SRDA: An Efficient Algorithm for Large-Scale Discriminant Analysis

Deng Cai, *Student Member*, *IEEE*, Xiaofei He, and Jiawei Han, *Senior Member*, *IEEE*

Abstract—Linear Discriminant Analysis (LDA) has been a popular method for extracting features that preserves 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 eigendecomposition, which can be computationally expensive in both 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 that 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 nonzero features in each sample. Extensive experimental results on four real-world data sets demonstrate the effectiveness and efficiency of our algorithm.

Index Terms—Linear Discriminant Analysis, spectral regression, dimensionality reduction.

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. [11]. However, these methods can only operate with small dimensionality, typically less than 100. The 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 a lower dimensional space, conventional indexing schemes can then be applied.

One of the most popular dimensionality reduction algorithms is the Linear Discriminant Analysis (LDA) [8], [10]. 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 eigendecomposition on the scatter matrices of the given training data. LDA has been widely used in many applications such as text processing [24] and face recognition [1]. However, the scatter matrices are dense, and the eigendecomposition 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 (for example, Principal Component Analysis (PCA) and Singular Value Decomposition (SVD)) are required to guarantee the nonsingularity of scatter matrices [1], [25], which further increase the time and memory costs. Therefore, 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 a significant computational advantage over LDA. Benefiting from recent progresses on spectral graph analysis, we analyze LDA from a graph embedding point of view that can be traced back to [15]. We show how the LDA solution can be obtained by solving a set of linear equations that links LDA and classical regression. Our approach combines spectral graph analysis and regression to provide an efficient and effective approach for discriminant analysis.

- *D. Cai and J. Han are with the Department of Computer Science, University of Illinois at Urbana Champaign, Siebel Center, 201 N. Goodwin Ave., Urbana, IL 61801. E-mail: {dengcai2, hanj}@cs.uiuc.edu.*
- *X. He is with Yahoo, 3333 W. Empire Avenue, Burbank, CA 91504. E-mail: hex@yahoo-inc.com.*

Manuscript received 17 May 2007; revised 23 Aug. 2007; accepted 28 Aug. 2007; published online 5 Sept. 2007.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2007-05-0224. Digital Object Identifier no. 10.1109/TKDE.2007.190669.

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, for example, the two-stage PCA+LDA approach [1] and LDA/Generalized SVD (GSVD) approaches [16], [25], can be unified in an 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 a small sample size situation, these solutions tend to overfit 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 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 nonzero features in each sample. It can be easily scaled to very large high-dimensional data sets.
- For the binary classification problem, LDA has been shown to be equivalent to the regression [14]. We extend this relation to the multiclass case. With regression as the building block, various kinds of regularization techniques can be easily incorporated in SRDA (for example, an L_1-norm regularizer to produce sparse projections [3]).
- Our algorithm may be conducted in the original space or in the reproducing kernel Hilbert space (RKHS) into which data points are mapped. This gives an efficient algorithm for Kernel Discriminant Analysis [2].

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 *SRDA* 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 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 (see Table 1 for a list of notations used in this paper). The objective function of LDA is given as follows:

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

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

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

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

where $\boldsymbol{\mu}$ is the total sample mean vector, m_k is the number of samples in the kth class, $\boldsymbol{\mu}^{(k)}$ is the average vector of the kth class, and $\mathbf{x}_i^{(k)}$ is the ith sample in the kth 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 $S_t = S_b + S_w$ [10]. The objective function of LDA in (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

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

where $tr()$ denotes matrix trace. The optimization problem in (5) is equivalent to finding the l eigenvectors of the following generalized eigenproblem 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 nonzero eigenvalues [10].

To get a stable solution of the above generalized eigenproblem, 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-known 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 PCA and SVD. Both Swets and Weng [23] and Belhumeur et al. [1] have utilized PCA+LDA for face recognition. Torkkola [24] implemented SVD+LDA for document classification. All these approaches use the LDA objective function in (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 and Park [16] solved the singularity problem of LDA by using GSVD. They rewrite the LDA objective function as the following equivalent form:

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

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 [25], 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) [9], [13]. However, $S_w + \lambda I$ is a very large dense matrix for high-dimensional data that incurs a high computational cost on directly solving the eigenproblem in (6). By noticing that the eigendecomposition of $S_w + \alpha I$ is the sum of the eigendecomposition of S_w and αI, Ye and Wang [27] developed an efficient algorithm to compute the projective functions of RDA. The computational cost of this approach will be comparable to those of two-stage PCA+LDA approaches.

