix $\mathbf{T}$ and its pseudo-inverse $\mathbf{T}^+$ can also be updated as the case when $n \geq d$, yet the computational cost would be much higher.

We also compare the computational complexity of LS-ILDA with that of the GSVD-ILDA [13] algorithm and the ILDA algorithm proposed in [6] (denoted as ILDA07 here). Table III summarizes the time complexity of the three incremental LDA algorithms for a single insertion.

Table III
COMPARISON OF TIME COMPLEXITY OF DIFFERENT ILDA ALGORITHMS

Algorithm	Time Complexity
LS-ILDA	$O(\min(n, d) \times d)$
GSVD-ILDA	$O(dsh + k^3)$
ILDA07	$O(k_T^3 + k_B^3 + dk_T k_B)$

In the complexity expression of GSVD-ILDA, k is the number of major components of the centered data matrix used in updating and h is a value larger than k and smaller than d. The value of k depends on how much the eigen-spectrum energy is used in each step. In the complexity expression of ILDA07, k_T is the number of components kept for total scatter matrix and k_B is the number of components kept for the between-class scatter matrix.

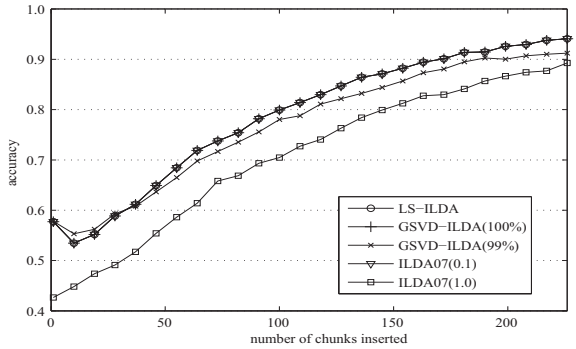
When the rank of the total scatter matrix is nearly full rank, if the performance of GSVD-ILDA and ILDA07 approximate the performance of batch LDA, k and k_T would be close to d and the computational cost would be greatly increased. This is observed in our experiments. LS-ILDA does not have such problem. Without relying on complicated eigen-analysis, LS-ILDA has a smaller magnitude of complexity. Overall, LS-ILDA gives exact least square solution and requires much less computation, which is verified in our experiments.

IV. EMPIRICAL STUDY

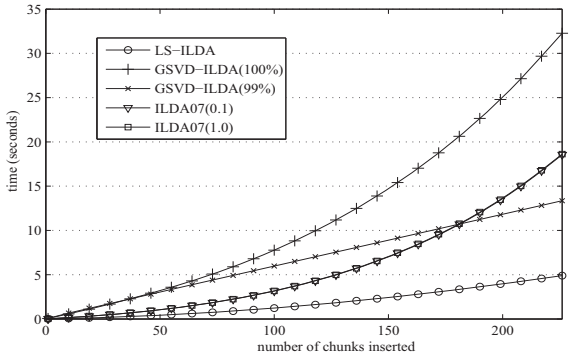
In the experiments we compare our LS-ILDA with GSVD-ILDA [13] and ILDA07 [6]. GSVD-ILDA has a parameter c controlling the percentage of eigen-spectrum energy kept in each update. ILDA07 has two similar parameters, that is, the threshold for significant components of the total scatter matrix and the threshold for significant components of the between-class scatter matrix. In [13], the performance of GSVD-ILDA with different parameter settings had been tested, yet the authors did not specify which setting is the best. The authors of ILDA07 did not give the best parameter setting either. In our experiments, the two parameters of ILDA07 use the same threshold t. We choose two parameter settings for the two algorithms. One setting enables the algorithms to achieve their best performance, while the other setting makes the algorithms more efficient.

