Report No. UIUCDCS-R-2007-2857 **UILU-ENG-2007-1758**

SRDA: An Efficient Algorithm for Large Scale Discriminant Analysis

by

Deng Cai, Xiaofei He, and Jiawei Han

May 2007

SRDA: An Efficient Algorithm for Large Scale Discriminant Analysis*

Deng Cai† Xiaofei He‡ Jiawei Han†

† Department of Computer Science, University of Illinois at Urbana-Champaign

‡ Yahoo! Research Labs

Abstract

Linear Discriminant Analysis (LDA) has been a popular method for extracting features which preserve class separability. The projection functions of LDA are commonly obtained by maximizing the between class covariance and simultaneously minimizing the within class covariance. It has been widely used in many fields of information processing, such as machine learning, data mining, information retrieval, and pattern recognition. However, the computation of LDA involves dense matrices eigen-decomposition which can be computationally expensive both in time and memory. Specifically, LDA has $O(mnt + t^3)$ time complexity and requires $O(mn + mt + nt)$ memory, where m is the number of samples, n is the number of features and $t = \min(m, n)$. When both m and n are large, it is infeasible to apply LDA. In this paper, we propose a novel algorithm for discriminant analysis, called *Spectral Regression Discriminant Analysis* (SRDA). By using spectral graph analysis, SRDA casts discriminant analysis into a regression framework which facilitates both efficient computation and the use of regularization techniques. Specifically, SRDA only needs to solve a set of regularized least squares problems and there is no eigenvector computation involved, which is a huge save of both time and memory. Our theoretical analysis shows that SRDA can be computed with $O(ms)$ time and $O(ms)$ memory, where $s(\leq n)$ is the average number of non-zero features in each sample. Extensive experimental results on four real world data sets demonstrate the effectiveness and efficiency of our algorithm.

1 Introduction

Dimensionality reduction has been a key problem in many fields of information processing, such as data mining, information retrieval, and pattern recognition. When data are represented as points in a high-dimensional space, one is often confronted with tasks like nearest neighbor search. Many methods have been proposed to index the data for fast query response, such as K-D tree, R tree, R^* tree, etc [6]. However, these methods can only operate with small dimensionality, typically less than 100. The

*The work was supported in part by the U.S. National Science Foundation NSF IIS-05-13678/06-42771 and NSF BDI-05-15813. Any opinions, findings, and conclusions or recommendations expressed here are those of the authors and do not necessarily reflect the views of the funding agencies.

effectiveness and efficiency of these methods drop exponentially as the dimensionality increases, which is commonly referred to as the "curse of dimensionality".

During the last decade, with the advances in computer technologies and the advent of the World Wide Web, there has been an explosion in the amount of digital data being generated, stored, analyzed, and accessed. Much of this information is multimedia in nature, including text, image, and video data. The multimedia data are typically of very high dimensionality, ranging from several thousands to several hundreds of thousand. Learning in such high dimensionality in many cases is almost infeasible. Thus, learnability necessitates dimensionality reduction. Once the high-dimensional data is mapped into lower-dimensional space, conventional indexing schemes can then be applied.

One of the most popular dimensionality reduction algorithms is Linear Discriminant Analysis (LDA) [3], [5]. LDA searches for the project axes on which the data points of different classes are far from each other while requiring data points of the same class to be close to each other. The optimal transformation (projection) of LDA can be computed by applying an eigen-decomposition on the scatter matrices of the given training data. LDA has been widely used in many applications such as text processing [19], face recognition [1]. However, the scatter matrices are dense and the eigen-decomposition could be very expensive in both time and memory for high dimensional large scale data. Moreover, to get a stable solution of LDA, the scatter matrices are required to be nonsingular which is not true when the number of features is larger than the number of samples. Some additional preprocessing steps (*e.g.*, PCA, SVD) are required to guarantee the non-singularity of scatter matrices [1], [20] which further increase the time and memory cost. Therefor, it is almost infeasible to apply LDA on large scale high dimensional data.

In this paper, we propose a novel algorithm for discriminant analysis, called *Spectral Regression Discriminant Analysis* (SRDA). SRDA is essentially developed from LDA but has significant computational advantage over LDA. Benefit from recent progresses on spectral graph analysis, we analyze LDA from a graph embedding point of view which can be traced back to [10]. We show how the LDA solution can be obtained by solving a set of linear equations which links LDA and classical regression. Our approach combines the spectral graph analysis and regression to provide an efficient and effective approach for discriminant analysis.

The points below highlight the contributions of this paper:

- The classical LDA is well analyzed from a new graph embedding point of view. The singularity issue in classical LDA is clearly analyzed and we show how various kinds of LDA extensions, *e.g.*, two-stage PCA+LDA approach [1] and LDA/GSVD approaches [11][20], can be unified in a SVD+LDA framework.

- The projective functions obtained by those classical LDA approaches and LDA/GSVD approaches are optimal with respect to the objective function. However, in small sample size situation, these solutions tend to over-fit the training data, and thus may not be optimal on the test set. The regularized solution of LDA usually achieves better performance.

- A new approach for discriminant analysis based on the graph embedding formulation of LDA is developed, which is called *Spectral Regression Discriminant Analysis* (SRDA). In SRDA, the transformation vectors are obtained by solving a set of linear regression problems which can be very efficient. Since it contains regression as a building block, SRDA provides a natural framework for *regularized* discriminant analysis.

- LDA has $O(mnt + t^3)$ time complexity and requires $O(mn + mt + nt)$ memory, where m is the number of samples, n is the number of features and $t = \min(m, n)$. When both m and n are large, it is infeasible to apply LDA. On the other hand, SRDA can be computed with $O(ms)$ time and $O(ms)$ memory, where $s(\leq n)$ is the average number of non-zero features in each sample. It can be easily scaled to very large high dimensional data sets.

The remainder of the paper is organized as follows. In Section 2, we provide a brief review of LDA and its variant extensions. Section 3 gives a detailed analysis of LDA from a graph embedding point of view. Section 4 introduces our proposed *Spectral Regression Discriminant Analysis* algorithm. The extensive experimental results are presented in Section 5. Finally, we provide some concluding remarks in Section 6.

2 A Brief Review of LDA

LDA seeks directions on which the data points of different classes are far from each other while requiring data points of the same class to be close to each other. Suppose we have a set of m samples $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m$, belonging to c classes. The objective function of LDA is as follows:

$$\mathbf{a}^* = \arg\max_{\mathbf{a}} \frac{\mathbf{a}^T S_b \mathbf{a}}{\mathbf{a}^T S_w \mathbf{a}}, \tag{1}$$

$$S_b = \sum_{k=1}^{c} m_k (\boldsymbol{\mu}^{(k)} - \boldsymbol{\mu})(\boldsymbol{\mu}^{(k)} - \boldsymbol{\mu})^T, \tag{2}$$

$$S_w = \sum_{k=1}^{c} \left(\sum_{i=1}^{m_k} (\mathbf{x}_i^{(k)} - \boldsymbol{\mu}^{(k)})(\mathbf{x}_i^{(k)} - \boldsymbol{\mu}^{(k)})^T \right), \tag{3}$$

where $\boldsymbol{\mu}$ is the total sample mean vector, m_k is the number of samples in the k-th class, $\boldsymbol{\mu}^{(k)}$ is the average vector of the k-th class, and $\mathbf{x}_i^{(k)}$ is the i-th sample in the k-th class. We call S_w the within-class scatter matrix and S_b the between-class scatter matrix.

Define $S_t = \sum_{i=1}^{m} (\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^T$ as the total scatter matrix and we have $S_t = S_b + S_w$ [5]. The objective function of LDA in Eqn. (1) is equivalent to

$$\mathbf{a}^* = \arg\max_{\mathbf{a}} \frac{\mathbf{a}^T S_b \mathbf{a}}{\mathbf{a}^T S_t \mathbf{a}}. \tag{4}$$

When l projective functions $A = [\mathbf{a}_1, \cdots, \mathbf{a}_l]$ are needed, the objective function of LDA can be written as

$$A^* = \arg\max_{A} \frac{tr(A^T S_b A)}{tr(A^T S_t A)}, \tag{5}$$

Table 1: Notations

Notations	Descriptions
m	the number of total training data points
n	the number of features
c	the number of classes
m_k	the number of data points in k-th class
$\mathbf{x}_i$	the i-th data point
$\mathbf{x}_i^{(k)}$	the i-th data point in the k-th class
$\boldsymbol{\mu}$	the total sample mean vector
$\boldsymbol{\mu}^{(k)}$	the mean vector of the k-th class
$\bar{\mathbf{x}}_i$	the i-th centered data point ($\bar{\mathbf{x}}_i = \mathbf{x}_i - \boldsymbol{\mu}$)
X	the data matrix
$\bar{X}$	the centered data matrix
S_b	the between-class scatter matrix
S_w	the within-class scatter matrix
S_t	the total scatter matrix
$\mathbf{a}$	the transformation vector
A	the transformation matrix

where $tr()$ denotes matrix trace. The optimization problem in Eq. (5) is equivalent to find the l eigenvectors of following generalized eigen-problem associated with maximum eigenvalues:

$$S_b\mathbf{a} = \lambda S_t\mathbf{a}. \tag{6}$$

Since the rank of S_b is bounded by $c-1$, there are at most $c-1$ eigenvectors corresponding to non-zero eigenvalues [5].

