THESIS

INFRARED IMAGING FACE RECOGNITION USING NONLINEAR KERNEL-BASED CLASSIFIERS

by

Dimitrios I. Domboulas

December 2004

Thesis Committee Supervisor:	Monique P. Fargues
Thesis Committee Members:	Roberto Cristi
	Gamani Karunasiri

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704–0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** December 2004	**3. REPORT TYPE AND DATES COVERED** Engineer's Thesis

4. TITLE AND SUBTITLE: Infrared Imaging Face Recognition Using Nonlinear Kernel-Based Classifiers	**5. FUNDING NUMBERS**
6. AUTHOR(S) Dimitrios I. Domboulas	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943–5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited	**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

In recent years there has been an increased interest in effective individual control and enhanced security measures, and face recognition schemes play an important role in this increasing market. In the past, most face recognition research studies have been conducted with visible imaging data. Only recently have IR imaging face recognition studies been reported for wide use applications, as uncooled IR imaging technology has improved to the point where the resolution of these much cheaper cameras closely approaches that of cooled counterparts. This study is part of an on–going research conducted at the Naval Postgraduate School which investigates the feasibility of applying a low cost uncooled IR camera for face recognition applications. This specific study investigates whether nonlinear kernel–based classifiers may improve overall classification rates over those obtained with linear classification schemes. The study is applied to a 50 subject IR database developed in house with a low resolution uncooled IR camera. Results show best overall mean classification performances around 98.55% which represents a 5% performance improvement over the best linear classifier results obtained previously on the same database. This study also considers several metrics to evaluate the impacts variations in various user–specified parameters have on the resulting classification performances. These results show that a low–cost, low–resolution IR camera combined with an efficient classifier can play an effective role in security related applications.

14. SUBJECT TERMS Face Recognition, Pattern Classification, Infrared, GDA, Distances, Eigenvectors	**15. NUMBER OF PAGES** 129
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540–01–280–5500 Standard Form 298 (Rev. 2–89) Prescribed by ANSI Std. 239–18

THIS PAGE INTENTIONALLY LEFT BLANK

INFRARED IMAGING FACE RECOGNITION USING NONLINEAR KERNEL-BASED CLASSIFIERS

Dimitrios I. Domboulas
Captain, Hellenic Air Force
B.S., in Electronic Engineering, Hellenic Air Force Academy, Athens, 1992

Submitted in partial fulfillment of the
requirements for the degrees of

ELECTRICAL ENGINEER

and

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 2004

Author:	Dimitrios I. Domboulas
Approved by:	Monique P. Fargues Thesis Committee Supervisor
	Roberto Cristi Thesis Committee Member
	Gamani Karunasiri Thesis Committee Member
	John P. Powers Chairman, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

In recent years there has been an increased interest in effective individual control and enhanced security measures, and face recognition schemes play an important role in this increasing market. In the past, most face recognition research studies have been conducted with visible imaging data. Only recently have IR imaging face recognition studies been reported for wide use applications, as uncooled IR imaging technology has improved to the point where the resolution of these much cheaper cameras closely approaches that of cooled counterparts. This study is part of an on–going research conducted at the Naval Postgraduate School which investigates the feasibility of applying a low cost uncooled IR camera for face recognition applications. This specific study investigates whether nonlinear kernel–based classifiers may improve overall classification rates over those obtained with linear classification schemes. The study is applied to a 50 subject IR database developed in house with a low resolution uncooled IR camera. Results show best overall mean classification performances around 98.55%, which represents a 5% performance improvement over the best linear classifier results obtained previously on the same database. This study also considers several metrics to evaluate the impacts variations in various user–specified parameters have on the resulting classification performances. These results show that a low–cost, low–resolution IR camera combined with an efficient classifier can play an effective role in security related applications.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

To my thesis advisor Dr. Monique P. Fargues, whose outstanding theoretical background, experience, and expertise was the “foundation stone” for the successful completion of the subject study’ s objectives. Her precious help, directions, inspiration, encouragement and patience made the subject study a worthwhile and memorable learning experience, and made one more individual adopt the following philosophical quotation, as presented below:

“The principle goal of education is to create men who are capable of doing new things, not simply repeating what other generations have done”.

Jean Piaget

To my parents, Ioannis and Anna and my wife Sofia, for their endless love and encouragement in the completion of this striving effort.

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

Numerous face recognition studies have been reported in the literature over the years due to their increasing usage in security, access control, biometric and other types of applications. In the past most research studies have been conducted in visible imaging data and only recently have IR imaging face recognition studies been reported. IR imaging offers the main advantage over visible imaging of being invariant to illumination changes. Good resolution IR imaging has been restricted to very expensive cooled devices which slowed the interest in IR face recognition down until the recent past. However, technological advances have increased the resolution of uncooled IR cameras to the point that their resolution closely approaches that of cooled devices at a fraction of the cost. This study extends research conducted earlier by Pereira [Pereira, 2002] and Lee [Lee, 2004], who collected an infrared (IR) face database using a low resolution IR camera and considered linear classifier approaches. This study explores the performance of a kernel–based nonlinear pattern classification algorithm applied to the same 50 subject database used by Lee.

The nonlinear kernel–based Generalized Discriminant Analysis (GDA) considered here is basically an extension of the Linear Discriminant Analysis (LDA), where the data is implicitly projected into a new higher dimensional nonlinear transformed domain prior to the application of the LDA step. As a result, the GDA approach takes advantages of the nonlinear characteristics of data, which LDA does not have access to for higher classification rates.

First, this study investigates several distance measures and selects the "best" one to apply to the data under investigation. It also considers various user–specified parameters such as the specific type of kernel function, the number of eigenvectors selected in the GDA step and investigates their resulting impacts on overall mean classification performances. As part of the study, the concept of distance maps is defined to visually evaluate and compare classifier performances, and apply concepts of confidence intervals, confidence scores, and rank–scores to assist the user in designing the "best" classifier.

The study implements a cross–validation variant to insure results derived are statistically meaningful, and software implementation steps are vectorized to minimize the associated computational load. Results show that the GDA implementation, when selecting the Mahalanobis Angular distance, a polynomial kernel of degree 2 and the 500 top eigenvectors for the GDA projection step, provides the best mean classification results (98.55%). These results correspond to a 5% improvement over the best derived with the linear Fisherface implementation, and show that significant improvement may be gained by implementing nonlinear kernel–based schemes. Results also show that a low–cost, low–resolution IR camera combined with an efficient classifier can play an effective role in security related applications.

I. INTRODUCTION

Numerous face recognition studies have been reported in the literature [Adini, 1997; Hearst, 1998; Liu, April 2004] over the years due to their increasing usage in security, access control, biometric and other types of applications. In the past, most research studies have been conducted in visible imaging data and only recently have infrared (IR) imaging face recognition studies been reported [Pereira, 2002; Chen, 2003; Socolinsky, 2003; Lee, 2004]. IR imaging offers the main advantage over visible imaging of being invariant to illumination changes [Chen, 2003]. Until the recent past, good resolution IR imaging was restricted to very expensive cooled cameras which lowered the interest in IR face recognition. However, technological advances have increased the resolution of uncooled IR cameras to the point that their resolution closely approaches that of cooled cameras at a fraction of the cost.

This study is an extension to the research conducted earlier by Pereira [Pereira, 2002] and Lee [Lee, 2004]. First, Pereira initialized the study. He developed an uncooled IR image capture system, generated a small database of 14 adult subjects, collected in a controlled indoor environment, and implemented two classic linear classification algorithms using Matlab, the Principal Component Analysis (PCA), and the Fisherface [Pereira, 2002]. Later Lee extended the database to 50 adult subjects and applied the same two linear classification algorithms [Lee, 2004]. This study extends the previous research to a kernel–based nonlinear classification approach, the Generalized Discriminant Algorithm, (GDA) and compares classification performances with those obtained using linear implementations.

The following sections briefly reviews the concepts of IR radiation, followed by a presentation of the procedures used for image collection.

A. THEORETICAL BACKGROUND

The IR radiation is the section of the electromagnetic spectrum whose wavelengths lie primarily between 1–100 μm. This wavelength range is above the visible spectrum range and below the microwave range [Dereniak, 1996]. Indicatively, Fig. 1.1 illustrates the spectrum, marking the ultraviolet, visible and IR regions.

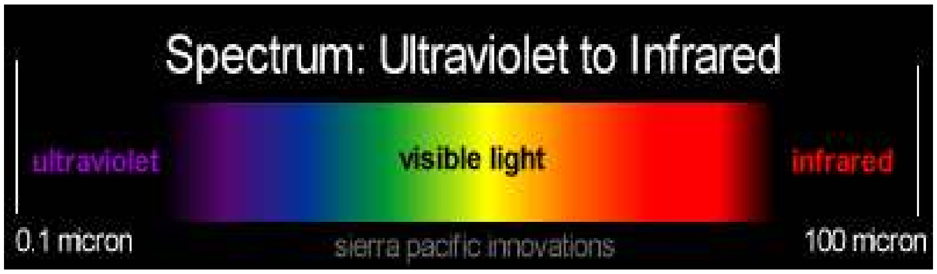


Figure 1.1. Visible Spectrum versus the Ultraviolet and the IR Ranges (From [Sierra Pacific Corp., 2004].).

Though the IR radiation is not visible, it can be perceived in the form of thermal energy, as all objects which have temperatures greater than 0 K emit thermal radiation. The amount of thermal energy an object emits depends on both the fourth power of its absolute temperature and its *emissivity*. Emissivity of an object indicates to which degree it emits thermal radiation, and it takes a value between 0 and 1. The larger the emissivity of an object is, the larger the thermal radiation it emits, and is usually a characteristic determined by the material structure of the object's surface (i.e., its molecular characteristics). A blackbody is defined to be the object that has emissivity value equal to 1. This ideal case of an object does not exist in nature.

The human body has an emissivity of about 0.98, at 310 K [Jones, 1998]. Additionally, each individual has a vein and tissue structure which is peculiar to him/her, resulting in the unique emission of a thermal radiation pattern. The temperature variation across the face of a typical person is about 7^o C. Since the spatial variation of one's facial temperature profile is not very abrupt, the low spatial resolution of the IR cameras generally should not affect the quality of the images. For this reason, the IR radiation emitted by the human face is ideal for high classification performance scores in face recognition applications.

An additional advantage of using IR images as compared to visible images for face recognition applications is that IR images are not affected by variations in the lighting of a particular environment while visible images tend to lose much of their information in poorly lit conditions [Chen, 2003]. However, apart from the low resolution, the main drawback of using IR images is the relatively high cost of IR cameras (purchase and maintenance). This additional cost is due to the fact that cooled IR cameras require cryogenic cooling in order to achieve high temperature resolution. In order to make such imagery more cost–effective, recent advances in uncooled infrared camera technology provide a less expensive means of acquiring infrared images.

Apart from face recognition applications, IR imaging has also been successfully used in many civilian and military applications such as Forward Looking Infrared (FLIR) systems for fighter aircrafts (F–14, F–16, F–18, etc.), helicopters, and IR targeting systems–missiles which use IR radiation to locate the targets [Sierra Pacific Corp., 2004].

B. EQUIPMENT USED

The IR face images were collected at the Naval Postgraduate School (NPS), using an uncooled IR camera (IR 160), manufactured by Infrared Solutions.

Even though the spatial resolution $(160 \times 120$ pixels) and temperature resolution (60 mK) provided by the camera are relatively low, the wavelength region in which it operates (8–14 μm) is critical for capturing wavelengths emitted by human body at about 310 K (which corresponds to peak wavelength of 10 μm). This phenomenon is illustrated in Fig. 1.2.

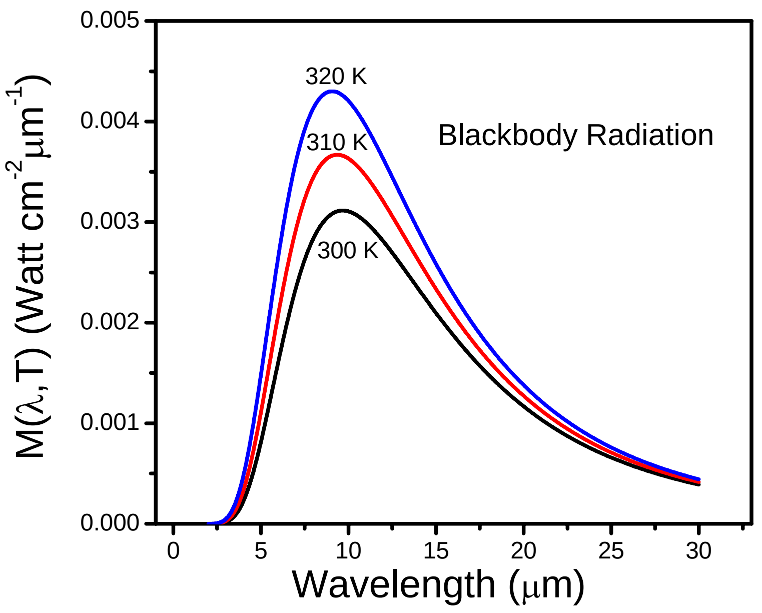


Figure 1.2. Blackbody spectrum for temperatures near that of the human body (From [Pereira, 2002].).

The camera used here is said to be uncooled because there is no requirement to cool the camera's sensor, which is a requirement for ordinary cooled cameras. The detailed specifications of the camera can be found in [Infrared Solutions Inc., 2004].

The IR–160 is connected to a monitor to allow for previewing the IR images before their capture. The camera's output produces IR images in a digital format. The digital image is then transmitted to a PC via an 8–bit, RS–232 serial port. Next, the collected images are cropped [Lee, 2004], reshaped in a column–vector format, and inserted in the database matrix (this process will be discussed further in Chapter IV).

The equipment used in the IR database collection is illustrated in Fig. 1.3. The detailed, geometric layout of the exact location of the individual posing for capturing his picture, his/her gazing directions towards marked spots on the facing wall, and the location of the pieces of equipment used (IR camera, monitor, PC) are shown in Figs. 1.4 and 1.5.

The nine spots on the wall, (forming a square with the center marked on it), correspond to the gazing positions of the individual, having a specific facial expression (neutral, smiling, or pronouncing the vowel "u"), and for which images were captured. In addition, a tenth image was collected with random gazing position, located within the defined square. Each group of ten images constitutes one out of three possible sections (facial expressions) collected for a given subject (class). Additional details regarding the organization of the IR image collection can be found in [Lee, 2004].

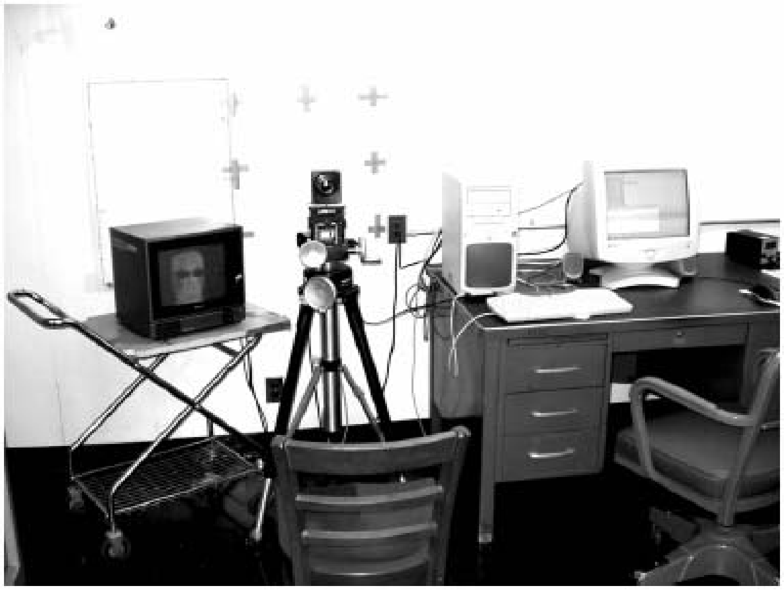


Figure 1.3. Equipment components used for IR data collection (From [Lee, 2004].).

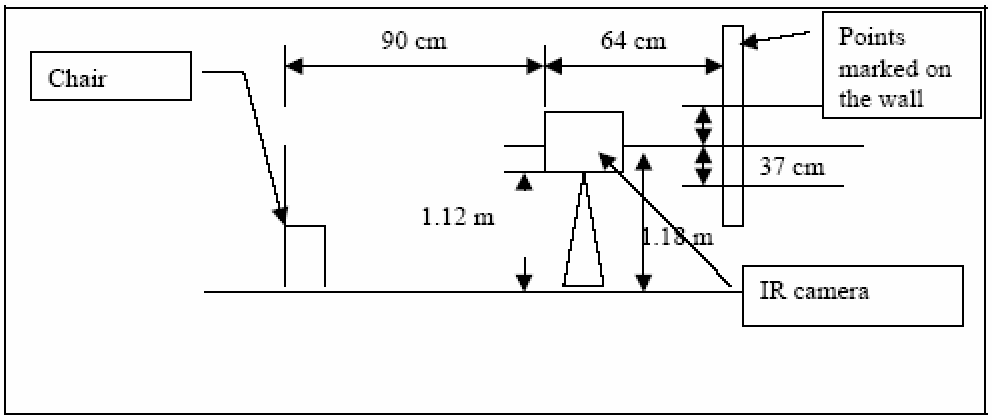


Figure 1.4. Lateral aspect of IR data collection system layout (From [Pereira, 2002].).

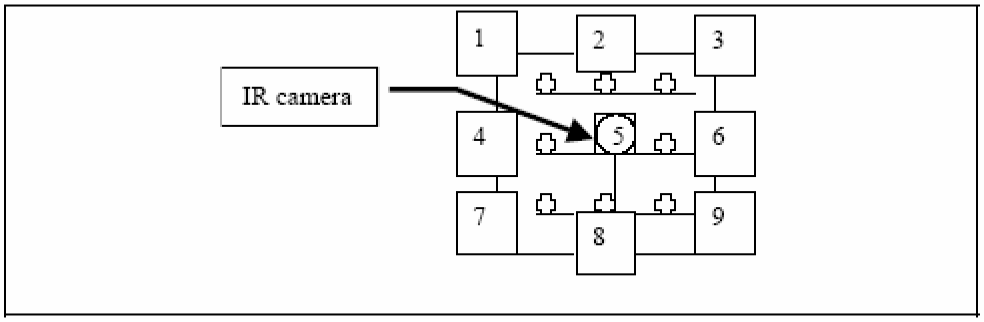


Figure 1.5. Forward aspect of IR data collection system layout (From [Pereira, 2002].).

C. THESIS OVERVIEW

This thesis consists of four chapters. Chapter I presents the basic idea behind the work conducted. Chapter II reviews the concepts of linear classifiers. Extensions to nonlinear classification schemes are described in Chapter III. Next, Chapter IV presents the experiments conducted on the face database and discusses results obtained. Finally, conclusions and recommendations for further research are presented in Chapter V. Mathematical details can be found in Appendix A. The software implemented during the study is included in Appendix B. Last, Appendix C includes typical performance measurement plots.

D. SUMMARY

This chapter presented the principles of IR imaging for face recognition applications. It also discussed some of the advantages of using IR over visible imaging and it presented some of the technical characteristics of the uncooled, IR–160 camera used in this study, together with some characteristics of the equipment used for the IR data collection. The following chapter presents the linear classification algorithms considered for benchmarking purposes against the nonlinear kernel based classifier approach, which is the main focus of this thesis.

II. LINEAR CLASSIFICATION ALGORITHMS

This chapter first presents a brief introduction to two classical linear classification schemes widely used in the pattern recognition community, the Principal Component Analysis (PCA) and the Linear Discriminant Analysis (LDA). Next, it presents a brief comparison between the two schemes.

