Rotational Linear Discriminant Analysis Technique for Dimensionality Reduction

Alok Sharma, *Member*, *IEEE*, and Kuldip K. Paliwal, *Member*, *IEEE*

Abstract—The linear discriminant analysis (LDA) technique is very popular in pattern recognition for dimensionality reduction. It is a supervised learning technique that finds a linear transformation such that the overlap between the classes is minimum for the projected feature vectors in the reduced feature space. This overlap, if present, adversely affects the classification performance. In this paper, we introduce prior to dimensionality-reduction transformation an additional rotational transform that rotates the feature vectors in the original feature space around their respective class centroids in such a way that the overlap between the classes in the reduced feature space is further minimized. As a result, the classification performance significantly improves, which is demonstrated using several data corpuses.

Index Terms—Rotational linear discriminant analysis, dimensionality reduction, classification error, fixed-point algorithm, probability of error.

1 Introduction

In a typical pattern recognition application, some characteristic properties (or features) of an object are measured, and the resulting feature vector is classified into one of the finite number of classes. When the number of features is relatively large, it becomes difficult to train a classifier using a finite amount of training data. In addition, the complexity of a classifier increases with the number of features used. In such situations, it becomes important to reduce the dimensionality of the feature space. There are a number of techniques proposed in the literature for dimensionality reduction [3], [7], [9]; the linear discriminant analysis (LDA) technique [7] is perhaps the most popular among them. LDA is a supervised learning technique that uses a linear transformation to project the feature vectors from the original feature space to a subspace in such a way that the overlap between the classes is minimized in the reduced feature space.

In order to provide an illustration, consider a two-dimensional feature space with three classes, as shown in Fig. 1a. When we use LDA to reduce the dimensionality to one, we get the orientation $\mathbf{W}$ along which the overlap between the classes is minimum. Note that the overlap between classes (though minimum) is still finite, and as a result, we get a finite amount of classification error. In order to reduce this classification error, we propose in this paper to use a (rotational) transform θ prior to LDA. The transform θ rotates the scatter (or spread) of each class around its own centroid; it is chosen in such a way that the overlap between the classes in the resulting LDA orientation is minimum (see Fig. 1b). It can be observed that classification error in Fig. 1b is less than that seen in Fig. 1a. Since we are using here the rotational transform with LDA, we call it the rotational LDA technique.

This paper is organized as follows: The conventional LDA method is described in Section 2. It provides a platform on which the rotational LDA method proposed in this paper is developed. Section 3 describes rotational LDA in detail. Section 4 describes experimental results where the rotational LDA method is evaluated with respect to its pattern classification performance on a number of data corpuses. A practical application of rotational LDA on a speaker identification task is described in Section 5. Conclusions are provided in Section 6.

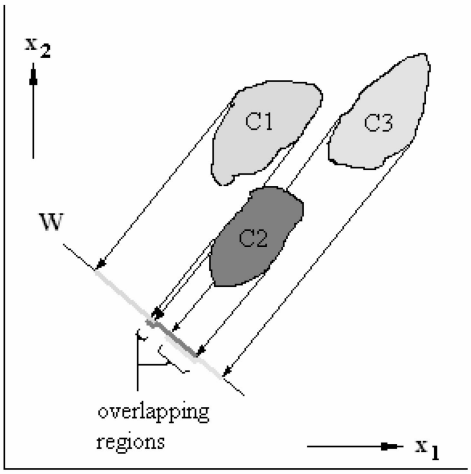
2 Linear Discriminant Analysis

LDA is a well-known technique for dimensionality reduction. It finds an orientation $\mathbf{W}$ that reduces high-dimensional feature vectors belonging to different classes to a lower dimensional feature space such that the projected feature vectors of a class on this lower dimensional space are well separated from the feature vectors of other classes. If the dimensionality reduction is from a d-dimensional $(\mathbf{R}^d)$ space to an h-dimensional $(\mathbf{R}^h)$ space (where $h < d$), then the size of the orientation matrix $\mathbf{W}$ is $d \times h$, and $\mathbf{W}$ has h column vectors known as the basis vectors. The orientation $\mathbf{W}$ is obtained by maximizing the Fisher's criterion function $J(\mathbf{W})$. This criterion function depends on three factors: orientation $\mathbf{W}$, within-class scatter matrix (S_W), and between-class scatter matrix (S_B). For a c-class problem, the value of h will be $c-1$ or less, a constraint due to S_B.

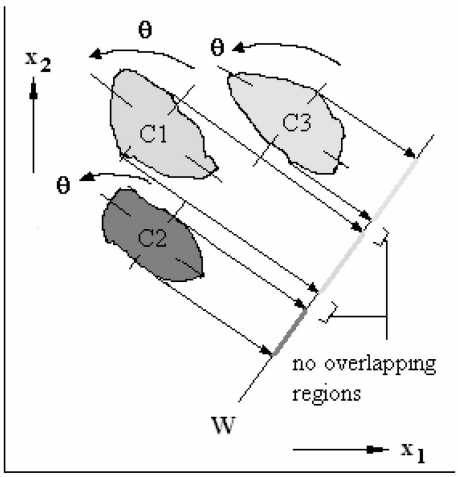
To define the LDA explicitly, let us consider a multiclass pattern classification problem with c classes. Let $\Omega = \{\omega_i : i = 1, 2, \ldots, c\}$ be the finite set of c class labels, where ω_i

- *A. Sharma is with the Signal Processing Laboratory, Griffith University, 170 Kessels Road, Nathan Campus, Nathan QLD 4111, Australia, and also with the School of Engineering and Physics, University of the South Pacific, Suva, Fiji. E-mail: sharma_al@usp.ac.fj.*
- *K.K. Paliwal is with the Signal Processing Laboratory, Griffith University, 170 Kessels Road, Nathan Campus, Nathan QLD 4111, Australia. E-mail: K.Paliwal@griffith.edu.au.*

Manuscript received 31 July 2007; revised 25 Feb. 2008; accepted 8 May 2008; published online 14 May 2008.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2007-07-0392. Digital Object Identifier no. 10.1109/TKDE.2008.101.



(a)



(b)

Fig. 1. A comparison between basic LDA and rotational LDA techniques.

denotes the ith class label. Let $\mathcal{X} = \{\mathbf{x}_j \in \mathbf{R}^d, j = 1, \ldots, n\}$ denote the set of n training samples (or feature vectors) in a d-dimensional space. The set $\mathcal{X}$ can be subdivided into c subsets $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_c$, where each subset $\mathcal{X}_i$ belongs to ω_i and consists of n_i feature vectors such that $n = \sum_{i=1}^{c} n_i$ and $\mathcal{X}_1 \cup \mathcal{X}_2 \cup, \ldots, \cup \mathcal{X}_c = \mathcal{X}$.

Let $\boldsymbol{\mu}_j$ be the centroid of $\mathcal{X}_j$ and $\boldsymbol{\mu}$ be the centroid of $\mathcal{X}$, then the between-class scatter matrix (S_B) is given as

$$S_B = \sum_{j=1}^{c} n_j (\boldsymbol{\mu}_j - \boldsymbol{\mu})(\boldsymbol{\mu}_j - \boldsymbol{\mu})^{\mathrm{T}}.$$

The within-class scatter matrix (S_W), which is the sum of c scatter matrices, is defined as

$$S_W = \sum_{i=1}^{c} M_i,$$

where $M_i = \sum_{\mathbf{x} \in \chi_i} (\mathbf{x} - \boldsymbol{\mu}_i)(\mathbf{x} - \boldsymbol{\mu}_i)^{\mathrm{T}}$.
Fisher's criterion as a function of $\mathbf{W}$ can be given as [7]

$$J(\mathbf{W}) = \frac{|\mathbf{W}^{\mathrm{T}} S_B \mathbf{W}|}{|\mathbf{W}^{\mathrm{T}} S_W \mathbf{W}|},$$

where $|\bullet|$ is the determinant. The orientation $\mathbf{W}$ is taken so that the Fisher's criterion function $J(\mathbf{W})$ is maximum. In a c-class problem, the LDA projects from a d-dimensional space to an h-dimensional space; i.e., $\mathbf{W} : \mathbf{x} \rightarrow \mathbf{y}$, or $\mathbf{y} = \mathbf{W}^{\mathrm{T}}\mathbf{x}$, where $\mathbf{x} \in \mathbf{R}^d$ and $\mathbf{y} \in \mathbf{R}^h$ such that $1 \leq h \leq c-1$. The orientation $\mathbf{W}$ is a rectangular matrix of size $d \times h$, which is the solution of the eigenvalue problem:

$$S_W^{-1} S_B \mathbf{w}_i = \lambda_i \mathbf{w}_i,$$

where $\mathbf{w}_i$ are the column vectors of $\mathbf{W}$ that correspond to the largest eigenvalues (λ_i).

In the conventional LDA technique, the Gaussian assumption is not required. However, the within-class scatter matrix (S_W) needs to be nonsingular. When the number of training samples is not adequate, this scatter matrix becomes singular. This occurs especially when the original feature space is very high. This drawback is considered to be the main problem of LDA and is known as the small sample size (SSS) problem [9]. The SSS problem has generated widespread interest among researchers, and a number of computational methods have been proposed in the literature to overcome this problem.