To get a stable solution of the above generalized eigen-problem, S_t is required to be nonsingular which is clearly not true when the number of features is larger than the number of samples. In the past few decades, various approaches have been proposed to solve this problem. One of the most well know approaches is to perform dimensionality reduction in two stages. LDA is performed after another stage of dimension reduction. Some popular methods for the first stage include Principle Component Analysis (PCA) and Singular Value Decomposition (SVD). Both Swets *et al.* [18] and Belhumeur *et al.* [1] have utilized PCA+LDA for face recognition. Torkkola [19] implemented SVD+LDA for document classification. All these approaches use the LDA objective function in Eqn. (1). Since the rank of S_w is bounded from above by $m-c$ [1], the PCA (SVD) step should reduce the dimension to at most $m-c$.

Recently, Howland *et al.* [11] solved the singularity problem of LDA by using Generalized Singular Value Decomposition (GSVD). They rewrite the LDA objective function as the following equivalent form:

$$A^* = \arg\max_A tr\left((A^T S_t A)^{-1}(A^T S_b A)\right),$$

which can be solved by the GSVD algorithm. One limitation of this method is the high computational cost of GSVD, especially for large and high-dimensional data sets. In [20], Ye extended such approach by solving the optimization problem using simultaneous diagonalization of the scatter matrices.

Another way to deal with the singularity of S_w is to apply the idea of regularization, by adding some constant values to the diagonal elements of S_w, as $S_w + \alpha I$, for some $\alpha > 0$. It is easy to see that $S_w + \alpha I$ is nonsingular. This approach is called Regularized Discriminant Analysis (RDA) [4], [8]. However, the $S_w + \lambda I$ is a very large dense matrix for high-dimensional data which incurs a high computational cost on directly solving the eigen-problem in Eqn (6). By noticing that the eigen-decomposition of $S_w + \alpha I$ is the sum of eigen-decomposition of S_w and αI, Ye *et al.* [22] developed an efficient algorithm to compute the projective functions of RDA. The computational cost of this approach will be comparable to those two stage PCA+LDA approaches.

The computation of all the above LDA extensions involves the SVD decomposition of the data matrix, which is computationally expensive in both time and memory for high dimensional large scale data sets. In some applications (*e.g.*, text processing), the data matrix is sparse which can be fit into the memory even with a large number of both samples and features. However, the singular vector matrices are dense, thus may not be able to be fit into the memory. In this case, all these LDA approaches can not be applied. To solve this problem, Ye *et al.* proposed a new algorithm called IDR/QR in which QR decomposition is applied rather than SVD [21]. Experiments on some data sets showed that IDR/QR is much more efficient than LDA and achieves comparable performance as LDA [21]. However, there is no theoretical relation between the optimization problem solved by IDR/QR and that of LDA. It is not clear under what situation IDR/QR can achieve similar or even better performance than LDA.

3 Computational Analysis of LDA

In this section, we provide a computational analysis of LDA. Our analysis is based on a graph embedding viewpoint of LDA which can be traced back to [10]. We start from analyzing the between-class scatter matrix S_b.

Let $\bar{\mathbf{x}}_i = \mathbf{x}_i - \boldsymbol{\mu}$ denote the centered data point and $\bar{X}^{(k)} = [\bar{\mathbf{x}}_1^{(k)}, \cdots, \bar{\mathbf{x}}_{m_k}^{(k)}]$ denote the centered data matrix of k-th class. We have

$$\begin{aligned}
S_b =& \sum_{k=1}^{c} m_k (\boldsymbol{\mu}^{(k)} - \boldsymbol{\mu})(\boldsymbol{\mu}^{(k)} - \boldsymbol{\mu})^T \\
=& \sum_{k=1}^{c} m_k \left(\frac{1}{m_k} \sum_{i=1}^{m_k} (\mathbf{x}_i^{(k)} - \boldsymbol{\mu}) \right) \left(\frac{1}{m_k} \sum_{i=1}^{m_k} (\mathbf{x}_i^{(k)} - \boldsymbol{\mu}) \right)^T \\
=& \sum_{k=1}^{c} \frac{1}{m_k} \left(\sum_{i=1}^{m_k} \bar{\mathbf{x}}_i^{(k)} \sum_{i=1}^{m_k} (\bar{\mathbf{x}}_i^{(k)})^T \right) \\
=& \sum_{k=1}^{c} \bar{X}^{(k)} W^{(k)} (\bar{X}^{(k)})^T
\end{aligned}$$

where $W^{(k)}$ is a $m_k \times m_k$ matrix with all the elements equal to $1/m_k$.

Let $\bar{X} = [\bar{X}^{(1)}, \cdots, \bar{X}^{(c)}]$ which is the centered data matrix and define a $m \times m$ matrix W as:

$$W = \begin{bmatrix} W^{(1)} & 0 & \cdots & 0 \\ 0 & W^{(2)} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & W^{(c)} \end{bmatrix} \tag{7}$$

We have

$$S_b = \sum_{k=1}^{c} \bar{X}^{(k)} W^{(k)} (\bar{X}^{(k)})^T = \bar{X} W \bar{X}^T. \tag{8}$$

Since $S_t = \bar{X}\bar{X}^T$, we have

$$S_w = S_t - S_b = \bar{X}(I - W)\bar{X}^T = \bar{X} L \bar{X}^T. \tag{9}$$

If we take the W as the edge weight matrix of a graph $\mathcal{G}$. W_{ij} is the weight of edge joining vertices i and j. $W_{ij} = 0$ indicates there is no edge between vertices i and j. Thus $L = I - W$ is called *graph Laplacian*[1] [2].

We have

$$rank(S_t) = rank(\bar{X}\bar{X}^T) \leq rank(\bar{X}) \leq \min(m-1, n).$$

Since S_t is size of $n \times n$, in the case of $n > m$, S_t is singular and the eigen-problem of LDA in Eqn. (6) can not be stably solved. With the new formulation of S_b, it is clear that we can use SVD to solve this singularity problem.

Suppose $rank(\bar{X}) = r$, the SVD decomposition of $\bar{X}$ is

$$\bar{X} = U \Sigma V^T \tag{10}$$

where $\Sigma = diag(\sigma_1, \cdots, \sigma_r)$ and $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$ are the singular values of $\bar{X}$, $U = [\mathbf{u}_1, \cdots, \mathbf{u}_r] \in \mathbb{R}^{n \times r}$ and $\mathbf{u}_i$'s are called left singular vectors, $V = [\mathbf{v}_1, \cdots, \mathbf{v}_r] \in \mathbb{R}^{m \times r}$ and $\mathbf{v}_i$'s are called right singular vectors. Let $\widetilde{X} = U^T \bar{X} = \Sigma V^T$ and $B = U^T A$, we have

$$\begin{aligned} A^T S_b A &= A^T \bar{X} W \bar{X}^T A = A^T U \Sigma V^T W V \Sigma U^T A \\ &= B^T \widetilde{X} W \widetilde{X}^T B \end{aligned}$$

and

$$\begin{aligned} A^T S_t A &= A^T \bar{X} \bar{X}^T A = A^T U \Sigma V^T V \Sigma U^T A \\ &= B^T \widetilde{X} \widetilde{X}^T B. \end{aligned}$$

Now, the objective function of LDA in (5) can be rewritten as:

$$B^* = \arg\max_{B} \frac{tr(B^T \widetilde{X} W \widetilde{X}^T B)}{tr(B^T \widetilde{X} \widetilde{X}^T B)},$$

[1] A subtlety needs to be addressed here. The graph Laplacian is actually defined as $L = D - W$, where D is a diagonal matrix with its (i,i)-element equals to the sum of the i-th column (or row, since W is symmetric) of W. With the W defined in Eqn. (7), we can easily see $D = I$.

and the columns of B^* are the eigenvectors of the following generalized eigen-problem associated with the non-zero eigenvalues:

$$\widetilde{X}W\widetilde{X}^T\mathbf{b} = \lambda\widetilde{X}\widetilde{X}^T\mathbf{b}. \tag{11}$$

Since $\widetilde{X}\widetilde{X}^T = \Sigma V^T(\Sigma V^T)^T = \Sigma^2$, the above eigen-problem can be stably solved. After we get B^*, the A^* can be obtained by

$$A^* = UB^* \tag{12}$$

Since $\bar{X}$ has zero mean, the SVD of $\bar{X}$ is exactly the same as the PCA of $\bar{X}$, and therefore the same as the PCA of X. Our analysis here justifies the rationale behind two-stage PCA+LDA approach. The Fisherface approach [1] keeps at most $m - c$ dimension in the PCA step to make S_w nonsingular, thus may lose some useful information. Our analysis shows that based on the modified but equivalent LDA objective function in Eqn. (4), we can keep all the non-zero eigenvalues in the PCA step which avoids information loss.