Four datasets are used in our experiments. The first is the ORL face database [3], which contains 400 faces of 40 individuals, 10 per individual. In our experiments, we use the version preprocessed by Cai [4]. The image size is 32×32. The second dataset is the Yale face database [5], which contains 165 gray-scale images of 15 individuals. The image size

[3] http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html
[4] http://www.cs.uiuc.edu/homes/dengcai2/Data/FaceData.html
[5] http://cvc.yale.edu/projects/yalefaces/yalefaces.html

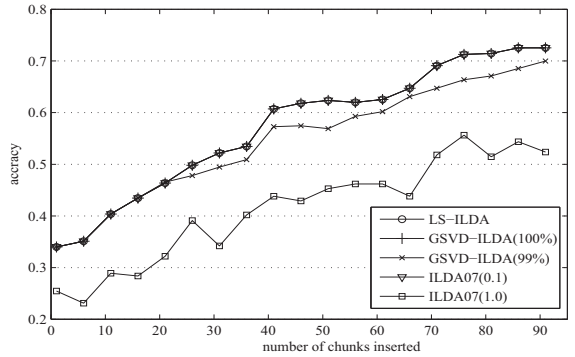


(a) Accuracy

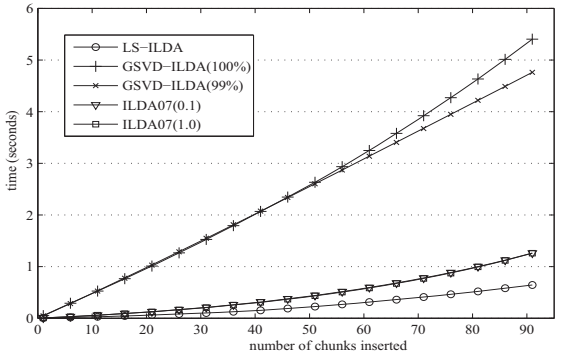


(b) Time (in seconds)

Figure 1. Comparison on the ORL database. The time cost of LS-ILDA is the smallest, while its accuracy is the highest (comparable to GSVD-ILDA [c=100%] and ILDA07 [t=0.1]).

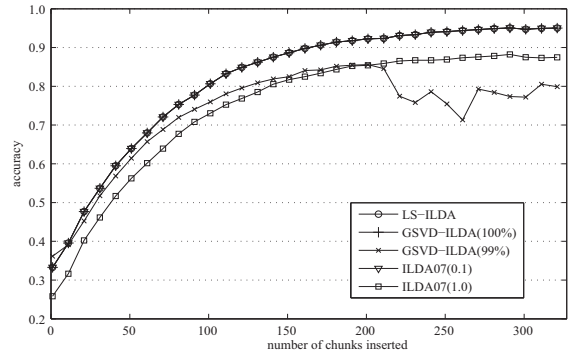


(a) Accuracy

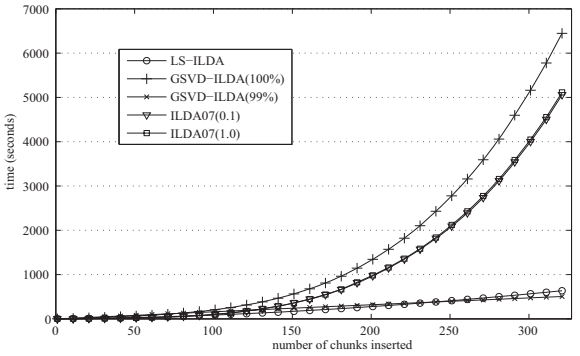


(b) Time (in seconds)

Figure 2. Comparison on the Yale database. The time cost of LS-ILDA is the smallest, while its accuracy is the highest (comparable to GSVD-ILDA [c=100%] and ILDA07 [t=0.1]).