A simple and direct way to avoid the singularity problem is to replace the inverse of the within-class matrix by its pseudoinverse [19], [25]. However, this does not guarantee the optimality of Fisher's criterion. Another way to overcome the singularity problem is through the regularized LDA method [8], [11], [16], which adds a small positive constant to the diagonal elements of S_W to make it nonsingular. This method is also suboptimal as Fisher's criterion is not exactly maximized. Swets and Weng [24] and Belhumeur et al. [2] have proposed a two-stage method (known as the Fisherface method) to avoid the singularity problem. In the first stage, the principal component analysis (PCA) is used for dimensionality reduction in such a way that the within-class matrix in the reduced dimensional subspace becomes nonsingular. In the second stage, the classical LDA is used to reduce the dimensionality further. The two-stage Fisherface method (also known as the PCA+LDA method) is suboptimal as the PCA used in the first-stage loses some discriminative information. In order to avoid this loss in discriminative information, two other recently proposed two-stage methods to overcome the SSS problem are the null space method [6] and the direct LDA method [28]. In the null space method, the first stage transforms the training data vectors to the null space of the within-class scatter matrix S_W (i.e., it discards the range space of S_W). In the second stage, the dimensionality is reduced by choosing h eigenvectors of the transformed between-class scatter matrix corresponding to the highest eigenvalues. In the direct LDA method, the first stage discards the null space of the between-class scatter matrix S_B (i.e., the training data vectors are transformed into the range space of S_B), and the second stage reduces the dimensionality to h by choosing h eigenvectors of the transformed within-class scatter matrix corresponding to the lowest eigenvalues. Though these two-stage methods (the null space method and the direct LDA method) provide better classification performance than the PCA+LDA method, these are still suboptimum as they optimize Fisher's criterion sequentially in two stages.

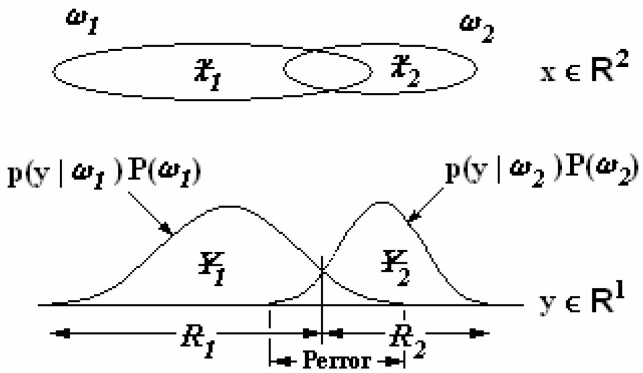


Fig. 2. Probability error for the two-class problem.

Here, we have provided a brief overview of some of the LDA methods to overcome the SSS problem; for other methods, see [5], [17], [18], [23], and [27]. It is important to note that though these methods overcome the SSS problem, they do not ensure the optimality of Fisher's criterion as done in the conventional LDA method and hence are suboptimal.

3 Rotational Linear Discriminant Analysis

3.1 Theory

This section provides the mathematical details of the rotational LDA method. Let us consider a multiclass pattern classification problem with c classes. Let $\Omega = \{\omega_i : i = 1, 2, \ldots, c\}$ be the finite set of c class labels. Let $\mathcal{X} = \{\mathbf{x}_j \in \mathbf{R}^d, j = 1, \ldots, n\}$ denote the set of n training samples (or feature vectors) in d-dimensional space.

Let $\mathcal{Y}_j \in \mathbf{R}^h$ (where $h < d$) be the reduced h-dimensional feature vector obtained from $\mathcal{X}_j \in \omega_j$ using LDA transformation and denote the set of reduced dimensional feature vectors by $\mathcal{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n\}$. Thus, $\mathcal{Y}_j \subset \mathcal{Y}$, and $\mathcal{Y}_1 \cup \mathcal{Y}_2 \cup \cdots \cup \mathcal{Y}_c = \mathcal{Y}$.

For illustration, consider a two-class problem ($c = 2$) as shown in Fig. 2, where $\mathcal{X}_1$ and $\mathcal{X}_2$ are the two subsets of feature vectors in the original two-dimensional space ($\mathbf{x} \in \mathbf{R}^2$). The class labels of these subsets are represented as ω_1 and ω_2, respectively. This original space is transformed to a lower one-dimensional subspace ($y \in \mathbf{R}^1$), producing transformed sample sets $\mathcal{Y}_1$ and $\mathcal{Y}_2$, which belong to the class labels ω_1 and ω_2, respectively. The transformation is conducted using an LDA transformation $\mathbf{W}$ of size $d \times h$ (in this illustration, $h = 1$); i.e., $\mathbf{W} : \mathbf{x} \rightarrow y$, or $y_j = \mathbf{W}^{\mathrm{T}}\mathbf{x}_j$ for $j = 1, \ldots, n$. The respective probability distributions of $\mathcal{Y}_1$ and $\mathcal{Y}_2$ are also shown in Fig. 2. The classifier divides the one-dimensional subspace into two regions $\mathcal{R}_1$ and $\mathcal{R}_2$. There are two possibilities in which a classification error could occur; either observation $y(\mathbf{W} : \mathbf{x} \rightarrow y)$ falls in the region $\mathcal{R}_1$ and the true class is ω_2 or y falls in the region $\mathcal{R}_2$ and the true class is ω_1. Since these events are mutually exclusive and exhaustive [7], we can define probability of error as

$$\begin{aligned}\mathrm{Perror} &= P(y \in \omega_2, \mathcal{R}_1) + P(y \in \omega_1, \mathcal{R}_2)\\ &= P(y \in \mathcal{R}_1|\omega_2)P(\omega_2) + P(y \in \mathcal{R}_2|\omega_1)P(\omega_1)\\ &= \int_{y \in \mathcal{R}_1} p(y|\omega_2)P(\omega_2)dy + \int_{y \in \mathcal{R}_2} p(y|\omega_1)P(\omega_1)dy,\end{aligned}$$

where $P(\omega_j)$ is the a priori probability of $\mathcal{Y}_j$. In a multiclass case, it would be easier to find the probability of being correct [7]. Therefore

$$\mathrm{Pcorrect} = \sum_{j=1}^{c} \int_{y \in \mathcal{R}_j} p(y|\omega_j)P(\omega_j)dy.$$

We can also compute the total probability of $\mathcal{Y}$ by evaluating the probability densities separately for each of $\mathcal{Y}_j$ and finally adding the computed densities; i.e.

$$\mathrm{Ptotal} = \sum_{j=1}^{c} \int_{y \in \mathcal{Y}_j} p(y|\omega_j)p(\omega_j)dy.$$

The Ptotal function is independent of regions $\mathcal{R}_j$; therefore, it will remain unchanged with respect to the values of $\mathcal{R}_j$. It can be observed that Pcorrect and Perror add up together to give Ptotal; i.e.

$$\mathrm{Ptotal} = \mathrm{Pcorrect} + \mathrm{Perror}.$$

Therefore, the probability error function can be written as

$$\begin{aligned}\mathrm{Perror} &= b - \mathrm{Pcorrect}\\ &= b - \sum_{j=1}^{c} \int_{y \in \mathcal{R}_j} p(y|\omega_j)P(\omega_j)dy,\end{aligned} \tag{1}$$

where b is a constant and is equal to Ptotal. In practice, we have to compute the integral in (1) using the data in the training set. Therefore, we approximate the integration operation by a summation operation as follows [1]:

$$A = \int f(x)dx = \lim_{n \rightarrow \infty} \sum_{k=1}^{n} f(x_k)\Delta x \approx \sum_{k=1}^{n} f(x_k)\Delta x. \tag{2}$$

Using this approximation (2), (1) can be rewritten as

$$\mathrm{Perror} = b - \sum_{j=1}^{c} \sum_{y \in \mathcal{R}_j} p(y|\omega_j)P(\omega_j)\Delta V, \tag{3}$$

where ΔV is a volume of a tiny hypercube. This ΔV is a scalar quantity and depends upon the transformed sample set $\mathcal{Y}$. Equation (3) is a probability error function for a scalar y, which can be extended to vector $\mathbf{y}$ simply by replacing a vector in place of scalar y.

Constant ΔV is independent of any class and therefore is taken outside from the summations of (3). The value of a priori probability $P(\omega_j) = n_j/n$ is substituted in (3); this yields Perror as

$$\mathrm{Perror} = b - k \sum_{j=1}^{c} n_j \sum_{y \in \mathcal{R}_j} p(\mathbf{y}|\omega_j), \tag{4}$$

where $k = \Delta V/n$ is a constant.

The basic LDA transformation ($\mathbf{y} = \mathbf{W}^{\mathrm{T}}\mathbf{x}$) has to be changed to account for the rotation of the original space. We introduce a rotational transform $\boldsymbol{\theta}$ prior to LDA and transform the feature vector $\mathbf{x}$ belonging to class ω_j as follows:

$$\mathbf{y} = \mathbf{W}^{\mathrm{T}}\left[\boldsymbol{\theta}^{\mathrm{T}}(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}_j}) + \boldsymbol{\mu}_{\mathbf{x}_j}\right], \tag{5}$$

where $\boldsymbol{\mu}_{\mathbf{x}_j}$ is the mean of $\mathcal{X}_j$.

