Class-Incremental Generalized Discriminant Analysis

Wenming Zheng

wenming.zheng@seu.edu.cn

Research Center for Learning Science, Southeast University, Nanjing, Jiangsu 210096, China

Generalized discriminant analysis (GDA) is the nonlinear extension of the classical linear discriminant analysis (LDA) via the kernel trick. Mathematically, GDA aims to solve a generalized eigenequation problem, which is always implemented by the use of singular value decomposition (SVD) in the previously proposed GDA algorithms. A major drawback of SVD, however, is the difficulty of designing an incremental solution for the eigenvalue problem. Moreover, there are still numerical problems of computing the eigenvalue problem of large matrices. In this article, we propose another algorithm for solving GDA as for the case of small sample size problem, which applies QR decomposition rather than SVD. A major contribution of the proposed algorithm is that it can incrementally update the discriminant vectors when new classes are inserted into the training set. The other major contribution of this article is the presentation of the modified kernel Gram-Schmidt (MKGS) orthogonalization algorithm for implementing the QR decomposition in the feature space, which is more numerically stable than the kernel Gram-Schmidt (KGS) algorithm. We conduct experiments on both simulated and real data to demonstrate the better performance of the proposed methods.

1 Introduction

Generalized discriminant analysis (GDA) was proposed by Baudat and Anour (2000) as the nonlinear extension of the classical linear discriminant analysis (LDA) (Duda & Hart, 1973) from input space to a high-dimensional feature space via the kernel trick (Vapnik, 1995; Schölkopf, Smola, & Müller, 1998). However, the GDA method often suffers from the so-called small sample size (SSS) problem (Chen, Liao, Ko, Lin, & Yu, 2000; Zheng, Zhao, & Zou, 2004a; Cevikalp & Wilkes, 2004; Cevikalp, Neamtu, Wilkes, & Barkana, 2005), where the dimensionality of the feature space nonlinearly mapped from the input space is generally much larger than the number of the training samples, such that the optimal discriminant vectors of GDA lie in the null space of the within-class scatter matrix (Zheng, Zhao, & Zou, 2004b). Mathematically, the standard approach of solving GDA is to solve the eigenvalues $\lambda$ and eigenvectors $\omega$, solutions of the following generalized...
eigenequation (Baudat & Anour, 2000; Zheng et al., 2004b): $S_B^\Phi \omega = \lambda S_W^\Phi \omega$, where $S_B^\Phi$ and $S_W^\Phi$ represent the between-class scatter matrix and the total scatter matrix in the feature space, respectively. However, as for the case of the small sample size problem, the optimal discriminant vectors of GDA can be found from the null space of $S_W^\Phi$ (Zheng et al., 2004b), where $S_W^\Phi$ denotes the within-class scatter matrix. A simple and efficient way of solving GDA in this case is to find an orthonormal basis of the subspace $S_I^\Phi(0) \cap S_W^\Phi(0)$ (Zheng, Zhao, & Zou, 2005), where $S_I^\Phi(0)$ and $S_W^\Phi(0)$ represent the null space of $S_I^\Phi$ and $S_W^\Phi$, respectively, and $S_I^\Phi(0)$ represents the complement of $S_I^\Phi(0)$.

Although it is tractable to solve GDA by utilizing the Mercer kernels, the common aspect of the previously proposed algorithms is the use of the singular value decomposition (SVD) (Baudat & Anour, 2000; Zheng et al., 2005; Yang, Frangi, Jin, & Yang, 2004; Liu, Wang, Li, & Tan, 2004). A major common drawback of these algorithms is the difficulty of designing an incremental solution for the eigenvalue problem. The other major drawback of these SVD-based algorithms is the numerical instability problem. This is because the eigenvalues determined by the eigenvalue decomposition approach may be very close to each other, which will result in instability of the eigenvector according to the perturbation theory (Fukunaga, 1990). Recently, Xiong, Ye, Li, Cherkassky, and Janardan (2005) proposed a kernel discriminant analysis algorithm via QR decomposition (KDA/QR) to reduce the computational complexity of kernel discriminant analysis (KDA). However, similar to the kernel direct discriminant analysis (KDDA) approach (Lu, Plataniotis, & Venetsanopoulos, 2003), this method finds the discriminant vectors of KDA by limiting attention to the range space of $S_B^\Phi$, which may not obtain the optimal discriminant vectors in terms of the Fisher discriminant criterion—in particular, the case of the small sample size problem (Zheng et al., 2005). The other drawback of KDA/QR is that it is only a batch method, which requires that all the training data be available before computing the discriminant vectors (Ye et al., 2004). Thus, it is still time-consuming to update the discriminant vectors when new data items are inserted into the training set.

In this article, we propose a computationally efficient and numerically stable algorithm for GDA as for the case of the small sample size problem. The proposed method can directly solve the optimal discriminant vectors of GDA by applying only QR decomposition. More important, the proposed method introduces an incremental technique to update the discriminant vectors when new data items are inserted into the training set, which is very desirable for designing a dynamic recognition system. Moreover, this article also proposes a modified kernel Gram-Schmidt (MKGS) orthogonalization algorithm for implementing the QR decomposition in the feature space, which is much more numerically stable in contrast to the kernel Gram-Schmidt (KGS) orthogonalization algorithm proposed by Wolf and Shashua (2003).
In the next section, we review the KGS algorithm and then propose the MKGS algorithm in section 3. In section 4, we propose the batch GDA algorithm and the class-incremental GDA algorithm, respectively, using the MKGS algorithm. In section 5, we present the feature extraction method for classification based on the proposed GDA algorithm. Section 6 is devoted to the experiments on both simulated and real data. The conclusion is given in the last section.

2 KGS Algorithm

Let $A$ be a matrix with $k$ columns $\alpha_1, \ldots, \alpha_k$, where $\alpha_i \in \mathbb{R}^n$ ($i = 1, \ldots, k$). Let $\Phi(\cdot)$ be a mapping that maps the elements of $\mathbb{R}^n$ into a high-dimensional Hilbert space $F$, that is, $\Phi : \mathbb{R}^n \to F$. Let

$$A^\Phi = [\Phi(\alpha_1), \ldots, \Phi(\alpha_k)].$$

Suppose that $\beta_1, \ldots, \beta_k$ are the equivalent orthonormal vectors corresponding to the columns of $A^\Phi$. Then $\beta_i$ ($i = 1, \ldots, k$) can be computed by using the following classical Gram-Schmidt (CGS) orthonormal procedure (Björck, 1994):

1. $\beta_1 = \Phi(\alpha_1)$.
2. Repeat for $j = 2, \ldots, k$.
   $$\beta_j = \Phi(\alpha_k) - \sum_{i=1}^{j-1} \frac{\beta_i^T \Phi(\alpha_j)}{\beta_i^T \beta_i} \beta_i.$$
3. Repeat for $j = 1, \ldots, k$,
   $$\beta_j = \beta_j / \|\beta_j\|,$$
   where $\| \cdot \|$ stands for the Euclidean distance norm.

However, directly computing the orthonormal vectors $\beta_i$ ($i = 1, \ldots, k$) is an intractable task because the mapping function $\Phi$ is hard to explicitly evaluate. Wolf and Shashua (2003) proposed an indirect approach to implement the above orthonormal procedure via the kernel trick (hereafter the KGS algorithm). More specifically, assume that $k(x, y)$ is the reproducing kernel defined on the feature space $F$ such that

$$k(x, y) = \langle \Phi(x), \Phi(y) \rangle = (\Phi(x))^T \Phi(y),$$

where $\langle \Phi(x), \Phi(y) \rangle$ stands for the inner product of $\Phi(x)$ and $\Phi(y)$. Then according to Wolf and Shashua (2003), the KGS algorithm can be summarized as follows:
KGS algorithm (Wolf & Shashua, 2003). Let $A^\Phi$ be a matrix with columns $\Phi(\alpha_1), \ldots, \Phi(\alpha_k)$, where $\Phi(\alpha_1), \ldots, \Phi(\alpha_k)$ are $k$ linearly independent vectors. Then the corresponding orthonormal vectors of the columns of $A^\Phi$ can be obtained using the following steps, where $s_j$ and $t_j (j = 1, \ldots, k)$ are $k$-dimensional vectors, $D$ is a $k \times k$ diagonal matrix, and $e_j = (0, \ldots, 1, \ldots, 0)^T$ is a $k$-dimensional vector where the $j$th item is 1.