The computation of all the above LDA extensions involves the SVD of the data matrix, which is computationally expensive in both time and memory for high-dimensional large-scale data sets. In some applications (for example, 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 and thus may not be able to be fit into the memory. In this case, all these LDA approaches cannot 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 [26]. Experiments on some data sets showed that IDR/QR is much more efficient than LDA and achieves comparable performance as LDA [26]. 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 that can be traced back to [15]. We start by 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 kth 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 the edge joining vertices i and j. $W_{ij} = 0$ indicates that there is no edge between vertices i and j. Thus, $L = I - W$ is called *graph Laplacian*[1] [7].

We have

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

Since S_t is of size $n \times n$, in the case of $n > m$, S_t is singular, and the eigenproblem of LDA in (6) cannot 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 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, and $V = [\mathbf{v}_1, \cdots, \mathbf{v}_r] \in \mathbb{R}^{m\times r}$ and $\mathbf{v}_i$s are called right singular vectors.

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 equal to the sum of the ith column (or row, since W is symmetric) of W. With the W defined in (7), we can easily see that $D = I$.

Substituting $\bar{X}$ in (5), we get

$$A^* = \arg\max_A \frac{tr(A^T U\Sigma V^T WV\Sigma U^T A)}{tr(A^T U\Sigma V^T V\Sigma U^T A)}.$$

We proceed to variable modification using $B = \Sigma U^T A$ and get

$$B^* = \arg\max_B \frac{tr(B^T V^T WVB)}{tr(B^T B)},$$

and the columns of B^* are the eigenvectors of $V^T WV$ associated with the nonzero eigenvalues.

After we get B^*, A^* can be obtained by solving a set of linear equation systems $\Sigma U^T A = B^*$. Notice that given U and B^*, there will be infinitely many solutions of A that satisfy this equations system.[2] Among all these solutions

$$A^* = U\Sigma^{-1}B^* \tag{11}$$

is obviously one of them and can be used as the transformation matrix of LDA.

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 the two-stage PCA+LDA approach. The Fisherface approach [1] keeps at most $m - c$ dimensions in the PCA step to make S_w nonsingular and thus may lose some useful information. Our analysis shows that based on the modified but equivalent LDA objective function in (4), we can keep all the nonzero 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 (11). The original feature vectors X is transformed into $Y = A^T X$, where the ith feature component of Y (ith row of Y) is denoted as $\mathbf{y}_i^T$, $\mathbf{y}_i = X^T\mathbf{a}_i$. Thus, $\mathbf{y}_i$ and $\mathbf{y}_j$ 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 that $(\mathbf{y}_i - \mathbf{e}\nu_i)^T(\mathbf{y}_j - \mathbf{e}\nu_j) = 0$ for $i \neq j$. We have

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

The last equation holds since $\mathbf{b}_i$s are eigenvectors of $V^T WV$ [12]. □

In this sense, the SVD+LDA approach described above can also be called Uncorrelated LDA (ULDA) [25].

2. This is true unless $n < m$ and $rank(\bar{X}) = n$. In this case, U will be an orthogonal matrix, and there is a unique solution of equation $\Sigma U^T A = B^*$, which is exactly $U\Sigma^{-1}B^*$.

3.1 Computational Complexity of LDA

Now, let us analyze the computational complexities of LDA. The main computation of LDA is solving the generalized eigenproblem:

$$\bar{X}W\bar{X}^T\mathbf{a} = \lambda\bar{X}\bar{X}^T\mathbf{a}. \tag{12}$$

Suppose we have the SVD of $\bar{X}$ shown in (10); we have

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

where $\mathbf{b} = \Sigma U^T\mathbf{a}$, and $V \in \mathbb{R}^{m\times r}$ is the 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 of $\bar{X}$ to get U, V, and Σ,
2. computing $\mathbf{b}$s, the eigenvectors of $V^T WV$, and
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 $V^T WV$. The following trick can be used to save computational cost. We denote the ith 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 that 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

$$\begin{aligned}
V^T WV &= \sum_{k=1}^{c}\frac{1}{l_k}\left(\sum_{i=1}^{l_k}\mathbf{z}_i^{(k)}\sum_{i=1}^{l_k}(\mathbf{z}_i^{(k)})^T\right)\\
&= \sum_{k=1}^{c} l_k\,\boldsymbol{\nu}^{(k)}(\boldsymbol{\nu}^{(k)})^T\\
&= HH^T.
\end{aligned} \tag{13}$$

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 $\bar{X}^T\bar{X}$ [22]. 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 (that is, *cross product*) applies this strategy [22]. 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, compared to directly calculating the eigenvectors of HH^T, computing the eigenvectors of $H^T H$ and then recovering the eigenvectors of HH^T can achieve a significant saving. The computational approach described here is exactly identical to the ULDA approach in [25].