By using this transformation matrix A^*, the features in the reduced space are uncorrelated to each other. We have the following theorem:

Theorem 1 *Let A be the transformation matrix of LDA calculated in Eq. (12). The original feature vectors X is transformed into $Y = A^TX$, where the i-th feature component of Y (i-th row of Y) is denoted as $\boldsymbol{y}_i^T$, $\boldsymbol{y}_i = X^T\boldsymbol{a}_i$. Thus, $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$ are uncorrelated, for any $i \neq j$.*

Proof Let $\nu_i = \text{mean}(\mathbf{y}_i) = \boldsymbol{\mu}^T\mathbf{a}_i$ and $\mathbf{e}$ be the vector of all ones, it is sufficient to prove $(\mathbf{y}_i - \mathbf{e}\nu_i)^T(\mathbf{y}_j - \mathbf{e}\nu_j) = 0$, for $i \neq j$. We have

$$\begin{aligned}
& (\mathbf{y}_i - \mathbf{e}\nu_i)^T(\mathbf{y}_j - \mathbf{e}\nu_j) \\
= \; & (X^T\mathbf{a}_i - \mathbf{e}\boldsymbol{\mu}^T\mathbf{a}_i)^T(X^T\mathbf{a}_j - \mathbf{e}\boldsymbol{\mu}^T\mathbf{a}_j) \\
= \; & (\bar{X}^T\mathbf{a}_i)^T(\bar{X}^T\mathbf{a}_j) \\
= \; & \mathbf{a}_i^T\bar{X}\bar{X}^T\mathbf{a}_j \\
= \; & \mathbf{b}_i^T\widetilde{X}\widetilde{X}^T\mathbf{b}_j = 0, \quad (i \neq j)
\end{aligned}$$

The last equation holds since $\mathbf{b}_i$'s are eigenvectors of eigen-problem (11) [7].

In this sense, this SVD+LDA approach described above can also be called Uncorrelated LDA (ULDA) [20].

3.1 Computational Complexity of LDA

Now let us analyze the computational complexities of LDA. The main computation of LDA is solve the generalized eigen-problem:

$$\bar{X}W\bar{X}^T\mathbf{a} = \lambda\bar{X}\bar{X}^T\mathbf{a}. \tag{13}$$

Suppose we have the SVD decomposition of $\bar{X}$ shown in Eqn. (10), we have

$$\begin{aligned}
&\bar{X}W\bar{X}^T\mathbf{a} = \lambda\bar{X}\bar{X}^T\mathbf{a}\\
\Rightarrow &U\Sigma V^TWV\Sigma U^T\mathbf{a} = \lambda U\Sigma\Sigma U^T\mathbf{a}\\
\Rightarrow &\Sigma^{-1}U^TU\Sigma V^TWV\Big(\Sigma U^T\mathbf{a}\Big) = \lambda\Sigma^{-1}U^TU\Sigma\Big(\Sigma U^T\mathbf{a}\Big)\\
\Rightarrow &V^TWV\mathbf{b} = \lambda\mathbf{b}
\end{aligned}$$

where $\mathbf{b} = \Sigma U^T\mathbf{a}$ and $V \in \mathbb{R}^{m\times r}$ is right singular matrix of $\bar{X}$. The above algebraic steps show that the LDA projective functions can be obtained through the following three steps:

1. SVD decomposition of $\bar{X}$ to get U, V and Σ.

2. Computing $\mathbf{b}$'s, the eigenvectors of V^TWV.

3. Computing $\mathbf{a} = U\Sigma^{-1}\mathbf{b}$.

Since there are at most $c-1$ projective functions in LDA, we do not need to compute all the eigenvectors of V^TWV. The following trick can be used to save computational cost. We denote the i-th row vector of V as $\mathbf{z}_i$, which corresponds to the data point $\mathbf{x}_i$. Let $\mathbf{z}_i^{(k)}$ denote the row vector of V which corresponds to $\mathbf{x}_i^{(k)}$. Define $\boldsymbol{\nu}^{(k)} = \frac{1}{l_k}\sum_{i=1}^{l_k}\mathbf{z}_i^{(k)}$ and $H = [\sqrt{l_1}\boldsymbol{\nu}^{(1)}, \cdots, \sqrt{l_c}\boldsymbol{\nu}^{(c)}] \in \mathbb{R}^{d\times c}$. We have

$$\begin{aligned}
V^TWV =&\sum_{k=1}^{c}\frac{1}{l_k}\left(\sum_{i=1}^{l_k}\mathbf{z}_i^{(k)}\sum_{i=1}^{l_k}(\mathbf{z}_i^{(k)})^T\right)\\
=&\sum_{k=1}^{c} l_k\ \boldsymbol{\nu}^{(k)}(\boldsymbol{\nu}^{(k)})^T\\
=&HH^T
\end{aligned}\tag{14}$$

It is easy to check that the left singular vectors of $\bar{X}$ (column vectors of U) are the eigenvectors of $\bar{X}\bar{X}^T$ and the right singular vectors of $\bar{X}$ (column vectors of V) are the eigenvectors of $\bar{X}^T\bar{X}$ [17]. Moreover, if U or V is given, then we can recover the other via the formula $\bar{X}V = U\Sigma$ and $U^T\bar{X} = \Sigma V^T$. In fact, the most efficient SVD decomposition algorithm (i.e. *cross-product*) applies this strategy [17]. Specifically, if $m \geq n$, we compute the eigenvectors of $\bar{X}\bar{X}^T$, which gives us U and can be used to recover V; If $m < n$, we compute the eigenvectors of $\bar{X}^T\bar{X}$, which gives us V and can be used to recover U. Since the matrix H is of size $r \times c$, where r is the rank of X and c is the number of classes. In most of the cases, r is close to $\min(m,n)$ which is far larger than c. Thus, comparing to directly calculate the eigenvectors of HH^T, compute the eigenvectors of H^TH then recover the eigenvectors of HH^T can achieve a significant saving. The computational approach described here is exactly identical to the ULDA approach in [20].

We use the term *flam* [16], a compound operation consisting of one addition and one multiplication, to measure the operation counts. When $m \geq n$, the calculation of $\bar{X}\bar{X}^T$ requires $\frac{1}{2}mn^2$ flam; Computing the eigenvectors of $\bar{X}\bar{X}^T$ requires $\frac{9}{2}n^3$ flam [17, 7]; Recovering V from U requires mn^2 flam by assuming r is close to $\min(m,n)$; Computing the c eigenvectors of HH^T requires $\frac{1}{2}nc^2 + \frac{9}{2}c^3 + nc^2$ flam; Finally, calculating $\mathbf{a}$'s from $\mathbf{b}$'s requiring n^2c. When $m < n$, we have the similar analysis. We conclude that the time complexity of LDA measured by flam is

$$\frac{3}{2}mnt + \frac{9}{2}t^3 + \frac{3}{2}tc^2 + \frac{9}{2}c^3 + t^2c$$

where $t = \min(m, n)$. Considering $c \ll t$, the time complexity of LDA can be written as $\frac{3}{2}mnt + \frac{9}{2}t^3 + O(t^2)$.

For the memory requirement, we need to store $\bar{X}$, U, V and $\mathbf{a}$'s. All sum together is

$$mn + nt + mt + cn$$

It is clear that LDA has cubic-time complexity with respect to $\min(m, n)$ and the memory requirement is $O(mn)$. When both m and n are large, it is not feasible to apply LDA. In the next section, we will show how to solve this problem with the new formulation of S_b.

4 Spectral Regression Discriminant Analysis

In order to solve the LDA eigen-problem in Eqn. (13) efficiently, we use the following theorem:

Theorem 2 *Let $\bar{\boldsymbol{y}}$ be the eigenvector of eigen-problem*

$$W\bar{\boldsymbol{y}} = \lambda\bar{\boldsymbol{y}} \tag{15}$$

with eigenvalue λ. If $\bar{X}^T\boldsymbol{a} = \bar{\boldsymbol{y}}$, then $\boldsymbol{a}$ is the eigenvector of eigen-problem in Eqn. (13) with the same eigenvalue λ.

Proof We have $W\bar{\mathbf{y}} = \lambda\bar{\mathbf{y}}$. At the left side of Eqn. (13), replace $\bar{X}^T\mathbf{a}$ by $\bar{\mathbf{y}}$, we have

$$\bar{X}W\bar{X}^T\mathbf{a} = \bar{X}W\bar{\mathbf{y}} = \bar{X}\lambda\bar{\mathbf{y}} = \lambda\bar{X}\bar{\mathbf{y}} = \lambda\bar{X}\bar{X}^T\mathbf{a}$$

Thus, $\mathbf{a}$ is the eigenvector of eigen-problem Eqn. (15) with the same eigenvalue λ.