1. Let $s_1 = t_1 = e_1$, $D_{11} = k(\alpha_1, \alpha_1)$, where $e_1 = (1, 0, \ldots, 0)^T$;
2. Repeat for $j = 2, \ldots, k$
   a. Compute $s_j = \left(\frac{t_1 k(\alpha_1, \alpha_j)}{D_{11}}, \ldots, \frac{\sum_{q=1}^{j-1} t_q k(\alpha_q, \alpha_j)}{D_{(j-1)(j-1)}}, 1, 0, \ldots, 0\right)^T$;
   b. Compute $t_j = (-t_1, \ldots, -t_{j-1}, e_j, 0, \ldots, 0)s_j$;
   c. Compute $D_{jj} = \sum_{p,q=1}^j t_p t_q k(\alpha_p, \alpha_q)$;
3. $R = D^{1/2}[s_1, \ldots, s_k]$;
4. $R^{-1} = [t_1, \ldots, t_k] D^{-1/2}$;

The columns of the matrix $[\beta_1, \ldots, \beta_k] = [\Phi(\alpha_1), \ldots, \Phi(\alpha_k)][t_1, \ldots, t_k] D^{-1/2}$ are the corresponding orthonormal vectors of the columns of $A^\Phi$.

3 MKGS Algorithm

The KGS algorithm proposed by Wolf and Shashua (2003) is essentially the kernelized version of the CGS algorithm. Thus, many numerical properties of CGS will be delivered to KGS. However, the experimental results by Rice (1966) and the theoretical analysis by Björck (1967) indicated that the CGS procedure is very sensitive to round-off errors. In other words, if the matrix $A^\Phi$ is ill conditioned, the computed vectors $\beta_1, \ldots, \beta_k$ will soon lose their orthogonality, and reorthogonalization will be needed. Thus, it is very desirable to modify the KGS algorithm to obtain a numerically superior algorithm for orthogonalizing the columns of $A^\Phi$.

It is notable that the modified Gram-Schmidt (MGS) orthogonalization procedure is numerically superior to CGS. More details can be found in Rice (1966) and Björck (1967). Thus, we will adopt the MGS procedure to modify the KGS algorithm. In general, the MGS procedure can be divided into two versions: the row-oriented procedure and the column-oriented procedure (Björck, 1994). According to Björck (1994), the two procedures are numerically equivalent, the operations and rounding errors are the same, and both produce the same numerical results. The main difference is that the column-oriented procedure is more appropriate to use when the orthogonalized vectors are sequentially obtained. Based on the column-oriented procedure, the orthonormal vectors $\beta_1, \ldots, \beta_k$ can be computed as follows (for simplicity, we use the notations: $\Phi(\alpha_i)^{(0)} = \Phi(\alpha_i)$, $i = 1, \ldots, k$):
1. \( \beta_1 = \Phi(\alpha_1)^{(0)}; \)
2. Repeat for \( j = 2, \ldots, k \)
a. Repeat for \( m = 1, \ldots, j - 1 \)
   \( \Phi(\alpha_j)^{(m)} = \Phi(\alpha_j)^{(m-1)} - \frac{\beta_m^T \Phi(\alpha_j)^{(m-1)} \beta_m}{\beta_m^T \beta_m}; \)
b. \( \beta_j = \Phi(\alpha_j)^{(j-1)}; \)
3. Repeat for \( j = 1, \ldots, k \)
   \( \beta_j = \beta_j / \| \beta_j \|; \)

Similar to the KGS algorithm, we implement the above procedure via the kernel trick, and hereafter the MKGS algorithm.

**MKGS Algorithm**

Let \( A^\Phi \) be a matrix with columns \( \Phi(\alpha_1), \ldots, \Phi(\alpha_k) \), where \( \Phi(\alpha_1), \ldots, \Phi(\alpha_k) \) are \( k \) linearly independent vectors. Then the corresponding orthonormal vectors of the columns of \( A^\Phi \) can be obtained using the following steps, where \( s_j \) and \( t_j \) \( (j = 1, \ldots, k) \) are \( k \)-dimensional vectors, \( D \) is a \( k \times k \) diagonal matrix, \( \Delta \) is a \( k \times k \) matrix, and \( e_j = (0, \ldots, 1, \ldots, 0)^T \) is a \( k \)-dimensional vector where the \( j \)th item is 1.

1. Let \( s_1 = t_1 = e_1, D_{11} = k(\alpha_1, \alpha_1), \Delta_{11} = k(\alpha_i, \alpha_1)(i = 1, \ldots, k); \)
2. Repeat for \( j = 2, \ldots, k \)
a. \( t_j^{(1)} = e_j; \)
b. Repeat for \( i = 1, \ldots, j - 1 \)
   \( s_{ji} = \sum_{p=1}^j \Delta_{pi} t_p^{(i)} / D_{ii}; \)
   \( t_j^{(i+1)} = t_j^{(i)} - s_{ji} t_i; \)
c. \( t_j = t_j^{(j)}; \)
d. Repeat for \( p = 1, \ldots, k \)
   \( \Delta_{pj} = \sum_{q=1}^j t_q k(\alpha_q, \alpha_p); \)
e. Compute \( D_{jj} = \sum_{p=1}^j \Delta_{pj} t_p; \)
3. \( R = D^{1/2} [s_1, \ldots, s_k] \), where \( s_i = [s_{i1}, s_{i2}, \ldots, s_{i(j-1)}, 1, 0, \ldots, 0]^T; \)
4. \( R^{-1} = [t_1, \ldots, t_k] D^{-1/2}; \)

The columns of the matrix \([\beta_1, \ldots, \beta_k] = [\Phi(\alpha_1), \ldots, \Phi(\alpha_k)] [t_1, \ldots, t_k] D^{-1/2} \) are the corresponding orthonormal vectors of the columns of \( A^\Phi \). By calculating the computational cost of each line of the above algorithm, we can easily obtain that the complexity of the MKGS algorithm is \( O(k^3) \).
Let $X = \{x^j_i\}_{i=1,\ldots,c; j=1,\ldots,N_i}$ be an $n$-dimensional training sample set with $N$ elements, where $c$ is the number of the classes and $N_i$ is the number of the samples in $i$th class. The between-class scatter matrix $S_B^\Phi$, the within-class scatter matrix $S_W^\Phi$, and the total scatter matrix $S_T^\Phi$ are respectively defined as:

$$S_B^\Phi = \sum_{i=1}^c N_i (u_i^\Phi - u^\Phi)(u_i^\Phi - u^\Phi)^T$$ \hspace{1cm} (4.1)$$

$$S_W^\Phi = \sum_{i=1}^c \sum_{j=1}^{N_i} (\Phi(x^j_i) - u_i^\Phi)(\Phi(x^j_i) - u_i^\Phi)^T$$ \hspace{1cm} (4.2)$$

$$S_T^\Phi = \sum_{i=1}^c \sum_{j=1}^{N_i} (\Phi(x^j_i) - u^\Phi)(\Phi(x^j_i) - u^\Phi)^T,$$ \hspace{1cm} (4.3)$$

where $x^T$ denotes the transpose of $x$, $\Phi(x^j_i)$ is the $j$th sample in the $i$th class, $u_i^\Phi$ is the mean of $i$th class samples, and $u^\Phi$ is the mean of all samples in $F$:

$$u_i^\Phi = \frac{1}{N_i} \sum_{j=1}^{N_i} \Phi(x^j_i), \quad u^\Phi = \frac{1}{N} \sum_{i=1}^c \sum_{j=1}^{N_i} \Phi(x^j_i).$$ \hspace{1cm} (4.4)$$