At the beginning when no rotation has taken place, the transformation $\boldsymbol{\theta}$ would be a $d \times d$ identity matrix, and (5) would reduce to the basic LDA transformation. To find the optimum rotation $\boldsymbol{\theta}$, it is required to differentiate the scalar function (4) with respect to the transformation matrix $\boldsymbol{\theta}$. The value of $\boldsymbol{\theta}$ that corresponds to the minimum of Perror would be the optimum rotation $\boldsymbol{\theta}$. From (4), we get

$$\frac{\partial}{\partial \boldsymbol{\theta}} \text{Perror} = -k \sum_{j=1}^{c} n_j \sum_{\mathbf{y} \in \mathcal{R}_j} \frac{\partial}{\partial \boldsymbol{\theta}} p(\mathbf{y}|\omega_j), \tag{6}$$

where $\mathbf{y}$ is from (5). The next thing is to find the probability distribution $p(\mathbf{y}|\omega_j)$ before differentiating it with respect to $\boldsymbol{\theta}$. One way to estimate $p(\mathbf{y}|\omega_j)$ is to use a parametric technique where we assume a functional form of the distribution characterized by a few parameters. Here, we assume $\mathbf{y}$ to be distributed as multidimensional Gaussian.[1] Differentiating the resulting $p(\mathbf{y}|\omega_j)$, we get

$$\frac{\partial}{\partial \boldsymbol{\theta}} p(\mathbf{y}|\omega_j) = \frac{\partial}{\partial \boldsymbol{\theta}} \left\{ \frac{1}{(2\pi)^{h/2} \left| \boldsymbol{\Sigma}_{\mathbf{y}_j} \right|^{1/2}} \exp\left(-\frac{1}{2} \left(\mathbf{y} - \boldsymbol{\mu}_{\mathbf{y}_j} \right)^{\mathrm{T}} \Sigma_{\mathbf{y}_j}^{-1} \left(\mathbf{y} - \boldsymbol{\mu}_{\mathbf{y}_j} \right) \right) \right\}, \tag{7}$$

where $\boldsymbol{\mu}_{\mathbf{y}_j}$ and $\boldsymbol{\Sigma}_{\mathbf{y}_j}$ are the mean and the covariance of $\mathcal{Y}_j$. Substituting (5) in (7), we get

$$\begin{aligned} &\frac{\partial}{\partial \theta} p(\mathbf{y}|\omega_j) \\ &= \frac{\partial}{\partial \boldsymbol{\theta}} p(\mathbf{x}, \boldsymbol{\theta}, \mathbf{W}|\omega_j) \\ &= \frac{\partial}{\partial \boldsymbol{\theta}} \left\{ \frac{1}{(2\pi)^{h/2} \left| \boldsymbol{\Sigma}_{\mathbf{y}_j} \right|^{1/2}} \right. \\ &\qquad \left. \exp\left(-\frac{1}{2} \left(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}_j} \right)^{\mathrm{T}} \boldsymbol{\theta} \mathbf{W} \boldsymbol{\Sigma}_{\mathbf{y}_j}^{-1} \mathbf{W}^{\mathrm{T}} \boldsymbol{\theta}^{\mathrm{T}} \left(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}_j} \right) \right) \right\}, \end{aligned} \tag{8}$$

where $\mathbf{x}$ is the corresponding vector of $\mathbf{y} \in \mathcal{R}_j$; i.e., vectors $\mathbf{x} \in \mathcal{X}_j$ are used to compute $\mathbf{y} \in \mathcal{R}_j$ (using (5)) in (8). Let us represent this correspondence relation by $\mathbf{x}(\mathbf{y} \in \mathcal{R}_j)$. The following lemma would help in solving (8).

Lemma 1. *Let the scalar function $\exp(-\frac{1}{2}u)$ be a differentiable function of a $d \times d$ square matrix $\boldsymbol{\theta}$. Suppose*

$$u = \boldsymbol{\Lambda}^{\mathrm{T}} \boldsymbol{\theta} \mathbf{W} \mathbf{B} \mathbf{W}^{\mathrm{T}} \boldsymbol{\theta}^{\mathrm{T}} \boldsymbol{\Lambda},$$

where $\boldsymbol{\Lambda}$ is any vector of size $d \times 1$, $\mathbf{B}$ is a square matrix of size $h \times h$, and $\mathbf{W}$ is a rectangular matrix of size $d \times h$ such that $h < d$. It can be assumed that both the matrices ($\mathbf{B}$ and $\mathbf{W}$) and the vector ($\boldsymbol{\Lambda}$) are independent of $\boldsymbol{\theta}$. Then, the gradient of $\exp(-\frac{1}{2}u)$ is defined as $\nabla_{\boldsymbol{\theta}} \exp(-\frac{1}{2}u) = -\frac{1}{2}\exp(-\frac{1}{2}u)\boldsymbol{\Lambda}\boldsymbol{\Lambda}^{\mathrm{T}}\boldsymbol{\theta}\mathbf{W}(\mathbf{B} + \mathbf{B}^{\mathrm{T}})\mathbf{W}^{\mathrm{T}}$.

1. It should be noted that the class-conditional probability density function $p(\mathbf{y}|\omega_j)$ is assumed here to be a multidimensional Gaussian function for analytical simplicity purposes. But this choice is justifiable from the central limit theorem as $\mathbf{y}$ is computed from $\mathbf{x}$ through a linear transformation (see (5)).

Proof 1. The derivative of scalar $\exp(-\frac{1}{2}u)$ with respect to matrix $\boldsymbol{\theta}$ can be given as

$$\begin{aligned} &\partial \exp\left(-\frac{1}{2}u\right) \\ &= -\frac{1}{2}\exp\left(-\frac{1}{2}u\right) trace\left[\partial(\boldsymbol{\Lambda}^{\mathrm{T}}\boldsymbol{\theta}\mathbf{W}\mathbf{B}\mathbf{W}^{\mathrm{T}}\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\Lambda})\right] \\ &= -\frac{1}{2}\exp\left(-\frac{1}{2}u\right)\Big\{ trace[\boldsymbol{\Lambda}^{\mathrm{T}}\partial\,\boldsymbol{\theta}\mathbf{W}\mathbf{B}\mathbf{W}^{\mathrm{T}}\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\Lambda})] \\ &\qquad + trace[\boldsymbol{\Lambda}^{\mathrm{T}}\boldsymbol{\theta}\mathbf{W}\mathbf{B}\mathbf{W}^{\mathrm{T}}\partial\,\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\Lambda}]\Big\} \\ &= -\frac{1}{2}\exp\left(-\frac{1}{2}u\right)\Big\{ trace[\boldsymbol{\Lambda}^{\mathrm{T}}\boldsymbol{\theta}\,\mathbf{W}\mathbf{B}^{\mathrm{T}}\mathbf{W}^{\mathrm{T}}\partial\,\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\Lambda})] \\ &\qquad + trace[\boldsymbol{\Lambda}^{\mathrm{T}}\boldsymbol{\theta}\mathbf{W}\mathbf{B}\mathbf{W}^{\mathrm{T}}\partial\,\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\Lambda}]\Big\} \\ &\qquad \left(\because tr(A^T) = tr(\mathrm{A})\right) \\ &= -\frac{1}{2}\exp\left(-\frac{1}{2}u\right)\left\{ trace\left[\boldsymbol{\Lambda}^{\mathrm{T}}\boldsymbol{\theta}\mathbf{W}(\mathbf{B} + \mathbf{B}^{\mathrm{T}})\mathbf{W}^{\mathrm{T}}\partial\,\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\Lambda})\right]\right\} \\ &= -\frac{1}{2}\exp\left(-\frac{1}{2}u\right)\left\{ trace\left[\boldsymbol{\Lambda}\boldsymbol{\Lambda}^{\mathrm{T}}\boldsymbol{\theta}\mathbf{W}(\mathbf{B} + \mathbf{B}^{\mathrm{T}})\mathbf{W}^{\mathrm{T}}\partial\,\boldsymbol{\theta}^{\mathrm{T}})\right]\right\} \\ &\qquad \left(\because tr(AD) = tr(DA)\right) \\ &\therefore \nabla_{\boldsymbol{\theta}}\exp\left(-\frac{1}{2}u\right) = -\frac{1}{2}\exp\left(-\frac{1}{2}u\right)\boldsymbol{\Lambda}\boldsymbol{\Lambda}^{\mathrm{T}}\boldsymbol{\theta}\mathbf{W}(\mathbf{B} + \mathbf{B}^{\mathrm{T}})\mathbf{W}^{\mathrm{T}}. \end{aligned}$$

□

Using Lemma 1, we can rewrite (8) as

$$\begin{aligned} \frac{\partial}{\partial \boldsymbol{\theta}} p(\mathbf{y}|\omega_j) = &\left[-\frac{1}{2(2\pi)^{h/2}|\boldsymbol{\Sigma}_{\mathbf{y}_j}|^{1/2}} \exp\left(-\frac{1}{2}u\right) \right] \\ &\cdot \left[\left(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}_j}\right)\left(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}_j}\right)^{\mathrm{T}} \boldsymbol{\theta}\mathbf{W}\left(\boldsymbol{\Sigma}_{\mathbf{y}_j}^{-1} + \boldsymbol{\Sigma}_{\mathbf{y}_j}^{-1\mathrm{T}}\right)\mathbf{W}^{\mathrm{T}} \right], \end{aligned} \tag{9}$$

where $u = (\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}_j})^{\mathrm{T}}\boldsymbol{\theta}\mathbf{W}\Sigma_{\mathbf{y}_j}^{-1}\mathbf{W}^{\mathrm{T}}\boldsymbol{\theta}^{\mathrm{T}}(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}_j})$. Substituting (9) in (6), we get

$$\begin{aligned} \frac{\partial}{\partial \boldsymbol{\theta}} \text{Perror} = k' \sum_{j=1}^{c} \frac{n_j}{|\boldsymbol{\Sigma}_{\mathbf{y}_j}|^{1/2}} \sum_{\mathbf{x}(\mathbf{y} \in \mathcal{R}_j)} &\exp\left(-\frac{1}{2}u\right) \\ &\times \left[\left(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}_j}\right)\left(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}_j}\right)^{\mathrm{T}} \boldsymbol{\theta}\mathbf{W}\left(\boldsymbol{\Sigma}_{\mathbf{y}_j}^{-1} + \boldsymbol{\Sigma}_{\mathbf{y}_j}^{-1\mathrm{T}}\right)\mathbf{W}^{\mathrm{T}} \right], \end{aligned} \tag{10}$$

where $k' = k/(2(2\pi)^{h/2})$. Equation (10) can also be written in the expectation form as