We use the term *flam* [21], 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 [22], [12], recovering V

from U requires mn^2 flam by assuming that 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, and, finally, calculating $\mathbf{a}$s from $\mathbf{b}$s requires n^2c flam. When $m<n$, we have a 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 that $c\ll t$, 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 summed up, this 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 eigenproblem in (12) efficiently, we use the following theorem:

Theorem 2. *Let $\bar{\mathbf{y}}$ be the eigenvector of eigenproblem*

$$W\bar{\mathbf{y}}=\lambda\bar{\mathbf{y}} \tag{14}$$

with eigenvalue λ. If $\bar{X}^T\mathbf{a}=\bar{\mathbf{y}}$, then $\mathbf{a}$ is the eigenvector of the eigenproblem in (12) with the same eigenvalue λ.

Proof. We have $W\bar{\mathbf{y}}=\lambda\bar{\mathbf{y}}$. At the left side of (12), replace $\bar{X}^T\mathbf{a}$ by $\bar{\mathbf{y}}$ and 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 the eigenproblem in (14) with the same eigenvalue λ. □

Theorem 2 shows that instead of solving the eigenproblem in (12), the LDA basis functions can be obtained through two steps:

1. Solve the eigenproblem in (14) to get $\bar{\mathbf{y}}$.
2. Find the $\mathbf{a}$ that satisfies $\bar{X}^T\mathbf{a}=\bar{\mathbf{y}}$. In reality, such $\mathbf{a}$ may not exist. A possible way is to find the $\mathbf{a}$ that can best fit the equation in the least squares sense:

$$\mathbf{a}=\arg\min_{\mathbf{a}}\sum_{i=1}^{m}(\mathbf{a}^T\bar{\mathbf{x}}_i-\bar{y}_i)^2, \tag{15}$$

where $\bar{y}_i$ is the ith element of $\bar{\mathbf{y}}$.

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

1. We will show later how the eigenproblem in (14) is *trivial* and we can directly get those eigenvectors $\bar{\mathbf{y}}$.
2. Compared to all the other LDA extensions, there is no dense matrix eigendecomposition or SVD decomposition involved. There exist many efficient iterative algorithms (for example, LSQR [19]) that can handle very large-scale least squares problems. Therefore, 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 (15) is *ill posed*. We may have an infinite number of solutions for the linear equation 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 $\mathbf{a}$:

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

This is the so-called regularization and is well studied in statistics. The regularized least squares is also called ridge regression [14]. $\alpha\geq 0$ 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 [14].

Now, let us analyze the eigenvectors of W, which is defined in (7). 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 one, where $\mathbf{e}^{(k)}=[1,1,\cdots,1]^T$. Also, there is only one nonzero eigenvalue of $W^{(k)}$ because the rank of $W^{(k)}$ is one. Thus, there are exactly c eigenvectors of W with the same eigenvalue 1. These eigenvectors are

$$\mathbf{y}_k=[\underbrace{0,\cdots,0}_{\sum_{i=1}^{k-1}m_i},\underbrace{1,\cdots,1}_{m_k},\underbrace{0,\cdots,0}_{\sum_{i=k+1}^{c}m_i}]^T\quad k=1,\cdots,c. \tag{17}$$

Since one 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 that there exists a vector $\mathbf{a}$ that satisfies the linear equation 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 (17). Therefore, we pick $\mathbf{e}$ as our first eigenvector of W and use the 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:

$$\{\bar{\mathbf{y}}_k\}_{k=1}^{c-1},\ (\bar{\mathbf{y}}_i^T\mathbf{e}=0,\ \ \bar{\mathbf{y}}_i^T\bar{\mathbf{y}}_j=0,\ i\neq j). \tag{18}$$

In the two-class case, the above procedure will produce one response vector:

$$\bar{\mathbf{y}}=[\underbrace{m/m_1,\cdots,m/m_1}_{m_1},\underbrace{-m/m_2,\cdots,-m/m_2}_{m_2}]^T. \tag{19}$$

This is consistent with the previous well-known result on the relationship between LDA and regression for the binary

problem [14]. Our approach here extends this relation to the multiclass case.