A. INTRODUCTION

The PCA and the LDA schemes have been used extensively and successfully in applications for image and speech recognition, machine learning, financial planning, and various other scientific areas [Martinez, 2001]. The database considered in the present face recognition study includes 50 adult subjects (i.e., classes), each having thirty infrared images. Identical samples and images were used by Lee in a study conducted in 2004 [Lee, 2004]. Two main types of class assignments are typically available in pattern recognition applications, namely the "closed set" and the "open set" implementations. Whereas the closed set implementation assumes that all testing trials belong to the database used to design the classification algorithm, no such assumption is made in the open set implementation. As a result, testing data trials may not necessarily be assigned to a given class in open set implementations. The present study was restricted to the closed set implementation type, i.e., all individuals tested were considered to be in the training database.

B. PRINCIPAL COMPONENT ANALYSIS (PCA) – EIGENFACE IMPLEMENTATION

1. Introduction

Karl Pearson first introduced the Principal Component Analysis in 1901 [Lee, 2004]. The PCA method was created initially for data compression, but has also been extensively used with success in pattern recognition applications. The basic idea behind PCA is to represent features extracted from the data in terms of the linearly independent eigenvectors obtained from the data's covariance matrix. Dimension reduction is obtained by approximating the feature vectors using a subset selected from the top eigenvectors (i.e., those associated with the eigenvalues with larger magnitude), thereby preserving most of the feature variance information. The PCA approach has been used extensively in classification applications even though it is, theoretically, not well–suited to deal with such problems because the ability to discriminate between individuals may often be contained in the small details (i.e., associated with eigenvectors with smaller magnitudes).

2. Algorithm Description

The PCA algorithm (also known as “Eigenspace Projection”, or “Karhunen Loeve” decomposition) seeks a linear projection direction which best represents the data in a norm sense [Fargues, 2001]. Research by [Yambor, 2000] presents a good illustration of this process and is discussed next. Recall that all available P images are stored in matrices of dimension $(m \times n)$. First, these images are reshaped into a set of column vectors $\{x^i\}_{i=1,\dots,P}$ of length $N = mn$. Next, the reshaped vectors are stacked column–wise, to form the $(N \times P)$ dimensional training image data matrix $X = \left[x^1, \dots, x^P\right]$. As a result, the overall mean image of length N is given by:

$$m = \frac{1}{P}\sum_{i=1}^{P} x^i, \qquad (2.1)$$

which leads to the corresponding i^{th} reshaped image vector, defined by the equation:

$$\overline{x}^i = x^i - m = \left[\overline{x}_1, \overline{x}_2, \overline{x}_3, \dots, \overline{x}_N\right]^T. \qquad (2.2)$$

As a result, the centered $(N \times P)$ dimensional training data matrix $\overline{X}$ is given by

$$\overline{X} = [\overline{x}^1 \mid \overline{x}^2 \mid \overline{x}^3 \mid \dots \mid \overline{x}^P], \qquad (2.3)$$

where the vertical bars denote that the column vectors $\{x^i\}_{i=1,\dots,P}$ are appended in sequence, in order to form the matrix $\overline{X}$. The N–dimensional data covariance matrix is given by:

$$\Omega = \overline{X}\overline{X}^T. \qquad (2.4)$$

Note that the covariance matrix, Ω, has up to $\min(P, N)$ non zero eigenvalues. Next, also recall that the eigenvector associated with the largest eigenvalue is that associated to the direction with the highest energy, and so on. Therefore, the PCA projection matrix V is made up from the top K eigenvectors v_K, where K is user–specified, as:

$$V = [v_1 \mid v_2 \mid .... \mid v_K]. \tag{2.5}$$

As a result, the projected centered data $\tilde{X}$ can be expressed as:

$$\tilde{X} = V^T \bar{X}. \tag{2.6}$$

Next, the projected data is used to compute the projected “class–centroids” as the means of the projected data associated with each class resulting in a class–centroid matrix of dimension $(K \times C)$, where C is the number of classes. Lastly, testing images are projected onto the smaller dimensional space using the projection matrix, V, defined with the training data. The classification decision is obtained by selecting as the class that one which is closest in norm to the projected testing image. Various types of norms may be applied, and in most cases, the simple Euclidean distance (Norm–2) is selected.

According to [Belhumeur, 1997], the PCA method yields projection directions that maximize the total scatter across all images. In order to find these projections, the training data covariance matrix has to be calculated and its eigenvalue–eigenvector decomposition performed. Eigenvectors associated with the top eigenvalues in magnitude constitute the projections’ directions mentioned above. However, PCA still retains unwanted information, e.g., that due to facial expressions and lighting. According to a study reported in [Adini, 1997], intra–class variations that were generated by altering the direction of one’s gaze and lighting conditions may be larger than inter–class variations. As discussed in [Pereira, 2004], classification errors may be reduced by discarding the first few top eigenvectors when dealing with a small IR database. However, Lee showed that such a trend does not extend as the database size [Lee, 2004]. Nevertheless, it remains true that PCA is not optimally designed to discriminate between classes. However, PCA can help to reduce problems associated with dimensionality due to its dimension reduction capability. This process will be more fully described in the next section.

Implementation of the PCA method requires defining the “Total–Scatter” matrix, S_T (i.e., the data covariance matrix) via the following equation:

$$S_T = \sum_{k=1}^{P} (x_k - \mu)(x_k - \mu)^T, \tag{2.7}$$

where x_k is an image reshaped as a column vector and μ is the mean of all training images.

The projection vectors matrix, W_{opt}, is chosen in such a way that it maximizes the determinant of the total scatter matrix of the projected training data matrix according to:

$$W_{opt} = \arg(\max_W |W^T S_T W|) = [w_1, w_2, \ldots w_m], \tag{2.8}$$

where $w_1, w_2, \ldots, w_m$ constitute the set of N–dimensional projection eigenvectors (sorted by descending eigenvalue magnitude that are obtained from S_T).

The PCA is implemented algorithmically using the following steps:

- Extract feature vectors from data images, which are obtained from the subjects and reshaped into a column–vector format.
- Organize the training data into a matrix form (training–data–matrix) by appending the column vectors, which represent the various training images, such that images of the same class are appended together. Consequently, all operations take place in a column–wise fashion, in the various steps of the algorithm.
- Calculate the covariance matrix of the training data matrix.
- Calculate the eigendecomposition of the covariance matrix.
- Sort the eigenvalues in decreasing order of their magnitude, along with their associated eigenvectors. Keep all eigenvectors that are associated with a particular eigenvalue magnitude above a user–defined threshold.
- Project the training data matrix onto the eigenvector subspace.
- Calculate the class centroids from the projected training data. The class centroids are used to characterize each specific class.
- Project the testing data onto the eigenvectors subspace.
- Calculate distances between each projected testing image to all class centroids.
- Assign each testing image to a specific class by selecting the class, which corresponds to the smallest distance between the projected testing and all previously–computed training class centroids.

C. FISHERFACE APPROACH

1. Introduction

The Linear Discriminant Analysis (LDA) algorithm, also known as Fisher Linear Discriminant (FLD) analysis, is based on a linear projection of the data onto a lower dimensional feature space, which identifies the projection that best discriminates among classes, rather than those directions that best represent the data [Martinez, 2001]. As a result, the LDA approach is less affected than PCA by variations in lighting and face expression [Belhumeur, 1997].

2. Algorithm Description

The LDA approach uses the concept of intra–class (or "Within–Class") and inter–class (or "Between–Class") scatter matrices [Yambor, 2000]. The Within–Class–Scatter matrix (WCS) S_i for class *i* is defined as:

$$S_i = \sum_{x \in X_i} (x - m_i)(x - m_i)^T, \tag{2.9}$$

where m_i is the mean image of class *i*. The overall WCS matrix, S_W, is defined as follows:

$$S_w = \sum_{i=1}^{C} S_i, \tag{2.10}$$

where *C* is the number of classes.

In a similar fashion, the Between–Class–Scatter matrix (BCS) S_B is defined as:

$$S_B = \sum_{i=1}^{C} n_i (m_i - m)(m_i - m)^T, \tag{2.11}$$

where *m* is the mean of all the training images and n_i is the number of images in the i^{th} class.

The LDA projection direction matrix, W_{opt} is defined as the matrix which maximizes the ratio of the determinant of the Between–Class–Scatter S_B to the determinant of the Within–Class–Scatter S_W, both defined on the training data [Belhumeur, 1997]. This is denoted by the equation, known as the "Rayleigh quotient":

$$W_{opt} = \arg\left(\max_W \frac{|W^T S_B W|}{|W^T S_W W|} \right). \tag{2.12}$$

Appendix A.1 shows that finding the solution for a particular Rayleigh quotient is equivalent to solving the following generalized eigenvalue–eigenvector problem, (provided that the WCS matrix S_W is non–singular):

$$S_B w_i = \lambda_i S_w w_i, i = 1, 2, ...k, \tag{2.13}$$

where λ_i is the i^{th} generalized eigenvalue, w_i is the i^{th} generalized eigenvector (obtained by solving the above generalized eigenproblem), and $W_{opt} = [w_1, w_2, ..., w_k]$ is the eigenvector matrix solution. Recall that there is a maximum potential occurrence of $C-1$ nonzero generalized eigenvalues, as reviewed in Appendix A.2 [Belhumeur, 1997]). These $C-1$ nonzero eigenvectors constitute the Fisher basis vectors, and are typically sorted by order of decreasing eigenvalues.

Note that the *N*–dimensional WCS matrix S_W has a maximum rank equal to $P-C,$ where *P* is the number of images in the training set, and *C* is the number of classes, (reviewed in Appendix A.3). In addition, the *N*–dimensional BCS matrix S_B has a maximum rank equal to $C-1,$ as this matrix is the combination of *C* matrices of rank 1. Note also that the number of images, *P,* is usually smaller than the number of pixels per image *N,* in face recognition applications which result in singular scatter matrices. When all of the above is taken into account, the Rayleigh quotient can no longer be solved directly [Belheumer, 1997]. A possible solution to the matrix S_W singularity problem is to reduce the data dimension, in order to insure that only non–singular scatter matrices are produced prior to applying the LDA scheme, [Belhumeur, 1997]. The resulting scheme is called the “Fisherface approach.”

Thus, in the Fisherface approach, original images are initially projected using the PCA approach in order to reduce the dimension of the projected data to $P-C$. Then, the LDA approach is applied to further reduce the data, to dimension $C-1.$ Next, class centroids are computed using the training data, as it was done for the PCA approach. Finally, testing images are projected onto the smaller dimensional space using the projection matrices defined in PCA and LDA steps. Final class decision is obtained by selecting for each testing image the class, which is closest in norm to the training data class centroids.

D. PCA AND LDA COMPARISONS

Typical implementations of PCA and LDA approaches can be compared by applying them to two 2–dimensional data classes (each one with five samples), as shown in Fig. 2.1, where the two classes are represented by stars and circles, respectively. Figure 2.1 also shows their 1–dimensional projections, as defined by the PCA (solid line) and the LDA (dashed line) approaches. This figure illustrates that the PCA approach causes projected classes to be interlaced, while the LDA approach best preserves discrimination between the projected classes. The respective Matlab code that performs the subject PCA LDA projection comparison is provided in Appendix B.

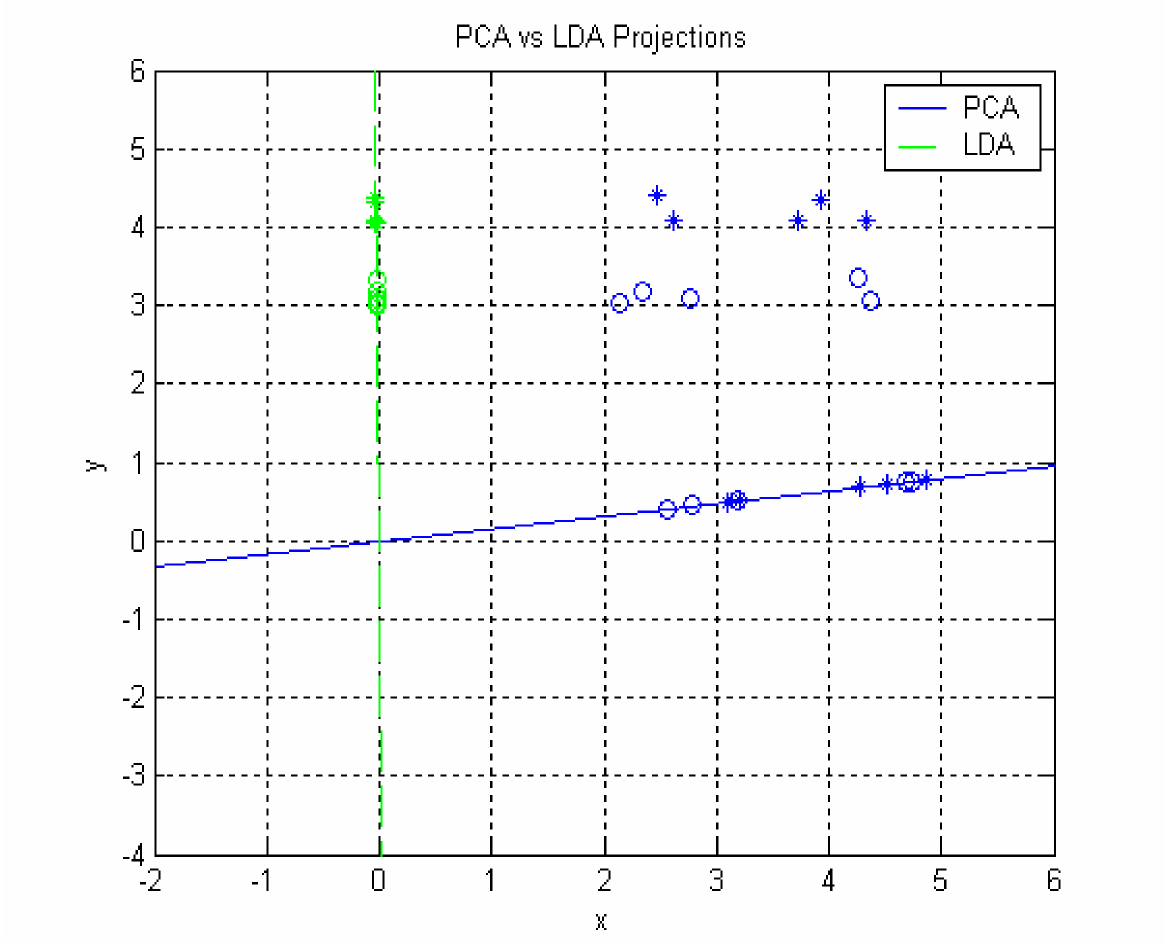


Figure 2.1. PCA and LDA projection directions for the two 2 classes, “o” and “*”.

E. CONCLUSIONS

This chapter discussed two basic linear classifiers, PCA and LDA. The chapter also illustrated the notion that LDA is better suited to classification problems than is PCA. Nevertheless, it should be recognized that either algorithm is successful for data for which linear projections are well matched to the data structure. Extension of the approaches to nonlinear classifiers is considered in the next chapter.

III. NONLINEAR CLASSIFICATION ALGORITHMS

This chapter extends the linear classifiers presented earlier to nonlinear schemes. First, we present the basic idea behind nonlinear classification approaches and introduce the concept of kernels, which is the key idea behind nonlinear schemes. Next, we describe the "kernel trick" which makes such procedures computationally feasible. Finally, we present the specific application of kernel–based implementations to the linear discriminant analysis called the Generalized Discriminant Analysis (GDA).

A. THEORETICAL BACKGROUND

Even though LDA is designed for discrimination applications, it is based on a linear projection approach. This may result in performance degradations when the data behavior is not well–suited to such design constraints, especially when classes are not linearly separable. A significant amount of research has been conducted recently in the area of kernel–based schemes, much of which has led to the implementation of nonlinear projections. The kernel theory was first used in pattern recognition applications by Aizerman, [Aizerman, 1964; Baudat, 2003]. The main kernel–based learning methods are the "Support Vector Machines" (SVM), the "Kernel Principal Component Analysis" (KPCA), and the "Kernel Fischer Discriminant Analysis" (KFD), also known as "Generalized Discriminant Analysis" (GDA), [Muller, 2001]. All of these methods have been widely applied in classification, regression, pattern and object recognition, optical character recognition (OCR), text categorization, time series prediction, gene expression profile analysis, protein and DNA analysis and other applications [Muller, 2001; Roobaert, 1999; Joachims, 1998; Brown, 2000].

Specifically, the following results have been recently reported when applying kernel–based schemes in:

- Face recognition: Various nonlinear discrimination algorithms have been developed [Baudat, 2000, 2003; Socolinsky, 2003; Liu, 2004] by taking into account nonlinear features of the training data, thereby extending classification to data which cannot be solved using linear algorithms. For example, the GDA method has been applied successfully with various kernel functions [Liu, April 2004; Liu, May 2004; Hearst, 1998].

- 3–Dimensional object recognition: the SVM algorithm has been successively applied to recognize 3–dimensional objects when a limited number of training views are available [Roobaert, 1999].
- Text categorization: The objective of this application is to assign documents into a set of defined document types, such as “news”, “historical”, “business”, “scientific”, “medical”, etc. [Joachims, 1998]. SVM approaches have been applied successfully to organize network data, document databases, or even to learn a user’s web browsing preferences [Joachims, 1998].
- Optical Character Recognition (OCR): The goal of this application is to identify handwritten samples. The first real world experiments were applied to OCR data using a basic SVM implementation and were shown to perform well (with error rates of only around 0.7%) [Muller, 2001].
- Gene expression profile and DNA and protein analysis: SVM approaches have also been applied to classify genetic functionality. Such applications require the extraction of data from the DNA strand relating to how a gene expresses itself [Brown, 2000; Muller, 2001].

1. Introduction to Kernel Theory

The basic idea behind Kernel methods is to map the data, $x \in X$, into a large dimensional feature space, *F,* via a function, φ, [Baudat, 2003], where the mapped data have characteristics which can be manipulated with simple computational methods. This mapping may seem counter–intuitive due to the issues raised by the *“curse of dimensionality”* [Muller, 2001]. Mapping essentially shows that the number of samples required to set–up an estimation problem increases exponentially, with the dimension of the sample space. Nevertheless, statistical learning theory also states that learning in the high dimensional feature space, *F,* can be simple if low–complexity criteria and specific mapping functions are selected. Therefore, the key issue when dealing with kernel–based schemes lies in the selection of the specific mapping φ. Rather than being carried out explicitly, this calculation is reformulated in terms of dot products applied to the data. Thus, data in the transformed space are expressed in terms of a kernel matrix containing comparisons of data pairs only, which makes the implementation of this process less expensive.

2. Introduction to the “Kernel Trick”

Assume that there are *m* training data [Scholkopf, 2000]:

$$X = \{x_1, x_2, ..., x_m\}. \qquad (3.1)$$

Also, assume that the training patterns, x_i, belong to one of the two classes with labels, y_i, defined as:

$$y_i \in \{-1, +1\}, \ \forall i = 1, 2, ..., m. \tag{3.2}$$

Therefore, the training data with label assignment can be expressed as:

$$(x_1, y_1), (x_2, y_2), ..., (x_m, y_m) \in X \times \{-1, +1\}. \tag{3.3}$$

Recall that the goal of learning applications is to assign a label (i.e., class) to an unlabeled data pattern. Therefore, its goal can be viewed as predicting the value of labels, y, for new unlabeled patterns, $x \in X$, in such a way that the ordered pair, (x, y), is as similar as possible to ordered pairs defined in the training data. Such a procedure requires the definition of a similarity measure in the transformed space. As a result, a similarity measure between two data points, x_1 and x_2, is implemented as:

$$\begin{aligned} &k : X \times X \rightarrow R, \\ &(x_1, x_2) \rightarrow \varphi(x_1) \cdot \varphi(x_2) = k(x_1, x_2), \end{aligned} \tag{3.4}$$

where φ represents the mapping function and can be rewritten in terms of a kernel function, $k,$ for specific choices of mapping, then implemented in terms of a dot product only. This process is known as the “kernel trick” [Scholkopf, 2000].