### 4.1 Batch GDA/MKGS Algorithm.

Let $S_B^\Phi(0)$ denote the null space of $S_B^\Phi$, $S_B^\Phi(0)$ and $S_W^\Phi(0)$ denote the complement of $S_B^\Phi(0)$ and $S_W^\Phi(0)$, respectively. Then, from the expressions of $S_B^\Phi$, $S_W^\Phi$, and $S_T^\Phi$ in equations 4.1, 4.2, and 4.3, we obtain

$$\overline{S_B^\Phi(0)} = \text{span}\{u_i^\Phi - u^\Phi|i = 1, \ldots, c\}$$ \hspace{1cm} (4.5)$$

$$\overline{S_W^\Phi(0)} = \text{span}\{\Phi(x^j_i) - u_i^\Phi|i = 1, \ldots, c; j = 1, \ldots, N_i\}$$ \hspace{1cm} (4.6)$$

$$\overline{S_T^\Phi(0)} = \text{span}\{\Phi(x^j_i) - u^\Phi|i = 1, \ldots, c; j = 1, \ldots, N_i\}. \hspace{1cm} (4.7)$$

Note that $\Phi(x^j_i) - u^\Phi = (\Phi(x^j_i) - u_i^\Phi) + (u_i^\Phi - u^\Phi)$. Thus, from equations 4.5, 4.6, and 4.7, we have

$$\overline{S_T^\Phi(0)} \subseteq \text{span}\{\Phi(x^j_i) - u_i^\Phi, u_i^\Phi - u^\Phi|i = 1, \ldots, c; j = 1, \ldots, N_i\}. \hspace{1cm} (4.8)$$

Moreover, we have the following two important theorems about $\overline{S_B^\Phi(0)}$ and $\overline{S_W^\Phi(0)}$:
Theorem 1. \( S_B^\Phi(0) \) can be spanned by \( u_i^\Phi - u_1^\Phi \) \((i = 2, \ldots, c)\), that is, \( S_B^\Phi(0) = \text{span}\{u_i^\Phi - u_1^\Phi\} | i = 2, \ldots, c\).

Theorem 2. \( S_W^\Phi(0) \) can be spanned by \( \Phi(x_i^j) - \Phi(x_1^j) \) \((i = 1, \ldots, c; j = 2, \ldots, N)\), that is, \( S_W^\Phi(0) = \text{span}\{\Phi(x_i^j) - \Phi(x_1^j)\} | i = 1, \ldots, c; j = 2, \ldots, N\}.

The proofs of theorems 1 and 2 are given in appendixes A and B, respectively.

Theorem 1 will be useful to design an incremental algorithm for updating the basis of \( S_B^\Phi(0) \) when new classes are inserted into the training set since it can be expressed as the span of the vectors \( u_i^\Phi - u_1^\Phi \) \((i = 2, \ldots, c)\), which does not depend on the ensemble mean \( u^\Phi \) of the training set. Similarly, theorem 2 will be useful to design an incremental algorithm for updating the basis of \( S_W^\Phi(0) \) when new instances are inserted into the existing classes of the training set since it can be expressed as the span of the vectors \( \Phi(x_i^j) - \Phi(x_1^j) \) \((i = 1, \ldots, c; j = 2, \ldots, N)\), which is not dependent on the class mean \( u_i^\Phi \) \((i = 1, \ldots, c)\). In the next section, we will show that theorems 1 and 2 are crucial for designing the class-incremental GDA/MKGS algorithm. Now let

\[
A^\Phi = [A_1^\Phi, \ldots, A_c^\Phi, A_{c+1}^\Phi] \tag{4.9}
\]

where the matrices \( A_i^\Phi \) are, respectively, defined by

\[
A_i^\Phi = [\Phi(x_1^3) - \Phi(x_i^3), \Phi(x_1^4) - \Phi(x_i^4), \ldots, \Phi(x_1^{N}) - \Phi(x_i^{N})] \tag{4.10}
\]

and

\[
A_{c+1}^\Phi = [u_2^\Phi - u_1^\Phi, u_3^\Phi - u_1^\Phi, \ldots, u_c^\Phi - u_1^\Phi]. \tag{4.11}
\]

From theorems 1 and 2 and equations 4.9, 4.10, and 4.11, we obtain that \( S_B^\Phi(0) \) and \( S_W^\Phi(0) \) can be respectively spanned by the first \( N - c \) columns and the last \( c - 1 \) columns of the matrix \( A^\Phi \). Moreover, from equation 4.8, we obtain that \( S_T^\Phi(0) \) lies in the span of the \( N - 1 \) columns of matrix \( A^\Phi \). Without loss of generality, we assume that \( \Phi(x_i^j) \) \((i = 1, \ldots, c; j = 1, \ldots, N)\) are linearly independent. Then we have the following theorem regarding the rank of \( S_T^\Phi(0) \):

Theorem 3. Suppose that \( \Phi(x_i^j) \) \((i = 1, \ldots, c; j = 1, \ldots, N)\) are linearly independent. Then the rank of \( S_T^\Phi(0) \) is \( N - 1 \), that is, \( \text{rank}(S_T^\Phi(0)) = N - 1 \).

The proof of theorem 3 is given in appendix C. Consider that \( S_T^\Phi(0) \) lies in the span of the \( N - 1 \) columns of matrix \( A^\Phi \). Thus, theorem 3 indicates that the \( N - 1 \) columns of the matrix \( A^\Phi \) form a basis of \( S_T^\Phi(0) \), where the first
\( N - c \) columns form a basis of \( \overline{S^\Phi_W}(0) \). Note that the optimal discriminant vectors of GDA lie in the subspace \( \overline{S^\Phi_T}(0) \cap \overline{S^\Phi_W}(0) \) as for the case of the small sample size problem (Zheng et al., 2004b). Thus, our goal is to get the basis of \( \overline{S^\Phi_T}(0) \cap \overline{S^\Phi_W}(0) \). This can be implemented by using the MKGS orthogonalization procedure.

Let \( A^\Phi_{ij} \) denote the \( p \)th column of matrix \( A^\Phi_i \). Then for \( i, j, m, n = 1, \ldots, c \) and \( p = 1, \ldots, N_i - 1 \) and \( q = 1, \ldots, N_j - 1 \), we have

\[
(A^\Phi_{ip})^T A^\Phi_{jq} = [\Phi(x_i^{p+1}) - \Phi(x_i^1)]^T [\Phi(x_j^{q+1}) - \Phi(x_j^1)]
\]

\[
= (\Phi(x_i^{p+1}))^T \Phi(x_j^{q+1}) - (\Phi(x_i^{p+1}))^T \Phi(x_j^1) - (\Phi(x_i^1))^T \Phi(x_j^{q+1}) + (\Phi(x_i^1))^T \Phi(x_j^1)
\]

\[
= k(x_i^{p+1}, x_j^{q+1}) - k(x_i^{p+1}, x_j^1) - k(x_i^1, x_j^{q+1}) + k(x_i^1, x_j^1)
\]

(4.12)

\[
(A^\Phi_{ip})^T u^\Phi_m = \frac{1}{N_m} \sum_{t=1}^{N_m} [\Phi(x_i^{p+1}) - \Phi(x_i^1)]^T \Phi(x_m^t)
\]

\[
= \frac{1}{N_m} \sum_{t=1}^{N_m} [k(x_i^{p+1}, x_m^t) - k(x_i^1, x_m^t)]
\]

(4.13)

\[
(u^\Phi_m)^T u^\Phi_n = \frac{1}{N_m} \left( \sum_{p=1}^{N_m} \Phi(x_m^p) \right)^T \frac{1}{N_n} \left( \sum_{q=1}^{N_n} \Phi(x_n^q) \right) = \frac{1}{N_m N_n} \sum_{p=1}^{N_m} \sum_{q=1}^{N_n} k(x_m^p, x_n^q).
\]