Theorem 2 shows that instead of solving the eigen-problem Eqn. (13), the LDA basis functions can be obtained through two steps:

1. Solve the eigen-problem in Eqn. (15) to get $\bar{\mathbf{y}}$.

2. Find $\mathbf{a}$ which satisfies $\bar{X}^T\mathbf{a} = \bar{\mathbf{y}}$. In reality, such $\mathbf{a}$ may not exist. A possible way is to find $\mathbf{a}$ which can best fit the equation in the least squares sense:

$$\mathbf{a} = \arg\min_{\mathbf{a}} \sum_{i=1}^{m} (\mathbf{a}^T\bar{\mathbf{x}}_i - \bar{y}_i)^2 \tag{16}$$

where $\bar{y}_i$ is the i-th element of $\bar{\mathbf{y}}$.

The advantages of this two-step approach are as follows:

1. We will show later how the eigen-problem in Eqn. (15) is *trivial* and we can directly get those eigenvectors $\bar{\mathbf{y}}$.

2. Comparing to all the other LDA extensions, there is no dense matrix eigen-decomposition or SVD decomposition involved. The technique to solve the least squares problem is already matured [7] and there exist many efficient iterative algorithms (*e.g.*, LSQR [14]) that can handle very large scale least squares problems. Therefor, the two-step approach can be easily scaled to large data sets.

In the situation that the number of samples is smaller than the number of features, the minimization problem (16) is *ill posed*. We may have infinite many solutions for the linear equations system $\bar{X}^T\mathbf{a} = \bar{\mathbf{y}}$ (the system is underdetermined). The most popular way to solve this problem is to impose a penalty on the norm of $\mathbf{a}$:

$$\mathbf{a} = \arg\min_{\mathbf{a}} \left(\sum_{i=1}^{m} \left(\mathbf{a}^T\bar{\mathbf{x}}_i - \bar{y}_i\right)^2 + \alpha\|\mathbf{a}\|^2 \right) \tag{17}$$

This is so called regularization and is well studied in statistics. The regularized least squares is also called ridge regression [9]. The $\alpha \geq 0$ is a parameter to control the amounts of shrinkage. Now we can see the third advantage of the two-step approach:

3 Since the regression was used as a building block, the regularization techniques can be easily incorporated and produce more stable and meaningful solutions, especially when there exist a large amount of features [9].

Now let us analyze the eigenvectors of W which is defined in Eqn. (7). The W is block-diagonal, thus, its eigenvalues and eigenvectors are the union of the eigenvalues and eigenvectors of its blocks (the latter padded appropriately with zeros). It is straightforward to show that $W^{(k)}$ has eigenvector $\mathbf{e}^{(k)} \in \mathbb{R}^{m_k}$ associated with eigenvalue 1, where $\mathbf{e}^{(k)} = [1, 1, \cdots, 1]^T$. Also there is only one non-zero eigenvalue of $W^{(k)}$ because the rank of $W^{(k)}$ is 1. Thus, there are exactly c eigenvectors of W with the same eigenvalue 1. These eigenvectors are

$$\mathbf{y}_k = [\,\underbrace{0, \cdots, 0}_{\sum_{i=1}^{k-1} m_i}, \underbrace{1, \cdots, 1}_{m_k}, \underbrace{0, \cdots, 0}_{\sum_{i=k+1}^{c} m_i}\,]^T \quad k = 1, \cdots, c \tag{18}$$

Since 1 is a repeated eigenvalue of W, we could just pick any other c orthogonal vectors in the space spanned by $\{\mathbf{y}_k\}$, and define them to be our c eigenvectors. Notice that, in order to guarantee there exists a vector $\mathbf{a}$ which satisfies the linear equations system $\bar{X}^T\mathbf{a} = \mathbf{y}$, $\mathbf{y}$ should be in the space spanned by the row vectors of $\bar{X}$. Since $\bar{X}\mathbf{e} = 0$, the vector of all ones $\mathbf{e}$ is orthogonal to this space. On the other hand, we can easily see that $\mathbf{e}$ is naturally in the space spanned by $\{\mathbf{y}_k\}$ in Eqn. (18). Therefor, we pick $\mathbf{e}$ as our first eigenvector of W and use Gram-Schmidt process to orthogonalize the remaining eigenvectors. The vector $\mathbf{e}$ can then be removed, which leaves us exactly $c-1$ eigenvectors of W, we denote them as follows:

$$\{\bar{\mathbf{y}}_k\}_{k=1}^{c-1}, \;(\bar{\mathbf{y}}_i^T\mathbf{e} = 0, \;\; \bar{\mathbf{y}}_i^T\bar{\mathbf{y}}_j = 0, \; i \neq j) \tag{19}$$

The two-step approach essentially combines the spectral analysis of the graph matrix W and regression techniques. Therefor, we named this new approach as *Spectral Regression Discriminant Analysis* (SRDA). In the following several subsections, we will provide the theoretical and computational analysis on SRDA and give the detailed algorithmic procedure.

4.1 Theoretical Analysis

In the following discussions, $\bar{\mathbf{y}}$ is one of the eigenvectors in Eqn. (19).

The regularized least squares problem of SRDA in Eqn. (17) can be rewritten in matrix form as:

$$\mathbf{a} = \arg\min_{\mathbf{a}} \left(\left(\bar{X}^T\mathbf{a} - \bar{\mathbf{y}}\right)^T \left(\bar{X}^T\mathbf{a} - \bar{\mathbf{y}}\right) + \alpha \mathbf{a}^T\mathbf{a} \right). \tag{20}$$

Requiring the derivative of right side with respect to $\mathbf{a}$ vanish, we get

$$\begin{aligned} \left(\bar{X}\bar{X}^T + \alpha I\right)\mathbf{a} &= \bar{X}\bar{\mathbf{y}} \\ \Rightarrow \ \mathbf{a} &= \left(\bar{X}\bar{X}^T + \alpha I\right)^{-1}\bar{X}\bar{\mathbf{y}} \end{aligned} \tag{21}$$

When $\alpha > 0$, this regularized solution will not satisfy the linear equations system $\bar{X}^T\mathbf{a} = \bar{\mathbf{y}}$ and $\mathbf{a}$ is also not the eigenvector of the LDA eign-problem in Eqn. (13). It is interesting and important to see the relationship between the projective function of ordinary LDA and SRDA. Specifically, we have the following theorem:

Theorem 3 *If $\bar{\mathbf{y}}$ is in the space spanned by row vectors of $\bar{X}$, the corresponding projective function $\mathbf{a}$ calculated in SRDA will be the eigenvector of eigen-problem in Eqn. (13) as α deceases to zero. Therefor, $\mathbf{a}$ will be one of the projective function of LDA.*

Proof See Appendix A.

When the number of features is larger than the number of samples, the sample vectors are usually linearly independent, *i.e.*, $rank(X) = m$. In this case, we have a stronger conclusion which is shown in the following corollary.

Corollary 4 *If the sample vectors are linearly independent,* i.e., *$rank(X) = m$, all the $c-1$ projective functions in SRDA will be identical to those of ULDA described in Section 3 as α deceases to zero.*

Proof See Appendix B.

It is easy to check that the values of the i-th and j-th entries of any vector $\mathbf{y}$ in the space spanned by $\{\mathbf{y}_k\}$ in Eqn. (18) are the same as long as $\mathbf{x}_i$ and $\mathbf{x}_j$ belong to the same class. Thus the i-th and j-th rows of $\bar{Y}$ are the same, where $\bar{Y} = [\bar{\mathbf{y}}_1, \cdots, \bar{\mathbf{y}}_{c-1}]$. Corollary (4) shows that when the sample vectors are linearly independent, the $c-1$ projective functions of LDA are exactly the solutions of the $c-1$ linear equations systems $\bar{X}^T\mathbf{a}_k = \bar{\mathbf{y}}_k$. Let $A = [\mathbf{a}_1, \cdots, \mathbf{a}_{c-1}]$ be the LDA transformation matrix which embeds the data points into the LDA subspace as:

$$A^T X = A^T(\bar{X} + \boldsymbol{\mu}\mathbf{e}^T) = \bar{Y}^T + A^T\boldsymbol{\mu}\mathbf{e}^T.$$

The columns of matrix $\bar{Y}^T + A^T\boldsymbol{\mu}\mathbf{e}^T$ are the embedding results of samples in the LDA subspace. Thus, the data points with the same label are corresponding to the same point in the LDA subspace when the sample vectors are linearly independent.

These projective functions are optimal in the sense of separating training samples with different labels. However, they usually overfit the training set thus may not be able to perform well for the test samples, thus the regularization is necessary.

4.2 The Algorithmic Procedure

Notice that, we need first to calculate the centered data matrix $\bar{X}$ in the algorithm. In some applications (*e.g.*, text processing), the data matrix is sparse which can be fit into the memory even with a large number of both samples and features. However, the center data matrix is dense, thus may not be able to be fit into the memory. Before we give the detailed algorithmic procedure of SRDA, we present a trick to avoid the center data matrix calculation first.