$$\frac{\partial}{\partial \boldsymbol{\theta}} \text{Perror} = k' \sum_{j=1}^{c} \frac{n_j^2}{\left|\boldsymbol{\Sigma}_{\mathbf{y}_j}\right|^{1/2}} \underset{\mathbf{x}(\mathbf{y} \in \mathcal{R}_j)}{E} \left[\mathrm{F}\left(\mathbf{x}, \boldsymbol{\theta}, \mathbf{W}, \boldsymbol{\mu}_{\mathbf{x}_j}, \boldsymbol{\Sigma}_{\mathbf{y}_j}\right) \right],$$

where

$$\begin{aligned} \mathrm{F}\left(\mathbf{x}, \boldsymbol{\theta}, \mathbf{W}, \boldsymbol{\mu}_{\mathbf{x}_j}, \boldsymbol{\Sigma}_{\mathbf{y}_j}\right) = &\left[\exp\left(-\frac{1}{2}u\right) \right] \\ &\cdot \left[\left(\mathbf{x} - \mu_{\mathbf{x}_j}\right)\left(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}_j}\right)^{\mathrm{T}} \boldsymbol{\theta}\mathbf{W} \right. \\ &\left. \times \left(\boldsymbol{\Sigma}_{\mathbf{y}_j}^{-1} + \boldsymbol{\Sigma}_{\mathbf{y}_j}^{-1\mathrm{T}}\right)\mathbf{W}^{\mathrm{T}} \right], \end{aligned}$$

and $E[F(\bullet)]$ is the expectation of $\mathrm{F}(\bullet)$ with respect to $\mathbf{x}$.

TABLE 1
The Rotational LDA Algorithm

1. Find the mean of each class $\boldsymbol{\mu}_{\mathbf{x}_j} \in \mathbf{R}^d$ for $j = 1 \ldots c$.
2. Initialize $\boldsymbol{\theta} \leftarrow \mathrm{I}_{d\times d}$, $\hat{\boldsymbol{\theta}} \leftarrow \mathrm{I}_{d\times d}$, $\mathrm{P}_0 = \mathrm{Perror} \leftarrow 100\%$ and set counter $m \leftarrow 0$.
3. while (true)
4. Increment counter $m \leftarrow m+1$.
5. Apply basic LDA to find orientation $\mathbf{W}$, the transformed samples $\mathbf{y} \in \mathbf{R}^h$ and $\boldsymbol{\mu}_{\mathbf{y}_j} \in \mathbf{R}^h$ using $\mathcal{X}$, i.e. $[\mathbf{y}, \mathbf{W}, \boldsymbol{\mu}_{\mathbf{y}_j}] \leftarrow$ basic_LDA_method ($\mathcal{X}$).
6. Perform classification (to find $\mathcal{R}_j$) and compute P_m (new Perror).
7. Check if P_m is decreasing
   if ($\mathrm{P}_m > \mathrm{P}_{m-1}$)
   Store $\mathbf{W}$, $\boldsymbol{\theta}$ and centroid of each class $\boldsymbol{\mu}_{\mathbf{y}_j} \in \mathbf{R}^h$.
   break
   end
8. Compute covariance $\boldsymbol{\Sigma}_{\mathbf{y}_j}$ (note covariance is computed here instead of step 5 since if the 'break' occurs then it would be useless to compute it at step 5).
9. Update rotation matrix
$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}\hat{\boldsymbol{\theta}}$$
$$\hat{\boldsymbol{\theta}} \leftarrow \sum_{j=1}^{c} \frac{n_j^2}{|\Sigma_{\mathbf{y}_j}|^{1/2}} \underset{\mathbf{x}(\mathbf{y}\in\mathcal{R}_j)}{E} [\mathrm{F}(\mathbf{x}, \hat{\boldsymbol{\theta}}, \mathbf{W}, \boldsymbol{\mu}_{\mathbf{x}_j}, \Sigma_{\mathbf{y}_j})]$$
10. Orthonormalize rotation matrix
$$\hat{\boldsymbol{\theta}} \leftarrow \hat{\boldsymbol{\theta}}(\hat{\boldsymbol{\theta}}^{\mathrm{T}}\hat{\boldsymbol{\theta}})^{-1/2}$$
$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}(\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\theta})^{-1/2}$$
11. Update $\mathcal{X}_j \leftarrow \hat{\boldsymbol{\theta}}^{\mathrm{T}}(\mathcal{X}_j - \boldsymbol{\mu}_{\mathbf{y}_j}) + \boldsymbol{\mu}_{\mathbf{y}_j}$ for $j = 1 \ldots c$
12. end

Since the topography of the original data should remain unchanged during rotation $\boldsymbol{\theta}$, the column vectors of $\boldsymbol{\theta}$ should be orthonormal. This means that the square matrix $\boldsymbol{\theta}$ is orthonormal; i.e., $\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\theta} = \mathrm{I}_{d\times d}$. This allows us to use a fixed-point algorithm [14], which is known to be faster and more reliable than the gradient algorithms. The fixed-point algorithm for rotation $\boldsymbol{\theta}$ is written as follows:

$$\boldsymbol{\theta} \propto \sum_{j=1}^{c} \frac{n_j^2}{\left|\Sigma_{\mathbf{y}_j}\right|^{1/2}} \underset{\mathbf{x}(\mathbf{y}\in\mathcal{R}_j)}{E} \left[\mathrm{F}\left(\mathbf{x}, \boldsymbol{\theta}, \mathbf{W}, \boldsymbol{\mu}_{\mathbf{x}_j}, \Sigma_{\mathbf{y}_j}\right)\right], \tag{11}$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}(\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\theta})^{-1/2}. \tag{12}$$

The values of $\mathbf{W}$ and $\Sigma_{\mathbf{y}_j}$ will be changing depending upon the rotation of the original feature space, whereas the class centroid $\boldsymbol{\mu}_{\mathbf{x}_j}$ will remain invariant for any such rotation since the rotation of the training vectors of a class is always with respect to its centroid $\boldsymbol{\mu}_{\mathbf{x}_j}$. The matrices $\mathbf{W}$ and $\Sigma_{\mathbf{y}_j}$ should be updated for every iteration of $\boldsymbol{\theta}$ for (11). The inverse of $\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\theta}$ in (12) is computed using eigenvalue decomposition. There are iterative methods for orthonormalization that avoid the matrix inverse and eigendecomposition. In that case, the rotation matrix $\boldsymbol{\theta}$ can be orthonormalized by using a symmetric orthonormalization procedure starting from a nonorthogonal matrix and continuing the iterative process until $\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\theta} \approx \mathrm{I}_{d\times d}$ [13].

TABLE 2
List of Parameters Computed during the Training Phase Which Will Be Required in the Testing Phase

Parameters	Unit Size	Total size for c classes
$\boldsymbol{\theta}$	$d \times d$	d^2
$\mathbf{W}$	$d \times h$	dh
$\boldsymbol{\mu}_{\mathbf{y}_j}$	$h \times 1$	ch

The value of Perror can be estimated more economically also for each of the iteration of (11) and (12) by applying the following equation instead of applying (1):

$$\begin{aligned}\mathrm{Perror} &= 1 - \frac{\sum_{j=1}^{c} \text{number of samples belongs to } \mathcal{R}_j \text{ given } \omega_j}{\text{total number of samples in } \mathcal{X}} \\ &= 1 - \frac{\sum_{j=1}^{c} (n_j | \mathcal{R}_j, \omega_j)}{n},\end{aligned} \tag{13}$$

where $(n_j|\mathcal{R}_j, \omega_j)$ denotes the number of samples that belong to the jth region $(\mathcal{R}_j)$ given class label ω_j. Region $\mathcal{R}_j$ of the training samples can be obtained by several methods. We have used the nearest neighbor classifier (with squared euclidean distance measure) for finding the regions.

3.2 Training Phase of Rotational LDA

The previous section has provided a mathematical description of the rotational LDA method. In this section, we provide its algorithmic description. As mentioned earlier, the rotational LDA algorithm is iterative in nature. It is given in Table 1. This algorithm computes parameters $\boldsymbol{\theta}$, $\mathbf{W}$, and $\boldsymbol{\mu}_{\mathbf{y}_j}$ $(j = 1, \ldots, c)$ (listed in Table 2) from the training data as follows:

Prior to the iterative process, compute $\boldsymbol{\mu}_{\mathbf{x}_j} \in \mathbf{R}^d$ $(j = 1, \ldots, c)$ from the training data and initialize the rotation $\boldsymbol{\theta}$ by a $d \times d$ identity matrix. In the first iteration, the first step is to find the orientation W (using $\boldsymbol{\theta}$ to be the identity matrix from the initialization step) by applying the basic LDA procedure. The obtained orientation $\mathbf{W}$ will be such that the overlap between classes is minimum in the reduced dimensional space (i.e., $\mathbf{W}$ maximizes Fisher's criterion). This transformation may produce overlapping of samples in the reduced dimensional space between adjacent classes that cannot be reduced any further by moving the direction (orientation) $\mathbf{W}$ around the origin in the reduced dimensional feature space. Let P_1 be the error with computed $\mathbf{W}$ and $\boldsymbol{\theta}$ from the previous iteration. In the second step, compute the rotation $\boldsymbol{\theta}$ in the original feature space using the fixed-point algorithm. By applying this rotation $\boldsymbol{\theta}$ in the original feature space, we get a reduced dimensional feature space with much less overlapping between the adjacent classes. With this $\boldsymbol{\theta}$, we go

TABLE 3
List of the Values of Parameters during an Example Run

Initial dimension $d = 2$, *reduced dimension* $h = 1$, *class* $= 3$, $P_0 = 100\%$