In order to be able to use dot products as similarity measurements, input patterns (i.e., vectors) should lie in a dot product domain. This means that the arguments of the kernel functions should be dot products of the original input element vectors. Thus, the new mapped space, F, includes the kernel function values, which have dot products as inputs. Generally, the space F is different from the original N–dimensional space.

Next, we present a basic example to illustrate the “kernel trick” [Muller, 2001].

Assume that there are two 2–dimensional classes, “x” and “o”, as shown on the left–hand plot of Fig. 3.1.

Figure 3.1 shows that the original two classes are nonlinearly separable. Consequently, a nonlinear mapping φ is required to potentially transform the input data into a higher dimensional space in order to enable the transformed data classes to be linearly separable. In this specific case, the mapping φ will be:

$$\varphi : R^2 \rightarrow R^3. \tag{3.5}$$

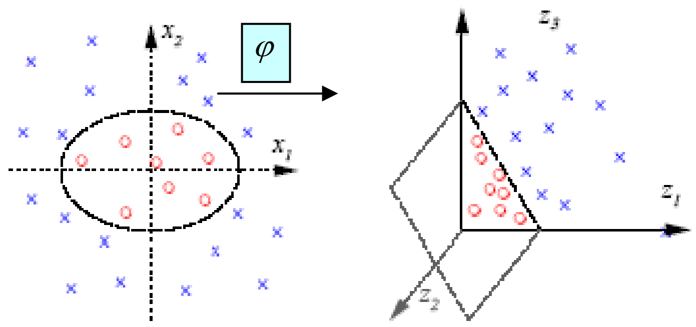


Figure 3.1. Illustration of a kernel–based, nonlinear transformation for a two–class problem. Left plot: original, 2–dimensional input space; Right plot: transformed 3–dimensional feature space. (From [Muller, 2001].).

A potentially nonlinear mapping that transforms the 2–dimensional element vectors into 3–dimensional element vectors can be expressed as:

$$(x_1, x_2) \rightarrow (z_1, z_2, z_3), \tag{3.6}$$

where the transformed space features are defined as the second order monomial terms:

$$(z_1, z_2, z_3) \equiv (x_1^2, \sqrt{2}x_1 x_2, x_2^2). \tag{3.7}$$

Note that the $\sqrt{2}$ present in the above term, $x_1 x_2$, is added for normalization purposes so that the nonlinear mapping can lead to a valid kernel expression, as shown below.

Next, let us define the following polynomial kernel function as:

$$k(x, y) \equiv (x \cdot y)^d, \tag{3.8}$$

where x and y are 2–dimensional element vectors (i.e., $x = (x_1, x_2)$, and $y = (y_1, y_2)$).

The second order kernel function is defined for d=2 as:

$$k(x, y) = (x \cdot y)^2. \tag{3.9}$$

For 2–dimensional vectors x and y, the kernel expression $k(x, y)$ may be rewritten as:

$$\begin{aligned} k(x,y) &= (x \cdot y)^2 \\ &= ((x_1, x_2)(y_1, y_2)^T)^2 \\ &= (x_1 y_1 + x_2 y_2)^2 \\ &= (x_1^2 y_1^2 + 2x_1 y_1 x_2 y_2 + x_2^2 y_2^2) \\ &= (x_1^2, \sqrt{2} x_1 x_2, x_2^2)(y_1^2, \sqrt{2} y_1 y_2, y_2^2)^T \\ &= (\varphi(x) \cdot \varphi(y)). \end{aligned} \tag{3.10}$$

The above equations show that the nonlinear mapping φ defined in Eq. (3.7) is performed implicitly via the kernel expression, $k(x,y)$, by the use of dot products in the input domain.

It can be shown that such nonlinear mapping potentially offers the following advantages [Scholkopf, 2000]:

- Similarity measures (dot products) in the transformed space, *F,* can be defined as:
$$k(x_1, x_2) = (\varphi(x_1) \cdot \varphi(x_2)). \tag{3.11}$$
- Linear algebra and analytic geometry concepts may be applied in the transformed space, facilitating data manipulation.
- Flexibility in selecting the mapping function, φ, allowing for the development of specialized algorithms.

3. Support Vector Machines (SVM)

SVM is a type of pattern classification and regression method that uses the concept of kernel functions, in conjunction with dot products [Scholkopf, 2000].

The basic idea behind SVM is to identify a hyperplane in a transformed high dimensional space that linearly separates the mapped classes in that space. This identification procedure is accomplished via special kernel functions that are applied to the input element vectors.

In order to clarify the concept at the basis of the SVM approach, consider the simple case of the classification of two nonlinearly separable classes. For this particular case SVM is used to select the optimum mapping φ of the input space into a space in which a class separating hyperplane can be defined. This type of hyperplane is illustrated in Fig. 3.2 and lies approximately in the middle of the shortest distance of the two classes' convex hulls. The support vectors define the exact location of the hyperplane. In

Fig. 3.2, one can observe the support vectors in this newly transformed space, *F*; these are a product of the elements of the two classes, the separating hyperplane, and the closest element vectors of each class to the hyperplane.

The key point behind the SVM approach is that the kernel function of the element vectors' dot products may be used directly, i.e., without the need to carry out the mapping of the inputs to the new high dimensional space explicitly.

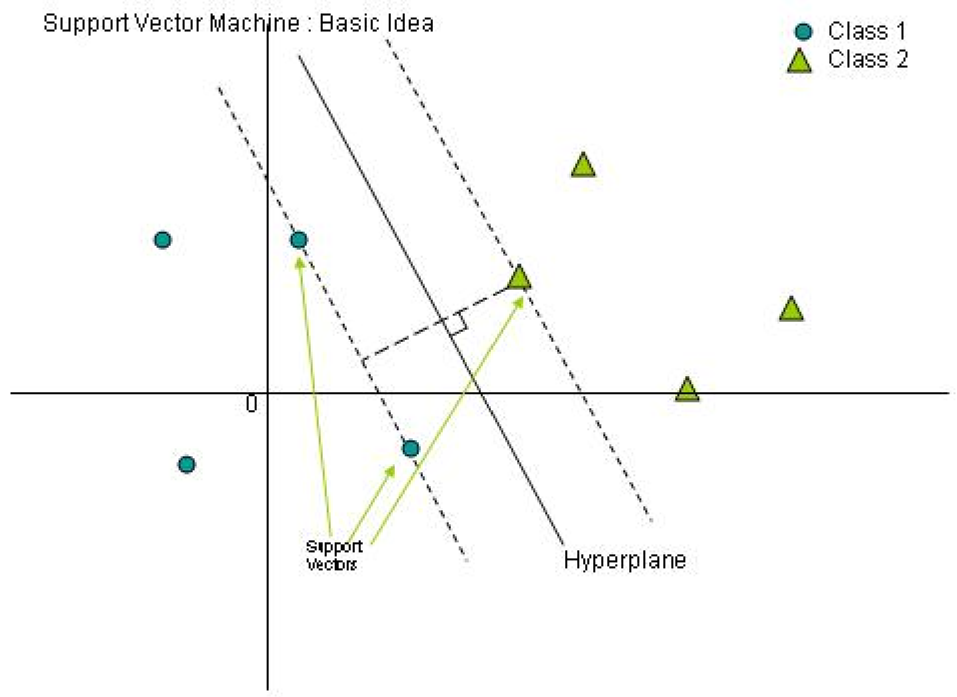


Figure 3.2. Basic idea of Support Vector Machines: Identification of the two–class optimum separating hyperplane in the new, high–dimensional transformed space. (After [Scholkopf, 2000].).

The hyperplane in the high–dimensional transformed space is actually a nonlinear decision boundary in the input space. The aforementioned decision boundary, when viewed in the original input space, is shown as the nonlinear curve in Fig. 3.3, with the support vectors (marked as double circles) on the two curved lines above and below the nonlinear decision boundary. In the subject figure, one class consists of circle elements and the other of discus–shaped elements.

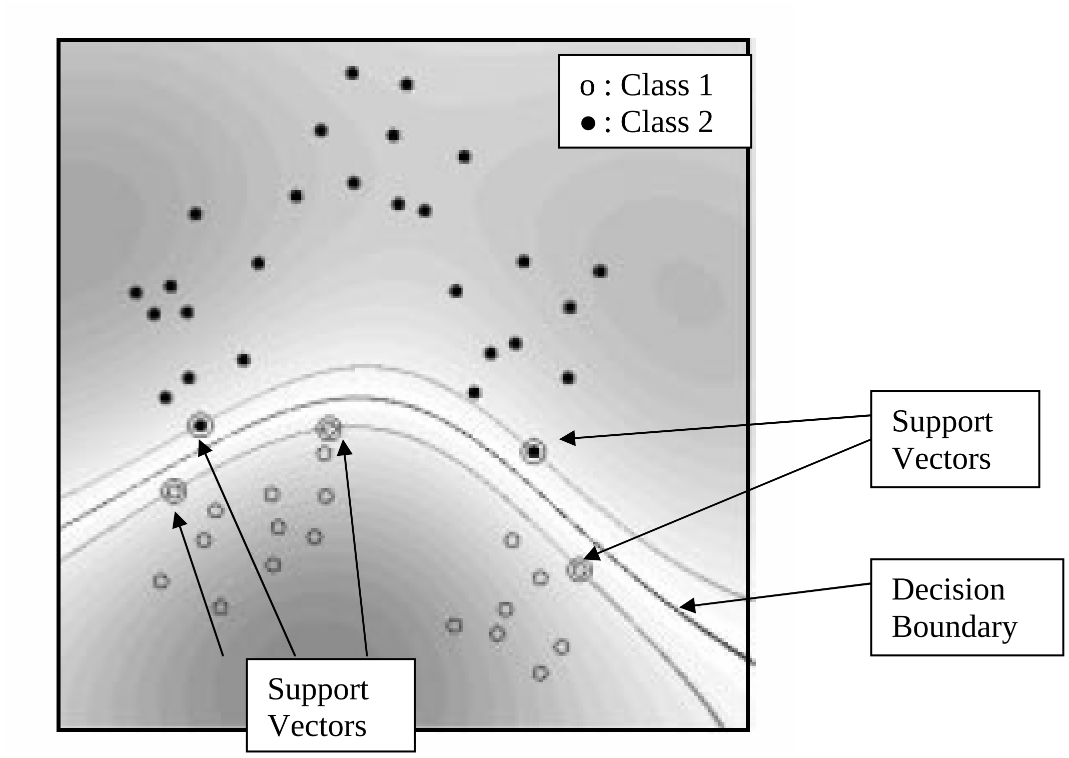


Figure 3.3. Original 2–dimensional two–class problem (classes, “o” and “●”), showing the support vectors and the nonlinear optimal boundary. (From [Scholkopf, 2000].).

The main advantages provided by the SVM method are [Scholkopf, 2000]:

- The SVM method can identify and eliminate data outliers.
- SVM schemes offer flexibility via the specific selection of the kernel function. They also allow for flexibility to choose the most suitable similarity function, and can handle large feature spaces.

The main drawbacks of the SVM method are [Scholkopf, 2000]:

- The size of the quadratic programming problem that results from the SVM algorithm can be quite large [Scholkopf, 2000], resulting in high computational load and the potential of numeric problems.
- The potential of high noise levels in the classes’ element vectors increases the SVM implementation complexity as additional conditions have to be met to minimize noise impacts on the classification accuracy, resulting in an increased computational load.

We note that optimization algorithms addressing the above issues have been recently proposed in the literature [Scholkopf, 2000].

4. Kernel PCA

The Kernel PCA executes a nonlinear mapping of the input space to a new high dimensional space, using the “kernel trick”. Next, the classic linear PCA is applied in the transformed space, resulting in a linear eigenvalue–eigenvector decomposition problem of a matrix, whose data elements are the values of the kernel function [Hearst, 1998]. Numerous results have been reported with this scheme which shows superior performance over the basic PCA implementation, [Scholkopf, 1998; Kwang, 2002; Lu, 2003].

B. GENERALIZED DISCRIMINANT ANALYSIS (GDA)

1. Introduction

Kernel concepts may be applied to classical algorithms, such as the PCA or LDA. Applying kernel theory to LDA means that the linear LDA algorithm is applied to a nonlinear transformed data, resulting in a nonlinear algorithm without the drawbacks present in a direct nonlinear transformation of the algorithm. Therefore, the GDA solves an ordinary eigenvalue–eigenvector decomposition, albeit of the transformed data instead.

Recall that the GDA is a supervised scheme. Therefore, issues dealing with algorithm generalization are still an issue, which in this case depends on the geometric characteristics of the training data, and not on its dimensionality. As a result, a judicious selection of the mapping function φ is a significant factor in insuring the reliability of the results and a lower computational load.

2. Algorithm Description

This section describes the GDA derivation and closely follows the presentation given by [Baudat, 2000]. Assume that x is a column vector representing a reshaped image of the data matrix X containing the M images included in the training dataset. Further, assume that X_i represents the matrix containing the n_i images of class i, where $i = 1, ..., N,$ and N is the total number of classes under study. Then, the total number of images contained in the training dataset is given by:

$$M = \sum_{i=1}^{N} n_i, \tag{3.12}$$

and the data covariance matrix C is defined as:

$$C = \frac{1}{M}\sum_{i=1}^{M} x_i x_i^T, \tag{3.13}$$

assuming that the training data is centered (i.e., has mean equal to 0).

Next, the input space *X* must be mapped into a Hilbert space, *F*, by using a nonlinear mapping function φ:

$$\begin{aligned} &\varphi : X \to F. \\ &x \to \varphi(x). \end{aligned} \tag{3.14}$$

Then, the covariance matrix in the resultant feature space, *F,* is given by:

$$V = \frac{1}{M}\sum_{j=1}^{M} \varphi(x_j)\varphi^T(x_j), \tag{3.15}$$

assuming that the data is centered in the transformed space.

Next, the covariance matrix *B* that is derived from the class centers expressed in the transformed space, *F*, is defined as:

$$B = \frac{1}{M}\sum_{i=1}^{N} n_i \bar{\varphi}_i \bar{\varphi}_i^T, \tag{3.16}$$

where $\bar{\varphi}_i$ represents the mean value of class *i,* and is defined as:

$$\bar{\varphi}_i = \frac{1}{n_i}\sum_{k=1}^{n_i} \varphi(x_{ik}), \tag{3.17}$$

where x_{ik} is element *k* of the class *i*.

The data covariance matrix may be expressed in the transformed space, *F*, as follows:

$$V = \frac{1}{M}\sum_{i=1}^{N}\sum_{k=1}^{n_i} \varphi(x_{ik})\varphi^T(x_{ik}). \tag{3.18}$$

Note that the data covariance matrix *V* defined above in Eq. (3.18) is expressed in terms of dot products only. The main advantage behind kernel–based schemes is that nonlinear algorithms may be implemented in terms of expressions involving only the dot product on the data considered, i.e., without having to explicitly define and use the actual nonlinear transform expression $\varphi(x).$ The dot product expression obtained for data x_i and x_j is defined by the following kernel function:

$$k_{ij} = k(x_i, x_j) = \varphi^T(x_i)\varphi(x_j). \tag{3.19}$$

When dealing with different data classes, the kernel expression becomes:

$$(k_{ij})_{pq} = \varphi^T(x_{pi})\varphi(x_{qj}), \tag{3.20}$$

where p and q are data classes.

Next, define the $\left(M \times M\right)$ dimensional matrix K as:

$$K = (K_{pq})_{\substack{p=1..N \\ q=1..N}}, \tag{3.21}$$

where K_{pq} are $\left(n_p \times n_q\right)$ dimensional sub-matrices defined as:

$$K_{pq} = (k_{ij}), \tag{3.22}$$

with $i = 1,...,n_p,\ j = 1,...,n_q,\ p = 1,...,N,$ and $q = 1,...,N.$

Next, define the M–dimensional block diagonal matrix W, where each block W_i is of dimension $\left(n_i \times n_i\right)$, and given by:

$$W_i = \frac{1}{n_i}\begin{pmatrix} 1 & \dots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{pmatrix}, \tag{3.23}$$

where n_i represents the number of the data samples of class i.

Recall that LDA maps the input space into one where the principal components maximize the ratio of the between–class–scatter matrix (also known as the inter–class inertia) and the within–class–scatter matrix (also known as the intra–class inertia). The use of kernel functions expands the LDA application to include the nonlinear transformed space. The principal components in the resultant space are nonlinearly related to the input variables and the kernel function, *k,* contributes to the creation of the function that nonlinearly separates the classes. Following the findings used in the LDA derivation, the

GDA maximizes the Rayleigh quotient $\frac{v^T Bv}{v^T Vv}$ of the between–class scatter matrix over the within–class scatter matrix, where these V and B matrices (defined previously in Eqs. (3.15) and (3.16)) are defined in the transformed space. Therefore, the LDA maximization problem is equivalent to solving the following eigenvalue–eigenvector decomposition problem:

$$\lambda Vv = Bv, \tag{3.24}$$

where λ and v are the resulting eigenvalues and eigenvectors, respectively.

The eigenvectors v can be expressed as a linear combination of elements expressed in the transformed space as:

$$v = \sum_{p=1}^{N} \sum_{q=1}^{n_p} a_{pq} \varphi(x_{pq}), \tag{3.25}$$

where $a_{pq}, {}_{p=1,\ldots N; q=1,\ldots n_p}$, are defined as the coefficients of the linear combination. Note that the coefficient vector $a = (a_{pq})_{p=1,\ldots N; q=1,\ldots n_p}$ can be expressed in a more compact form as $a = (a_p)_{p=1,\ldots N}$, where $a_p = (a_{pq})_{q=1,\ldots n_p}$ is defined as the coefficient of the vector v in class p.

Baudat showed that the Rayleigh quotient expressed in the transformed space, may be rewritten as [Baudat, 2000]:

$$\lambda = \frac{v^T Bv}{v^T Vv} = \frac{a^T KWKa}{a^T KKa}. \tag{3.26}$$

Therefore, maximizing the above Rayleigh quotient can be executed by solving the following generalized eigenproblem:

$$KWK\alpha = \lambda KK\alpha. \tag{3.27}$$

Note that the matrix K is symmetric and positive definite when the kernel type selected satisfies Mercer theorem [Scholkpof, 1998]. Therefore, K may be expressed as:

$$K = P\Gamma P^T, \tag{3.28}$$

where P and Γ are the eigenvector and nonsingular diagonal eigenvalue matrices, respectively. In addition, P has full rank when K is positive definite and the eigenvector matrix P is unitary, i.e., $PP^T = P^T P = I$.