We have:

$$\begin{aligned} & \arg\min_{\mathbf{a}} \sum_{i=1}^{m} (\mathbf{a}^T\bar{\mathbf{x}}_i - \bar{y}_i)^2 \\ = \quad & \arg\min_{\mathbf{a}} \sum_{i=1}^{m} (\mathbf{a}^T\mathbf{x}_i - \mathbf{a}^T\boldsymbol{\mu} - \bar{y}_i)^2 \end{aligned}$$

If we append a new element "1" to each $\mathbf{x}_i$, the scalar $\mathbf{a}^T\boldsymbol{\mu}$ can be absorbed into $\mathbf{a}$ and we have

$$\arg\min_{\mathbf{a}'} \sum_{i=1}^{m} ((\mathbf{a}')^T\mathbf{x}'_i - \bar{y}_i)^2$$

where both $\mathbf{a}'$ and $\mathbf{x}'_i$ are $(n+1)$-dimensional vectors. By using this trick, we can avoid the computation of centered data matrix which can save the memory a lot for sparse data processing.

Given a set of data points $\mathbf{x}_1, \cdots, \mathbf{x}_m \in \mathbf{R}^n$ which belong to c classes. Let m_k denote the number of samples in the k-th class ($\sum_{k=1}^{c} m_k = m$). The algorithmic procedure of SRDA is as follows.

1. **Responses generation**: Let

$$\mathbf{y}_k = [\underbrace{0, \cdots, 0}_{\sum_{i=1}^{k-1} m_i}, \underbrace{1, \cdots, 1}_{m_k}, \underbrace{0, \cdots, 0}_{\sum_{i=k+1}^{c} m_i}]^T \quad k = 1, \cdots, c$$

and $\mathbf{y}_0 = [1, 1, \cdots, 1]^T$ denotes a vector of all ones. Take $\mathbf{y}_0$ as the first vector and use Gram-Schmidt process to orthogonize $\{\mathbf{y}_k\}$. Since $\mathbf{y}_0$ is in the subspace spanned by $\{\mathbf{y}_k\}$, we will obtain $c-1$ vectors

$$\{\bar{\mathbf{y}}_k\}_{k=1}^{c-1}, \; (\bar{\mathbf{y}}_i^T\mathbf{y}_0 = 0, \;\; \bar{\mathbf{y}}_i^T\bar{\mathbf{y}}_j = 0, \; i \neq j)$$

2. **Regularized least squares**: Append a new element "1" to each $\mathbf{x}_i$ which will be still denoted as $\mathbf{x}_i$ for simplicity. Find $c-1$ vectors $\{\mathbf{a}_k\}_{k=1}^{c-1} \in \mathbb{R}^{n+1}$, where $\mathbf{a}_k$ is the solution of regularized least squares problem:
$$\mathbf{a}_k = \arg\min_{\mathbf{a}} \left(\sum_{i=1}^{m} (\mathbf{a}^T \mathbf{x}_i - \bar{y}_i^k)^2 + \alpha \|\mathbf{a}\|^2 \right) \tag{22}$$
where $\bar{y}_i^k$ is the i-th element of $\bar{\mathbf{y}}_k$.

3. **Embedding to $c-1$ dimensional subspace**: The $c-1$ vectors $\{\mathbf{a}_k\}$ are the basis vectors of SRDA. Let $A = [\mathbf{a}_1, \cdots, \mathbf{a}_{c-1}]$ which is a $(n+1) \times (c-1)$ transformation matrix. The samples can be embedded into $c-1$ dimensional subspace by
$$\mathbf{x} \rightarrow \mathbf{z} = A^T \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}$$

4.3 Computational Complexity Analysis

In this section, we provide a computational complexity analysis of SRDA. Our analysis considers both time complexity and memory cost. The term *flam*, a compound operation consisting of one addition and one multiplication, is used for presenting operation counts [16].

The computation of SRDA involves two steps: responses generation and regularized least squares. The cost of the first step is mainly the cost of Gram-Schmidt method, which requires $(mc^2 - \frac{1}{3}c^3)$ flam and $mc + c^2$ memory [16].

We have two ways to solve the $c-1$ regularized least squares problems in Eqn. (22):

- Differentiate the residual sum of squares with respect to components of $\mathbf{a}$ and set the results to zero, which is the textbook way to minimize a function. The result is a linear system called the *normal equations* [16], as shown in Eqn. (21)

- Use iterative algorithm LSQR [14].

These two approaches have different complexity and we provide the analysis below separately.

4.3.1 Solving Normal Equations

As shown in Eqn. (21), the normal equations of regularized least squares problem in Eqn (22) are
$$(XX^T + \alpha I)\mathbf{a}_k = X\bar{\mathbf{y}}_k \tag{23}$$

The calculation of XX^T requires $\frac{1}{2}mn^2$ flam and the calculation of $c-1$ $X\bar{\mathbf{y}}_k$ requires cmn flam. Since the matrix $XX^T + \alpha I$ is positive definite, it can be factored uniquely in the form $XX^T + \alpha I = R^T R$, where R is upper triangular with positive diagonal elements. This is so called Cholesky decomposition and it requires $\frac{1}{6}n^3$ flam [16]. With this Cholesky decomposition, the $c-1$ linear equations can be solved within cn^2 flam [16]. Thus, the computational cost of solving regularized least squares by normal equations is

$$\frac{1}{2}mn^2 + cmn + \frac{1}{6}n^3 + cn^2.$$

When $n > m$, we can further decrease the cost. In the proof of Theorem 3, we used the concept of pseudo inverse of a matrix [15], which is denoted as $(\cdot)^+$. We have [15]:

$$X^+ = \lim_{\alpha \to 0}(X^T X + \alpha I)^{-1} X^T = \lim_{\alpha \to 0} X(XX^T + \alpha I)^{-1}.$$

Thus, the normal equations in Eqn. (23) can be solve by solving the following two linear equations system when α decreasing to zero:

$$\begin{aligned} (X^T X + \alpha I)\mathbf{c}_k &= \bar{\mathbf{y}}_k \\ \mathbf{a}_k &= X\mathbf{c}_k \end{aligned} \qquad (24)$$

The cost of solving $c-1$ linear equations system in Eqn. (24) is

$$\frac{1}{2}nm^2 + \frac{1}{6}m^3 + cm^2 + cmn.$$

Finally, the time cost of SRDA (including the responses generation step) by solving normal equations is:

$$mc^2 - \frac{1}{3}c^3 + \frac{1}{2}mnt + cmn + \frac{1}{6}t^3 + ct^2.$$

where $t = \min(m, n)$. Considering $c \ll t$, this time complexity can be written as $\frac{1}{2}mnt + \frac{1}{6}t^3 + O(t^2) + O(mn)$.

We also need to store X, XX^T (or $X^T X$), $\mathbf{y}_k$ and the solutions $\mathbf{a}_k$. Thus, the memory cost of SRDA by solving normal equations is:

$$mn + t^2 + mc + nc$$

4.3.2 Iterative Solution with LSQR

The LSQR is an iterative algorithm designed to solve large scale sparse linear equations and least squares problems [14]. In each iteration, LSQR needs to compute two matrix-vector products in the form of $X\mathbf{p}$ and $X^T\mathbf{q}$. The remaining work load of LSQR in each iteration is $3m + 5n$ flam [13]. Thus, the time cost of LSQR in each iteration is $2mn + 3m + 5n$. If LSQR stops after k iterations, the total time cost is $k(2mn + 3m + 5n)$. LSRQ converges very fast [14]. In our experiments, 20 iterations are enough. Since we need to solve $c - 1$ least squares problems, the time cost of SRDA with LSQR is

$$k(c-1)(2mn + 3m + 5n),$$

which can be simplified as $2kcmn + O(m) + O(n)$.

Besides storing X, LSQR needs $m + 2n$ memory [13]. We need to store the $\mathbf{a}_k$. Thus, the memory cost of SRDA with LSQR is:

$$mn + m + 2n + cn.$$

Table 2: Computational complexity of LDA and SRDA

Algorithm			operation counts (*flam* [16])	memory
LDA			$\frac{3}{2}mnt + \frac{9}{2}t^3$	$mn + nt + mt$
SRDA	Solving normal equations		$\frac{1}{2}mnt + \frac{1}{6}t^3$	$mn + t^2$
	Iterative solution with LSQR	dense	$2kcmn$	mn
		sparse	$2kcms + 5kcn$	$ms + (2 + c)n$

m: the number of data samples $\quad$ n: the number of features

t: $\min(m, n)$ $\quad$ c: the number of classes

k: the number of iterations in LSQR

s: the average number of non-zero features for one sample

which can be simplified as $mn + O(m) + O(n)$.

When the data matrix is sparse, the above computational cost can be further reduced. Suppose each sample has around only $s \ll n$ non-zero features, the time cost of SRDA with LSQR is $2kcsm + 5kcn + O(m)$ and the memory cost is $sm + (2 + c)n + O(m)$.