$\boldsymbol{\mu}_{\mathbf{x}_1} = [63.6283,\ 96.0867]^{\mathrm{T}}$, $\boldsymbol{\mu}_{\mathbf{x}_2} = [50.5850,\ 43.0275]^{\mathrm{T}}$, $\boldsymbol{\mu}_{\mathbf{x}_3} = [86.8150,\ 104.3275]^{\mathrm{T}}$

Iteration m	Perror (P_m)	$\mathbf{W}$	$\boldsymbol{\theta}$	$\hat{\boldsymbol{\theta}}$				Figure
					$\boldsymbol{\mu}_{\mathbf{y}_1}$	$\boldsymbol{\mu}_{\mathbf{y}_2}$	$\boldsymbol{\mu}_{\mathbf{y}_3}$	
1	21.1667	$\begin{bmatrix} 0.8837 \\ -0.4680 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0.9070 & -0.4211 \\ 0.4211 & 0.9070 \end{bmatrix}$	11.2658	24.5688	27.9005	3a
					$\boldsymbol{\Sigma}_{\mathbf{y}_1}$	$\boldsymbol{\Sigma}_{\mathbf{y}_2}$	$\boldsymbol{\Sigma}_{\mathbf{y}_3}$	
					23.4744	8.7760	9.2393	
2	18.7222	$\begin{bmatrix} 0.5107 \\ -0.8598 \end{bmatrix}$	$\begin{bmatrix} 0.9070 & -0.4211 \\ 0.4211 & 0.9070 \end{bmatrix}$	$\begin{bmatrix} 0.8396 & -0.5432 \\ -0.5432 & -0.8396 \end{bmatrix}$	$\boldsymbol{\mu}_{\mathbf{y}_1}$	$\boldsymbol{\mu}_{\mathbf{y}_2}$	$\boldsymbol{\mu}_{\mathbf{y}_3}$	3b
					-50.1910	-11.1612	-45.3634	
					$\boldsymbol{\Sigma}_{\mathbf{y}_1}$	$\boldsymbol{\Sigma}_{\mathbf{y}_2}$	$\boldsymbol{\Sigma}_{\mathbf{y}_3}$	
					27.2460	13.7915	8.2005	
3	0.6667	$\begin{bmatrix} 0.9173 \\ 0.3981 \end{bmatrix}$	$\begin{bmatrix} 0.9903 & -0.1391 \\ -0.1391 & -0.9903 \end{bmatrix}$	$\begin{bmatrix} 0.7260 & -0.6877 \\ -0.6877 & -0.7260 \end{bmatrix}$	$\boldsymbol{\mu}_{\mathbf{y}_1}$	$\boldsymbol{\mu}_{\mathbf{y}_2}$	$\boldsymbol{\mu}_{\mathbf{y}_3}$	3c
					96.6242	63.5340	121.1749	
					$\boldsymbol{\Sigma}_{\mathbf{y}_1}$	$\boldsymbol{\Sigma}_{\mathbf{y}_2}$	$\boldsymbol{\Sigma}_{\mathbf{y}_3}$	
					24.7971	10.2408	8.3090	

to the second iteration and compute $\mathbf{W}$ in the first step (which gives error $P_2 < P_1$) and $\boldsymbol{\theta}$ in the second step. This iterative process is continued as long as the reduction in error between successive iterations is significant. For the data sets used in this paper, we have observed that this algorithm needs only a few iterations (typically two to four) to converge.

We know that the rotational LDA algorithm is an iterative algorithm, and it rotates the original feature vectors with respect to the centroid of their own class separately, until the minimum overlapping error is obtained. It would be interesting to see what happens to the original feature vectors $\mathbf{x} \in \chi_j \in \mathbf{R}^d$ after the mth iteration of the algorithm. Let us denote the rotated feature vectors after the first iteration as $\mathbf{x}_1$. It can then be expressed in terms of the original feature vectors $\mathbf{x}$ as

$$\text{Iteration } 1: \quad \mathbf{x}_1 = \boldsymbol{\theta}_1^{\mathrm{T}}\left(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}_j}\right) + \boldsymbol{\mu}_{\mathbf{x}_j}. \tag{14}$$

After the second iteration, the feature vectors would be

$$\text{Iteration } 2: \quad \mathbf{x}_2 = \boldsymbol{\theta}_2^{\mathrm{T}}\left(\mathbf{x}_1 - \boldsymbol{\mu}_{\mathbf{x}_j}\right) + \boldsymbol{\mu}_{\mathbf{x}_j}. \tag{15}$$

Similarly, after the mth iteration, feature vectors can be given as

$$\text{Iteration } m: \quad \mathbf{x}_m = \boldsymbol{\theta}_m^{\mathrm{T}}\left(\mathbf{x}_{m-1} - \boldsymbol{\mu}_{\mathbf{x}_j}\right) + \boldsymbol{\mu}_{\mathbf{x}_j}. \tag{16}$$

It should be noted that the location of the centroid of a class is not changing since the rotation of feature vectors is always with respect to the centroid of their own class.

Substituting (14) in (15), we get

$$\mathbf{x}_2 = \boldsymbol{\theta}_2^{\mathrm{T}}\boldsymbol{\theta}_1^{\mathrm{T}}\left(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}_j}\right) + \boldsymbol{\mu}_{\mathbf{x}_j}.$$

Similarly, we can say that

$$\mathbf{x}_m = \boldsymbol{\theta}_m^{\mathrm{T}} \ldots \boldsymbol{\theta}_2^{\mathrm{T}}\boldsymbol{\theta}_1^{\mathrm{T}}\left(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}_j}\right) + \boldsymbol{\mu}_{\mathbf{x}_j},$$

or

$$\mathbf{x}_m = \boldsymbol{\theta}^{\mathrm{T}}(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}_j}) + \boldsymbol{\mu}_{\mathbf{x}_j},$$

where $\boldsymbol{\theta} = \theta_1, \theta_2, \ldots, \theta_m$. Some issues related to the computational complexity of the training phase are discussed in the Appendix.

3.3 An Example of the Training Phase of Rotational LDA

This section provides an example that illustrates the training phase of the rotational LDA algorithm. For this purpose, the Sat-Image data set from the UCI repository [4] is used. The Sat-Image data set consists of six distinct classes with 36 features. It consists of 4,435 feature vectors for training purposes and 2,000 feature vectors for testing purposes. However, in this example, we have used only the first three classes (with 600 training vectors from Class 1, 400 from Class 2, and 800 from Class 3) and consider only the first two features. Thus, we have here 1,800 feature vectors in two-dimensional space for training. The rotational LDA algorithm is applied to these training vectors, and the values of the resulting parameters ($\mathbf{W}$, $\boldsymbol{\theta}$ and centroid of each class $\boldsymbol{\mu}_{\mathbf{y}_j} \in \mathbf{R}^h$) are given in Table 3.

In this example, the dimensionality of the original feature space is two, and it is reduced to one for recognition and/or classification purposes. It can be seen in Table 3 that the algorithm converged at the third iteration. At the first iteration, there is no rotation of the original feature space, only the conventional LDA method is applied to compute the value of orientation $\mathbf{W}$. The overlapping error is noted to be 21.17 percent at the first iteration (without any rotation). When the first rotation is applied at the second iteration, the error reduces to 18.72 percent and to only 0.67 percent at the third iteration (on the application of second rotation). This example is also illustrated in Figs. 3a, 3b, and 3c for iterations 1, 2, and 3, respectively. In all the three figures, the projection of feature vectors is illustrated from a two-dimensional space onto a one-dimensional

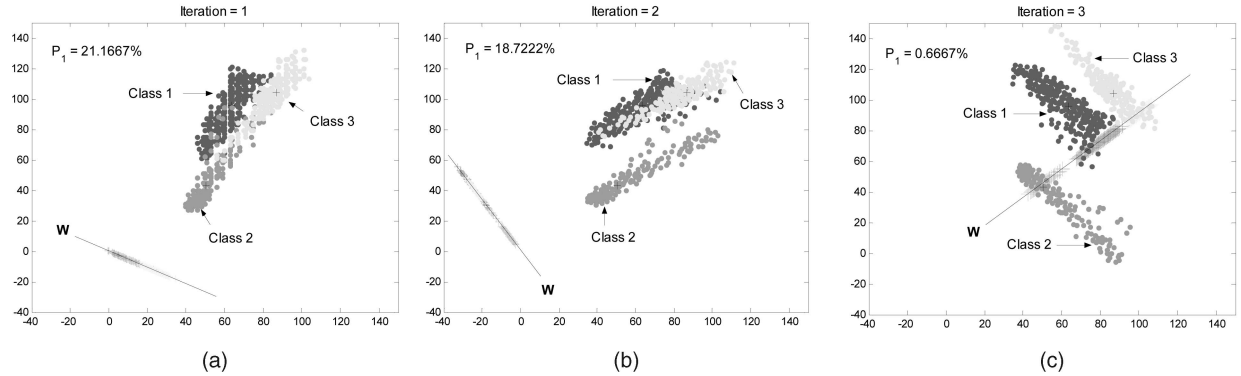


Fig. 3. Illustration of the transformation from a two-dimensional feature space to a one-dimensional feature plane using the rotational LDA method on a three-class problem.

subspace for all the three classes. Fig. 3a depicts the projection when only the basic LDA method is applied (i.e., iteration 1). Thereafter, rotation is applied, which is depicted in Figs. 3b and 3c. It is quite clear from all the three figures that the rotation does help in minimizing the overlapping error significantly in the transformed space, which is not possible by the LDA method.