At this point, replacing the matrix K by its eigendecomposition in Eq. (3.27) leads to:

$$(P\Gamma P^T)W(P\Gamma P^T)\alpha = \lambda(P\Gamma P^T)(P\Gamma P^T)\alpha. \tag{3.29}$$

Recall that the eigenvector matrix P is unitary. Thus, Eq. (3.29) becomes:

$$(P\Gamma P^T)W(P\Gamma P^T)\alpha = \lambda(P\Gamma)(\Gamma P^T)\alpha. \tag{3.30}$$

Using the fact that the eigenvalue matrix P and Γ have full rank (i.e., are invertible matrices) when the kernel type is selected to satisfy Mercer theorem, Eq. (3.30) may be simplified to:

$$(P^T)W(P\Gamma P^T)\alpha = \lambda(\Gamma P^T)\alpha. \tag{3.31}$$

In order to simplify the following calculations, a new variable β is defined as:

$$\beta = \Gamma P^T \alpha. \tag{3.32}$$

Then, replacing Eq. (3.32) into Eq. (3.31) leads to:

$$(P^T)WP\beta = \lambda\beta. \tag{3.33}$$

Therefore, maximizing the Rayleigh quotient shown in Eq. (3.26) above may be obtained by first, solving the eigenproblem derived in Eq. (3.33) in order to obtain β; and second, by computing the coefficient vector α from β, as $a = P\Gamma^{-1}\beta$. Note that the actual a coefficients are not unique, as only the direction of the eigenvectors obtained by solving an eigenproblem is unique. The a coefficients can be computed by setting the vectors v to norm 1. Thus, taking into consideration Eq. (3.26), the following expression is obtained:

$$v^T v = 1 \tag{3.34}$$

$$\Rightarrow v^T v = \sum_{p=1}^{N}\sum_{q=1}^{n_p}\sum_{l=1}^{N}\sum_{h=1}^{n_l}\alpha_{pq}\alpha_{lh}\varphi^T(x_{pq})\varphi(x_{lh}) = 1. \tag{3.35}$$

Therefore,

$$v^T v = \sum_{p=1}^{N} \sum_{l=1}^{N} \alpha_P{}^T K_{pl} \alpha_l = 1, \tag{3.36}$$

which yields:

$$v^T v = \alpha^T K \alpha = 1, \tag{3.37}$$

where the dimension of K_{pl} is equal to $\left(n_p \times n_l\right)$, as the dimensions of α_p and α_l are $\left(n_p \times 1\right)$ and $\left(n_l \times 1\right)$, respectively. Thus, the coefficients α need to be divided by $\sqrt{a^T K \alpha}$, in order to normalize the vectors v to norm 1.

Finally, testing data z may be projected into the nonlinear feature space by projecting them as:

$$v^T \varphi(z) = \sum_{p=1}^{N} \sum_{q=1}^{n_p} \alpha_{pq} k(x_{pq}, z). \tag{3.38}$$

3. Kernel Selection

Recall that the basic idea of the main kernel–based schemes is to project the data into a feature space via a nonlinear projection φ and to apply linear schemes in that new space. The kernel function required for this implementation is expressed in terms of dot products as:

$$k(x, y) = \varphi(x) \cdot \varphi(y). \tag{3.39}$$

Currently, several types of kernels are being used in kernel–based implementations. The three most significant are:

- The polynomial kernel of the form: $k(x, y) = (x \cdot y)^d$, where $d \geq 1$ has been used extensively. Note that the GDA implementation reverts back to the classical LDA implementation when $d = 1$. In addition, [Liu, 2004] recently investigated polynomial kernel functions with fractional powers (i.e., when $0 < d < 1$), which can be applied to face recognition applications, even though such studies may not necessarily satisfy the Mercer theorem, and may not be considered as valid kernel functions. Nevertheless, Liu showed in his simulations, (when using 30 features or more on visible imaging data), that fractional degree polynomial kernels lead to higher recognition rates than do those obtained with integer–degree polynomial kernels [Liu, 2004].

- The Gaussian kernel function defined as:

$$k(x,y) = \mathrm{e}^{(-\frac{\|x-y\|^2}{\sigma})}, \tag{3.40}$$

where the parameter, σ, has to be selected arbitrarily.

- The sigmoid kernel defined as:

$$k(x,y) = \tanh(k(xy) + \vartheta), \tag{3.41}$$

where $0 < k$ and $\vartheta < 0.$ The sigmoid kernel function has been successfully used in practice, (especially in SVM applications), even though it does not actually define a positive semi–definite Gram matrix and thus, is not a valid kernel function [Liu, 2004].

4. Application to Iris Data

The GDA scheme is first applied to a classical dataset commonly used in classification applications for benchmarking purposes, namely the “Iris” dataset [Baudat, 2000]. The specific Iris data consists of four real parameters of the flower Iris divided into three classes, each with 50 elements of dimension four. The Iris data consists of: the sepal length, the sepal width, the petal length, and the petal width, with all measurements expressed in millimeters. The three classes represent three different types of Iris flowers: Iris setosa, Iris versicolor, and Iris virginica. The Iris dataset is interesting, as one class is linearly separable from the other two, whereas the other two are not linearly separable from each other. As a result, linear classifiers are unable to separate the three classes very well. Next, we will show that the GDA is able to separate all three classes well.

Using the regular LDA leads to two overlapping projected classes, with only one being separable from the other two. Such overlapping is shown in Figs. 3.4 and 3.5, which plot the Iris data 1–and 2–dimensional projections in the LDA–derived directions [Baudat, 2000].

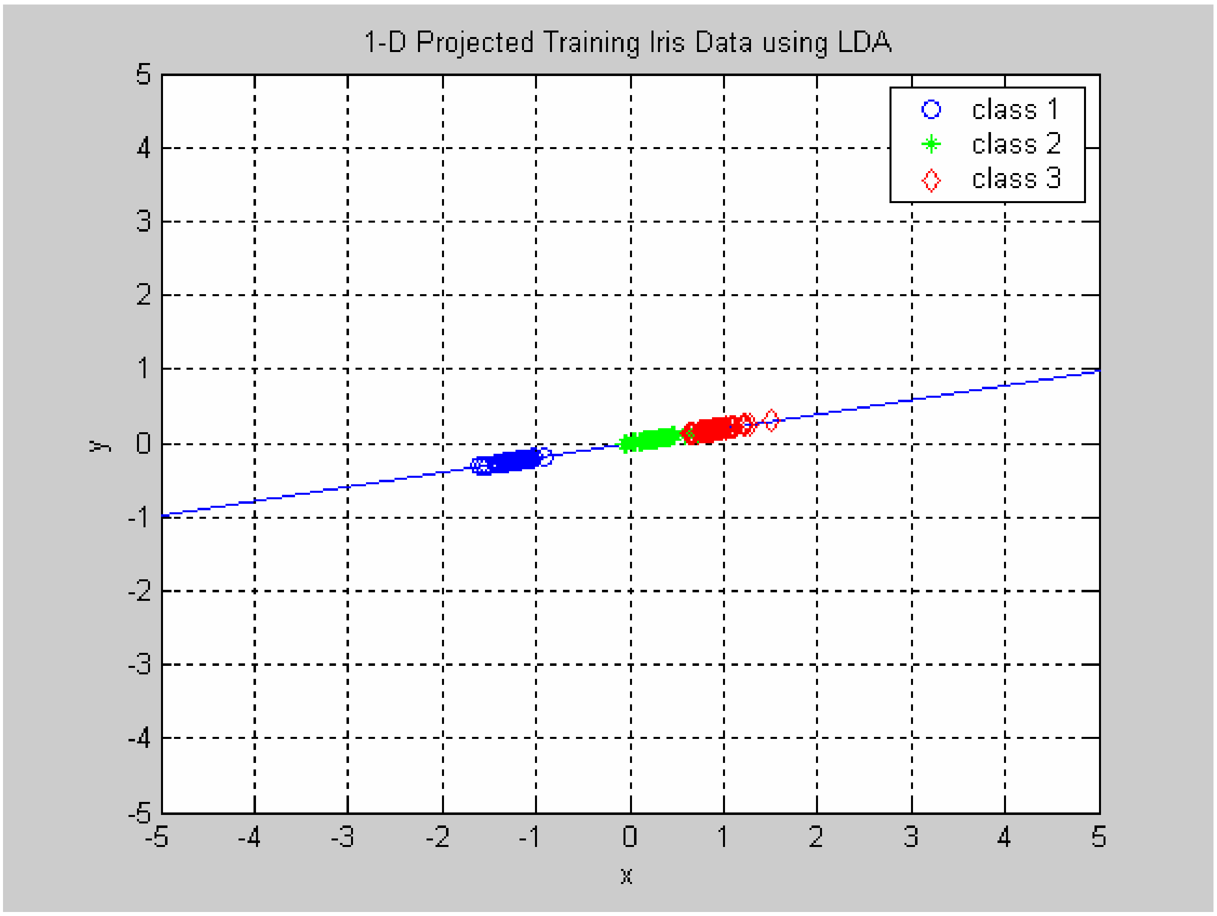


Figure 3.4. 1–Dimensional LDA projection of Iris data. Classes 2 and 3 are not separable.

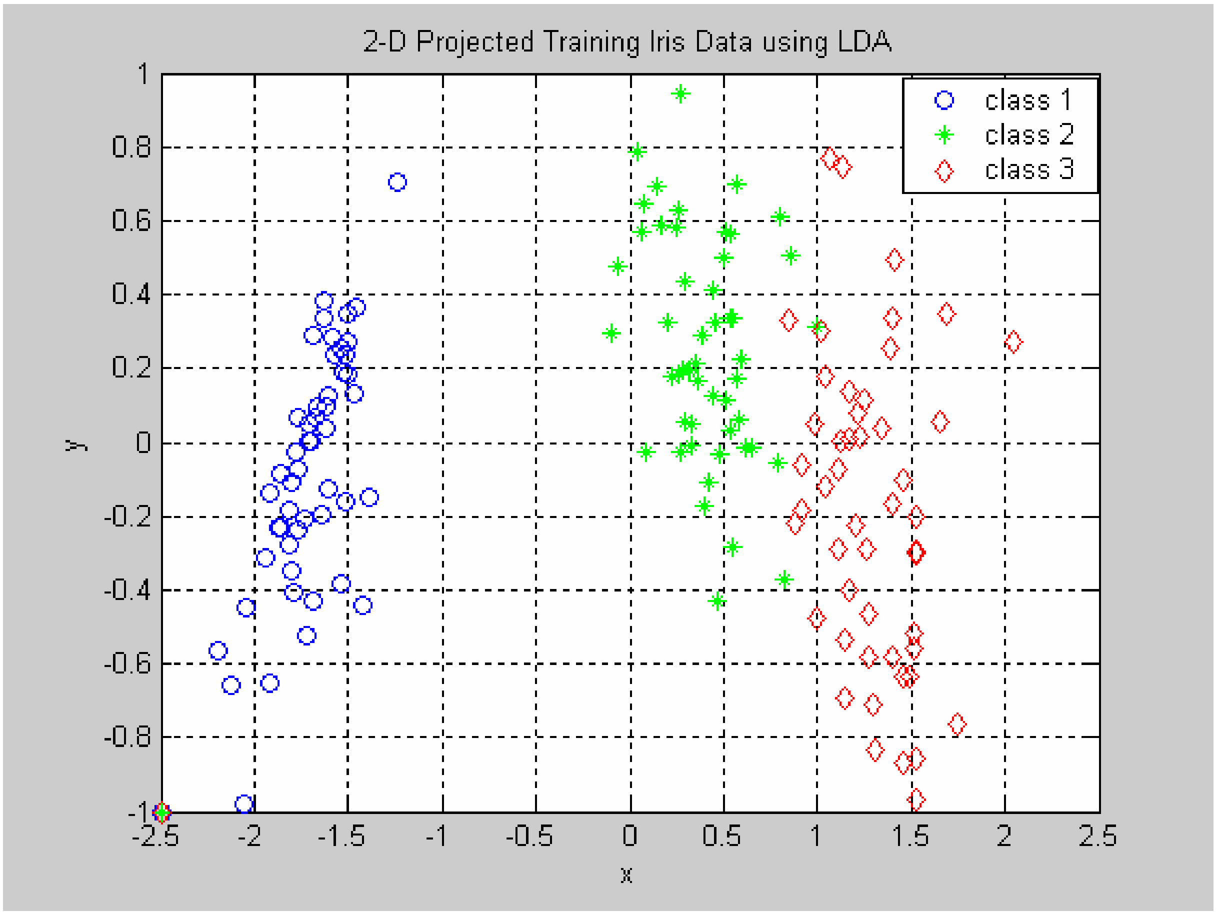


Figure 3.5. 2–Dimensional LDA projection of Iris data. Classes 2 and 3 are not separable.

Next, Figs. 3.6, 3.7, and 3.8 plot the 2–dimensional class projections obtained using the GDA with a Gaussian kernel, [Baudat, 2000], where different values of the arbitrary spread parameter σ were selected $(\sigma = 0.3, 0.7, 7)$. The plots were produced using the software available from [Baudat, October 2000]. Results demonstrate that all three projected classes are widely separated from the other two. It should be noted that the projected data spread increases with increasing values of the spread parameter, which may result in nonlinearly separable projected data, as shown in Fig. 3.8.

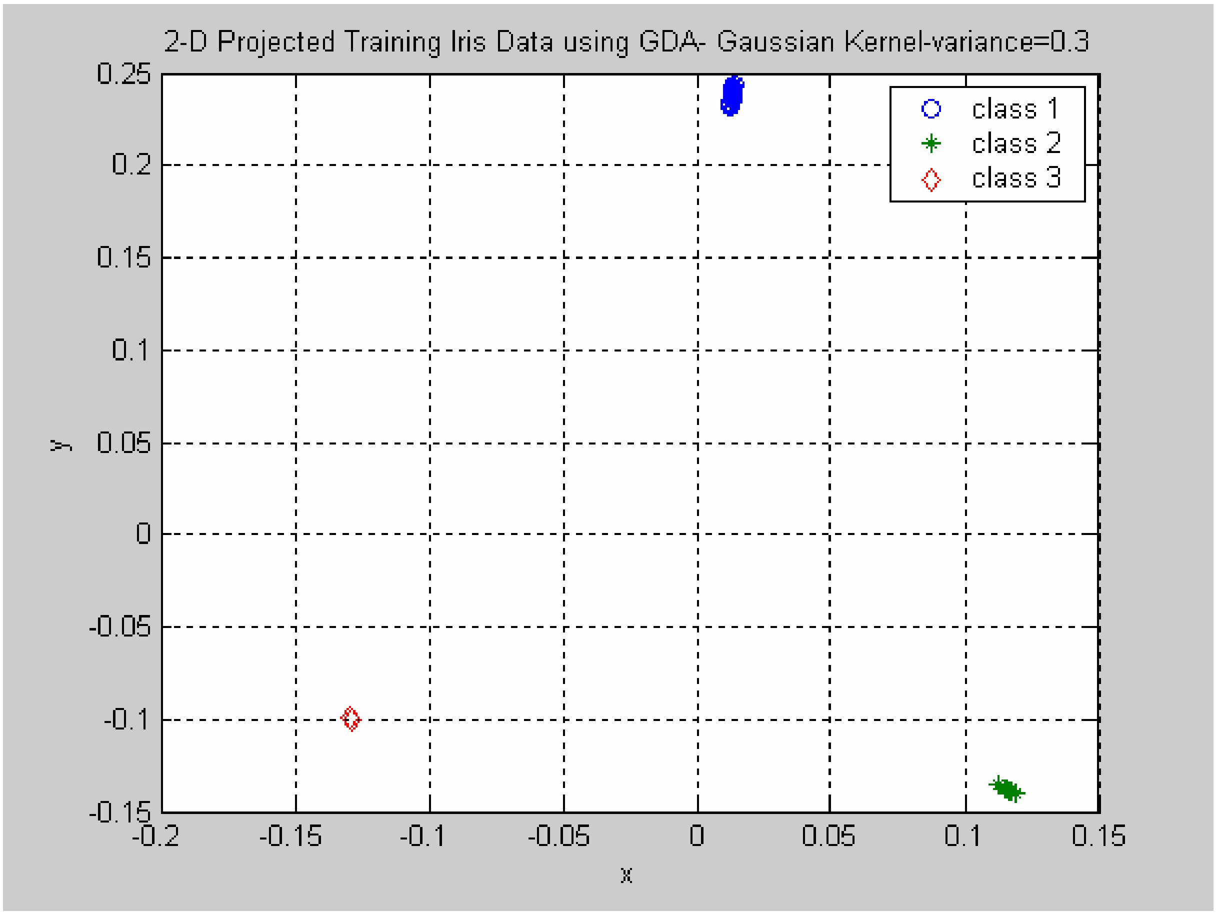


Figure 3.6. GDA projection of Iris data. All classes are separable, but the element vectors within the classes are not discernible (σ = 0.3➔ Small variation.).

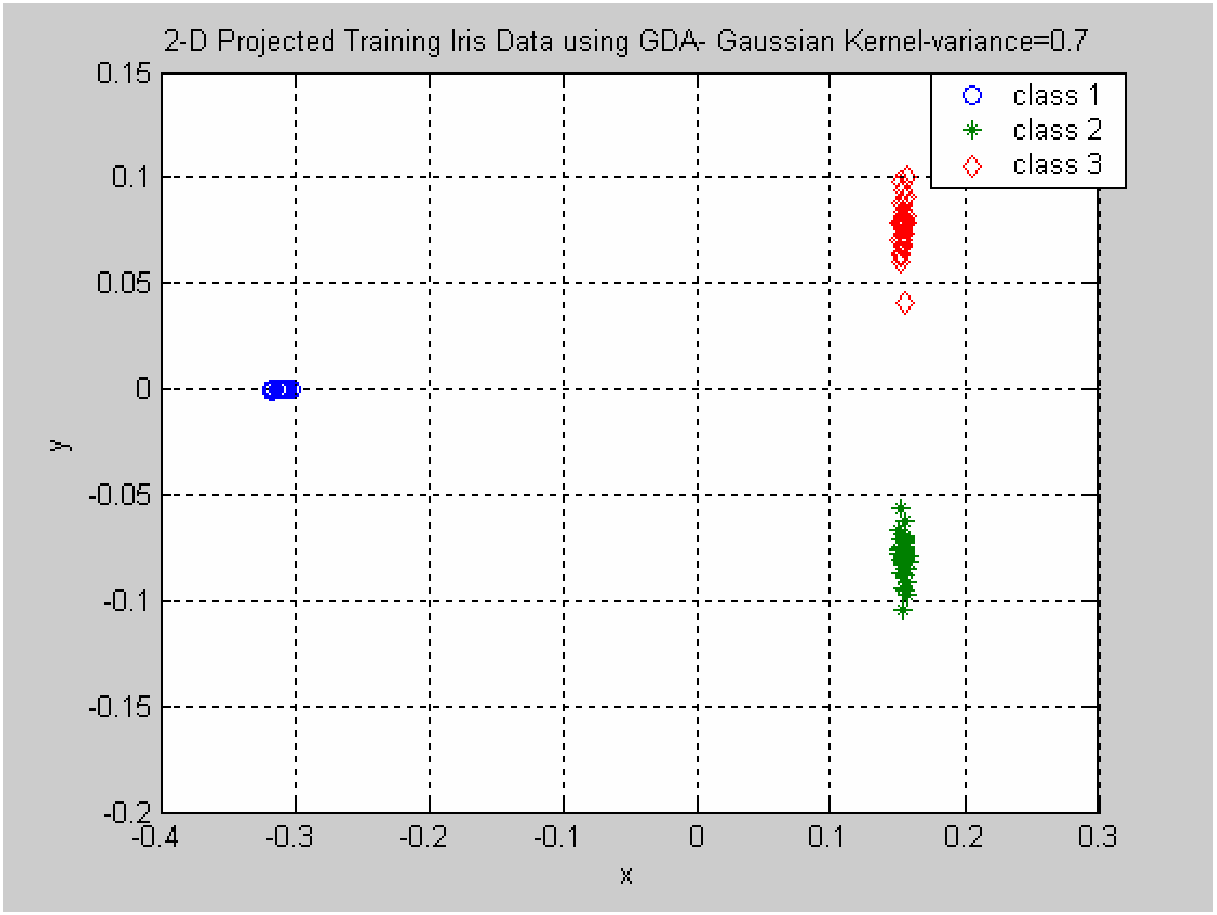


Figure 3.7. GDA projection of Iris data. All classes are separable (σ = 0.7➔ Average variation.).