4.3.3 Summary

We summarize our complexity analysis results in Table 2, together with the complexity results of LDA. For simplicity, we only show the dominant part of the time and memory costs. The main conclusions include:

- SRDA (by solving normal equations) is always faster than LDA. It is easy to check that when $m = n$, we get the maximum speedup, which is 9.
- LDA has cubic-time complexity with respect to $\min(m, n)$. When both m and n are large, it is not feasible to apply LDA. SRDA (iterative solution with LSQR) has linear-time complexity with both m and n. It can be easily scaled to high dimensional large data sets.
- In many high dimensional data processing tasks *e.g.*, text processing, the data matrix is sparse. However, LDA needs to calculate centered data matrix $\bar{X}$ which is dense. Moreover, the left and right singular matrices are also dense. When both m and n are large, the memory limit will restricts the ordinary LDA algorithms (*e.g.*, PCA+LDA, ULDA, RLDA) to be applied.
- On the other hand, SRDA (iterative solution with LSQR) can fully explore the sparseness of the data matrix and gain significant computational saving on both time and memory. SRDA can successfully applied as long as the data matrix X can be fit into the memory.
- Even the data matrix X is too large to be fit into the memory, SRDA can still be applied with some reasonable disk I/O. This is because in each iteration of LSQR, we only need to calculate two

Table 3: Statistics of the four data sets

dataset	size (m)	dimensionality (n)	# of classes (c)
PIE	11560	1024	68
Isolet	6237	617	26
MNIST	4000	784	10
20Newsgroup	18941	26214	20

matrix-vector products in the form of $X\mathbf{p}$ and $X^T\mathbf{q}$, which can be easily implemented with X and X^T stored on the disk.

5 Experimental Results

In this section, we investigate the performance of our proposed SRDA algorithm for classification. All of our experiments have been performed on a P4 3.20GHz Windows XP machines with 2GB memory.

5.1 Datasets

Four datasets are used in our experimental study, including face, handwritten digit, spoken letter and text databases. The important statistics of these datasets are summarized below (see also Table 3):

- The CMU PIE face database[2] contains 68 subjects with 41,368 face images as a whole. The face images were captured under varying pose, illumination and expression. We choose the five near frontal poses (C05, C07, C09, C27, C29) and use all the images under different illuminations and expressions, thus we get 170 images for each individual. All the face images are manually aligned and cropped. The cropped images are 32×32 pixels, with 256 gray levels per pixel. The features (pixel values) are then scaled to [0,1] (divided by 256). For each individual, $l(= 10, 20, 30, 40, 50, 60)$ images are randomly selected for training and the rest are used for testing.

- The Isolet spoken letter recognition database[3] contains 150 subjects who spoke the name of each letter of the alphabet twice. The speakers are grouped into sets of 30 speakers each, and are referred to as isolet1 through isolet5. For the purposes of this experiment, we chose isolet 1&2 which contain 3120 examples (120 examples per class) as the training set, and test on isolet 4&5 which contains 3117 examples (3 example is missing due to the difficulties in recording). A random subset with $l(= 20, 30, 50, 70, 90, 110)$ examples per letter from the isolet 1&2 were selected for training.

- The MNIST handwritten digit database[4] has a training set of 60,000 samples (denoted as set A), and a testing set of 10,000 samples (denoted as set B). In our experiment, we take the first 2,000 samples from the set A as our training set and the first 2,000 samples from the set B as our test

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

[3] http://www.ics.uci.edu/~mlearn/MLSummary.html

[4] http://yann.lecun.com/exdb/mnist/

set. Each digit image is of size 28 × 28 and there are around 200 samples of each digit in both training and test sets. A random subset with $l(= 30, 50, 70, 100, 130, 170)$ samples per digit from training set are selected for training.

- The popular 20 Newsgroups[5] is a data set collected and originally used for document classification by Lang [12]. The "bydate" version is used in our experiment. The duplicates and newsgroup-identifying headers are removed which leaves us 18,941 documents, evenly distributed across 20 classes. This corpus contains 26,214 distinct terms after stemming and stop word removal. Each document is then represented as a term-frequency vector and normalized to 1. A random subset with $l(= 5\%, 10\%, 20\%, 30\%, 40\%, 50\%)$ samples per category are selected for training and the rest are used for testing.

The first three data sets have relatively smaller numbers of features and the data matrices are dense. The last data set has a very large number of features and the data matrix is sparse.

5.2 Compared algorithms

Four algorithms which are compared in our experiments are listed below:

1. Uncorrelated LDA (ULDA)[20], which was also analyzed in Section 3.
2. Regularized LDA (RLDA) [4]. Solving the singularity problem by adding some constant values to the diagonal elements of S_w, as $S_w + \alpha I$, for some $\alpha > 0$. In [22], Ye *et al.* proposed an efficient algorithm to calculate the solution of RLDA.
3. Spectral Regression Discriminant Analysis (SRDA), our approach proposed in this paper.
4. IDR/QR [21], a LDA variation in which QR decomposition is applied rather than SVD. Thus, IDR/QR is very efficient.

We compute the closed form solution of SRDA (by solving normal equations) for the first three data sets and use LSQR [14] to get the iterative solution for 20Newsgroup. The iteration number in LSQR is set to be 15. Notice that there is a parameter α which controls smoothness of the estimator in both RLDA and SRDA. We simply set the value of α as 1, and the effect of parameter selection will be discussed later.

5.3 Results

The classification error rate as well as the the running time (second) of computing the projection functions for each method on the four data sets are reported on the Table (4 ∼ 11) respectively. These results are also showed in the Figure (1 ∼ 4). For each given l (the number of training samples per class), we average the results over 20 random splits and report the mean as well as the standard deviation.

The main observations from the performance comparisons include:

[5] http://people.csail.mit.edu/jrennie/20Newsgroups/

Table 4: Classification error rates on PIE (mean±std-dev%)

Train Size	ULDA	RLDA	SRDA	IDR/QR
10×68	31.8±1.1	19.1±1.2	19.5±1.3	23.1±1.4
20×68	20.5±0.8	10.9±0.7	10.8±0.7	16.0±1.1
30×68	10.9±0.5	8.7±0.7	8.4±0.7	13.7±0.8
40×68	8.2±0.4	7.2±0.5	6.9±0.4	11.9±0.6
50×68	7.2±0.4	6.6±0.4	6.3±0.4	11.4±0.7
60×68	6.4±0.3	6.0±0.3	5.7±0.2	10.8±0.5

Table 5: Computational time on PIE (s)

Train Size	ULDA	RLDA	SRDA	IDR/QR
10×68	4.291	4.725	0.235	0.126
20×68	7.626	7.728	0.685	0.244
30×68	7.887	7.918	0.903	0.359
40×68	8.130	8.178	1.126	0.488
50×68	8.377	8.414	1.336	0.527
60×68	8.639	8.654	1.573	0.675

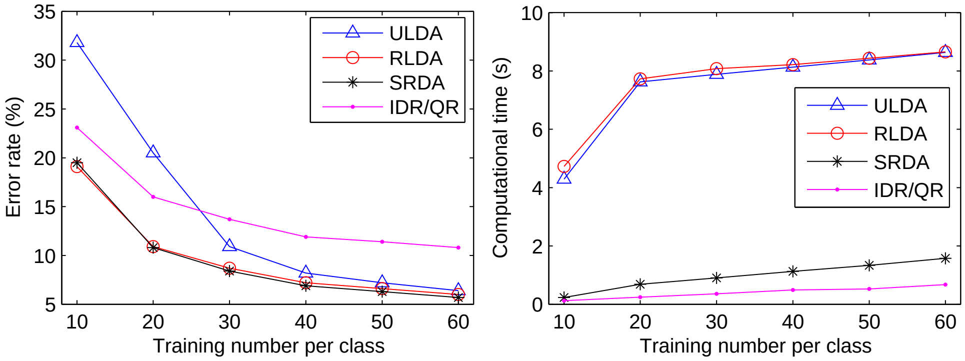


Figure 1: Error rate and computational time as functions of number of labeled samples per class on PIE.

- Both ULDA and RLDA need SVD decomposition of the data matrix. They can be applied when $\min(m,n)$ is small (the first three data sets). The 20Nesgroups has a very large number of features ($n = 26214$). ULDA needs the memory to store the centered data matrix and the left singular matrix, which are both dense and with size of $m \times n$ [20]. With the size of training sample (m) increases, these matrices can not be fit into memory and ULDA thus can not be applied. The situation of RLDA is even worse since it needs store a left singular matrix with size of $n \times n$ [22]. The IDR/QR algorithm only need to solve a QR decomposition of matrix with size of $n \times c$ and an Eigen-decomposition of matrix with size $c \times c$, where c is number of classes [21]. Thus, IDR/QR is very efficient. However, it still needs to store the centered data matrix which can not be fit into memory when both m and n are large (In the case of using more than 40% samples in 20Newsgroups as training set). SRDA only needs to solve $c - 1$ regularized least squares problems which make it almost as efficient as IDR/QR. Moreover, it can fully explore the sparseness of the data matrix

Table 6: Classification error rates on Isolet (mean±std-dev%)

Train Size	ULDA	RLDA	SRDA	IDR/QR
20×26	54.1±1.5	9.4±0.4	9.5±0.5	11.4±0.5
30×26	27.7±1.0	8.3±0.6	8.4±0.7	10.2±0.7
50×26	11.4±0.6	7.5±0.3	7.5±0.3	9.3±0.4
70×26	8.9±0.4	7.0±0.3	7.1±0.3	8.9±0.3
90×26	7.8±0.3	6.7±0.2	6.8±0.2	8.5±0.3
110×26	7.2±0.2	6.5±0.1	6.6±0.2	8.3±0.2