The above two-step approach essentially combines the spectral analysis of the graph matrix W and the regression technique. Therefore, we named this new approach as SRDA. It is important to note that our approach can be generalized by constructing the graph matrix W in the unsupervised or semisupervised way. Please see [4], [5], [6] for more details.

4.1 Theoretical Analysis

In the following discussions, $\bar{\mathbf{y}}$ is one of the eigenvectors in (18).

The regularized least squares problem of SRDA in (16) can be rewritten in matrix form as

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

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

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

When $\alpha > 0$, this regularized solution will not satisfy the linear equation system $\bar{X}^T\mathbf{a} = \bar{\mathbf{y}}$, and $\mathbf{a}$ is also not the eigenvector of the LDA eigenproblem in (12). 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 the eigenproblem in (12) as α deceases to zero. Therefore, $\mathbf{a}$ will be one of the projective functions 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, that is, $rank(X) = m$. In this case, we have a stronger conclusion, which is shown in the following corollary:

Corollary 4. *If the sample vectors are linearly independent, that is, $rank(X) = m$, 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 ith and jth entries of any vector $\mathbf{y}$ in the space spanned by $\{\mathbf{y}_k\}$ in (17) are the same as long as $\mathbf{x}_i$ and $\mathbf{x}_j$ belong to the same class. Thus, the ith and jth 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 equation 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 that 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, data points with the same label correspond 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 and 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 (for example, 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 and 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

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

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 the centered data matrix, which can save the memory a lot for sparse data processing.

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

1. **Response 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 $\mathbf{y}_0 = [1, 1, \cdots, 1]^T$ denote a vector of ones. Take $\mathbf{y}_0$ as the first vector and use the 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 the regularized least squares problem:

$$\mathbf{a}_k = \arg\min_{\mathbf{a}} \left(\sum_{i=1}^{m} (\mathbf{a}^T \mathbf{x}_i - \bar{y}_i^k)^2 + \alpha \|\mathbf{a}\|^2 \right), \qquad (22)$$

where $\bar{y}_i^k$ is the ith 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 the $(c-1)$-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 [21].

The computation of SRDA involves two steps: response generation and regularized least squares. The cost of the first step is mainly the cost of the Gram-Schmidt method, which requires $(mc^2 - \frac{1}{3}c^3)$ flam and $mc + c^2$ memory [21].

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

- Differentiate the residual sum of squares with respect to the 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* [21], as shown in (21).
- Use iterative algorithm LSQR [19].

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

4.3.1 Solving Normal Equations

As shown in (21), the normal equations of the regularized least squares problem in (22) are

$$(XX^T + \alpha I)\mathbf{a}_k = X\bar{\mathbf{y}}_k. \qquad (23)$$

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 the upper triangular with positive diagonal elements. This is the so-called Cholesky decomposition, and it requires $\frac{1}{6}n^3$ flam [21]. With this Cholesky decomposition, the $c-1$ linear equations can be solved within cn^2 flam [21]. 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 the pseudoinverse of a matrix [20], which is denoted as $(\cdot)^+$. We have [20]

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

Thus, the normal equations in (23) can be solved by solving the following two linear equation systems when α is 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 equation systems in (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 that $c \ll t$, 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

LSQR is an iterative algorithm designed to solve large-scale sparse linear equations and least squares problems [19]. 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 workload of LSQR in each iteration is $3m + 5n$ flam [18]. 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 [19]. 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 [18]. We need to store $\mathbf{a}_k$. Thus, the memory cost of SRDA with LSQR is

$$mn + m + 2n + cn,$$

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 $s \ll n$ nonzero 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 the following:

- 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 nine.

TABLE 2
Computational Complexity of LDA and SRDA

Algorithm			operation counts (*flam* [21])	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
n: the number of features
t: $\min(m, n)$
c: the number of classes
k: the number of iterations in LSQR
s: the average number of non-zero features for one sample

- 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, for example, 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 restrict the ordinary LDA algorithms (for example, PCA+LDA, ULDA, and 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 be successfully applied as long as the data matrix X can be fit into the memory.
- Even when 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 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 Pentium 4 3.20-GHz Windows XP machine with 2 Gbytes of memory. For the purpose of reproducibility, we provide our algorithms and data sets used in these experiments at http://www.cs.uiuc.edu/homes/dengcai2/Data/data.html.