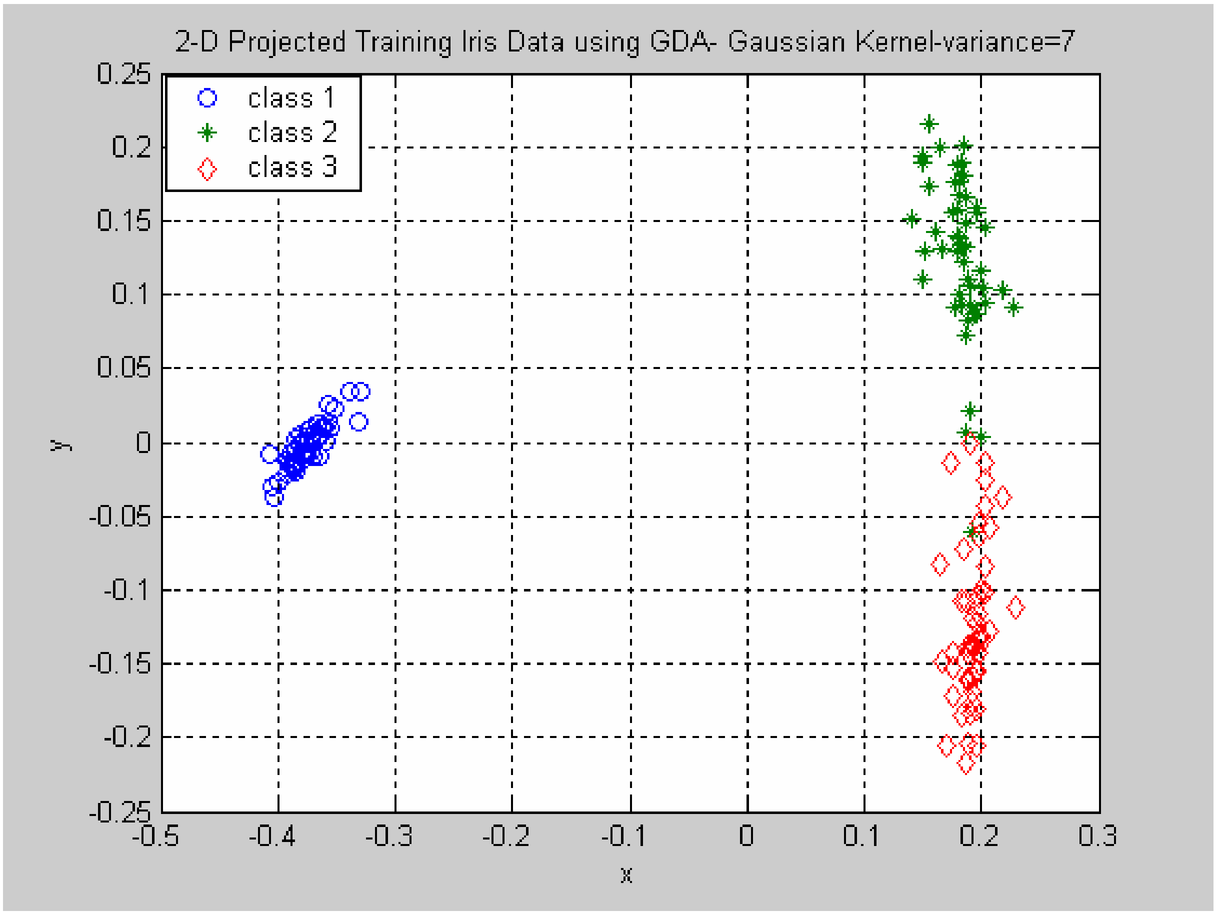


Figure 3.8. GDA projection of Iris data. The two classes are not entirely separable ($\sigma = 7$ ➔Large variation.).

C. CONCLUSIONS

This chapter presented the basic concepts behind kernel–based algorithms and their applications to classification problems. It also described the GDA implementation, which constitutes the main research tool of this study.

The following chapter discusses the application of the GDA to the IR face database.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. GDA RESULTS

This chapter considers the application of the GDA approach to the IR face database. First, it investigates how the specific type of distance metrics selected in the algorithm may affect classification performances. Second, the concepts of rank score and confidence interval are reviewed and it is shown how they can be applied toward analyzing and comparing classifier performances.

Next, we apply the above distance measures to the GDA implementation considered in this study. Specifically, we investigate the link between distance metrics and resulting classification performances obtained using the IR database. We also investigate the links between kernel function type, and number of eigenvectors selected in the GDA projection operation and the resulting classifier performances.

A. THEORETICAL BACKGROUND

1. Introduction

First, this section considers various types of distances commonly used in pattern classification applications. Second, it reviews the concepts of rank score, confidence interval, and confidence score. These parameters are used to evaluate how specific distances, kernel types, and number of eigenvectors extracted from the kernel matrix K influence the classifier performances.

2. Distances Considered

According to [Duda, 2001], pattern classification is a one–to–one assignment data to a class belonging to a set of pre–defined element classes. This assignment is based on the concept of *nearest neighbor classifier*, which is in turn based on the minimization of a metric or distance function between testing data and each class centroid.

Distances are measures which evaluate the difference between two vectors v and w. Many types of distances have been commonly used in engineering applications, such as the Euclidian Norm, Norm 1, etc. Some distances are known to perform better than others depending on the data and the specific pattern classification algorithm considered.

The following section investigates five different distance measures commonly used in pattern classification applications which are considered in this study. We will select the distance leading to best overall classification performances in the rest of the study.

The following five distances are considered, where it is assumed that the vectors *v* and *w* are real–valued and of dimension $(N \times 1)$ [Socolinsky, 2003]:

1. Euclidean Norm 2:

$$L^2(v,w) = \sqrt{\sum_{i=1}^{N} (v_i - w_i)^2}\ , \tag{4.1}$$

2. Mahalanobis:

$$M(v,w) = (v-w)^T \Sigma^{-1} (v-w), \tag{4.2}$$

where Σ is the data covariance matrix defined as [Bishop, 1995]:

$$\Sigma = \mathrm{E}\left[\left(v-w\right)\left(v-w\right)^T \right], \tag{4.3}$$

3. Mahalanobis Norm 1:

$$M^1(v,w) = \sum_{i=1}^{N} \sigma_i^{-\frac{1}{2}} \,|\, v_i - w_i \,|, \tag{4.4}$$

where σ_i is an estimate of the data variance along the i^{th} coordinate direction, and v_i, w_i are the i^{th} vector coordinates of *v* and *w* respectively,

4. Mahalanobis Norm 2:

$$M^2(v,w) = \sqrt{\sum_{i=1}^{N} \sigma_i^{-1} (v_i - w_i)^2}\ , \tag{4.5}$$

5. Mahalanobis Angular:

$$M^{ang}(v,w) = \arccos \frac{\sum_{i=1}^{N} \sigma_i^{-1} v_i w_i}{L^2(0,v) L^2(0,w)}. \tag{4.6}$$

3. Rank Score

The notion of rank score (also known as cumulative matching score) is based on the probability of guessing the data class assignment correctly [Philips, 1996]. As a result, a rank score equal to one (referred to as rank–1 score) is defined as the probability of making a correct class assignment. Accordingly, a rank score equal to 2 (i.e., rank–2 score) corresponds to the probability of making a correct class assignment in the top (i.e., most likely) two decisions, and rank–N score refers to as the probability of making a correct assignment within the top N candidates. Extending this concept, a rank–C score defined on a C–class data is equal to 1, as all potential classes are included. In addition, a classifier with a 100% classification accuracy has a rank score plot with rank–1 score equal to 1. Therefore, the rank score value, expressed as a function of the rank index, is a monotonically increasing function in [0, 1], and is commonly used in evaluating classifier performances [Philips, 1996].

Assume that Fig. 4.1 represents the rank score obtained for some classifier. This example shows that the probability of making a correct class decision (obtained for rank–1 score) is equal to 0.8. Next, the probability of making a correct decision in the first or second top decisions (represented with the rank–2 score quantity) is equal to 0.84. The example also shows that all correct guesses are contained in the first top five guesses, as the rank–5 score is equal to 1. In the following sections, references to specific classification performance measurements will refer to rank–1 score values, unless specified otherwise.

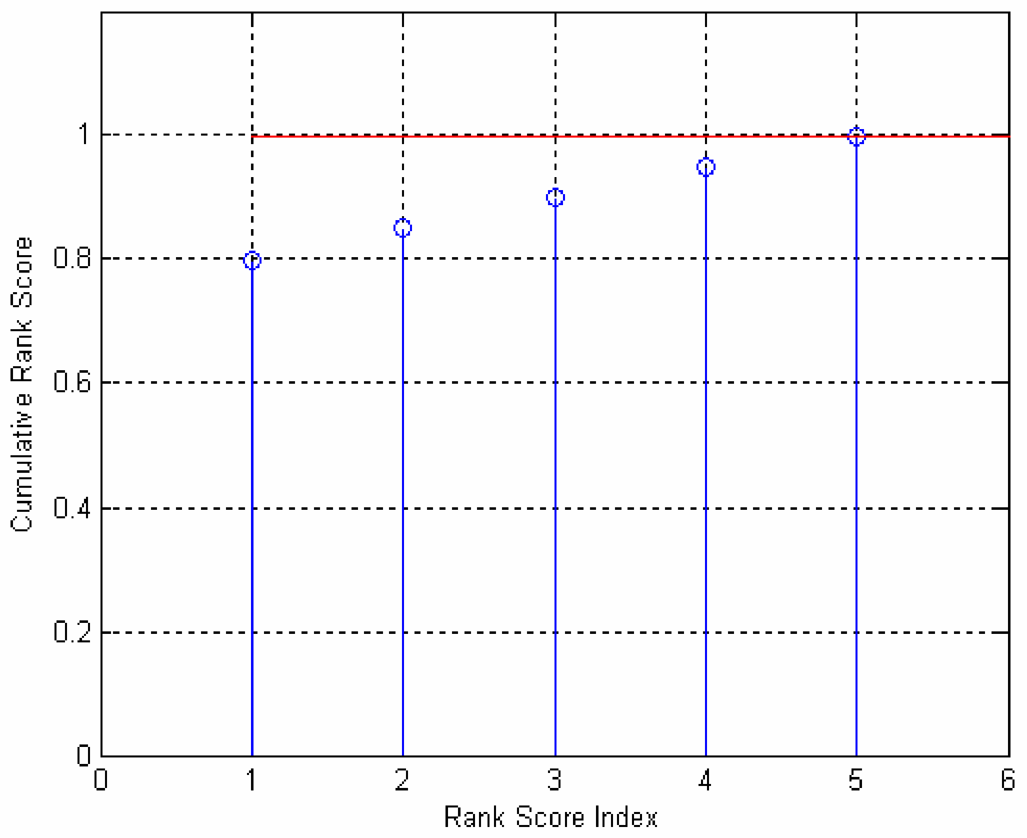


Figure 4.1. Rank score example.

4. Confidence Interval Definitions

Recall that the sample mean computed from finite data only approximates the (true) population mean, with accuracy depending on the data size [Jain, 1991]. A useful statistical measure in addition to the sample mean is the associated confidence interval, which provides the user with a measure of reliability of the observed mean. Ideally, a confidence interval width should be as small as possible, because the true mean lies within the confidence interval with some predefined probability and thus it is easier to estimate its true value.

Next, the concept of confidence interval (CI) is briefly reviewed. Assume that there are n observations $\left[x_{1,}x_2,...,x_n\right]$ obtained for one trial of the whole population of observations (i.e., measurements) [Jain, 1991]. Further, assume that the population has true mean μ, standard deviation σ, and sample mean $\bar{\mu}$. Note that the true mean μ is a deterministic number, whereas the sample mean $\bar{\mu}$ is a random number. As a result, sample means computed from different data trials will be close to the true mean, but will most likely be different from each other.

The fact that the sample size is finite means that $\bar{\mu}$ is just an estimate of the true mean μ. Thus, it is quite useful to estimate the corresponding confidence interval $[b_1, b_2]$ defined around μ, as this information gives the user a measure of confidence in the estimated true mean. The values b_1, b_2 are called the CI lower and upper bounds, respectively. The measure of confidence is defined as $1-\alpha$, where α is called the significance level, and $100 \cdot (1-\alpha)$ is called the confidence level. This definition means that there is $100 \cdot (1-\alpha)\%$ probability that the true mean, computed from multiple trials, lies in the derived confidence interval $[b_1, b_2]$. However, one trial can be selected to estimate the confidence interval associated with the sample data mean $\bar{\mu}$, provided the sample size n is large enough, as a result of the central limit theorem. In such a case, $\bar{\mu}$ can be shown to be approximately Gaussian with mean μ and standard deviation $\frac{\sigma}{\sqrt{n}}$, where n and σ are the sample size and standard deviation of the data sequence, respectively.

It can be shown that the data mean confidence interval can be expressed generally as:

$$[b_1, b_2] = \left[\bar{\mu} - t_{\left[1-\frac{\alpha}{2};n-1\right]} \frac{\sigma}{\sqrt{n}}, \bar{\mu} + t_{\left[1-\frac{\alpha}{2};n-1\right]} \frac{\sigma}{\sqrt{n}} \right], \tag{4.7}$$

where $t_{\left[1-\frac{\alpha}{2};n-1\right]}$ is obtained from the Student's t-distribution, where $n-1$ represents the number of degrees of freedom and n is the sample size. However, for large sample sizes, the t–distribution values can be approximated by the normal distribution values. Therefore, given that the sample size is 1000 for this study, the CI is computed using the normal distribution as:

$$[b_1, b_2] = \left[\bar{\mu} - z_{1-\frac{\alpha}{2}} \frac{\sigma}{\sqrt{n}}, \bar{\mu} + z_{1-\frac{\alpha}{2}} \frac{\sigma}{\sqrt{n}} \right]. \tag{4.8}$$

In this study we select a 95% confidence interval, i.e., $\alpha = 0.05$, which leads to a value of $z_{1-\frac{\alpha}{2}} = z_{1-\frac{0.05}{2}} = 1.960$, [Jain, 1991]. As a result, the corresponding 95% confidence interval is defined as:

$$[b_1, b_2] = \left[\bar{\mu} - 1.960\frac{\sigma}{\sqrt{n}}, \bar{\mu} + 1.960\frac{\sigma}{\sqrt{n}}\right], \tag{4.9}$$

where σ and $\bar{\mu}$, and n are the standard deviation, the estimated mean, and the sample size of the measured data x. The value of $z_{1-\frac{\alpha}{2}}$ for the Gaussian distribution can be found in look–up tables contained in statistical textbooks and others [Jain, 1991].

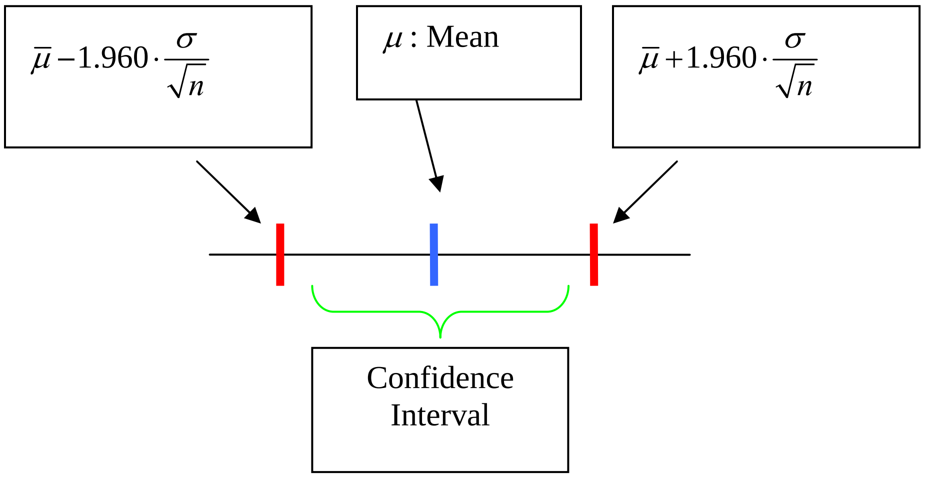

$\bar{\mu} - 1.960 \cdot \frac{\sigma}{\sqrt{n}}$
μ : Mean
$\bar{\mu} + 1.960 \cdot \frac{\sigma}{\sqrt{n}}$
Confidence Interval

Figure 4.2. Mean confidence interval.

5. Confidence Score

Confidence scores are scalar quantities defined in the range [0, 1] which provide an evaluation of the confidence in the class decision. As a result, the confidence score u_{ij} of image j represents the likelihood that it belongs to class i. The confidence score value is defined for a C–class problem as:

$$u_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{d_{ij}}{d_{kj}}\right)^{\frac{2}{(m-1)}}}, \tag{4.10}$$

where d_{ij} represents the distance between image *j* and the i^{th} class centroid, and *m* is a weighting exponent which can take values between $[1,\infty]$ [Jang, 1997]. We selected $m=2$ in this study. Various distance types may be selected in the confidence score definition. In this study we selected the Mahalanobis Angular distance as it closely approximates classification performances obtained with the Mahalanobis distance at a reduced computational cost. Further details regarding the distance metric selection are discussed in Chapter IV.B.4.

6. Kernel Matrix *K* Eigen–Decomposition Issues

An important parameter in this study is the number of top eigenvectors of the kernel matrix used in the derivation of the GDA projection vectors. Recall that eigenvectors associated with "negligible" eigenvalues may be discarded in the computation of the GDA projection vectors ν *'s*, as defined in Chapter III.B.2, since they are expected to have little impact on the resulting projected data. Further, recall that the kernel matrix size is directly related to the number of training data samples (900), resulting in a (900×900) matrix for our study. Figure 4.3 plots the eigenvalues as a function of the eigenvalue index obtained from the eigen–decomposition of the kernel matrix *K,* for one typical iteration in the database. Note that eigenvalue magnitudes decrease quite rapidly, and that those corresponding to index 500 or higher do not seem to contribute significantly to the kernel matrix structure. Further, note that truncating the eigen–decomposition can result in significant computational savings.

Therefore, we investigated whether truncating the kernel matrix *K* eigen–decomposition specifically affected the resulting classification performances with the following experiments:

- Keep all eigenvectors assigned to eigenvalues with value greater than or equal to the maximum eigenvalue divided by 1000 (unless specified otherwise, this is the referred to as the "default case" as was proposed by Baudat [Baudat, October 2000]. For practical purposes, simulations have shown that this choice comes down to keeping about the top first 700 eigenvectors, i.e., the eigenvectors associated with the top 700 eigenvalues in magnitude).
- Keep the top 100 eigenvectors.
- Keep the top 150 eigenvectors.

- Keep the top 200 eigenvectors.
- Keep the top 250 eigenvectors.
- Keep the top 500 eigenvectors.

The results are presented in Chapter IV.B.