(4.14)

Let \( K \) be an \( N - 1 \) by \( N - 1 \) matrix defined by

\[
K = (K_{ij})_{i=1,\ldots,c+1; j=1,\ldots,c+1},
\]

(4.15)

where

\[
K_{ij} = (A^\Phi_i)^T A^\Phi_j.
\]

(4.16)

Let \( (K_{ij})_{pq} \) denote the element in the \( p \)th row and \( q \)th column of the matrix \( K_{ij} \). Then for \( i, j = 1, \ldots, c, m, n = 1, \ldots, c - 1, \) and \( p = 1, \ldots, N_j - 1 \) and \( q = 1, \ldots, N_j - 1 \), we have

\[
(K_{ij})_{pq} = (A^\Phi_p)^T A^\Phi_{jq}
\]

(4.17)

\[
(K_{(c+1)j})_{mq} = (K_{j(c+1)})_{qm} = (A^\Phi_{jq})^T (u^\Phi_{m+1} - u^\Phi_1) = (A^\Phi_{jq})^T u^\Phi_{m+1} - (A^\Phi_{jq})^T u^\Phi_1
\]

(4.18)
According to equations 4.12 to 4.19, we can easily calculate the matrix $K$. Let $K(i, j)$ denote the element in the $i$th row and $j$th column of $K$. According to the MKGS algorithm, we obtain the following batch GDA/MKGS algorithm, where $s_j$ and $t_j$ ($j = 1, \ldots, N - 1$) are $N - 1$ dimensional vectors, $D$ is an $N - 1$ by $N - 1$ diagonal matrix, and $\Delta$ is an $N - 1$ by $N - 1$ matrix:

**Batch GDA/MKGS Algorithm**

1. Let $s_1 = t_1 = e_1$, $D_{11} = K(1, 1)$, $\Delta_{i1} = K(i, 1)(i = 1, \ldots, N - 1)$;
2. Repeat for $j = 2, \ldots, N - 1$
   a. $t_j^{(1)} = e_j$;
   b. Repeat for $i = 1, \ldots, j - 1$
      $s = \sum_{p=1}^{j-1} \Delta_{pi} t_p^{(i)}$;
      $t_j^{(i+1)} = t_j^{(i)} - s t_i$;
   c. $t_j = t_j^{(j)}$
   d. Repeat for $p = 1, \ldots, N - 1$
      $\Delta_{pj} = \sum_{q=1}^{j} t_q^{(j)} K(q, p)$;
   e. Compute $D_{jj} = \sum_{p=1}^{j} \Delta_{pj} t_p^{(j)}$;
3. $[\beta_{N-c+1}, \ldots, \beta_{N-1}] = A^\Phi [t_{N-c+1}, \ldots, t_{N-1}] (D_{ij})_{i,j=N-c+1,\ldots,N-1}^{1/2}$

The $c - 1$ vectors $\beta_{N-c+1}, \ldots, \beta_{N-1}$ form an orthonormal basis of $S^\Phi_y(0) \cap S^\Phi_w(0)$, which are referred to as the discriminant vectors of GDA as for the case of the small sample size problem. By calculating the computational cost of the above algorithm, we obtain that the complexity of the batch GDA/MKGS algorithm is $O(N^3)$.

### 4.2 Class-Incremental GDA/MKGS Algorithm

This section aims to design an incremental algorithm for updating the discriminant vectors of GDA/MKGS when new data items are inserted into the training set. We consider two distinct cases of the inserted instances: (1) the instances belong to a new class, and (2) the instances belong to an existing class.

#### 4.2.1 Insertion of a New Class

Recalling that we have $c$ classes, let ${x_{c+1,j}^j | j = 1, \ldots, N_{c+1}}$ be the $(c + 1)$th class being inserted, where $N_{c+1}$ is
the number of the new training samples. In this case, the expression in
 equation 4.9 can be rewritten as

$$\lambda^\phi = [A^\phi_1, A^\phi_c, \tilde{A}^\phi_{c+1}, \tilde{A}^\phi_{c+2}], \quad (4.20)$$

where $A^\phi_i$ ($i = 1, \ldots, c$) are defined in equation 4.10, $\tilde{A}^\phi_{c+1}$ and $\tilde{A}^\phi_{c+2}$ are
respectively defined as follows:

$$\tilde{A}^\phi_{c+1} = [\Phi(x^2_{c+1}) - \Phi(x^1_{c+1}), \Phi(x^3_{c+1}) - \Phi(x^1_{c+1}), \ldots, \Phi(x^N_{c+1}) - \Phi(x^1_{c+1})] \quad (4.21)$$

$$\tilde{A}^\phi_{c+2} = [u_2^\phi - u_1^\phi, u_3^\phi - u_1^\phi, \ldots, u_{c+1}^\phi - u_1^\phi]. \quad (4.22)$$

The kernel matrix $K$ in equation 4.15 is replaced by

$$\tilde{K} = (\tilde{A}^\phi)^T \tilde{A}^\phi. \quad (4.23)$$

The elements of $\tilde{K}$ can be calculated by utilizing the kernel function.

According to the batch GDA/MKGS algorithm, we have the following
algorithm of updating the new discriminant vectors when the $(c+1)$th
class is inserted into the training set, where $\tilde{K}(i, j)$ denotes the element in
the $i$th row and $j$th column of $\tilde{K}$, $s_j$ and $\tilde{i}_j$ ($j = 1, \ldots, N + N_{c+1} - 1$) are
$N + N_{c+1} - 1$-dimensional vectors, $\tilde{D}$ is an $N + N_{c+1} - 1$ by $N + N_{c+1} - 1$
diagonal matrix, and $\tilde{A}$ is an $N + N_{c+1} - 1$ by $N + N_{c+1} - 1$ matrix.

**Class-Incremental GDA/MKGS Algorithm 1: Updating Discriminant
Vectors with the Insertion of the $(c+1)$th Class**

1. Repeat for $j = 1, \ldots, N - c$
   a. Compute $\tilde{i}_j = (t_j^T, 0, \ldots, 0)^T$;
   b. Compute $\tilde{D}_{jj} = D_{jj}$;
   c. Repeat for $i = 1, \ldots, N - c$
      $\tilde{\Delta}_{ij} = \Delta_{ij}$;
   d. Repeat for $i = N - c + 1, \ldots, N + N_{c+1} - 1$
      $\tilde{A}_{ij} = \sum_{q=1}^{N - c} \tilde{t}_{qj} \tilde{K}(q, i)$;
2. Repeat for $j = N - c + 1, \ldots, N + N_{c+1} - 1$
   a. $t_j^{(1)} = e_j$;
   b. Repeat for $i = 1, \ldots, j - 1$
      $s = \sum_{q=1}^{i=1} \Delta_{pq} t_j^{(q)}$;
      $t_j^{(i+1)} = t_j^{(i)} - s \tilde{i}_j$;
   c. $\tilde{i}_j = t_j^{(i)}$. 

d. Repeat for $p = 1, \ldots, N + N_{c+1} - 1$
   \[ \tilde{\Delta}_{pj} = \sum_{q=1}^{j} \tilde{t}_{qj} \tilde{K}(q, p); \]

3. \[
[\beta_{N+N_{c+1}-c}, \ldots, \beta_{N+N_{c+1}-1}] = \tilde{A}^{\Phi}[\tilde{t}_{N+N_{c+1}-c}, \ldots, \tilde{t}_{N+N_{c+1}-1}] (\tilde{D}_{ij})^{1/2};
\]

The $c$ vectors $\beta_{N+N_{c+1}-c}, \ldots, \beta_{N+N_{c+1}-1}$ are the new discriminant vectors of GDA after the $(c + 1)$th class is inserted into the training set. According to calculating the computational cost of the above algorithm, we obtain that the complexity of the class-incremental GDA/MKGS algorithm for updating discriminant vectors with the insertion of the $(c + 1)$th class is $O((N_{c+1} + c)(N + N_{c+1})^2)$.