3.4 Classification Phase of Rotational LDA

The strategy of classifying or testing a test vector using the rotational LDA algorithm is slightly different from the basic LDA algorithm. In rotational LDA, each feature vector in the original d-dimensional space is rotated by the $d \times d$ rotational matrix $\boldsymbol{\theta}$ around its own class centroid (see (5)). It is possible to carry out this rotational transform on each feature vector in the training data set as the class labels of all the training feature vectors are known (since the rotational LDA method is a supervised learning method). But it is not possible to do it for a test vector because we do not know its class label (and, thus, its class centroid). However, this problem can be solved if we can get an estimate of this centroid. For this, we take L test vectors that belong to the same class as the test vector and use their mean value as an estimate of the class centroid. We will discuss this issue further in Section 5, where we use the rotational LDA method in a practical application. The procedure followed in the classification phase is outlined in Table 4.

TABLE 4
Classification Phase of the Rotational LDA Algorithm

1. Take L test vectors $\hat{\mathcal{X}}_L = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_L\}$ such that $\hat{\mathcal{X}}_L \in \omega_k$ where k is unknown.
2. Find the rotation of these L test vectors with respect to the centroid of $\hat{\mathcal{X}}_L$ i.e.
$$\mathbf{x}_{rot} = \boldsymbol{\theta}^{\mathrm{T}}(\mathbf{x} - \boldsymbol{\mu}_L) + \boldsymbol{\mu}_L$$
where $\mathbf{x} \in \hat{\mathcal{X}}_L$ and $\boldsymbol{\mu}_L = \frac{1}{L}\sum_{j=1}^{L} \mathbf{x}_j$.
3. Transform $\mathbf{x}_{rot}$ to lower dimensional space by $\mathbf{W}$
$$\mathbf{y} = \mathbf{W}^{\mathrm{T}} \mathbf{x}_{rot}$$
4. Associate the class label of the closest centroid to $\mathbf{y}$
$$k = \arg\min_{j=1}^{c} \| \mathbf{y} - \boldsymbol{\mu}_{\mathbf{y}_j} \|^2$$

In Table 4, we are using the nearest centroid classifier (with squared euclidean distance measure) to classify the test vectors. If we want, we can also use a Bayesian classifier for this purpose. In that case, step 4 in Table 4 will be replaced by a Bayes decision rule as

$$k = \arg\max_{j=1}^{c} \left[p(\mathbf{y}|\omega_j) P(\omega_j) \right],$$

where $p(\mathbf{y}|\omega_j)$ is the class-conditional probability density function of $\mathbf{y}$ and $P(\omega_j)$ is the a priori probability of class ω_j, as described in Section 3.

The computational complexity of the classification phase is measured here in terms of flop counts, where each floating-point arithmetic operation (addition, subtraction, multiplication, or division) is counted as one flop unit. It is listed in Table 5. Here, the computational complexity of step 2 is dominating the other steps. Thus, the total computational complexity of the classification phase is estimated to be $O(2d^2 + dL + d)$ flop counts.

4 Experimental Results and Discussion

In this section, we conduct experiments where we apply the proposed rotational LDA algorithm on a number of publicly available data corpuses and compare it with the conventional LDA method in terms of classification performance. We use here a nearest centroid classifier (with squared

TABLE 5
Computational Complexity of the Classification Phase of the Rotational LDA Algorithm

No. of step from table 4	Computational complexity
step 2	$O(2d^2 + dL + d)$
step 3	$O(2dh)$
step 4	$O(3ch + c)$

TABLE 6
Details of the Data Sets Used for the Experimentation

Name of dataset	Number of classes	Dimensions d	Number of vectors used in training phase	Number of vectors used in testing phase
Sat-Image	6	36	4435	2000
Mfeat-Fourier coefficients	10	76	1500	500
Mfeat-Zernike moments	10	47	1500	500
Mfeat-pixel averages	10	240	1500	500
Mfeat-KL	10	64	1500	500
TIMIT	10	39	9357	3222
Waveform	3	21	3600	1400

euclidean distance measure) to find the class label of a test vector. For evaluating the performance of the proposed LDA method, we use here the following seven data corpuses: Sat-Image data set [4], Waveform data set [4], TIMIT[2] data set [10], and multiple features (Mfeat) data sets for Karhunen-Loéve coefficients, Fourier coefficients, Zernike moments, and pixel averages [4], [15]. The relevant details of these data sets useful for defining the experimental setup are listed in Table 6.

To conduct the experimentation, original feature vectors are reduced from a d-dimensional space to an h-dimensional space, where $h = 1, 2, \ldots, c-1$ since $d > c$ for all the databases. The "classification error in percentage" (ε) is reported for all the h dimensions. The lesser is the classification error, the better is the performance of the algorithm. For rotational LDA, $L = 10$ vectors belonging to the same class as the test vector to estimate the class centroid around which the test vector is rotated by $\boldsymbol{\theta}$ (using (5)). Perror is also reported for each of the iterations until convergence is reached. The results are provided in Table 7 for all the databases. We also report in this table results when no LDA method is used (i.e., no dimensionality reduction is done) and the nearest centroid classifier (with squared euclidean distance measure) is applied in the original d-dimensional space to get the classification performance. These results are reported in the last row of this table (below the dotted line) for each data set. In Table 7, there are some blank spaces (marked as "$-$") under iteration-III columns; this means that the rotational LDA algorithm converged prior to iteration III (i.e., at iteration II).

In Table 7, it is evident that the rotational LDA method is performing far better than the LDA technique in terms of reducing the classification error for all the data corpuses. We also observe from this table that the rotational algorithm also performs much better than the case when no dimensionality reduction is done, even though the former case uses a much smaller number of features than the later case (i.e., $h \leq c - 1 < d$).

2. From the TIMIT corpus, a set of 10 distinct vowels are extracted; then, each vowel is divided into three segments, and each segment is used in getting mel-frequency cepstral coefficients with energy-delta-acceleration (MFCC_E_D_A) feature vectors [26].

The minimum classification error for the Sat-Image data set produced by LDA is 19.2 percent, whereas it is only 1.1 percent by the rotational LDA algorithm. Similarly, for Mfeat-Fourier coefficients, Mfeat-Zernike moments, Mfeat-pixel averages, Mfeat-Karhunen-Loéve coefficients, TIMIT, and Waveform databases, the minimum classification error produced by LDA are 19.0 percent, 19.8 percent, 4.2 percent, 5.0 percent, 11.2 percent, and 17.8 percent, respectively, whereas that by the rotational LDA are 3.8 percent, 12.0 percent, 0.0 percent, 0.2 percent, 5.7 percent, and 4.1 percent, respectively. Thus, the minimum classification error rates are better for the rotational LDA for all the data sets used in the paper. It can also be observed that rotational LDA is producing a better classification error rate at very low-dimensional space (one or two) for all the data sets except for the TIMIT data set. Nonetheless, the classification error for TIMIT reduces gradually and becomes better than the LDA algorithm when dimension h is increased. In general, we expect the error rate to decrease with an increase in the value of h, but it is possible that after a certain value of h, the error may start increasing. This is due to the "curse of dimensionality" occurring in pattern recognition due to the limited size of the training data [7]. This effect has been observed in Mfeat data. If we could make the size of the training data to be arbitrarily large, we would expect the error rate to decrease with an increase in the value of h.

Thus, we can conclude that the rotational LDA method performs better than the conventional LDA method. However, it is important to note here that the rotational LDA technique assumes the class-conditional probability density function $p(\mathbf{y}|\omega_j)$ to be multidimensional Gaussian for analytical simplicity purposes. Though this is justifiable from the central limit theorem as $\mathbf{y}$ is computed from $\mathbf{x}$ through a linear transformation (see (5)) and the performance of the rotational LDA method is good (as seen from Table 7) with this Gaussian assumption, one may wonder that we get this good performance with the relatively large training sizes of databases (as shown in Table 6). A natural question that comes to mind at this stage is how will the rotational LDA method perform when the training size (i.e., the number of vectors used for training) is reduced; i.e., we want to know how the Gaussian assumption affects the

TABLE 7
A Comparison of Algorithms Using Classification Error in Percentage (ε) as a Prototype

Sat-Image

h	LDA	Rotational LDA			
			Perror – iterations		
	ε	ε	I	II	III
1	50.9	18.9	49.36	0.18	0.14
2	27.4	2.5	25.32	0	-
3	19.2	1.8	17.79	0	-
4	19.2	1.6	17.61	0	-
5	19.2	1.1	17.54	0	-

With no dim. reduction, $\varepsilon = 23.4$.

Mfeat – pixel averages

h	LDA	Rotational LDA			
			Perror – iterations		
	ε	ε	I	II	III
1	56.2	47.2	52.80	7.67	7.00
2	27.2	11.2	22.13	0.07	0
3	15.6	0	10.07	0	-
4	8.8	0	4.00	0	-
5	7.2	0	2.47	0	-
6	6.2	0	2.13	0	-
7	4.2	0	1.73	0	-
8	4.8	0	1.53	0	-
9	5.6	0	1.60	0	-

With no dim. reduction, $\varepsilon = 6.2$.

Waveform

h	LDA	Rotational LDA			
			Perror – iterations		
	ε	ε	I	II	III
1	38.1	10.4	40.22	28.11	4.92
2	17.8	4.1	16.03	9.75	1.22

With no dim. reduction, $\varepsilon = 20$.

Mfeat – Fourier coefficients

h	LDA	Rotational LDA			
			Perror – iterations		
	ε	ε	I	II	III
1	55.2	43.0	56.33	28.00	-
2	31.0	24.8	30.40	16.27	-
3	25.6	10.2	24.00	6.60	6.13
4	24.2	8.8	19.33	6.27	-
5	23.2	7.2	18.73	3.60	4.27
6	20.8	4.4	17.00	5.27	-
7	20.0	3.8	16.80	4.53	-
8	19.0	6.8	16.33	5.73	5.27[3]
9	19.4	7.4	14.67	2.00	-

With no dim. reduction, $\varepsilon = 20.2$.