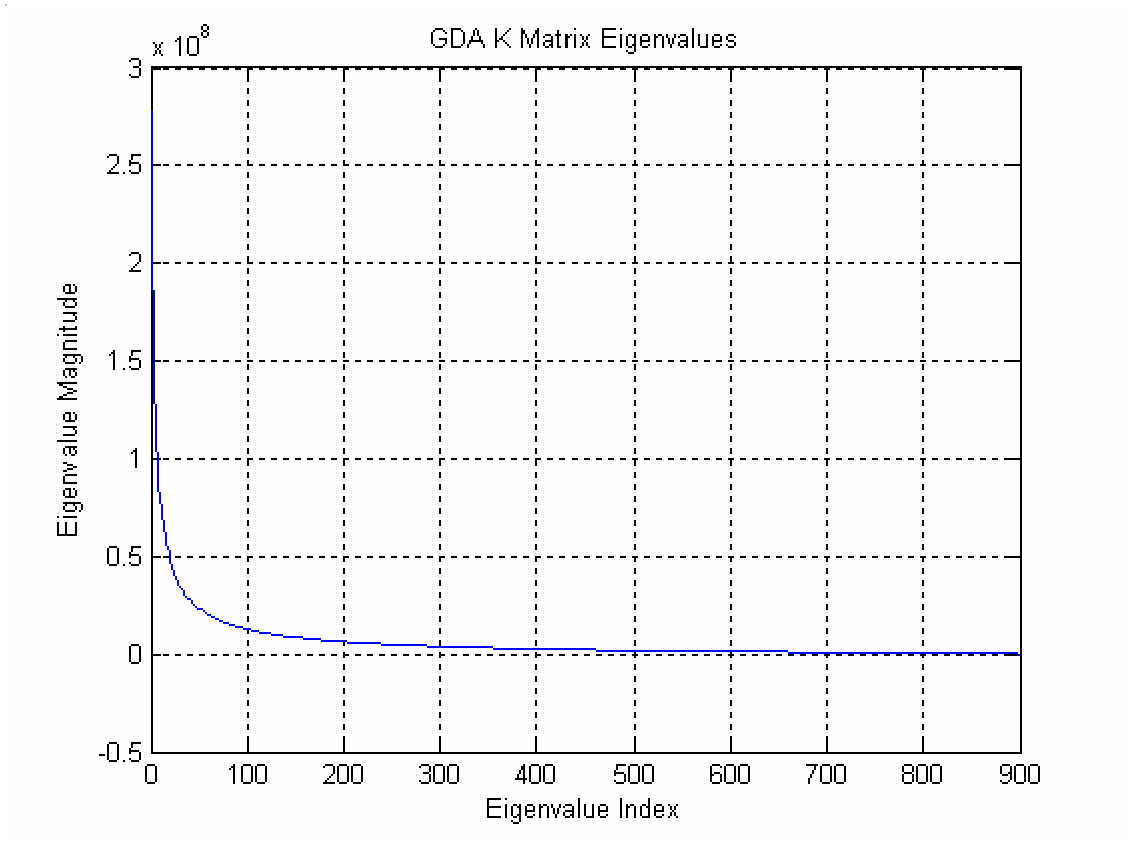


Figure 4.3. Eigenvalues of the kernel matrix *K*.

7. Kernel Function Types Considered

Various types of polynomials may be used in kernel–based schemes, as mentioned earlier. First, the Gaussian kernel was selected, as it has been used with success in numerous applications. Next, polynomials of different degrees were considered such as:

- Polynomial of degree 1: $k(x, y) = (x \cdot y)$, which is equivalent to the LDA implementation [Baudat, 2000],
- Polynomial of degree 2: $k(x, y) = (x \cdot y)^2$,
- Polynomial of degree 3: $k(x, y) = (x \cdot y)^3$.

Results showed the Gaussian kernel led to similar classification performances as that obtained for the polynomial of degree 2, for the selected spread value $\sigma = 7000$. In addition, it was noted that selecting the polynomial kernel allowed us to efficiently vectorize the Matlab code, initially written by Baudat and Anouar [Baudat, October 2000], and significantly reduce the associated computational load. Details regarding the vectorization operations are presented in Appendix B. Specifically, it was noted that the Matlab implementation with Gaussian kernel took about 7.5 to 8 minutes per iteration, while the vectorized Matlab implementation with polynomial kernel of degree 2 took about 75 seconds per iteration in the database, using a 3–GHz PC.

B. EXPERIMENT DESCRIPTION

1. Data Selection Procedures

The database considered in this study is that previously used in [Lee, 2004], and specific details regarding the collection process and data manipulation can be found within. The database includes 50 classes (adult subjects). Each subject was asked to gaze at one of ten points on the wall behind the camera to introduce angle and tilt variations in the subject pose. Each class has the following three sections:

1. Photographed in a neutral pose.
2. Photographed pronouncing the vowel “u”.
3. Photographed while smiling.

As a result, the complete database consists of $(50 \times 10 \times 3) = 1500$ images. All IR images were then cropped to retain only the middle section of the face, which excluded the chin, some of the forehead, and the ears, resulting in cropped images of dimension equal to (45×60) pixels. Next, cropped images were reshaped as column vectors of dimension (2700×1), resulting in an overall database matrix of size (2700×1500).

The data were split into nonoverlapping training and testing sets, and 40% and 60% of the data per class were selected for testing and training sets, respectively. We implemented a cross–validation variant to estimate classification performances based on resampling and averaged over multiple repetitions of each experiment (referred to as an iteration in this study). Therefore, for each iteration, images in each section and class are randomly permuted, and the first four resulting images (40%) of each section and class were selected to create the Testing Image Data Matrix (TEIDM), while the last six images (60%) were selected to form the Training Image Data Matrix (TRIDM). As a result, the TEIDM and TRIDM have dimensions equal to $\left(2700\times600\right)$ and $\left(2700\times900\right)$, respectively.

2. Number of Iterations per Each Individual Experiment

A significant body of research dealing with small sample size experiments has been conducted in the research community, and most studies consider some type of averaging operations to reduce the impact of the data split between training and testing sets. In this study, we implemented a cross–validation variant based on resampling. We considered 500 and 1000 iterations (i.e., repetitions of each experiment) and selected 1000 iterations to collect sufficient reliable statistical data regarding classification performances.

3. Software Implementation

Four main categories of experiments were conducted in this study:

1. Implementation of the Fisherface algorithm, using the GDA with polynomial kernel of degree 1, to benchmark results against those obtained earlier by Lee [Lee, 2004],
2. Selection of the best distance metric to be applied in the classifier. This selection was conducted using 3–dimensional distance map plots, which are discussed later in Chapter IV.B.4.
3. Computation of rank scores and overall classification performance per class for 40/60 cross–validation variant and 1000 iterations.
4. Computation of 95% confidence intervals for rank–1 to rank–5 scores mean classification results.

All the software code developed for these simulations is included in Appendix B. Note that the software implementation used in this study was either written by the author or modified from that developed earlier by Baudat [Baudat, October 2000]. The main steps associated with a given iteration are listed in Table 4.1.

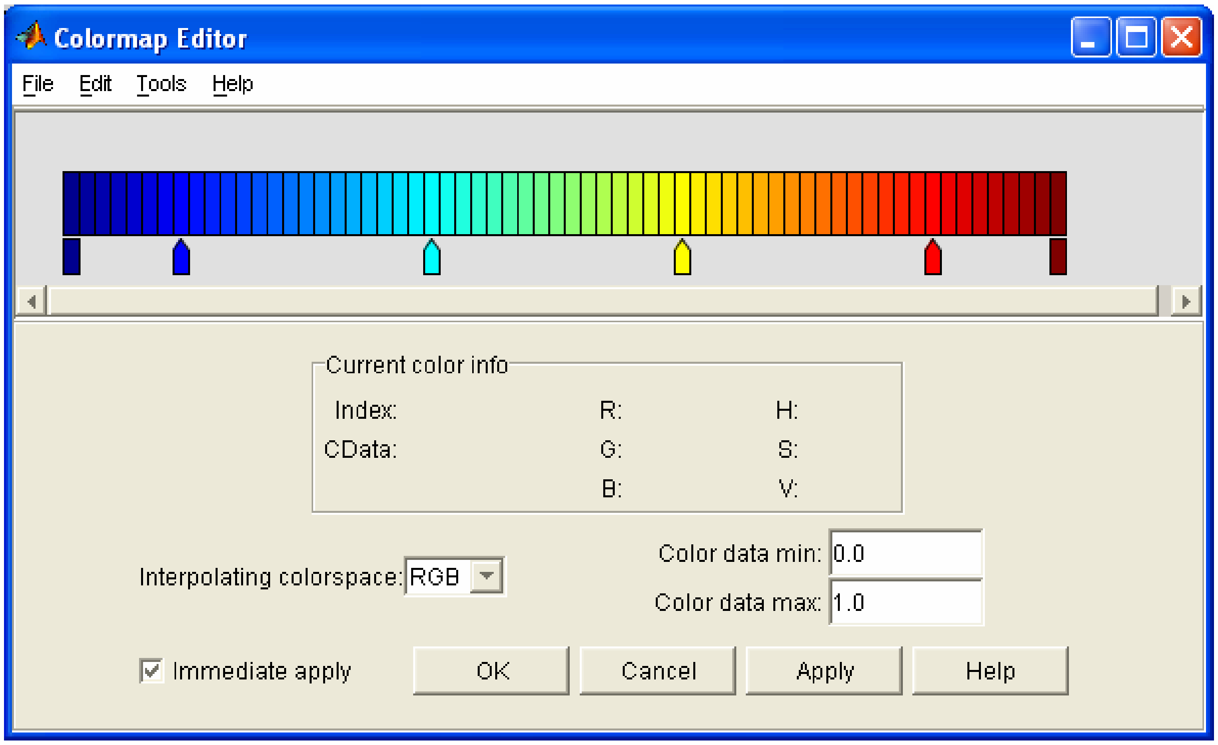
Data Selection	1. Execute one random permutation of the images within each section and class, as discussed in Chapter IV.B.1 above. 2. Generate TRIDM and TEIDM matrices, as described in Chapter IV.B.1.
GDA steps	3. Apply the GDA algorithm to the TRIDM to create the kernel matrix *K*. 4. Compute the eigen–decomposition of the kernel matrix *K*. 5. Project TRIDM and TEIDM matrices onto the kernel matrix *K*.
Classification steps	6. Compute the class centroids from the projected TRIDM information. 7. Compute the projected training images covariance matrix (needed for the Mahalanobis distance considered in the study). 8. Compute the distance between projected training data and class centroids and between projected testing data and class centroids. 9. For each distance considered and projected training and testing data sample, make class decision by selecting as class that which leads to the smallest distance between projected data and class centroids.

Table 4.1. Main software steps required for a given iteration.

4. Distance and Eigenvector Impacts on Classification Results

Recall that the classifier scheme analyzed in this study belongs to the family of distance classifiers. Thus, the specific choice of distance may be an important parameter in the resulting algorithm performance. As a result, we investigated how the selection of the specific distance influenced resulting classifier performances. In addition, the specific number of eigenvectors selected in the GDA kernel matrix *K* is also an important user–specified parameter. Therefore, we also investigated the impact the number of eigenvectors extracted from the kernel matrix *K* has on resulting classifier performances. To visualize the impact these two parameters have on the classifier performances, we generated 3–dimensional training and testing distance maps, which plot the distances between each training (testing) image and all training data centroids, respectively for the five distance types discussed earlier in Chapter IV.A.2, while varying the number of top eigenvectors in the kernel matrix *K*.

Figure 4.4 illustrates the Matlab color code variations used in Figs. 4.5 to 4.14., which investigate the classifier performance as a function of the distance used. Table 4.2 lists all experiments conducted according to the various types of distances used, the type of images selected (testing/training), and the number of kernel matrix *K* eigenvectors selected for the projections. These simulations were repeated ten times and Figs. 4.5 to 4.14 represent a typical result, (as significant consistency was observed between trials).



Current color info

Index: R: H:

CData: G: S:

B: V:

Interpolating colorspace: RGB

Color data min: 0.0

Color data max: 1.0

Immediate apply [✓]

Figure 4.4. Matlab color code scaling.

Type of Distance used	**Training images, Default**	**Training images, 250**	**Testing images, Default**	**Testing images, 250**
Norm–2	X	X	X	X
Mahalanobis	X	X	X	X
Mahalanobis Norm–2	X	X	X	X
Mahalanobis Norm–1	X	X	X	X
Mahalanobis Angular	X	X	X	X

Table 4.2. Distance maps computed. "Default" or "250" refers to the number of kernel matrix eigenvectors kept in the GDA algorithm step.

Each figure corresponds to a specific distance and number of kernel matrix K eigenvectors used in the GDA projection step, and is subdivided into four sections. The two sub–figures in the top row represent the 3–dimensional distance maps obtained for training images (left) and testing images (right), respectively. The bottom row represents the corresponding contour plots. Distance maps generated are of dimension $(X \times Y)$, where X represents the number of training (equal to 900), or testing (equal to 600) images on the x axis, and Y (equal to 50) represents the number of classes on the y axis. Finally, the vertical z axis represents the measured distance between each sample and all training data class centroids.

The distance maps are set up so that diagonal values represent the distance between each image and its own class centroid, while off–diagonal elements represent distances between images and centroids from other classes. As a result, one would expect to see small distances (i.e., blue colors) on the diagonal elements and larger ones (i.e., towards red colors) on the off–diagonal elements to represent good classification behavior. The “perfect” classifier should exhibit a distance map showing uniform deep blue diagonal elements and uniform red off–diagonal elements. As a result, these distance maps make it possible to evaluate the resulting classifier performances visually by noting:

- How uniformly deep blue diagonal elements are (deep blue color on the main diagonal insures the smallest distances are observed between images and centroids from their own class).
- How red off–diagonal elements are (red on the off–diagonal elements insures larger distances between images and other class centroids are observed).
- How uniformly red the off–diagonal elements are (uniformly red color on the off–diagonal elements indicate low variance in the results).

In addition, results show that classification performances increase with the number of kernel matrix eigenvectors selected in the GDA computation, except for one specific case discussed later. Results also show that better performances are observed when dealing with training images than for testing images, as expected because the centroids are derived from the training data itself.

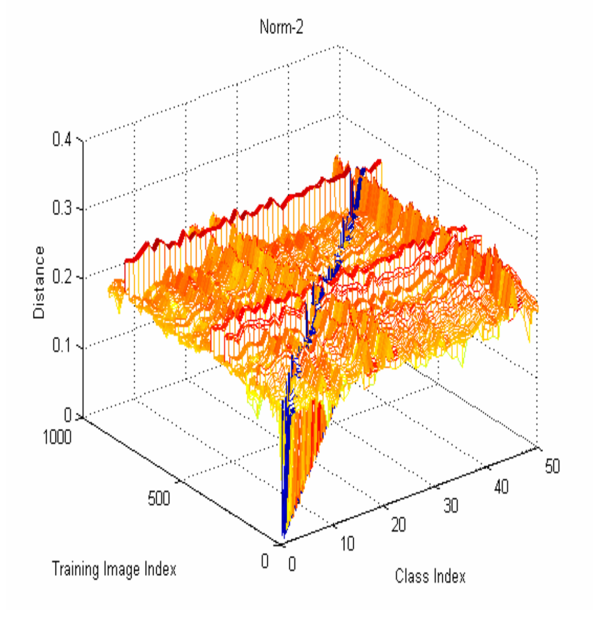
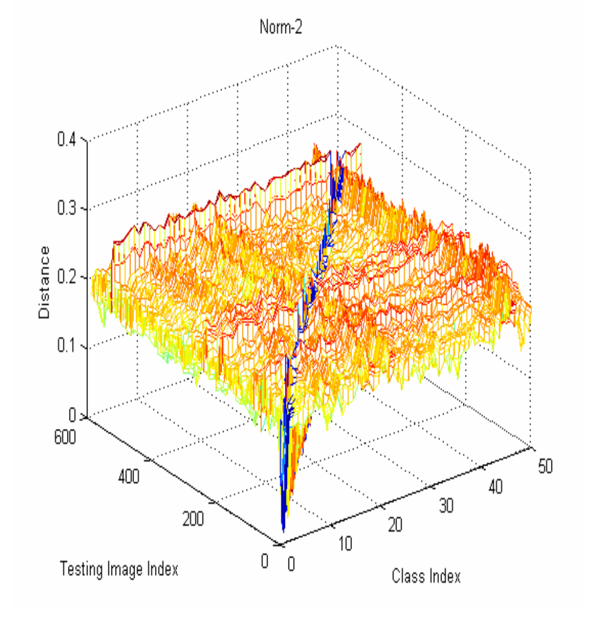
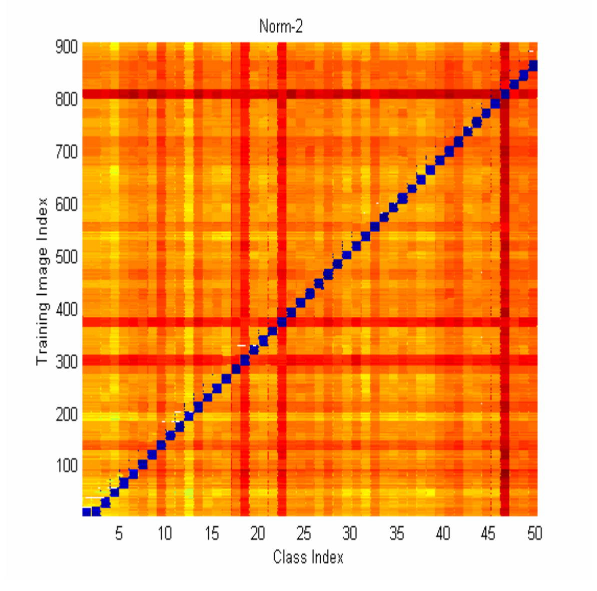
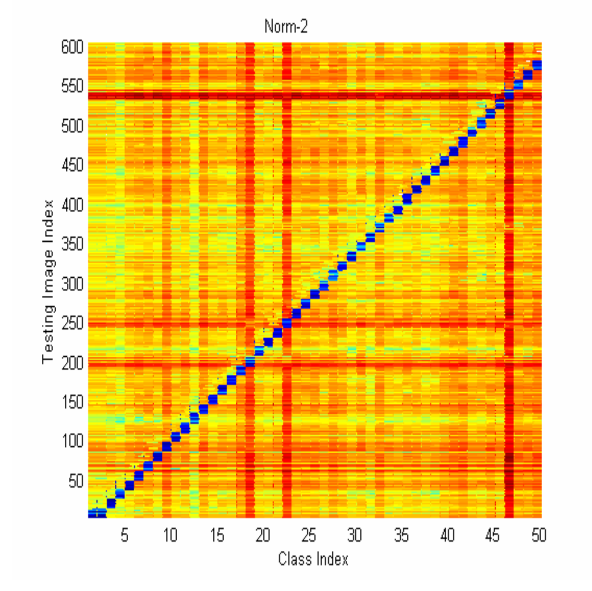
TRAINING IMAGES	TESTING IMAGES
	
	

Figure 4.5. 3–Dimensional plots–contours using Norm–2 distance and default number of eigenvectors.