5.1 Data Sets

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

- The Carnegie Mellon University (CMU) PIE face database[3] contains 68 subjects with 41,368 face images in total. The face images were captured under varying pose, illumination, and expression. We choose the five near-frontal poses (C05, C07, C09, C27, and 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[4] 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 isolets 1 through 5. For the purposes of this experiment, we chose isolets 1 and 2, which contain 3,120 examples (120 examples per class) as the training set, and test on isolets 4 and 5, which contains 3,117 examples (three examples are missing due to the difficulties in recording). A random subset with $l(= 20, 30, 50, 70, 90, 110)$ examples per letter from isolets 1 and 2 was selected for training.
- The MNIST handwritten digit database[5] has a training set of 60,000 samples (denoted as set A) and a test set of 10,000 samples (denoted as set B). In our experiment, we take the first 2,000 samples from set A as our training set and the first 2,000 samples from set B as our test set. Each digit image is of size 28×28, and there are around 200 samples of each digit in both the training and test sets. A random subset with $l(= 30, 50, 70, 100, 130, 170)$ samples per digit from the training set was selected for training.
- The popular 20 Newsgroups[6] is a data set collected and originally used for document classification by Lang [17]. The "bydate" version is used in our experiment. The duplicates and newsgroup-identifying headers are removed, which leaves us with 18,846 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 one. A random subset with $l(=$ 5 percent, 10 percent,

3. http://www.ri.cmu.edu/projects/project_418.html.
4. http://www.ics.uci.edu/~mlearn/MLSummary.html.
5. http://yann.lecun.com/exdb/mnist/.
6. http://people.csail.mit.edu/jrennie/20Newsgroups/.

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	18846	26214	20

TABLE 4
Classification Error Rates on PIE (Mean ± Std-Dev Percent)

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

20 percent, 30 percent, 40 percent, and 50 percent) 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

The four algorithms that are compared in our experiments are listed below:

1. ULDA [25], which was also analyzed in Section 3.
2. Regularized LDA (RLDA) [9]. 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 [27], Ye and Wang proposed an efficient algorithm to compute the solution of RLDA.
3. SRDA, our approach proposed in this paper.
4. IDR/QR [26], an LDA variation in which QR decomposition is applied rather than SVD. Thus, IDR/QR is very efficient.

TABLE 5
Computational Time on PIE (Seconds)

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

TABLE 6
Classification Error Rates on Isolet (Mean ± Std-Dev Percent)

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 (Seconds)

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

TABLE 8
Classification Error Rates on MNIST (Mean ± Std-Dev Percent)

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

We compute the closed-form solution of SRDA (by solving normal equations) for the first three data sets and use LSQR [19] to get the iterative solution for 20Newsgroup. The iteration number in LSQR is set to be 15. Notice that there is a parameter α that controls the smoothness of the estimator in both RLDA and SRDA. We simply set the value of α as one, and the effect of parameter selection will be discussed later.

5.3 Results

The classification error rate and the runtime (second) of computing the projection functions for each method on the four data sets are reported in Tables 4, 5, 6, 7, 8, 9, 10, and 11, respectively. These results are also shown in Figs. 1, 2, 3, and 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 the following:

- Both ULDA and RLDA need the SVD of the data matrix. They can be applied when $\min(m, n)$ is small (the first three data sets). The 20Nesgroups data set has a very large number of features $(n = 26,214)$. ULDA needs the memory to store the centered data matrix and the left singular matrix, which are both dense and have a size of $m \times n$ [25]. With the increase in the size of the training sample (m), these matrices cannot be fit into the memory, and ULDA thus cannot be applied. The situation of RLDA is

TABLE 9
Computational Time on MNIST (Seconds)

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

TABLE 10
Classification Error Rates on 20Newsgroups
(Mean ± Std-Dev Percent)

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 (Seconds)

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	–

even worse since it needs to store a left singular matrix with a size of $n \times n$ [27]. The IDR/QR algorithm only needs to solve a QR decomposition of a matrix with a size of $n \times c$ and an eigendecomposition of a matrix with a size of $c \times c$, where c is number of classes [26]. Thus, IDR/QR is very efficient. However, it still needs to store the centered data matrix, which cannot be fit into the memory when both m and n are large (in the case of using more than 40 percent of the samples in 20Newsgroups as the 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 and gain significant computational saving in both time and memory.