4.2.2 Insertion of a New Instance from an Existing Class. Suppose that $x$ is an instance being inserted into the $i$th $(1 \leq i \leq c)$ class. For simplicity of notation, we denote $x$ by $x_{N_{i} + 1}$ since there are $N_{i}$ samples in the $i$th class. Then the mean of the $i$th class, denoted by $\tilde{u}_{i}^{\Phi}$, is expressed as

\[
\tilde{u}_{i}^{\Phi} = \frac{1}{N_{i} + 1} \sum_{j=1}^{N_{i}+1} \Phi(x_{i}^{j}).
\]  

(4.24)

Without loss of the generality, we assume that $i > 1$. Then the expression in equation 4.9 can be rewritten as

\[
\tilde{A}^{\Phi} = [A_{1}^{\Phi}, \ldots, A_{c}^{\Phi}, \Phi(x_{i}^{N_{i}+1}) - \Phi(x_{i}^{1}), \tilde{A}_{c+1}^{\Phi}],
\]

(4.25)

where $A_{i}^{\Phi}$ $(i = 1, \ldots, c)$ are defined in equation 4.10, and $\tilde{A}_{c+1}^{\Phi}$ is defined as

\[
\tilde{A}_{c+1} = [u_{2}^{\Phi} - u_{1}^{\Phi}, \ldots, u_{i-1}^{\Phi} - u_{1}^{\Phi}, \tilde{u}_{i}^{\Phi} - u_{1}^{\Phi}, u_{i+1}^{\Phi} - u_{1}^{\Phi}, \ldots, u_{c}^{\Phi} - u_{1}^{\Phi}].
\]

(4.26)

The new kernel matrix, denoted by $\tilde{K}$, is expressed as

\[
\tilde{K} = (\tilde{A}^{\Phi})^{T} \tilde{A}^{\Phi}.
\]

(4.27)

According to the batch GDA/MKGS algorithm, we have the following incremental algorithm of updating the new discriminant vectors with the insertion of an instance in $i$th class, where $\tilde{K}(i, j)$ denotes the element in the $i$th row and $j$th column of $\tilde{K}$, $\tilde{s}_{j}$ and $\tilde{t}_{j}$ $(j = 1, \ldots, N)$ are $N$-dimensional vectors, $\tilde{D}$ is an $N$ by $N$ diagonal matrix, and $\tilde{\Delta}$ is an $N$ by $N$ matrix.
Class-Incremental GDA/MKGS Algorithm 2: Updating Discriminant Vectors with the Insertion of a New Instance in \(i\)th (\(i \leq c\)) Class

1. Repeat for \(j = 1, \ldots, N - c\)
   a. Compute \(\tilde{t}_j = (t_j^T, 0, \ldots, 0)^T\);
   b. Compute \(\tilde{D}_{jj} = D_{jj}\);
   c. Repeat for \(i = 1, \ldots, N - c\)
      \(\Delta_{ij} = \Delta_{ij};\)
   d. Repeat for \(i = N - c + 1, \ldots, N\)
      \(\tilde{\Delta}_{ij} = \sum_{q=1}^{N-c} \tilde{t}_{iq} K(q, i)\)

2. Repeat for \(j = N - c + 1, \ldots, N\)
   a. \(t_j^{(1)} = e_j;\)
   b. Repeat for \(i = 1, \ldots, j - 1\)
      \(s = \sum_{p=1}^{i} \tilde{\Delta}_{pi};\)
      \(t_j^{(i+1)} = t_j^{(i)} - s \tilde{t};\)
   c. \(\tilde{t}_j = t_j^{(j)};\)
   d. Repeat for \(p = 1, \ldots, N\)
      \(\tilde{\Delta}_{pj} = \sum_{q=1}^{j} \tilde{t}_{pq} K(q, p);\)
   e. Compute \(\tilde{D}_{jj} = \sum_{p=1}^{j} \tilde{\Delta}_{pj} \tilde{t}_{pj};\)

3. \([\beta_{N+2-c}, \ldots, \beta_{N}] = \tilde{A}^{1/2}(\tilde{\beta}_{N+2-c}, \ldots, \tilde{\beta}_{N}(\tilde{D}_{ij})_{i,j=N+2-c,N};\)

The \(c - 1\) vectors \(\beta_{N+2-c}, \ldots, \beta_{N}\) are the new discriminant vectors of GDA after the new instance of \(i\)th class is inserted into the training set. By calculating the computational cost of the above algorithm, we obtain that the complexity of the class-incremental GDA/MKGS algorithm for updating discriminant vectors with the insertion of a new instance is \(O(cN^2)\).

5 Feature Extraction for Classification

Let \(\Phi(X) = [\Phi(x_1^1) \cdots \Phi(x_1^{N_1}) \cdots \Phi(x_c^1) \cdots \Phi(x_c^{N_c})]\). Then we have

\[
\Phi(x_i^j) = \Phi(X) e_{k+j}
\] (5.1)

where \(k = \sum_{l=1}^{i-1} N_l\). From equations 4.4, 4.10, and 5.1, we have

\[
u_i^\Phi = \frac{1}{N_i} \sum_{t=1}^{N_i} \Phi(X) e_{k+t} = \frac{\Phi(X)}{N_i} \sum_{t=1}^{N_i} e_{k+t} = \Phi(X) \tilde{L}_i
\] (5.2)

\[
A_i^\Phi = \Phi(X)[e_{k+2} - e_{k+1}, e_{k+3} - e_{k+1}, \ldots, e_{k+N_i} - e_{k+1}] (i = 1, \ldots, c).
\] (5.3)
where \( L_i = \frac{1}{N_i} \sum_{t=1}^{N} e_{k+t} \). From equations 4.11 and 5.2, we have

\[
A_\Phi^{c+1} = \Phi(X)[L_2 - L_1, L_3 - L_1, \ldots, L_c - L_1].
\]  

(5.4)

Combining equations 4.9, 5.3, and 5.4, we obtain

\[
A_\Phi = [A_\Phi^1, \ldots, A_\Phi^c, A_\Phi^{c+1}] = \Phi(X)P.
\]

(5.5)

where

\[
P = [e_2 - e_1, \ldots, e_{N_i} - e_1, e_{N_i+2} - e_{N_i+1} \ldots, e_N - e_{N-N_i+1}, L_2 - L_1, \ldots, L_c - L_1].
\]

(5.6)

Thus, according to the batch GDA/MKGS algorithm, the projection matrix of GDA can be expressed as

\[
W_{GDA/MKGS} = [\beta_{N-c+1}, \ldots, \beta_{N-1}] = \Phi(X)P_T,
\]

(5.7)

where

\[
T = \begin{bmatrix}
\frac{1}{D(N-c+1)(N-c+1)} & \cdots & \frac{1}{D(N-1)(N-1)}
\end{bmatrix}.
\]

(5.8)

The projection of a test point \( \Phi(x_{test}) \) onto \( W_{GDA/KMGS} \) can be calculated by

\[
y_{test} = W_{GDA/MKGS}^T \Phi(x_{test}) = T^T P^T K_{test},
\]

(5.9)

where

\[
K_{test} = [k(x_1^1, x_{test}), k(x_1^2, x_{test}), \ldots, k(x_c^N, x_{test})]^T.
\]

(5.10)

Note that the discriminant vectors of GDA/MKGS lie in the subspace \( S_W^\Phi(0) \). Thus, we have

\[
W_{GDA/MKGS}^T S_W^\Phi = 0.
\]

(5.11)

From equations 4.2 and 5.11, we have

\[
W_{GDA/MKGS}^T [\Phi(x_1^j) - u_1^\Phi, \ldots, \Phi(x_{N_i}^j) - u_1^\Phi, \ldots, \Phi(x_c^N) - u_c^\Phi] = 0.
\]

(5.12)

From equation 5.12, we obtain that

\[
W_{GDA/MKGS}^T \Phi(x_i^j) = W_{GDA/MKGS}^T u_i^\Phi (j = 1, \ldots, N_i; i = 1, \ldots, c).
\]