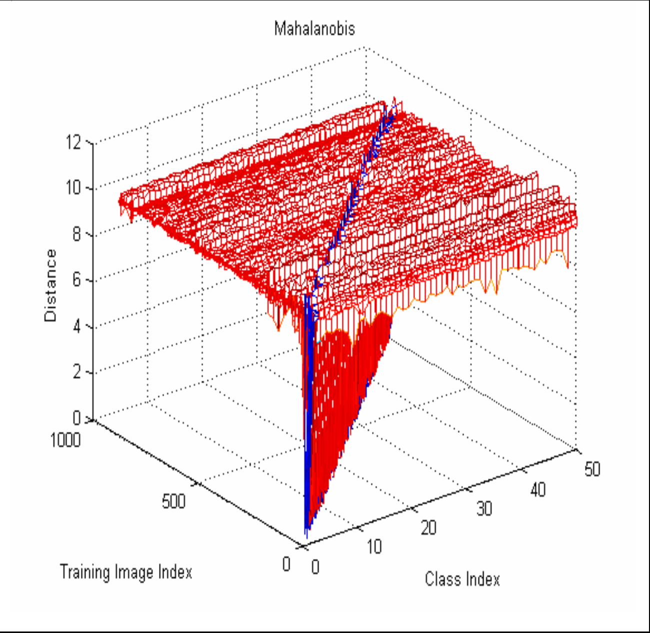
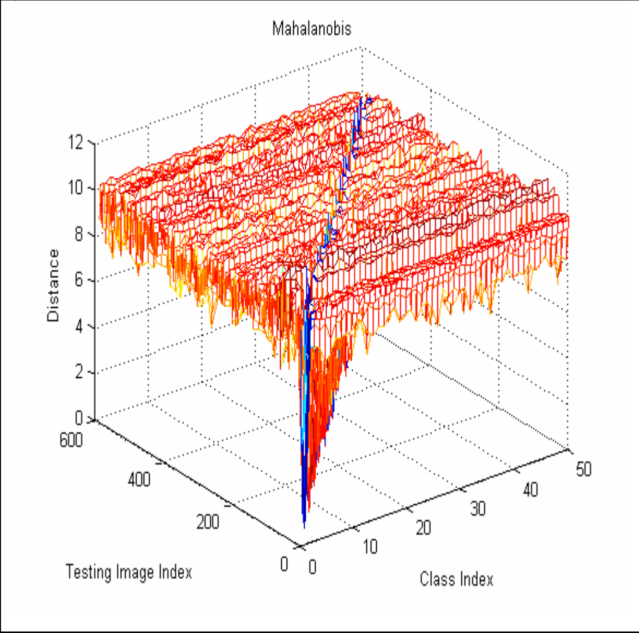
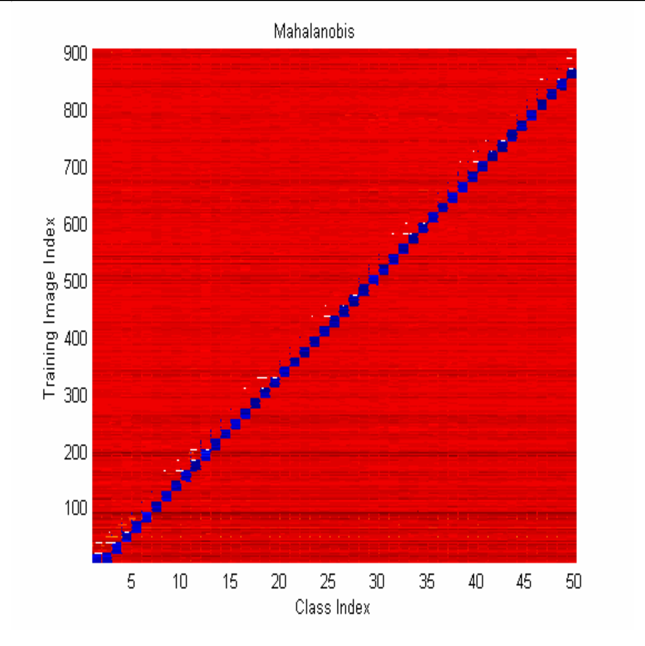
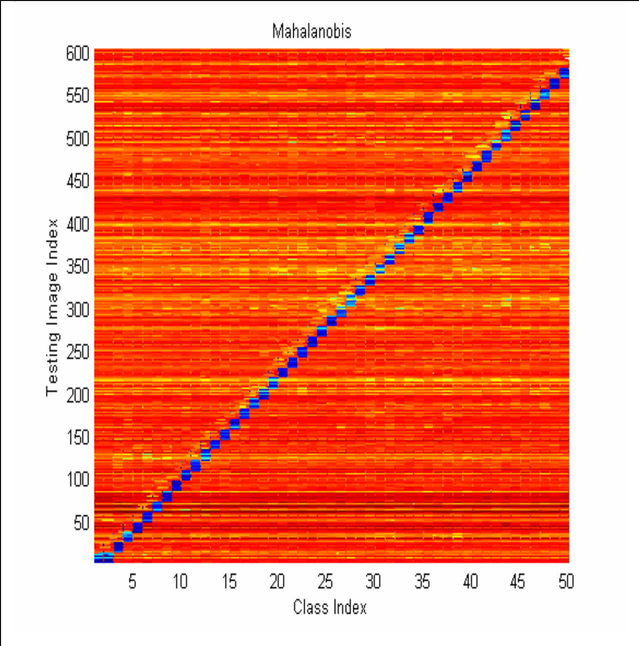
TRAINING IMAGES	TESTING IMAGES
	
	

Figure 4.6. 3–Dimensional plots–contours using Mahalanobis distance and default number of eigenvectors.

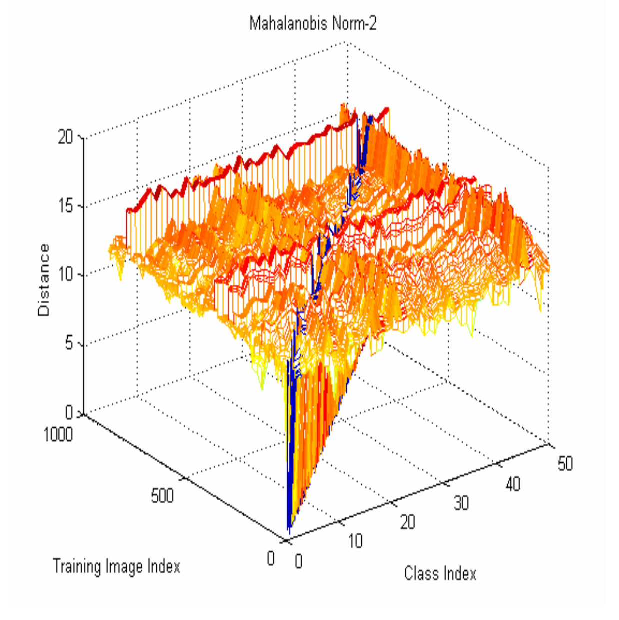
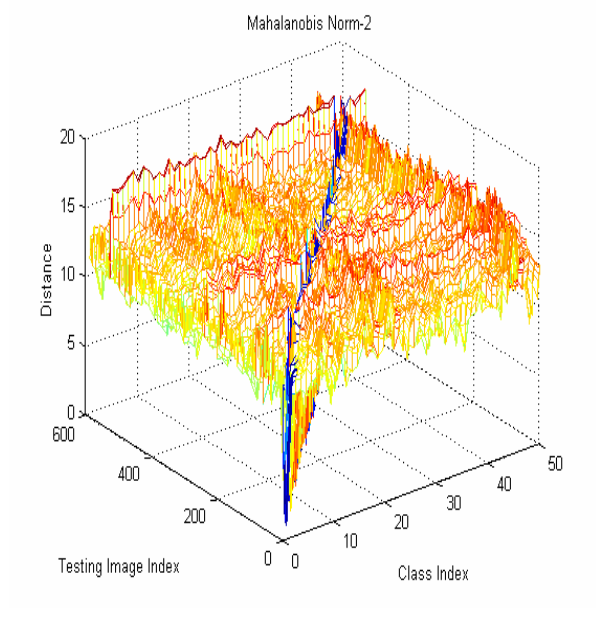
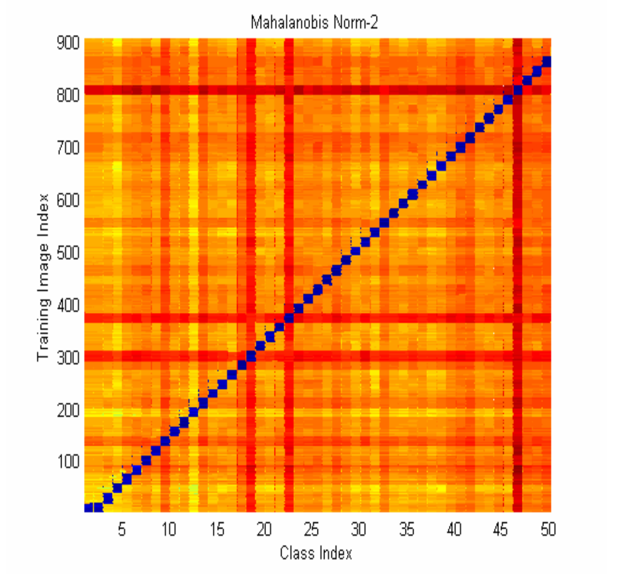
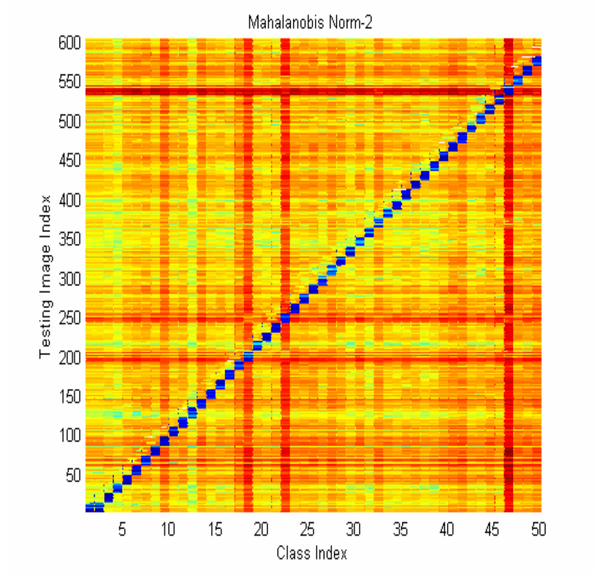
TRAINING IMAGES	TESTING IMAGES
	
	

Figure 4.7. 3–Dimensional plots–contours using Mahalanobis Norm–2 distance and default number of eigenvectors.

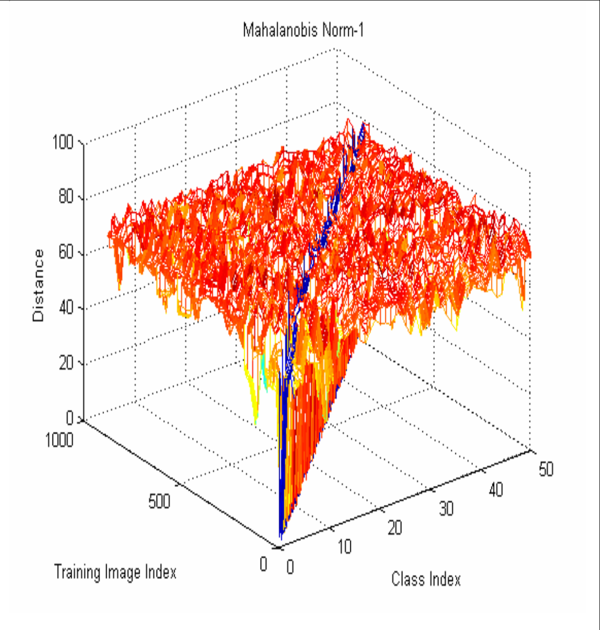
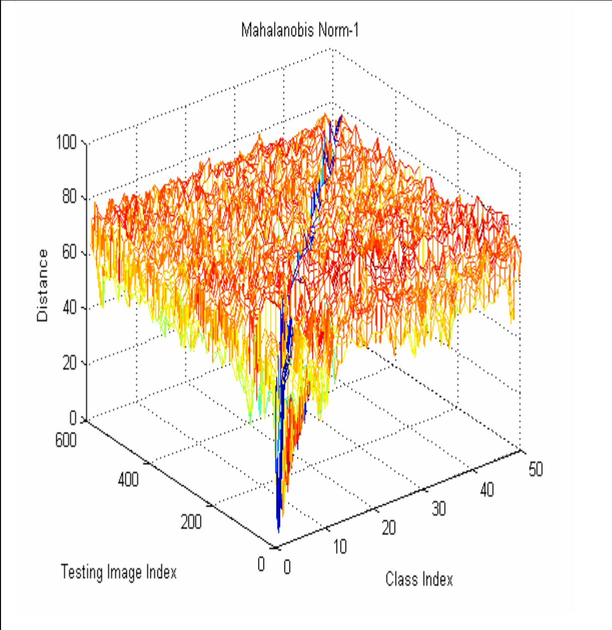
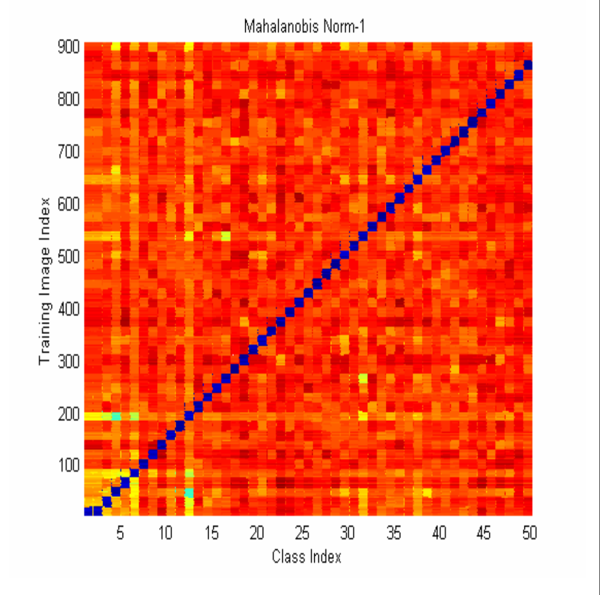
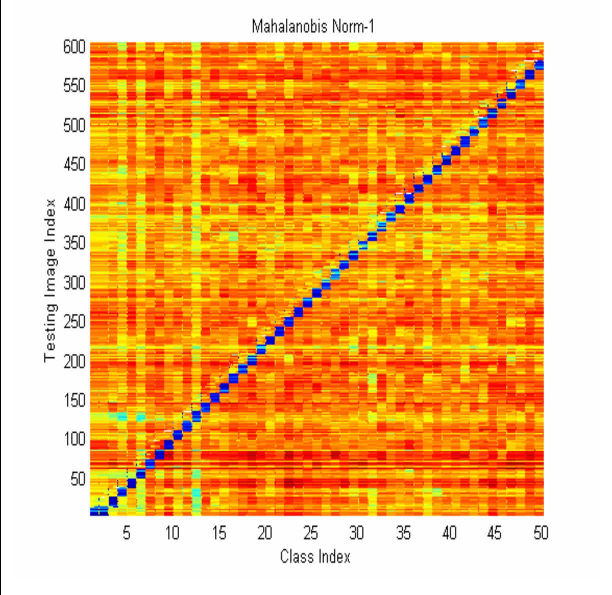
TRAINING IMAGES	TESTING IMAGES
	
	

Figure 4.8. 3–Dimensional plots–contours using Mahalanobis Norm–1 distance and default number of eigenvectors.

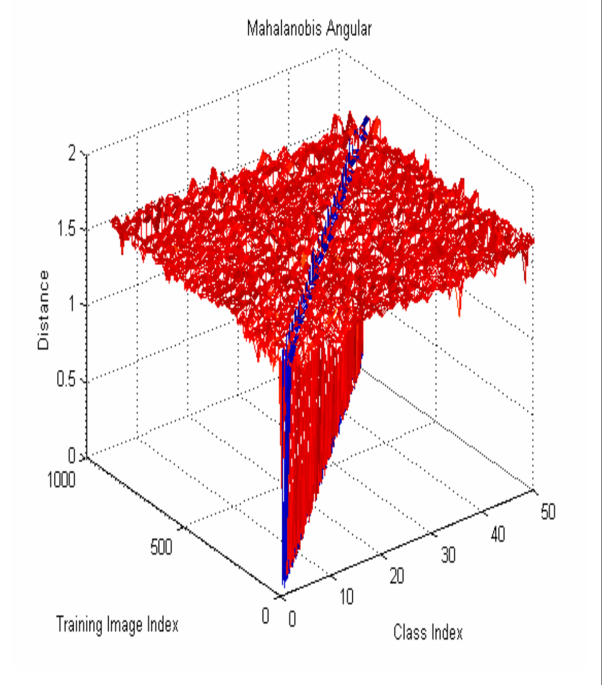
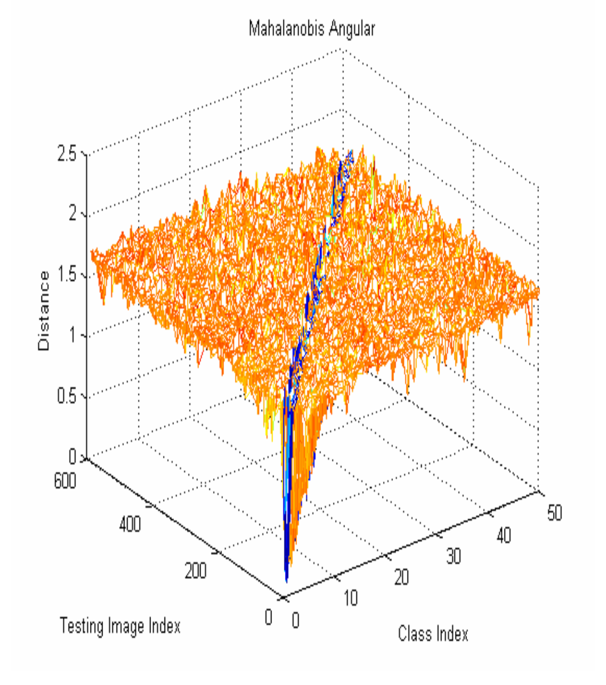
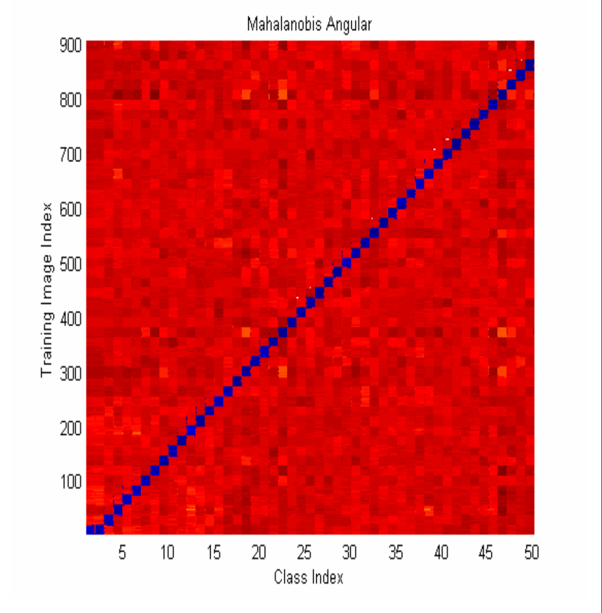
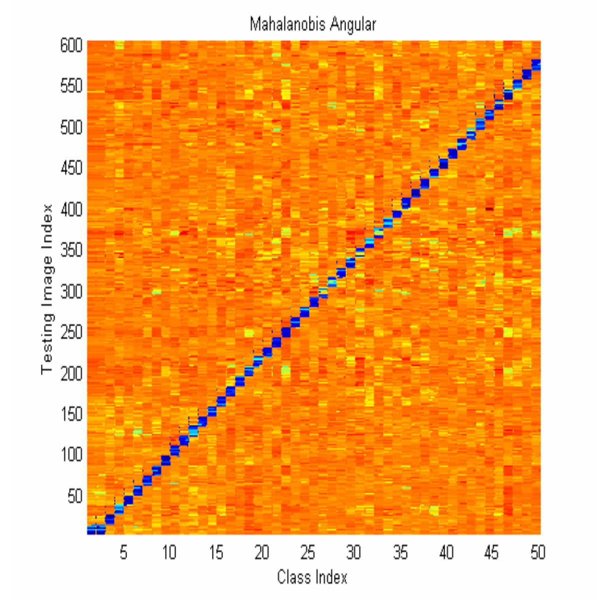
TRAINING IMAGES	TESTING IMAGES
	
	

Figure 4.9. 3–Dimensional plots–contours using Mahalanobis Angular distance and default number of eigenvectors.