Table 7: Computational time on Isolet (s)

Train Size	ULDA	RLDA	SRDA	IDR/QR
20×26	1.351	1.403	0.096	0.056
30×26	1.629	1.653	0.148	0.059
50×26	1.764	1.766	0.204	0.092
70×26	1.861	1.869	0.265	0.134
90×26	1.935	1.941	0.322	0.177
110×26	2.007	2.020	0.374	0.269

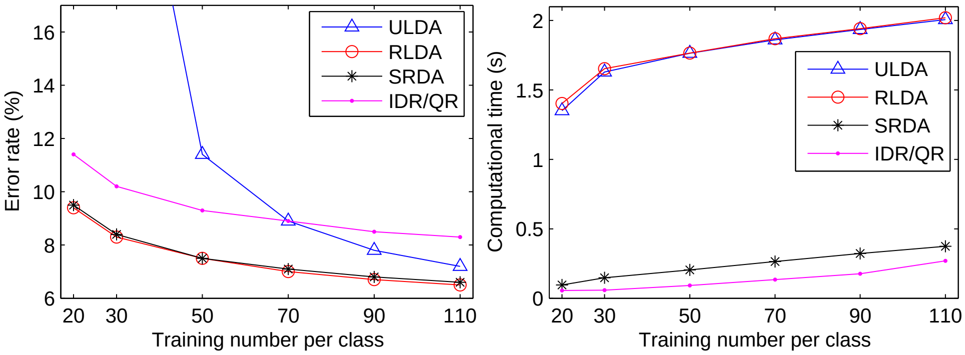


Figure 2: Error rate and computational time as functions of number of labeled samples per class on Isolet.

and gain significant computational saving on both time and memory.

- The ULDA seeks the projective functions which are optimal on the training set. It does not consider the possible overfitting in small sample size case. RLDA and SRDA are regularized versions of LDA. The Tikhonov regularizer is used to control the model complexity. In all the test cases, RLDA and SRDA are significantly better than other ULDA, which suggests that overfitting is a very crucial problem which should be addressed in LDA model.

- Although IDR/QR is developed from LDA idea, there is no theoretical relation between the optimization problem solved by IDR/QR and that of LDA. In all the four data sets, RLDA and SRDA significantly outperform IDR/QR.

- Considering both accuracy and efficiency, SRDA is the best choice among four of the compared

Table 8: Classification error rates on MNIST (mean±std-dev%)

Train Size	ULDA	RLDA	SRDA	IDR/QR
30×10	48.1±1.5	23.4±1.4	23.6±1.4	26.8±1.6
50×10	73.3±2.2	21.5±1.2	21.9±1.2	26.1±1.7
70×10	62.1±7.3	20.4±0.9	20.8±0.8	24.9±1.1
100×10	43.1±3.3	19.5±0.5	19.7±0.5	24.7±0.7
130×10	45.5±9.7	18.8±0.5	19.0±0.6	24.2±0.9
170×10	38.4±8.0	18.1±0.3	18.5±0.5	24.0±0.6

Table 9: Computational time on MNIST (s)

Train Size	ULDA	RLDA	SRDA	IDR/QR
30×10	0.389	0.817	0.035	0.023
50×10	1.645	1.881	0.092	0.042
70×10	2.341	2.429	0.180	0.062
100×10	2.498	2.622	0.268	0.154
130×10	2.528	2.673	0.317	0.168
170×10	2.636	2.713	0.379	0.211

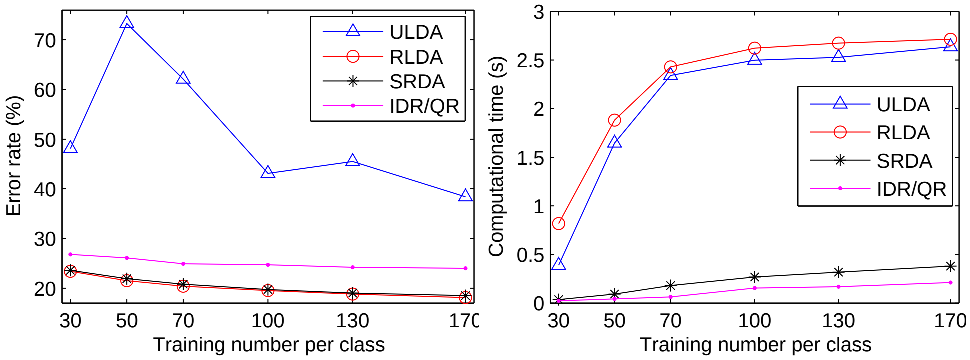


Figure 3: Error rate and computational time as functions of number of labeled samples per class on MNIST.

algorithms. It provides an efficient and effective discriminant analysis solution for large scale data sets.

5.4 Parameter selection for SRDA

The $\alpha \geq 0$ is an essential parameter in our SRDA algorithm which controls the smoothness of the estimator. We empirically set it to be 1 in the previous experiments. In this subsection, we try to examine the impact of parameter α on the performance of SRDA.

Figure (5) shows the performance of SRDA as a function of the parameter α. For convenience, the X-axis is plotted as $\alpha/(1+\alpha)$ which is strictly in the interval $[0,1]$. It is easy to see that SRDA can achieve significantly better performance than ULDA and IDR/QR over a large range of α. Thus, the

Table 10: Classification error rates on 20Newsgroups (mean±std-dev%)

Train Size	ULDA*	RLDA*	SRDA	IDR/QR*
5%	28.0±0.6	–	27.3±0.5	33.0±0.9
10%	22.7±0.6	–	21.3±0.5	29.0±0.4
20%	–	–	16.0±0.3	25.9±0.4
30%	–	–	13.8±0.2	25.2±0.4
40%	–	–	12.4±0.2	–
50%	–	–	11.4±0.2	–

Table 11: Computational time on 20Newsgroups (s)

Train Size	ULDA*	RLDA*	SRDA	IDR/QR*
5%	61.84	–	16.47	5.705
10%	224.9	–	19.23	11.77
20%	–	–	22.93	20.18
30%	–	–	26.84	32.75
40%	–	–	31.24	–
50%	–	–	36.51	–

*ULDA (RLDA, IDR/QR) can not be applied as the size of training set increases due to the memory limit.

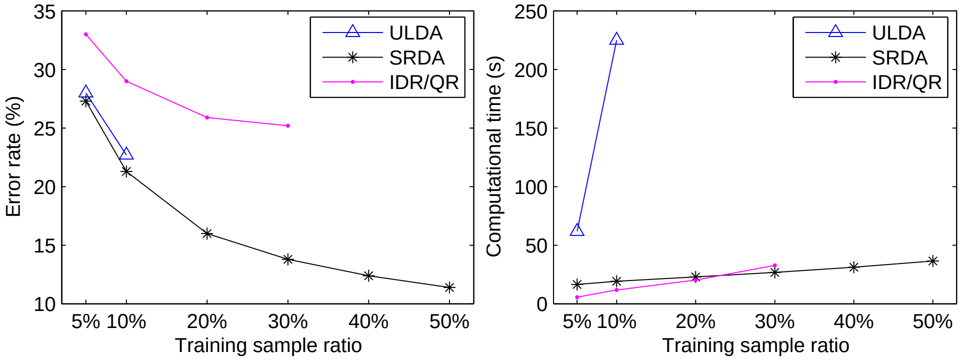
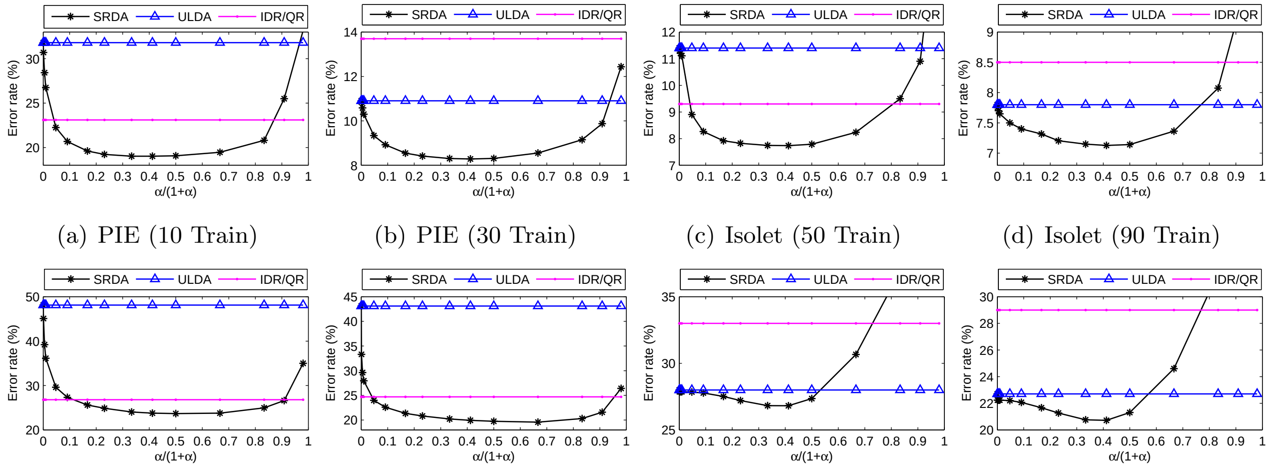


Figure 4: Error rate and computational time as functions of number of labeled samples per class on 20Newsgroup.

parameter selection is not a very crucial problem in SRDA algorithm.