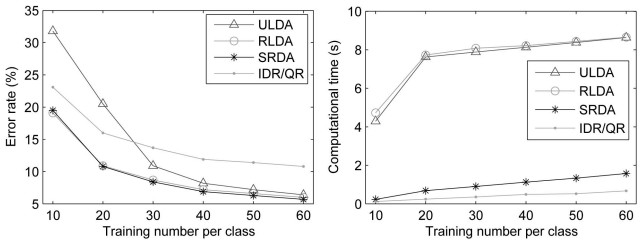

Fig. 1. Error rate and computational time as functions of the number of labeled samples per class on PIE.

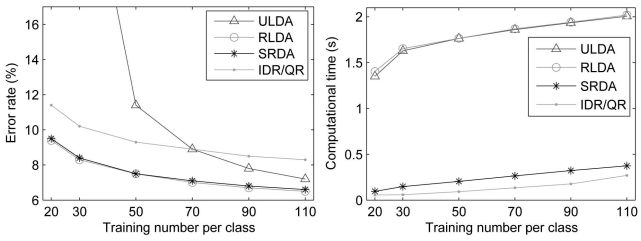

Fig. 2. Error rate and computational time as functions of the number of labeled samples per class on Isolet.

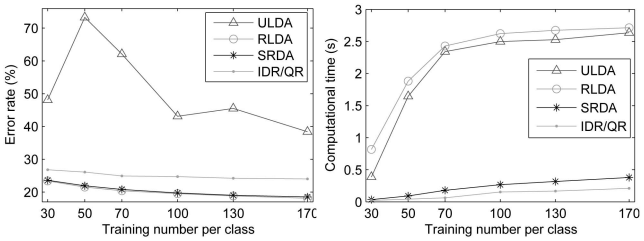

Fig. 3. Error rate and computational time as functions of the number of labeled samples per class on MNIST.

- ULDA seeks the projective functions that are optimal on the training set. It does not consider the possible overfitting in the 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 ULDA, which suggests that overfitting is a very crucial problem that should be addressed in the LDA model.
- Although IDR/QR is developed from the 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 the four compared algorithms. It provides an efficient and effective discriminant analysis solution for large-scale data sets.

5.4 Parameter Selection for SRDA

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

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

6 Conclusions

In this paper, we propose a novel algorithm for discriminant analysis, called SRDA. Our algorithm is developed from a

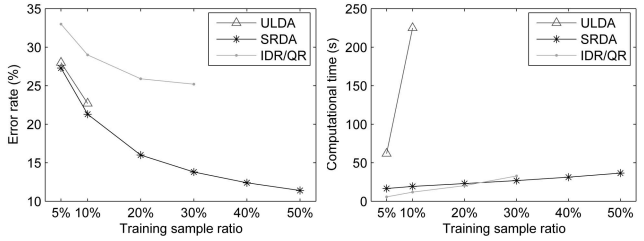

Fig. 4. Error rate and computational time as functions of the number of labeled samples per class on 20Newsgroups.