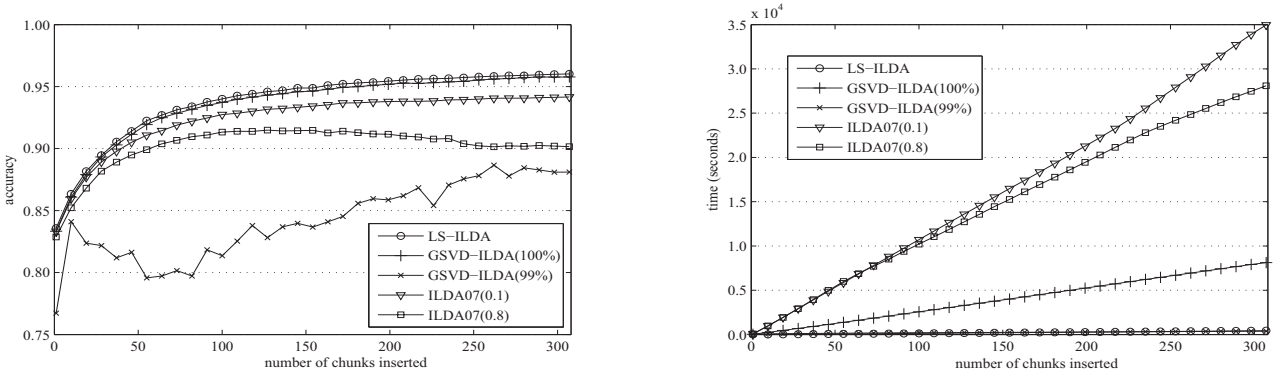


(a) Accuracy



(b) Time (in seconds)

Figure 3. Comparison on the AR database. The time cost of LS-ILDA is among the smallest (comparable to GSVD-ILDA [c=99%]), while its accuracy is the highest (comparable to GSVD-ILDA [c=100%] and ILDA07 [t=0.1]).



(a) Accuracy

(b) Time (in seconds)

Figure 4. Comparison on the PIE database. The time cost of LS-ILDA is the smallest (comparable to GSVD-ILDA [c=99%]), while its accuracy is the highest (comparable to GSVD-ILDA [c=100%]).

is 32×32. The third dataset is the AR database [8]. The version we used contains 2,600 faces of 100 individuals. The image size is 66×48. The fourth dataset is the CMU-PIE database [6]. In our experiments, only five near frontal poses are included. There are overall 11,554 images, 170 per individual except for the 38th individual. These images are resized to 32×32. This is a relatively large dataset.

All the datasets are split by 2:1, with two thirds used for training and the other one third for testing. On each dataset, a proportion of training samples are used as the initial data, and all the other training samples are incrementally inserted in random order. For each algorithm, this process is repeated for ten times, and the average results are recorded.

On ORL, one image per individual from the training data is randomly drawn out as initial data, and the remaining training samples are added chunk by chunk for training in random order. The chunk size is set to 1 and overall there are 226 insertions. For this dataset, c is set to 100% and 99% for GSVD-ILDA, while t is set to 0.1 and 1.0 for ILDA07. Figure 1 shows that LS-ILDA achieves the highest accuracy with the lowest time cost among the compared algorithms. It is also interesting to see that the three highest accuracy curves of the three algorithms superpose each other. The three curves correspond to GSVD-ILDA with $c = 100\%$, ILDA07 with $t = 0.1$ and LS-ILDA. When GSVD-ILDA and ILDA07 include minor components in their updating, they obtain the exact solution of LDA; LS-ILDA give the exact least square solution to LDA. These solutions reach exactly the same accuracy on this dataset. This indicates that the solution given by LS-ILDA is really a solution of LDA. Yet, when GSVD-ILDA and ILDA07 use other parameter settings, their performances become lower.

On Yale, one image per individual from the training data is randomly drawn out as initial data, and the remaining training samples are inserted one by one, with 95 insertions. On this dataset, c for GSVD-ILDA and t for ILDA07 are set to the same settings as that used for ORL. Similar results are obtained on this dataset, as shown in Figure 2. LS-ILDA achieves the highest accuracy with the lowest time cost. The accuracy curve of LS-ILDA is almost as same as that of GSVD-ILDA with $c = 100\%$ and ILDA07 with $t = 0.1$, which verifies that the solution of LS-ILDA is really a solution of LDA.

On AR, one image per individual from the training data is randomly drawn out as initial data, and the remaining training samples are added chunk by chunk for training in random order. In each insertion, the chunk size is set to 5, and there are overall 327 insertions. On this dataset, c for GSVD-ILDA and t for ILDA07 are set to the same settings as that used before. From Figure 3 we can see that the highest accuracy curves of LS-ILDA, GSVD-ILDA with $c = 100\%$ and ILDA07 with $t = 0.1$ are exactly the same, which verifies the goodness of the LS-ILDA solution again. Note that to achieve such high accuracy, LS-ILDA spends far less time cost than GSVD-ILDA with $c = 100\%$ and ILDA07 with $t = 0.1$. Although the time cost of GSVD-ILDA with $c = 99\%$ is comparable with that of LS-ILDA, its performance is much worse than LS-ILDA.