(5.13)
Equation 5.13 means that the training data of each class are projected onto the same point in the projection space. Now let $X^\phi_i$ denote the $i$th class data set in the feature space, and let $y^i_1$ and $\bar{y}^i_1$ represent the respective projection of $\Phi(x^i_j)$ and $u^\phi_i$ onto the projection matrix $W_{GDA/MKGS}$, that is,

$$y^i_1 = W^T_{GDA/MKGS} \Phi(x^i_j), \quad \bar{y}^i_1 = W^T_{GDA/MKGS} u^\phi_i. \quad (5.14)$$

From equations 5.13 and 5.14, we have

$$y^1_1 = y^2_1 = \cdots = y^i_1 = \cdots = y^{N_i}_1 = \bar{y}^i_1. \quad (5.15)$$

Based on the nearest-neighbor rule, we define the distance between the projections of the test point $\Phi(x_{test})$ and the data set $X^\phi_i$ as follows:

$$d_p(\Phi(x_{test}), X^\phi_i) = \min\{\|y_{test} - y^j_1\|, \quad j = 1, \ldots, N_i. \quad (5.16)$$

Combining equations 5.16 with 5.15, we obtain that

$$d_p(\Phi(x_{test}), X^\phi_i) = \|y_{test} - y^1_1\|. \quad (5.17)$$

Therefore, based on the nearest-neighbor rule, the associated class index of the test point can be obtained as follows:

$$c^* = \arg \min_i d_p(\Phi(x_{test}), X^\phi_i) = \arg \min_i \|y_{test} - y^1_i\|. \quad (5.18)$$

6 Experiments

We will use simulated and real data, respectively, to demonstrate the efficiency of the proposed method in this section. In the first example, we use toy data to show that MKGS is numerically superior to KGS for implementing the QR decomposition. The second example aims to demonstrate the better performance of the class-incremental GDA/MKGS algorithm when used for a dynamic face recognition system. In the third example, we use a handwritten digital character recognition experiment to demonstrate the better performance of the proposed algorithm. All of these experiments are run on the platform of IBM personal computer with MATLAB software. The monomial kernel and the gaussian kernel are used in the experiments, which are respectively defined as follows:

- Monomial kernel: $k(x, y) = (x^T y)^d$, where $d$ is the monomial degree
- Gaussian kernel: $k(x, y) = \exp(-\|x - y\|^2 / \sigma)$, where $\sigma$ is the parameter of the gaussian kernel
6.1 Toy Example. In this example, we use toy data to test the performance of MKGS and KGS for implementing the QR decomposition. The GDA algorithm via KGS (GDA/KGS) include the batch GDA/KGS algorithm and the class-incremental GDA/KGS algorithm for updating discriminant vectors with the insertion of the \((c + 1)\) class. These two algorithms are given in appendices D and E, respectively.

Four clusters of artificial two-dimensional data sets are generated by the function \(y = x^2 + 0.5 + \varepsilon\), where the \(x\) values have a uniform distribution in \([-1, 1]\), and \(\varepsilon\) is a uniformly distributed random number on the interval \([0, 0.5]\). For each cluster, we generate 100 samples. The gaussian kernel with parameter \(\sigma = 0.1\) is used over the experiment to calculate the kernel matrix. The experiment has three steps:

1. Choose the samples in the first two clusters as training data to compute discriminant vector using the batch GDA/MKGS algorithm and the batch GDA/KGS algorithm, respectively. Then compute and display the projections of the test data onto the computed discriminant vector. Figures 1a and 2a, respectively, display the experimental results, where the figure shows the feature values (indicated by shade of gray) and contour lines of identical feature values.

2. Insert the third cluster into the training set, and then compute two discriminant vectors for discriminating the three clusters using the class-incremental GDA/MKGS algorithm and the class-incremental GDA/KGS algorithm, respectively. Figures 1b and 1c display the projections of the test data onto the two discriminant vectors of GDA/MKGS, while Figures 2b and 2c display the projections of the test data onto the two discriminant vectors of GDA/KGS.

3. Insert the fourth cluster into the training set, and then compute three discriminant vectors for discriminating the four clusters by using the class-incremental GDA/MKGS algorithm and the class-incremental GDA/KGS algorithm, respectively. Figures 1d through 1f display the projections of the test data onto the three discriminant vectors of GDA/MKGS, while Figures 2d through 2f display the projections of the test data onto the three discriminant vectors of GDA/KGS.

By contrast with the experimental results between Figure 1 and Figure 2, we can see that the GDA/MKGS algorithm achieves much better performance than the GDA/KGS algorithm. The projections of the toy data can be nicely separated in Figure 1, whereas they could not be well separated in Figure 2.

6.2 Face Recognition. This example aims to demonstrate the better performance of the class-incremental GDA/MKGS algorithm in terms of recognition accuracy and training time for dynamic face recognition task. We use the AR face database (Martinez & Benavente, 1998) to perform
Figure 1: Features extraction by class-incremental GDA/MKGS. (a) Projections onto the discriminant vector computed from the first two clusters using the batch GDA/MKGS algorithm. (b–c) Projections onto the two discriminant vectors computed from the first three clusters using the class-incremental GDA/MKGS algorithm; (d–f) Projections onto the three discriminant vectors computed from the four clusters using the class-incremental GDA/MKGS algorithm.

This experiment. The AR face database consists of over 3000 facial images of 126 subjects. Each subject contains 26 facial images recorded in two different sessions separated by two weeks, and each session consists of 13 images. The original image size is $768 \times 576$ pixels, and each pixel is represented by 24 bits of RGB color values. Figure 3 shows the 26 images for one subject; images 1 to 13 were taken in the first session and images 14 to 26 in the second session. Among the 126 subjects, we randomly select 70 subjects (50 males and 20 females) for this experiment. Similar to Cevikalp et al. (2005), we use only the nonoccluded images (those numbered 1 to 7 and 14 to 20) for the experiment. Before the experiment, all images are centered and cropped to the size of $468 \times 476$ pixels, and then are down-sampled into the size of $100 \times 100$ pixels.
Figure 2: Features extraction by class-incremental GDA/KGS. (a) Projections onto the discriminant vector computed from the first two clusters using the batch GDA/KGS algorithm. (b–c) Projections onto the two discriminant vectors computed from the first three clusters using the class-incremental GDA/KGS algorithm. (d–f) Projections onto the three discriminant vectors computed from the four clusters using the class-incremental GDA/KGS algorithm.

We use the twofold cross-validation method (Fukunaga, 1990) to perform the experiment: divide all the images into two subsets, and use one subset as the training set and the other as the test set. After performing the experiment, we swap the training set and the test set and repeat the experiment. Considering that our experiment aims to demonstrate the better performance of the class-incremental GDA/MKGS algorithm for dynamic face recognition task in terms of recognition accuracy and training time, our experiment will focus more on demonstrating the dynamic recognition procedures when new subjects are inserted into the training set. More specifically, we will design the two trials of the twofold cross-validation as follows.

In the first trial of the twofold cross validation, we choose 61 subjects among the 70 subjects for the experiment; we use the seven images
numbered 1, 2, 3, 4, 14, 15, and 16 in each subject as training images and the other seven images numbered 5, 6, 7, 17, 18, 19, and 20 in each subject as test images. The discriminant vectors are computed using the batch GDA/MKGS algorithm. The test recognition rate of the test images is then calculated based on the nearest-neighbor classifier. After finishing the recognition, we choose one subject from the remaining images and insert the seven images numbered 1, 2, 3, 4, 14, 15, and 16 into the training set and the other seven images into the test set. Then we update the discriminant vectors using the class-incremental GDA/MKGS algorithm and recalculate the test recognition rate. This procedure is repeated until all 70 subjects are included in the training set and the test set.

In the second trial, we swap the training images and the test images for each subject; that is, for each subject, we use the seven images numbered 1, 2, 3, 4, 14, 15, and 16 as test images and the other seven images numbered 5, 6, 7, 17, 18, 19, and 20 as training images, and then perform the same recognition procedure as in the first trial.