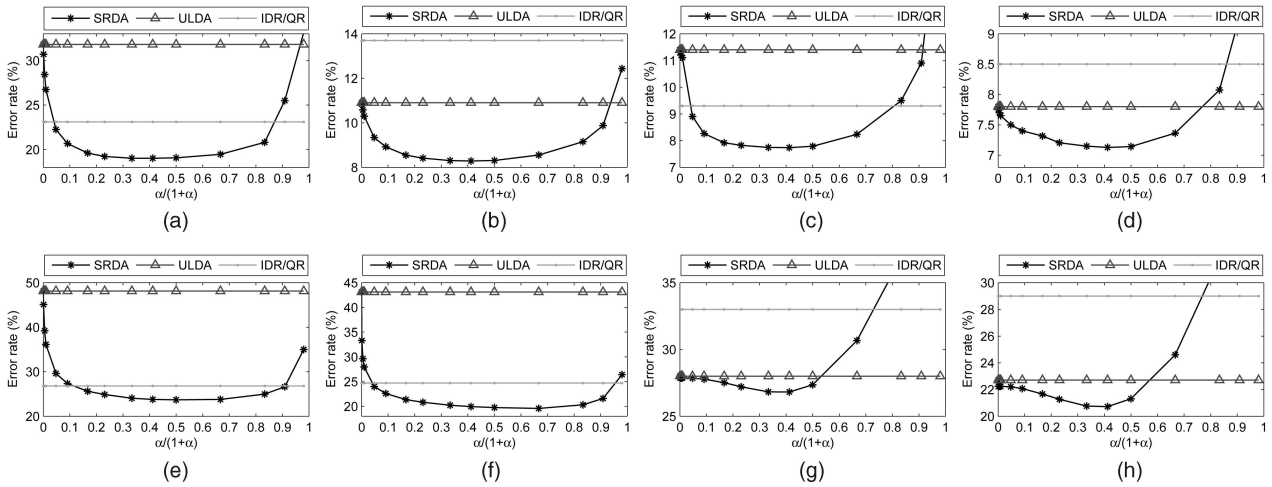

Fig. 5. Model selection of SRDA on (a) PIE (10 train), (b) PIE (30 train), (c) Isolet (50 train), (d) Isolet (90 train), (e) MNIST (30 train), (f) MNIST (100 train), (g) 20Newsgroups (5 percent train), and (h) 20Newsgroups (10 percent train). The curve shows the test error of SRDA with respect to $\alpha/(1+\alpha)$. The other two lines show the test errors 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.

graph embedding viewpoint of the LDA problem. It combines 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 best of our knowledge, our proposed SRDA algorithm is the first one that 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. If $rank(\bar{X}) = r$, 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 the row vectors of $\bar{X}$; therefore, $\bar{\mathbf{y}}$ is in the space spanned by the 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

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

To continue our proof, we need to introduce the concept of the pseudoinverse of a matrix [20], which we denote as $(\cdot)^+$. Specifically, the pseudoinverse of the matrix $\bar{X}$ can be computed in 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 [20].

Thus, the regularized least squares solution of SRDA is

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

Combined with (26), we have

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

By Theorem 2, $\mathbf{a}$ is the eigenvector of the eigenproblem in (12). □

APPENDIX 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 the 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 (18). 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 the row vectors of $\bar{X}$. By Theorem 3, all $c - 1$ corresponding $\mathbf{a}_k$ of SRDA are eigenvectors of the eigenproblem in (12) as α decreases to zero. They are

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

Consider the eigenvectors of V^TWV. Since the $c - 1$ vectors $\bar{\mathbf{y}}_k$ are also in the space spanned by the

column vectors of V, eigenvector $\mathbf{b}_k$ will be the solution of the linear equation system $V\mathbf{b}_k = \bar{\mathbf{y}}_k$. The column vectors of V are linearly independent; thus, $\mathbf{b}_k$ is unique, and

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

Thus, the projective functions of LDA in Section 3 are

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

□

ACKNOWLEDGMENTS

The work was supported in part by the US National Science Foundation Grants IIS-05-13678 and BDI-05-15813 and MIAS (a DHS Institute of Discrete Science Center for Multimodal Information Access and Synthesis).

REFERENCES

[1] P.N. Belhumeur, J.P. Hepanha, and D.J. Kriegman, "Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 19, no. 7, pp. 711-720, July 1997.

[2] D. Cai, X. He, and J. Han, "Efficient Kernel Discriminant Analysis via Spectral Regression," *Proc. Int'l Conf. Data Mining (ICDM '07),* 2007.

[3] D. Cai, X. He, and J. Han, "Spectral Regression: A Unified Approach for Sparse Subspace Learning," *Proc. Int'l Conf. Data Mining (ICDM '07),* 2007.

[4] D. Cai, X. He, and J. Han, "Spectral Regression: A Unified Subspace Learning Framework for Content-Based Image Retrieval," *Proc. ACM Conf. Multimedia,* 2007.

[5] D. Cai, X. He, and J. Han, "Spectral Regression for Efficient Regularized Subspace Learning," *Proc. 11th Int'l Conf. Computer Vision (ICCV '07),* 2007.

[6] D. Cai, X. He, W.V. Zhang, and J. Han, "Regularized Locality Preserving Indexing via Spectral Regression," *Proc. 16th ACM Int'l Conf. Information and Knowledge Management (CIKM '07),* 2007.

[7] F.R.K. Chung, "Spectral Graph Theory," *CBMS Regional Conf. Series in Math.,* vol. 92, 1997.

[8] R.O. Duda, P.E. Hart, and D.G. Stork, *Pattern Classification,* second ed. Wiley-Interscience, 2000.

[9] J.H. Friedman, "Regularized Discriminant Analysis," *J. Am. Statistical Assoc.,* vol. 84, no. 405, pp. 165-175, 1989.

[10] K. Fukunaga, *Introduction to Statistical Pattern Recognition,* second ed. Academic Press, 1990.

[11] V. Gaede and O. Günther, "Multidimensional Access Methods," *ACM Computing Surveys,* vol. 30, no. 2, pp. 170-231, 1998.