Mfeat –Karhunen Loéve coefficients

h	LDA	Rotational LDA			
			Perror – iterations		
	ε	ε	I	II	III
1	52.2	56.0	53.40	30.07	29.87
2	29.8	12.8	27.53	6.07	4.33
3	17.2	2.4	14.67	1.47	0.13
4	9.6	1.0	7.27	2.8	0.47[4]
5	7.6	0.2	5.40	0	-
6	6.0	0.2	3.87	0	-
7	5.4	0.8	2.87	0	-
8	5.0	0.4	3.00	0	-
9	5.2	0.2	3.07	0	-

With no dim. reduction, $\varepsilon = 6.4$.

Mfeat – Zernike moments

h	LDA	Rotational LDA			
			Perror – iterations		
	ε	ε	I	II	III
1	60.2	50.0	59.80	0	-
2	43.6	32.0	41.27	0	-
3	39.2	28.0	36.53	0	-
4	36.6	12.0	34.33	0	-
5	21.6	14.0	24.53	0	-
6	21.0	16.0	24.20	0	-
7	20.2	22.0	22.60	0	-
8	19.8	14.0	21.40	0	-
9	19.8	14.0	21.53	0	-

With no dim. reduction, $\varepsilon = 24.6$.

TIMIT – vowels

h	LDA	Rotational LDA			
			Perror – iterations		
	ε	ε	I	II	III
1	58.7	62.5	59.66	0	-
2	28.1	33.7	29.58	0	-
3	18.4	34.6	17.79	0	-
4	16.1	18.2	15.20	0	-
5	12.6	24.1	12.78	0	-
6	12.0	18.7	11.70	0	-
7	11.3	9.3	11.06	0	-
8	11.3	14.2	11.01	0	-
9	11.2	5.7	10.99	0	-

With no dim. reduction, $\varepsilon = 24.7$.

[3] The process stops here at 4th iteration. The Perror value is 3.6 for the 4th iteration.

[4] The process stops here at 5th iteration. The Perror are 0.4 and 0.0 for iterations 4 and 5 respectively.

performance when the training data size is small. In order to answer this question, we use the Sat-Image data corpus and investigate the classification performance for different sizes of training data. The results are shown in Fig. 4, where we show the classification error as a function of the training data size (i.e., the number of vectors in the training data set). We can see in this figure that the classification performance of the rotational LDA method does not deteriorate as the training data size decreases by a factor of $\sim$50 (from 4,435 to 89 training vectors). We also observe in this figure that the rotational LDA method is resulting in better classification performance than the conventional LDA method. Thus, we can say that the Gaussian assumption made in the rotational LDA method holds for small training data sizes as well.

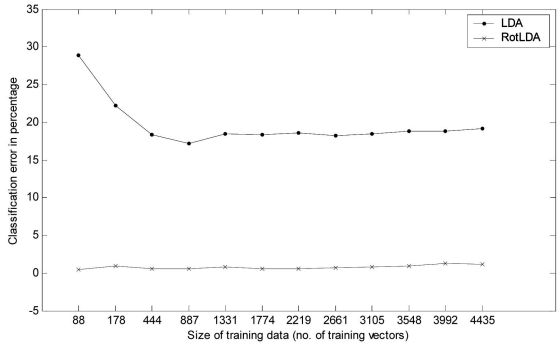


Fig. 4. Effect of training data size on the classification error of LDA and rotational LDA.

TABLE 8
Relevant Details of the Speaker Identification Experiment

Sampling frequency (Fs)	44.1 kHz	Number of average frames per utterance	35
Frame analysis window	1024 samples	Number of speakers	6
Windowing function	Hamming window	Number of repetition of the word in training phase	15
Number of filters in filterbank	32	Number of repetition of the word in testing phase	10
Shape of the filter	triangular shaped in mel domain	Number of cepstral coefficients excluding 0th coefficient	12

5 A Practical Application of Rotational LDA: Speaker Identification

In this section, we illustrate the practical application of the rotational LDA method in text-dependent speaker identification. A speaker identification system [12] uses the speech utterance of a person (who speaks a prespecified keyword or sentence) to find his/her identity from a group of prerecorded persons. The speech utterance is analyzed framewise, and an acoustic front end is used to measure a few characteristic properties (or features) from the signal of each frame. In the current state-of-the-art speaker identification systems, the mel frequency cepstral coefficients (MFCCs) are commonly used as features [12], [20]. Thus, a frame is represented by a d-dimensional feature vector with d MFCCs as its components. In the experiment reported in this section, the speaker identification system makes a decision about the speaker's identity for each frame.

In the rotational LDA method, a test vector that we want to classify is rotated around the centroid of its own class. Since the information about the class centroid is not known (as we do not know to which class the test vector belongs), we try to estimate the class centroid. The speaker identification application studied in this section allows us to get a reasonable estimate of the class centroid. Here, we assume that a test frame (that we are trying to classify) and its neighboring frames from a given utterance belong to the same class. We use the mean of the L neighboring frames as an estimate of the class centroid. Note that a similar situation occurs in face recognition applications [21], where one can find the class-centroid information by dividing a given image of a face into blocks, representing each block by its local features and using the neighboring blocks to estimate the class centroid.

In order to carry out the speaker identification experiment, we record a database from six speakers, where each speaker utters the prespecified word "money" 25 times. We manually remove the silence portions at the beginning and the end of each utterance. For each speaker, we use 15 utterances for training the speaker identification system and 10 utterances for testing. The speech signal is processed framewise, where the frames are updated every $\sim$25 ms. For each frame, 12 MFCCs are computed from the speech signal. The relevant details of the experiment are listed in Table 8.

As mentioned earlier, we make a decision about the speaker identity using each frame; i.e., we classify each frame represented by a d-dimensional MFCC feature vector (with $d = 12$) into one of the $c(= 10)$ classes. The speaker identification system is trained and tested using the same procedure as described earlier in Section 3 (see Tables 1 and 4). During the testing (classification) phase, we use L neighboring frames around the test frame ($L/2$ preceding + $L/2$ following frames) for estimating its class centroid used in Table 4. The speaker identification results from the LDA and rotational LDA methods in terms of classification error are listed in Table 9. We also provide in this table the values of Perror at different iterations for the rotational LDA method to illustrate the convergence of the method. It is evident from this table that the rotational LDA method converges in three to four iterations. Also, it provides better speaker identification performance than the LDA method.

6 Conclusions

The LDA method has been commonly used in the pattern recognition literature for dimensionality reduction. It finds a linear transformation that reduces the dimensionality of the original feature space such that the overlap between the classes is minimized for the projected vectors in the reduced dimensional space. In this paper, we introduce a rotational transform prior to the dimensionality reduction transformation step and call the resulting procedure as the rotational LDA method. The rotational transform rotates the feature vectors in the original feature space around their respective

TABLE 9
Speaker Identification Results as a Function of Reduced Dimension for LDA and Rotational LDA

Speaker Recognition						
h	LDA	Rotational LDA				
		Perror – iterations				
	ε	ε	I	II	III	IV
1	52.37	33.23	52.38	35.48	26.50	-
2	26.69	14.09	28.25	17.23	7.23	-
3	17.89	7.40	17.49	5.56	4.54	-
4	17.04	12.29	16.64	9.77	8.91	3.12
5	14.84	6.75	14.60	4.28	-	-

class centroids. We have mathematically derived a procedure that finds the rotational transform, as well as the dimensionality-reduction transform, such that the overlap between the classes is minimized in the reduced feature space. We have provided illustrative examples that demonstrate that the rotational LDA method is more effective in reducing this overlap between the classes than the LDA method. By introducing rotational transform in the original feature space, the rotational LDA method can be envisioned as a more generalized version of the LDA method.

We have conducted experiments on a number of publicly available data corpuses to evaluate the pattern classification performance of the rotational LDA method. We have reported results that demonstrate that the rotational LDA method outperforms the LDA method on all the data corpuses.

APPENDIX

SOME COMPUTATIONAL ISSUES RELATED TO THE TRAINING PHASE OF ROTATIONAL LDA

We have seen that the rotational LDA method is more effective in reducing the overlap between classes in the reduced-dimensional space than the conventional LDA method and hence results in better classification performance. However, this improvement in performance comes with some additional computational cost during the training phase of the rotational LDA method. Some of these computational issues are listed as follows:

1. The rotational LDA algorithm (Table 1) computes the basic (conventional) LDA at each stage of iteration. This means that the algorithm is computing the within-class scatter matrix ($\mathbf{S}_W$) and the between class-scatter matrix ($\mathbf{S}_b$) for each iteration.
2. It can also be observed in Table 1 that the training feature vectors are updated at each iteration.
3. The use of an expectation operator in (11) (or step 9 in Table 1) increases the computational cost.

We provide here some suggestions to address these computational issues. For the first computational issue, the procedure can be modified such that it will not compute $\mathbf{S}_W$ and $\mathbf{S}_b$ for each step of the iteration process. We can modify these matrices by investigating how $\mathbf{S}_W$ and $\mathbf{S}_b$ alter during the iteration process. It is known that $\mathbf{S}_W$ is the sum of scatter matrices $\mathbf{M}_j$ [7], i.e.

$$\mathbf{S}_W = \sum_{j=1}^{c} \mathbf{M}_j,$$

where $\mathbf{M}_j = \sum_{\mathbf{x} \in \aleph_j} (\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}_j})(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}_j})^{\mathrm{T}}$.