For comparison, we conduct the same experiment using other commonly used face recognition methods, including the Eigenfaces method (Turk & Pentland, 1991), the Fisherfaces method (Belhumeur, Hespanha, & Kriegman, 1997), the LDA method via the MGS algorithm (LDA/MGS; Zheng, Zou, & Zhao, 2004), the KPCA method (Schölkopf et al., 1998), and the standard GDA method (Baudat, 2000), respectively. The monomial kernel with degree $d = 2$ and gaussian kernel with $\sigma = 1e8$ are used in the

Figure 3: Images of one subject in the AR face database.
Figure 4: Average recognition rates of various systems with respect to the number of the training classes in the AR face recognition experiment, where the monomial kernel with degree $d = 2$ is used.

experiments. Figure 4 shows the average recognition rates of the two trials with respect to the number of the training classes when the monomial kernel with degree $d = 2$ is used, and Figure 5 shows the average recognition rates when the gaussian kernel with $\sigma = 1e8$ is used. As we can see from Figures 4 and 5, the GDA/MKGS method achieves the best recognition rate among the commonly used face recognition methods. Moreover, in order to demonstrate the incremental technique of the proposed algorithm, we compare the average training time between the batch GDA/MKGS algorithm and the class-incremental GDA/MKGS algorithm of updating the discriminant vectors when a new class is inserted into the training set. Figure 6 shows the experimental results. From Figure 6 we can clearly see that the class-incremental GDA/MKGS approach saves much more training time than the batch GDA/MKGS approach.

6.3 Handwritten Digital Character Recognition. In this example, we conduct handwritten digital character recognition experiment to further demonstrate the better recognition performance of the proposed algorithm.
We use the handwritten digits database of the U.S. Postal Service (USPS) collected from mail envelopes in Buffalo as the experimental data. The USPS database contains 7291 training points and 2007 test points of dimensionality 256 (Schölkopf et al., 1998). We choose the first 1500 training points as training data and all 2007 test points as test data for experiment. For the comparison, this experiment is also conducted using the PCA method, the LDA method, the KPCA method, and the traditional GDA method, respectively, where we use the monomial kernel with a different degree in each trial to calculate the kernel matrix for the kernel-based algorithms. The nearest-neighbor classifier is used over the experiments for classification task. Table 1 shows the best average recognition rates of various systems. From Table 1, we can see that the GDA/MKGS method achieves the best recognition rate among the systems. Moreover, we also compare the average recognition rates among the kernel-based algorithms with respect to the different choice of the monomial degree. The experimental results are shown in Figure 7. It can be clearly seen from Figure 7 that the GDA/MKGS
algorithm can significantly improve the performance of the traditional GDA algorithm.

7 Conclusion

In this work, an efficient algorithm has been presented to solve the discriminant vectors of GDA in the case of the small sample size problem. By applying QR decomposition rather than SVD, the proposed algorithm is computationally fast compared to the traditional GDA algorithms. More important, the proposed algorithm introduces an effective technique to update the discriminant vectors of GDA when new classes are inserted into
the training set, which is very desirable for designing dynamic recognition systems. Moreover, we have proposed a modified KGS algorithm in this work to replace the KGS algorithm proposed by Wolf and Shashua (2003) for implementing the QR decomposition in the feature space. This algorithm has turned out to be much more numerically stable than the KGS algorithm.

Experiments on both simulated and real data sets demonstrated the better performance of the class-incremental GDA/MKGS algorithm. On the simulated toy data, our experiment demonstrated the superiority of MKGS to KGS for implementing the QR decomposition. And the face recognition experiments on the AR face database and handwritten digital character recognition on the USPS database demonstrated the high recognition rates of the proposed GDA/MKGS algorithm in contrast to other commonly used face recognition methods. Moreover, our face experiments demonstrated the computational advantage of the class-incremental GDA/MKGS algorithm when new classes are dynamically inserted into the training set.
Appendix A: Proof of Theorem 1

Proof. Because $u_i^\Phi - u_1^\Phi = (u_i^\Phi - u^\Phi) - (u_1^\Phi - u^\Phi)$, we get

$$\text{span}\{u_i^\Phi - u_1^\Phi | i = 2, \ldots, c\} \subseteq \text{span}\{u_i^\Phi - u_1^\Phi | i = 1, \ldots, c\}. \quad (A.1)$$

Note that $u^\Phi - u_1^\Phi = \frac{1}{N} \sum_{i=1}^{c} N_i (u_i^\Phi - u_1^\Phi) = \frac{1}{N} \sum_{i=2}^{c} N_i (u_i^\Phi - u_1^\Phi)$. Thus, we get

$$\text{span}\{u_i^\Phi - u_1^\Phi | i = 2, \ldots, c\} = \text{span}\{u^\Phi - u_1^\Phi, u_i^\Phi - u_1^\Phi | i = 2, \ldots, c\}. \quad (A.2)$$

Because $u_i^\Phi - u^\Phi = (u_i^\Phi - u_1^\Phi) - (u^\Phi - u_1^\Phi)$, we get

$$\text{span}\{u_i^\Phi - u^\Phi | i = 1, \ldots, c\} \subseteq \text{span}\{u_i^\Phi - u_1^\Phi, u_i^\Phi - u_1^\Phi | i = 2, \ldots, c\}. \quad (A.3)$$

From equations A.2 and A.3, we obtain

$$\text{span}\{u_i^\Phi - u_1^\Phi | i = 1, \ldots, c\} \subseteq \text{span}\{u_i^\Phi - u_1^\Phi | i = 2, \ldots, c\}. \quad (A.4)$$

From equations A.1 and A.4, we obtain

$$\text{span}\{u_i^\Phi - u^\Phi | i = 1, \ldots, c\} = \text{span}\{u_i^\Phi - u_1^\Phi | i = 2, \ldots, c\}. \quad (A.5)$$

Combining equations 4.1 and A.5, we obtain

$$\overline{S_B^\Phi(0)} = \text{span}\{u_i^\Phi - u_1^\Phi | i = 2, \ldots, c\}.$$

Appendix B: Proof of Theorem 2

Proof. Note that

$$\Phi(x_i^j) - \Phi(x_1^j) = (\Phi(x_i^j) - u_i^\Phi) - (\Phi(x_1^j) - u_1^\Phi). \quad (B.1)$$

Thus, we obtain

$$\text{span}\{\Phi(x_i^j) - \Phi(x_1^j) | j = 2, \ldots, N_i\} \subseteq \text{span}\{\Phi(x_i^j) - u_i^\Phi | j = 1, \ldots, N_i\}. \quad (B.2)$$

Note that

$$\sum_{j=2}^{N_i} (\Phi(x_i^j) - \Phi(x_1^j)) = \sum_{j=1}^{N_i} (\Phi(x_i^j) - \Phi(x_1^j)) = N_i (u_i^\Phi - \Phi(x_1^j)). \quad (B.3)$$
Thus, we get

\[
\text{span}\{ \Phi(x_i^j) - \Phi(x_i^1) \mid j = 2, \ldots, N_i \} = \text{span}\{ u_i^\Phi - \Phi(x_i^1), \Phi(x_i^j) - \Phi(x_i^1) \mid j = 2, \ldots, N_i \}. \tag{B.4}
\]

Because \( \Phi(x_i^j) - u_i^\Phi = (\Phi(x_i^j) - \Phi(x_i^1)) - (u_i^\Phi - \Phi(x_i^1)) \), we get

\[
\text{span}\{ \Phi(x_i^j) - u_i^\Phi \mid j = 1, \ldots, N_i \} \subseteq \text{span}\{ u_i^\Phi - \Phi(x_i^1), \Phi(x_i^j) - \Phi(x_i^1) \mid j = 2, \ldots, N_i \}. \tag{B.5}
\]