[12] G.H. Golub and C.F.V. Loan, *Matrix Computations,* third ed. Johns Hopkins Univ. Press, 1996.

[13] T. Hastie, A. Buja, and R. Tibshirani, "Penalized Discriminant Analysis," *Annals of Statistics,* vol. 23, pp. 73-102, 1995.

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

[15] X. He, S. Yan, Y. Hu, P. Niyogi, and H.-J. Zhang, "Face Recognition Using Laplacianfaces," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 27, no. 3, pp. 328-340, Mar. 2005.

[16] P. Howland and H. Park, "Generalizing Discriminant Analysis Using the Generalized Singular Value Decomposition," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 26, no. 8, pp. 995-1006, Aug. 2004.

[17] K. Lang, "Newsweeder: Learning to Filter Netnews," *Proc. 12th Int'l Conf. Machine Learning (ICML '95),* pp. 331-339, 1995.

[18] C.C. Paige and M.A. Saunders, "Algorithm 583 LSQR: Sparse Linear Equations and Least Squares Problems," *ACM Trans. Math. Software,* vol. 8, no. 2, pp. 195-209, June 1982.

[19] C.C. Paige and M.A. Saunders, "LSQR: An Algorithm for Sparse Linear Equations and Sparse Least Squares," *ACM Trans. Math. Software,* vol. 8, no. 1, pp. 43-71, Mar. 1982.

[20] R. Penrose, "A Generalized Inverse for Matrices," *Proc. Cambridge Philosophical Soc.,* vol. 51, pp. 406-413, 1955.

[21] G.W. Stewart, "Basic Decompositions," *Matrix Algorithms,* vol. 1, SIAM, 1998.

[22] G.W. Stewart, "Eigensystems," *Matrix Algorithms,* vol. 2, SIAM, 2001.

[23] D.L. Swets and J. Weng, "Using Discriminant Eigenfeatures for Image Retrieval," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 18, no. 8, pp. 831-836, Aug. 1996.

[24] K. Torkkola, "Linear Discriminant Analysis in Document Classification," *Proc. IEEE Int'l Conf. Data Mining Workshop Text Mining,* 2001.

[25] J. Ye, "Characterization of a Family of Algorithms for Generalized Discriminant Analysis on Undersampled Problems," *J. Machine Learning Research,* vol. 6, pp. 483-502, 2005.

[26] J. Ye, Q. Li, H. Xiong, H. Park, R. Janardan, and V. Kumar, "IDR/QR: An Incremental Dimension Reduction Algorithm via QR Decomposition," *Proc. 10th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD '04),* pp. 364-373, 2004.

[27] J. Ye and T. Wang, "Regularized Discriminant Analysis for High Dimensional, Low Sample Size Data," *Proc. 12th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD '06),* pp. 454-463, 2006.

[28] J. Ye, "Least Squares Linear Discriminant Analysis," *Proc. 24th Int'l Conf. Machine Learning (ICML '07),* 2007.

Deng Cai received the BEng and MEng degrees in automation from Tsinghua University, China, in 2000 and 2003, respectively. He is currently working toward the PhD degree in the Department of Computer Science, University of Illinois, Urbana-Champaign. His research interests include machine learning, data mining, and information retrieval. He is a student member of the IEEE.

Xiaofei He received the BS degree from Zhejiang University, China, in 2000 and the PhD degree from the University of Chicago in 2005, both in computer science. He is currently a research scientist at Yahoo! Research. His research interests include machine learning, information retrieval, computer vision, and multimedia.

Jiawei Han is a professor in the Department of Computer Science, University of Illinois, Urbana-Champaign. He has been working on research into data mining, data warehousing, stream data mining, spatiotemporal and multimedia data mining, biological data mining, social network analysis, text and Web mining, and software bug mining, with more than 300 conference proceedings and journal publications. He has chaired or served in many program committees of international conferences and workshops. He also served or is serving on the editorial boards for *Data Mining and Knowledge Discovery*, *IEEE Transactions on Knowledge and Data Engineering*, *Journal of Computer Science and Technology*, and *Journal of Intelligent Information Systems*. He is currently serving as the founding editor in chief of *ACM Transactions on Knowledge Discovery from Data* and on the board of directors for the Executive Committee of the ACM Special Interest Group on Knowledge Discovery and Data Mining (SIGKDD). He is a fellow of the ACM and a senior member of the IEEE. He has received many awards and recognition, including the ACM SIGKDD Innovation Award in 2004 and the IEEE Computer Society Technical Achievement Award in 2005.