6 Conclusions

In this paper, we propose a novel algorithm for discriminant analysis, called *Spectral Regression Discriminant Analysis* (SRDA). Our algorithm is developed from a graph embedding viewpoint of LDA problem. It combines the spectral graph analysis and regression to provide an efficient and effective approach for discriminant analysis. Specifically, SRDA only needs to solve a set of regularized least squares problems and there is no eigenvector computation involved, which is a huge save of both time and memory. To the



(a) PIE (10 Train) (b) PIE (30 Train) (c) Isolet (50 Train) (d) Isolet (90 Train)

(e) MNIST (30 Train) (f) MNIST (100 Train) (g) 20Newsgroup (5% Train) (h) 20Newsgroup (10% Train)

Figure 5: Model selection of SRDA on PIE (a, b), Isolet (c, d), MNIST (e, f) and 20Newsgroup (g h). The curve shows the test error of SRDA with respect to $\alpha/(1+\alpha)$. The other two lines show the test error of ULDA and IDR/QR. It is clear that SRDA can achieve significantly better performance than ULDA and IDR/QR over a large range of α.

best of our knowledge, our proposed SRDA algorithm is the first one which can handle very large scale high dimensional data for discriminant analysis. Extensive experimental results show that our method consistently outperforms the other state-of-the-art LDA extensions considering both effectiveness and efficiency.

APPENDIX

A Proof of Theorem 3

Proof Suppose $rank(\bar{X}) = r$, the SVD decomposition of $\bar{X}$ is

$$\bar{X} = U\Sigma V^T \tag{25}$$

where $\Sigma = diag(\sigma_1, \cdots, \sigma_r)$, $U \in \mathbb{R}^{n\times r}$, $V \in \mathbb{R}^{m\times r}$ and we have $U^TU = V^TV = I$. The $\bar{\mathbf{y}}$ is in the space spanned by row vectors of $\bar{X}$, therefor, $\bar{\mathbf{y}}$ is in the space spanned by column vectors of V. Thus, $\bar{\mathbf{y}}$ can be represented as the linear combination of the column vectors of V. Moreover, the combination is unique because the column vectors of V are linearly independent. Suppose the combination coefficients are $b_1, \cdots, b_r$. Let $\mathbf{b} = [b_1, \cdots, b_r]^T$, we have:

$$\begin{aligned} V\mathbf{b} = \bar{\mathbf{y}} \quad &\Rightarrow \quad V^TV\mathbf{b} = V^T\bar{\mathbf{y}} \\ &\Rightarrow \quad \mathbf{b} = V^T\bar{\mathbf{y}} \\ &\Rightarrow \quad VV^T\bar{\mathbf{y}} = \bar{\mathbf{y}} \end{aligned} \tag{26}$$

To continue our proof, we need to introduce the concept of pseudo inverse of a matrix [15], which we denote as $(\cdot)^+$. Specifically, pseudo inverse of the matrix $\bar{X}$ can be computed by the following two ways:

$$\bar{X}^+ = V\Sigma^{-1}U^T$$

and

$$\bar{X}^+ = \lim_{\alpha\to 0}(\bar{X}^T\bar{X} + \alpha I)^{-1}\bar{X}^T$$

The above limit exists even if $\bar{X}^T\bar{X}$ is singular and $(\bar{X}^T\bar{X})^{-1}$ does not exist [15].

Thus, the regularized least squares solution of SRDA

$$\begin{aligned} \mathbf{a} &= \left(\bar{X}\bar{X}^T + \alpha I\right)^{-1}\bar{X}\bar{\mathbf{y}} \\ &\overset{\alpha\to 0}{=} (\bar{X}^T)^+\bar{\mathbf{y}} \\ &= U\Sigma^{-1}V^T\bar{\mathbf{y}} \end{aligned}$$

Combined with Eqn. (26), we have

$$\begin{aligned} \bar{X}^T\mathbf{a} &= V\Sigma U^T\mathbf{a} \\ &= V\Sigma U^T U\Sigma^{-1}V^T\bar{\mathbf{y}} = VV^T\bar{\mathbf{y}} = \bar{\mathbf{y}} \end{aligned}$$

By Theorem (2), $\mathbf{a}$ is the eigenvector of eigen-problem in Eqn. (13).

B Proof of Corollary 4

Proof Since the m data points $\mathbf{x}_i$'s are linearly independent, we have $rank(\bar{X}) = m-1$. Also we have $\bar{X}\mathbf{e} = 0$. The space spanned by row vectors of $\bar{X}$ is orthogonal to $\mathbf{e}$ and have dimension $m-1$. Let us examine the $c-1$ vectors $\bar{\mathbf{y}}_k$ in Eqn. (19). We have $\bar{\mathbf{y}}_k \in \mathbb{R}^m$ and $\bar{\mathbf{y}}_k^T\mathbf{e} = 0$. Thus, all $c-1$ vectors $\bar{\mathbf{y}}_k$ are in the space spanned by row vectors of $\bar{X}$. By Theorem (3), all $c-1$ corresponding $\mathbf{a}_k$ of SRDA are eigenvectors of eigen-problem in Eqn. (13) as α decreases to zero. They are

$$\mathbf{a}_k^{SRDA} = U\Sigma^{-1}V^T\bar{\mathbf{y}}_k.$$

Consider the eigen-problem in Eqn. (11), since the $c-1$ vectors $\bar{\mathbf{y}}_k$ are also in the space spanned by row vectors of $\widetilde{X} = U^T\bar{X} = \Sigma V^T$, eigenvector $\mathbf{b}_k$ will be the solution of linear equations system $\widetilde{X}^T\mathbf{b}_k = \bar{\mathbf{y}}_k$. The row vectors of $\widetilde{X} = \Sigma V^T$ are linearly independent, thus $\mathbf{b}_k$ is unique and

$$\mathbf{b}_k = \Sigma^{-1}V^T\bar{\mathbf{y}}_k.$$

Thus, the projective functions of LDA in Section 3

$$\mathbf{a}_k^{ULDA} = U\mathbf{b}_k = U\Sigma^{-1}V^T\bar{\mathbf{y}}_k = \mathbf{a}_k^{SRDA}$$

References

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

[2] F. R. K. Chung. *Spectral Graph Theory*, volume 92 of *Regional Conference Series in Mathematics*. AMS, 1997.

[3] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience, Hoboken, NJ, 2nd edition, 2000.

[4] J. H. Friedman. Regularized discriminant analysis. *Journal of the American Statistical Association*, 84(405):165–175, 1989.

[5] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, 2nd edition, 1990.

[6] V. Gaede and O. Günther. Multidimensional access methods. *ACM Comput. Surv.*, 30(2):170–231, 1998.

[7] G. H. Golub and C. F. V. Loan. *Matrix computations*. Johns Hopkins University Press, 3rd edition, 1996.

[8] T. Hastie, A. Buja, and R. Tibshirani. Penalized discriminant analysis. *Annals of Statistics*, 23:73–102, 1995.

[9] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York: Springer-Verlag, 2001.

[10] X. He, S. Yan, Y. Hu, P. Niyogi, and H.-J. Zhang. Face recognition using laplacianfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(3):328–340, 2005.

[11] P. Howland and H. Park. Generalizing discriminant analysis using the generalized singular value decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(8):995–1006, 2004.

[12] K. Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339, 1995.

[13] C. C. Paige and M. A. Saunders. Algorithm 583 LSQR: Sparse linear equations and least squares problems. *ACM Transactions on Mathematical Software*, 8(2):195–209, June 1982.

[14] C. C. Paige and M. A. Saunders. LSQR: An algorithm for sparse linear equations and sparse least squares. *ACM Transactions on Mathematical Software*, 8(1):43–71, March 1982.

[15] R. Penrose. A generalized inverse for matrices. In *Proceedings of the Cambridge Philosophical Society*, volume 51, pages 406–413, 1955.

[16] G. W. Stewart. *Matrix Algorithms Volume I: Basic Decompositions.* SIAM, 1998.

[17] G. W. Stewart. *Matrix Algorithms Volume II: Eigensystems.* SIAM, 2001.

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

[19] K. Torkkola. Linear discriminant analysis in document classification. In *Proc. IEEE ICDM Workshop Text Mining*, 2001.

[20] J. Ye. Characterization of a family of algorithms for generalized discriminant analysis on undersampled problems. *Journal of Machine Learning Research*, (6):483–502, 2005.

[21] J. Ye, Q. Li, H. Xiong, H. Park, R. Janardan, and V. Kumar. IDR/QR: an incremental dimension reduction algorithm via QR decomposition. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'04)*, pages 364–373, 2004.

[22] J. Ye and T. Wang. Regularized discriminant analysis for high dimensional, low sample size data. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'06)*, pages 454–463, 2006.