From equations B.4 and B.5, we get

\[
\text{span}\{ \Phi(x_i^j) - u_i^\Phi \mid j = 1, \ldots, N_i \} \subseteq \text{span}\{ \Phi(x_i^j) - \Phi(x_i^1) \mid j = 2, \ldots, N_i \}. \tag{B.6}
\]

From equations B.2 and B.6, we obtain

\[
\text{span}\{ \Phi(x_i^j) - u_i^\Phi \mid j = 1, \ldots, N_i \} = \text{span}\{ \Phi(x_i^j) - \Phi(x_i^1) \mid j = 2, \ldots, N_i \}. \tag{B.7}
\]

From equation B.7, we obtain

\[
\text{span}\{ \Phi(x_i^j) - u_i^\Phi \mid i = 1, \ldots, c; j = 1, \ldots, N_i \}
\]

\[
= \text{span}\{ \Phi(x_i^j) - \Phi(x_i^1) \mid i = 1, \ldots, c; j = 2, \ldots, N_i \}. \tag{B.8}
\]

From equations 4.2 and B.8, we obtain

\[
\overline{S^\phi_{W_0}(0)} = \text{span}\{ \Phi(x_i^j) - \Phi(x_i^1) \mid i = 1, \ldots, c; j = 2, \ldots, N_i \}.
\]

**Appendix C: Proof of Theorem 3**

**Proof.** Because \( \Phi(x_i^j)(i = 1, \ldots, c; j = 1, \ldots, N_i) \) are linearly independent, we have

\[
\text{rank} (\Phi(x_1^1), \Phi(x_1^2), \ldots, \Phi(x_c^{N_c})) = \sum_i N_i = N. \tag{C.1}
\]
Moreover, we have

\[
\text{rank}\left(\Phi(x_1^1), \Phi(x_1^2), \ldots, \Phi(x_c^N)\right) = \text{rank}\left(\Phi(x_1^1), \Phi(x_1^2) - \Phi(x_1^1), \ldots, \Phi(x_c^N) - \Phi(x_1^1)\right). \tag{C.2}
\]

From equations C.1 and C.2, we obtain that

\[
\text{rank}\left(\Phi(x_1^1), \Phi(x_1^2) - \Phi(x_1^1), \ldots, \Phi(x_c^N) - \Phi(x_1^1)\right) = N. \tag{C.3}
\]

Equation C.3 means that the \(N\) vectors \(\Phi(x_1^1), \Phi(x_1^2) - \Phi(x_1^1), \ldots, \Phi(x_c^N) - \Phi(x_1^1)\) are linearly independent. Thus, we obtain that

\[
\text{rank}\left(\Phi(x_1^2) - \Phi(x_1^1), \ldots, \Phi(x_c^N) - \Phi(x_1^1)\right) = N - 1. \tag{C.4}
\]

Moreover, we have

\[
\text{rank}\left(\Phi(x_1^1) - u^\Phi, \Phi(x_1^2) - u^\Phi, \ldots, \Phi(x_c^N) - u^\Phi\right) = \text{rank}\left(\Phi(x_1^1) - u^\Phi, \Phi(x_1^2) - \Phi(x_1^1), \ldots, \Phi(x_c^N) - \Phi(x_1^1)\right) \geq N - 1. \tag{C.5}
\]

From equations C.4 and C.5, we obtain that

\[
\text{rank}\left(\Phi(x_1^1) - u^\Phi, \Phi(x_1^2) - u^\Phi, \ldots, \Phi(x_c^N) - u^\Phi\right) \geq N - 1. \tag{C.6}
\]

Note that \(\sum_{i=1}^c \sum_{j=1}^{N_i} (\Phi(x_i^j) - u^\Phi) = 0\). Thus, we have

\[
\text{rank}\left(\Phi(x_1^1) - u^\Phi, \Phi(x_1^2) - u^\Phi, \ldots, \Phi(x_c^N) - u^\Phi\right) \geq N - 1. \tag{C.7}
\]

Combining equations C.6 and C.7, we obtain that

\[
\text{rank}\left(\Phi(x_1^1) - u^\Phi, \Phi(x_1^2) - u^\Phi, \ldots, \Phi(x_c^N) - u^\Phi\right) = N - 1. \tag{C.8}
\]

Note that \(\overline{S_T^\Phi(0)} = \text{span}\{\Phi(x_i^j) - u^\Phi| i = 1, \ldots, c; j = 1, \ldots, N_i\}\). Thus, from equation C.8, we obtain that \(\text{rank}(\overline{S_T^\Phi(0)}) = N - 1\).

**Appendix D: Batch GDA/KGS Algorithm**

1. Let \(s_1 = t_1 = e_1, D_{11} = K(1, 1)\);
2. Repeat for \(j = 2, \ldots, N - 1\)
a. Compute 
\[ s_j = \left( \frac{t_{11}K(1,j)}{D_{11}}, \ldots, \frac{\sum_{q=1}^{j-1} t_{(j-1),q}K(q,j)}{D_{(j-1)(j-1)}}, 1, 0, \ldots, 0 \right)^T \];

b. Compute 
\[ t_j = (-t_1, \ldots, -t_{j-1}, 1, 0, \ldots, 0)s_j; \]
c. Compute 
\[ D_{jj} = \sum_{p,q=1}^{j} t_{pj}t_{qj}K(p,q); \]

3. 
\[ [\beta_{N-c+1}, \ldots, \beta_{N-1}] = A^\Phi [t_{N-c+1}, \ldots, t_{N-1}] (D_{jj})_{i,j=N-c+1,\ldots,N-1}^{-1/2} \]

The \( c - 1 \) vectors \( \beta_{N-c+1}, \ldots, \beta_{N-1} \) are the discriminant vectors of GDA as for the case of small sample size problem.

Appendix E: Class-Incremental GDA/KGS Algorithm: Updating Discriminant vectors with the Insertion of the \((c + 1)\)th Class

1. Repeat for \( j = 1, \ldots, N - c \)
   a. Compute \( \tilde{t}_j = (t_j^T, 0, \ldots, 0)^T; \)
   b. Compute \( \tilde{D}_{jj} = D_{jj}; \)

2. Repeat for \( j = N - c + 1, \ldots, N + N_c + 1 - 1 \)
   a. Compute \( \tilde{s}_j = \left( \frac{\tilde{t}_{11}K(1,j)}{\tilde{D}_{11}}, \ldots, \frac{\sum_{q=1}^{j-1} \tilde{t}_{(j-1),q}K(q,j)}{\tilde{D}_{(j-1)(j-1)}}, 1, 0, \ldots, 0 \right)^T; \)
   b. Compute \( \tilde{t}_j = (-\tilde{t}_1, \ldots, -\tilde{t}_{j-1}, 1, 0, \ldots, 0)\tilde{s}_j; \)
   c. Compute \( \tilde{D}_{jj} = \sum_{p,q=1}^{j} \tilde{t}_{pj}\tilde{t}_{qj}K(p,q); \)

3. 
\[ [\beta_{N+N_c+1-c}, \ldots, \beta_{N+N_c+1-1}] = \tilde{A}^\Phi [\tilde{t}_{N+N_c+1-c}, \ldots, \tilde{t}_{N+N_c+1-1}] (\tilde{D}_{jj})_{i,j=N+N_c+1-c,\ldots,N+N_c+1-1}^{-1/2} \]

The \( c \) vectors \( \beta_{N+N_c+1-c}, \ldots, \beta_{N+N_c+1-1} \) are the new discriminant vectors of GDA after the \((c + 1)\)th class is inserted into the training set.

Acknowledgments

I thank the anonymous reviewers for their valuable comments and suggestions. I also thank Songcan Chen from the Department of Computer Science and Engineering, University of Aeronautics and Astronautics, China, for his kind discussions and valuable advice. This work was supported in part by the National Natural Science Foundations of China under grants 60503023 and 60473035, and in part by the Jiangsu Natural Science Foundations under grants BK2005407 and BK2005122.

References


Received January 10, 2005; accepted August 19, 2005.