After rotation $\hat{\boldsymbol{\theta}}$, feature vector x will change according to (16); this would change the scatter matrix as

$$\hat{\mathbf{M}}_j = \hat{\boldsymbol{\theta}}^{\mathrm{T}} \left[\sum_{x \in \aleph_j} \left(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}_j} \right) \left(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}_j} \right)^{\mathrm{T}} \right] \hat{\boldsymbol{\theta}} = \hat{\boldsymbol{\theta}}^{\mathrm{T}} \mathbf{M}_j \hat{\boldsymbol{\theta}}.$$

Therefore, the modified within-class scatter matrix will become

$$\hat{\mathbf{S}}_W = \hat{\boldsymbol{\theta}}^{\mathrm{T}} \mathbf{S}_W \hat{\boldsymbol{\theta}}.$$

On the other hand, $\mathbf{S}_b$ depends on the class centroids and the total mean vector [7], which would not change during the rotation. Therefore, $\mathbf{S}_b$ will remain unchanged with respect to rotation. Thus, the new value of orientation $\mathbf{W}$ can be computed directly from rotation $\hat{\boldsymbol{\theta}}$ and the previous value of $\mathbf{S}_W$ using eigenvalue decomposition; i.e.

$$\mathbf{S}_W \leftarrow \hat{\boldsymbol{\theta}}^{\mathrm{T}} \mathbf{S}_W \hat{\boldsymbol{\theta}},$$
$$\mathbf{S}_b w_i = \lambda_i S_W w_i,$$

where w_i are the column vectors of $\mathbf{W}$ corresponding to λ_i. This would save processing time in computing these matrices for each of the iterations. It should be noted here that though we have used orientation $\mathbf{W}$ from the LDA technique, one could apply some other techniques instead of LDA [22] together with the rotational method for the improvement of recognition and/or classification. In that case, the optimization criteria will be different, depending upon the technique then used.

The second computational issue can be addressed by introducing some kind of weighting coefficients that would update the parameters depending upon the rotation and their previous values.

For the third computational issue, the expectation operation can be omitted by introducing an online or adaptive version of the algorithm, where parameters may be updated for every feature vector $\mathbf{x}$ instead of taking the class average of feature vectors.

REFERENCES

[1] H. Anton, *Calculus*. John Wiley & Sons, 1995.

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

[3] C.M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[4] C.L. Blake and C.J. Merz, *UCI Repository of Machine Learning Databases*, Dept. of Information and Computer Science, Univ. of California, Irvine, http://www.ics.uci.edu/~mlearn, 1998.

[5] H. Cevikalp, M. Neamtu, M. Wlkes, and A. Barkana, "Discriminative Common Vectors for Face Recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 27, no. 1, pp. 4-13, Jan. 2005.

[6] L.-F. Chen, H.-Y.M. Liao, M.-T. Ko, J.-C. Lin, and G.-J. Yu, "A New LDA-Based Face Recognition System Which Can Solve the Small Sample Size Problem," *Pattern Recognition*, vol. 33, pp. 1713-1726, 2000.

[7] R.O. Duda and P.E. Hart, *Pattern Classification and Scene Analysis*. John Wiley & Sons, 1973.

[8] J.H. Friedman, "Regularized Discriminant Analysis," *J. Am. Statistical Assoc.*, vol. 84, pp. 165-175, 1989.

[9] K. Fukunaga, *Introduction to Statistical Pattern Recognition*. Academic Press, 1990.

[10] S.G. Garofalo, L.F. Lori, F.M. William, F.G. Jonathan, P.S. David, and D.L. Nancy, "The DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus CDROM," *NIST*, 1986.

[11] Y. Guo, T. Hastie, and R. Tinshirani, "Regularized Discriminant Analysis and Its Application in Microarrays," *Biostatistics*, vol. 8, no. 1, pp. 86-100, 2007.

[12] X. Huang, A. Acero, and H.-W. Hon, *Spoken Language Processing*. Prentice Hall, 2001.

[13] A. Hyvärinen, "Fast and Robust Fixed-Point Algorithms for Independent Component Analysis," *IEEE Trans. Neural Networks*, vol. 10, no. 3, pp. 626-634, 1999.

[14] A. Hyvärinen and E. Oja, "A Fast Fixed-Point Algorithm for Independent Component Analysis," *Neural Computation*, vol. 9, no. 7, pp. 1483-1492, 1997.

[15] A.K. Jain, R.P.W. Duin, and J. Mao, "Statistical Pattern Recognition: A Review," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 22, no. 1, pp. 4-37, Jan. 2000.
[16] W.J. Krzanowski, P. Jonathan, W.V. McCarthy, and M.R. Thomas, "Discriminant Analysis with Singular Covariance Matrices: Methods and Applications to Spectroscopic Data," *Applied Statistics,* vol. 44, pp. 101-115, 1995.
[17] R. Lotlikar and R. Kothari, "Fractional-Step Dimensionality Reduction," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 22, no. 6, pp. 623-627, June 2000.
[18] J. Lu, K.N. Plataniotis, and A.N. Venetsanopoulos, "Face Recognition Using LDA-Based Algorithms," *IEEE Trans. Neural Networks,* vol. 14, no. 1, pp. 195-200, 2003.
[19] S. Raudys and R.P.W. Duin, "On Expected Classification Error of the Fisher Linear Classifier with Pseudo-Inverse Covariance Matrix," *Pattern Recognition Letters,* vol. 19, nos. 5-6, pp. 385-392, 1998.
[20] D.A. Reynolds, T.F. Quatieri, and R.B. Dunn, "Speaker Verification Using Adapted Gaussian Mixture Models," *Digital Signal Processing,* vol. 10, no. 1-3, pp. 19-41, 2000.
[21] C. Sanderson and K.K. Paliwal, "Identity Verification Using Speech and Face Information," *Digital Signal Processing,* vol. 14, no. 5, pp. 449-480, 2004.
[22] A. Sharma, K.K. Paliwal, and G.C. Onwubolu, "Class-Dependent PCA, LDA and MDC: A Combined Classifier for Pattern Classification," *Pattern Recognition,* vol. 39, no. 7, pp. 1215-1229, 2006.
[23] A. Sharma and K.K. Paliwal, "A Gradient Linear Discriminant Analysis for Small Sample Sized Problem," *Neural Processing Letters,* vol. 27, no. 1, pp. 17-24, 2008.
[24] D.L. Swets and J. Weng, "Using Discriminative Eigenfeatures for Image Retrieval," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 18, no. 8, pp. 831-836, Aug. 1996.
[25] Q. Tian, M. Barbero, Z.H. Gu, and S.H. Lee, "Image Classification by the Foley-Sammon Transform," *Optical Eng.,* vol. 25, no. 7, pp. 834-840, 1986.
[26] S. Young, G. Evermann, T. Hain, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland, *The HTK Book Version 3.2.* Cambridge Univ., 2002.
[27] 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.
[28] H. Yu and J. Yang, "A Direct LDA Algorithm for High-Dimensional Data—With Application to Face Recognition," *Pattern Recognition,* vol. 34, pp. 2067-2070, 2001.

Alok Sharma received the BTech degree from the University of the South Pacific (USP), Suva, Fiji, in 2000 and the MEng degree, with an academic excellence award, and the PhD degree in the area of pattern recognition from Griffith University, Brisbane, Australia, in 2001 and 2006, respectively. He is currently serving as an academic and the head of the Division of Electrical/Electronics, School of Engineering and Physics, USP. He is also with the Signal Processing Laboratory, Griffith University. He participated in various projects carried out in conjunction with Motorola (Sydney), Auslog Pty. Ltd. (Brisbane), CRC Micro Technology (Brisbane), and the French Embassy (Suva). He also received several grant packages. Some of them are from the French Embassy, the University Research Committee, Fiji, and the SPAS Research Committee, Fiji. His research interests include pattern recognition, computer security, and human cancer classification. He reviewed several articles from journals like *IEEE Transactions on Neural Networks*, *IEEE Transaction on Systems, Man, and Cybernetics, Part A: Systems and Humans*, *IEEE Journal on Selected Topics in Signal Processing*, *IEEE Transactions on Knowledge and Data Engineering*, *Computers & Security*, and *Pattern Recognition*. He is a member of the IEEE.

Kuldip K. Paliwal received the BS degree from Agra University, Agra, India, in 1969, the MS degree from Aligarh Muslim University, Aligarh, India, in 1971, and the PhD degree from Bombay University, Bombay, India, in 1978. He has been carrying out research in the area of speech processing since 1972. He has worked at a number of organizations including the Tata Institute of Fundamental Research, Bombay, India, the Norwegian Institute of Technology, Trondheim, Norway, the University of Keele, United Kingdom, AT&T Bell Laboratories, Murray Hill, New Jersey, AT&T Shannon Laboratories, Florham Park, New Jersey, and Advanced Telecommunication Research Laboratories, Kyoto. Since July 1993, he has been a professor in the School of Microelectronic Engineering, Griffith University, Brisbane, Australia. His current research interests include speech recognition, speech coding, speaker recognition, speech enhancement, face recognition, image coding, pattern recognition, and artificial neural networks. He has published more than 250 papers in these research areas. He has coedited two books: *Speech Coding and Synthesis* (published by Elsevier) and *Speech and Speaker Recognition: Advanced Topics* (published by Kluwer Academic Publishers). He was an associate editor of the *IEEE Transactions on Speech and Audio Processing* during the periods 1994-1997 and 2003-2004. He also served as an associate editor of the *IEEE Signal Processing Letters* from 1997 to 2000. He is currently serving the journal *Speech Communication* (published by Elsevier) as its editor in chief. He has served in the IEEE Signal Processing Society's Neural Networks Technical Committee as a founding member from 1991 to 1995 and the Speech Processing Technical Committee from 1999 to 2003. He was the general cochair of the 10th IEEE Workshop on Neural Networks for Signal Processing (NNSP '00). He received the IEEE Signal Processing Society's Best (Senior) Paper Award in 1995 for his paper on LPC quantization. He is a Fellow of the Acoustical Society of India and a member of the IEEE.