On CMU-PIE, one fifth of the training data are randomly draw out as initial data, and the remaining training samples are added chunk by chunk for training in random order. The chunk size is set to 20. The samples in one chunk is inserted one by one for the LS-ILDA algorithm. On this datset, c for GSVD-ILDA is set to 100% and 99%, and t for ILDA07 is set to 0.1 and 0.8. Figure 4 shows that LS-ILDA achieves the highest accuracy. The overall 6,932 incremental samples are inserted to the training set of LS-ILDA one by one. It is impressive that after 6,932 updates, LS-ILDA can still give very precise solution. GSVD-ILDA with $c = 100\%$ achieves a comparable accuracy, yet its time cost is much higher than that of LS-ILDA. It can be observed from Figure 4 that the

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

time cost of LS-ILDA is the lowest; the time cost of GSVD-ILDA with $c = 99\%$ is comparable yet its accuracy is much worse than that of LS-ILDA. Note that the experimental results show that the time cost of each update does not rely on the number of samples when LS-ILDA using the second updating method.

V. Conclusion

In this paper we propose the LS-ILDA algorithm to solve the incremental LDA problem. LS-ILDA incrementally updates the least square solution to LDA with exactness. The least square solution is the multiplication of pseudo-inverse of the centered data matrix and the indicator matrix [10], which gives us a chance to update the solution without eigen-analysis. The updates of related matrices and the final solution involve only simple matrix operations, such as multiplication between matrix and vector, and multiplication between vector and scalar, and addition between matrices and between vectors. The updates are exact, and do not lead to any theoretical deviation.

To reduce the computational cost, LS-ILDA contains two updating methods. When the number of samples is smaller than the number of dimensions, the data matrix and its pseudo-inverse are kept after each update; if the number of samples is not smaller than the dimension, the pseudo-inverse of the scatter matrix is kept, and the computational cost will not increase with the number of samples. The time complexity of one update in LS-ILDA is $O(min(n, d) \times d)$ when there are n samples in d dimensions.

Experiments on four datasets are performed to compare LS-ILDA with two recently developed incremental LDA algorithms, GSVD-ILDA [13] and ILDA07 [6]. The results show that LS-ILDA achieves the highest accuracy with the lowest time cost. The accuracy of LS-ILDA is as exact as that of the best performance achieved by GSVD-ILDA and ILDA07, which verifies that the solution obtained by LS-ILDA is really the solution to LDA. Moreover, to achieve the highest accuracy, the time cost of LS-ILDA is much smaller than that of the compared methods.

In the future, we will try to theoretically analyze the error bounds of LS-ILDA, and apply LS-ILDA to other application domains.

Acknowledgment

This work was supported by the National Science Foundation of China (60635030, 60721002), the Jiangsu Science Foundation (BK2008018), the Jiangsu 333 High-Level Talent Cultivation Program and the MSRA IST Program. Li-Ping Liu is now working at the Alibaba Co., Beijing, China.

References

[1] J. K. Baksalary, O. M. Baksalary, and G. Trenkler. A revisitation of formulae for the Moore-Penrose inverse of modified matrices. *Linear Algebra and Its Applications*, 372:207–224, 2003.

[2] C. Chatterjee and V. P. Roychowdhury. On self-organizing algorithms and networks for class-separability features. *IEEE Transactions on Neural Networks*, 8(3):663–678, 1997.

[3] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley, New York, NY, 2nd edition, 2000.

[4] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, New York, NY, 2nd edition, 1990.

[5] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, 3rd edition, 1996.

[6] T.-K. Kim, S.-F. Wong, B. Stenger, J. Kittler, and R. Cipolla. Incremental linear discriminant analysis using sufficient spanning set approximations. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Minneapolis, MN, 2007.

[7] J. Mao and A. K. Jain. Artificial neural networks for feature extraction and multivariatedata projection. *IEEE Transactions on Neural Networks*, 6(2):296–317, 1995.

[8] M. Martinez and R. Benavente. The AR face database. Technical Report 24, CVC, 1998.

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

[10] J. Ye. Least squares linear discriminant analysis. In *Proceedings of the 24th International Conference on Machine Learning*, pages 1087–1094, Corvallis, OR, 2007.

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

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

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