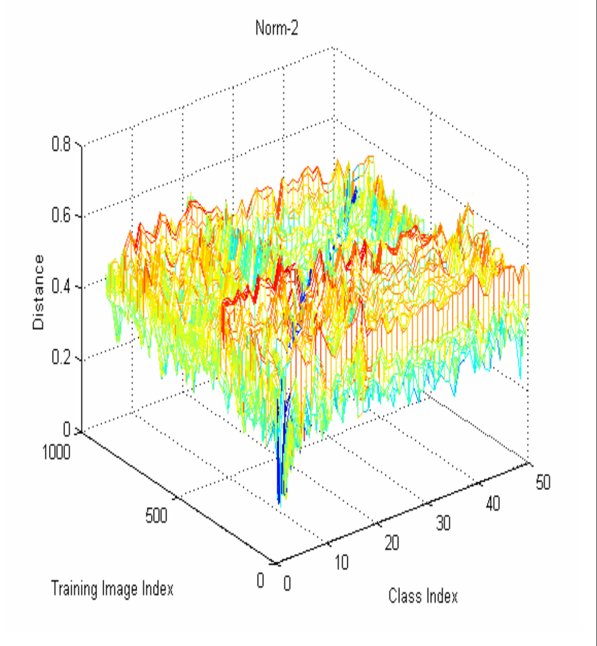
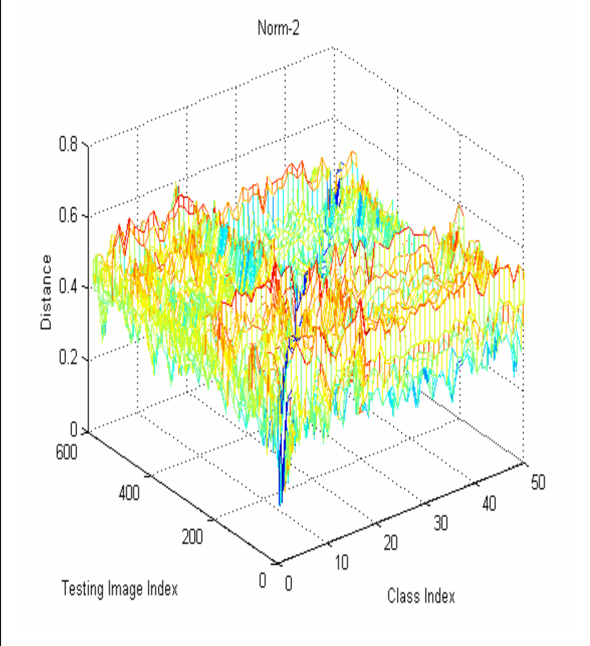
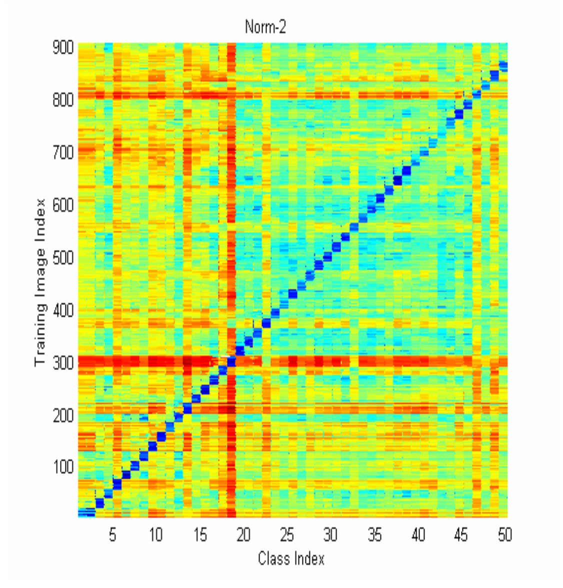
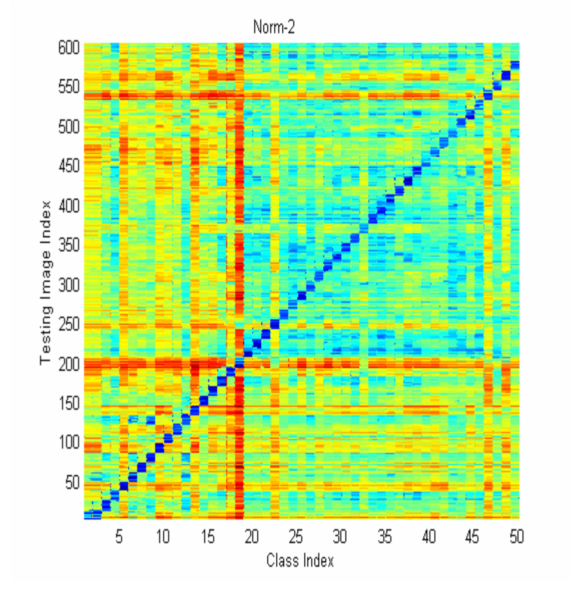
TRAINING IMAGES	TESTING IMAGES
	
	

Figure 4.10. 3–Dimensional plots–contours using Norm–2 distance and 250 eigenvectors.

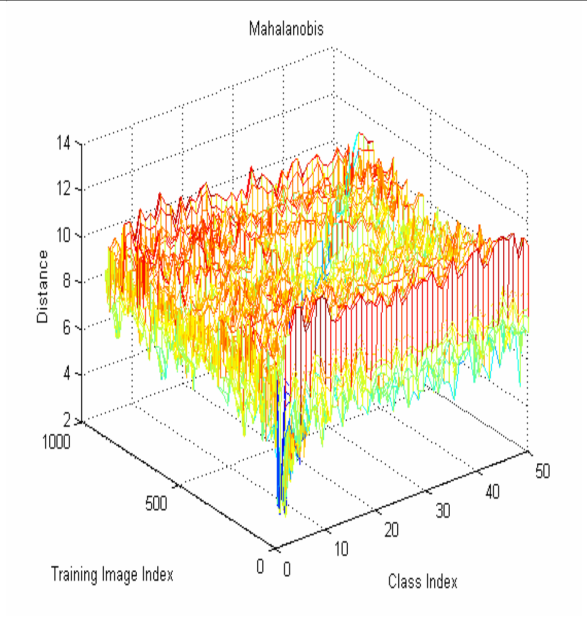
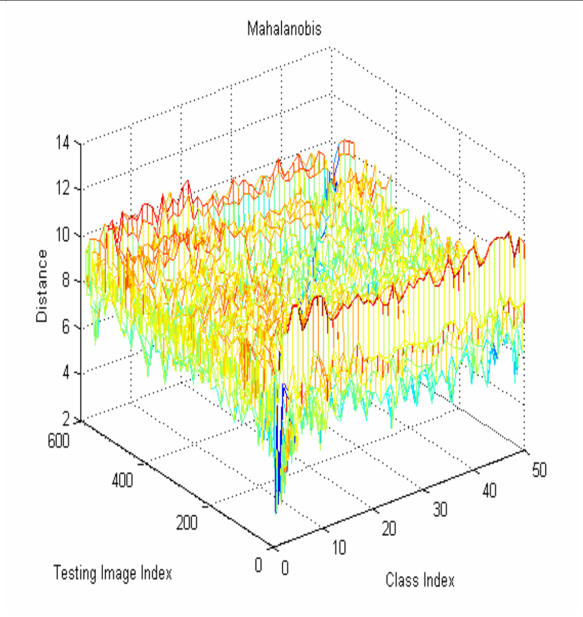
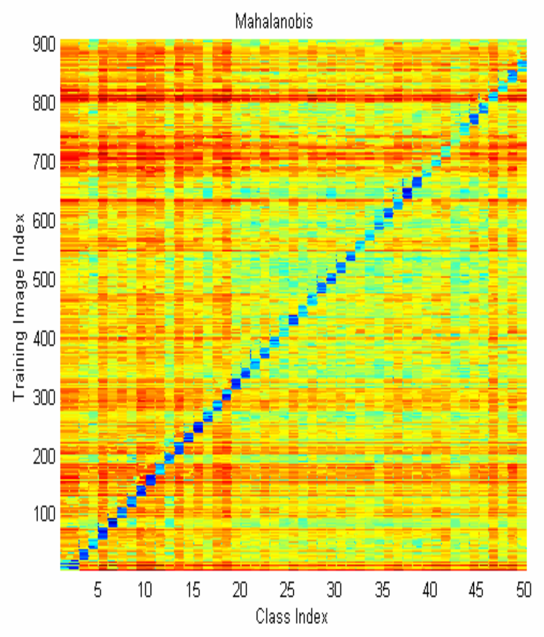
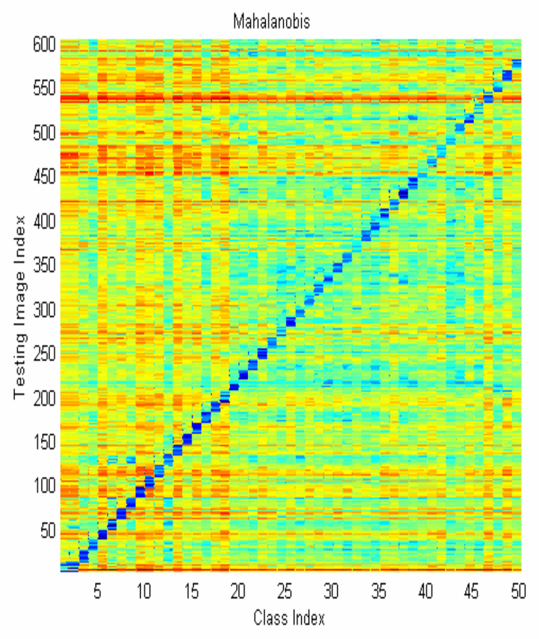
TRAINING IMAGES	TESTING IMAGES
	
	

Figure 4.11. 3–Dimensional plots–contours using Mahalanobis distance and 250 eigen-vectors.

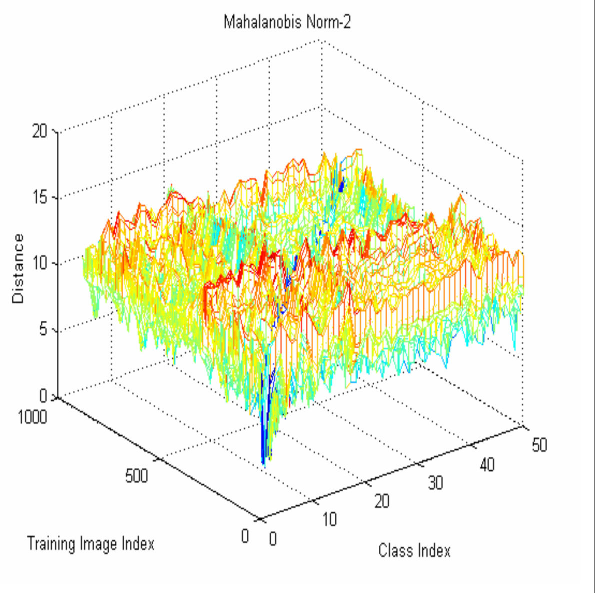
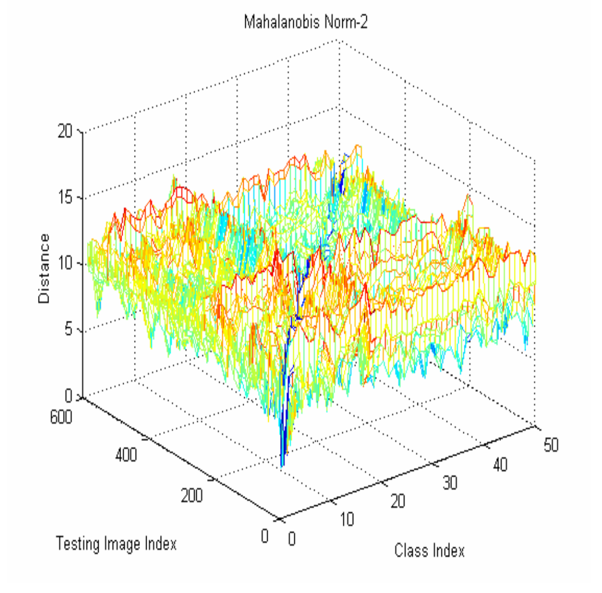
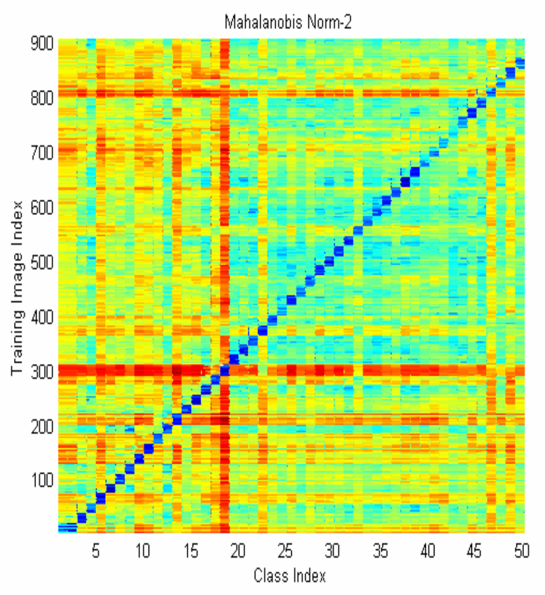
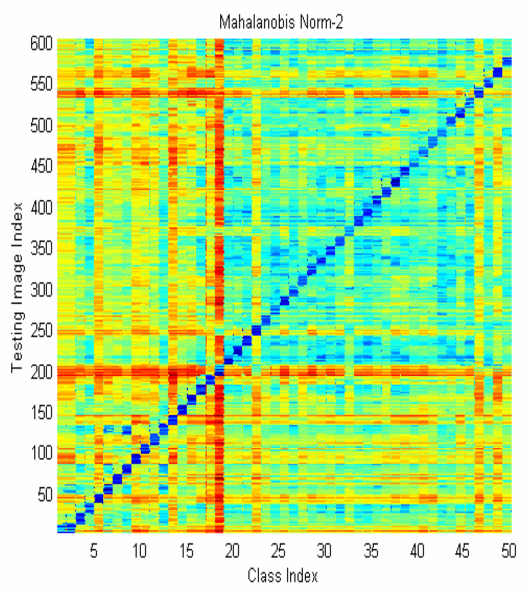
TRAINING IMAGES	TESTING IMAGES
	
	

Figure 4.12. 3–Dimensional plots–contours using Mahalanobis Norm–2 distance and 250 eigenvectors.

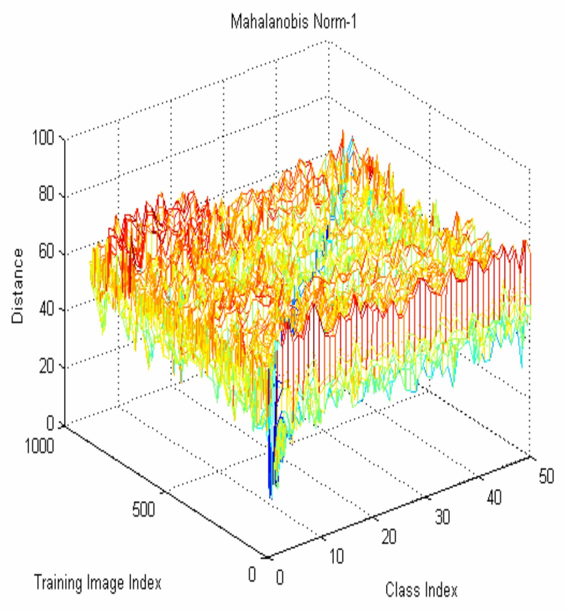
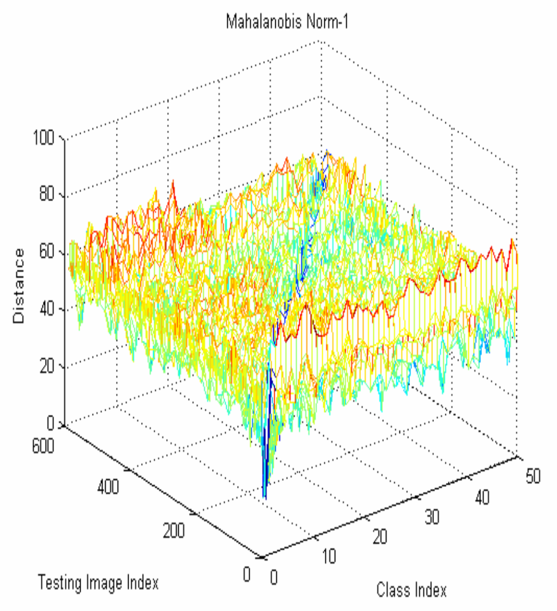
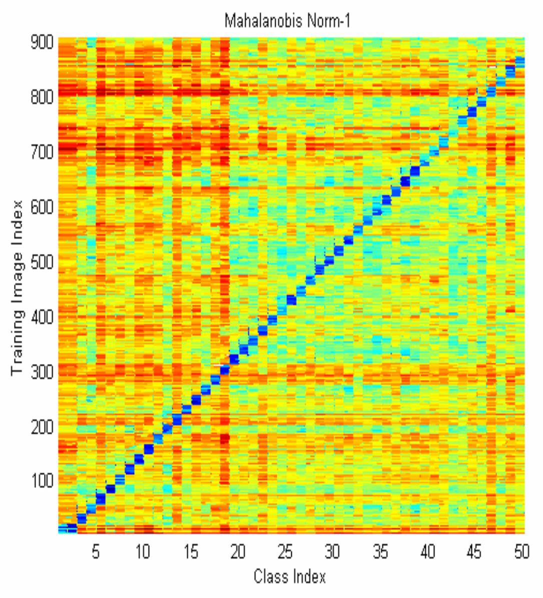
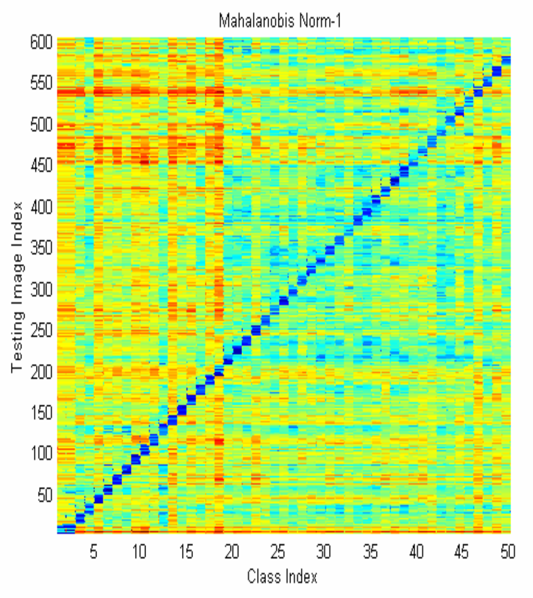
TRAINING IMAGES	TESTING IMAGES
	
	

Figure 4.13. 3–Dimensional plots–contours using Mahalanobis Norm–1 distance and 250 eigenvectors.

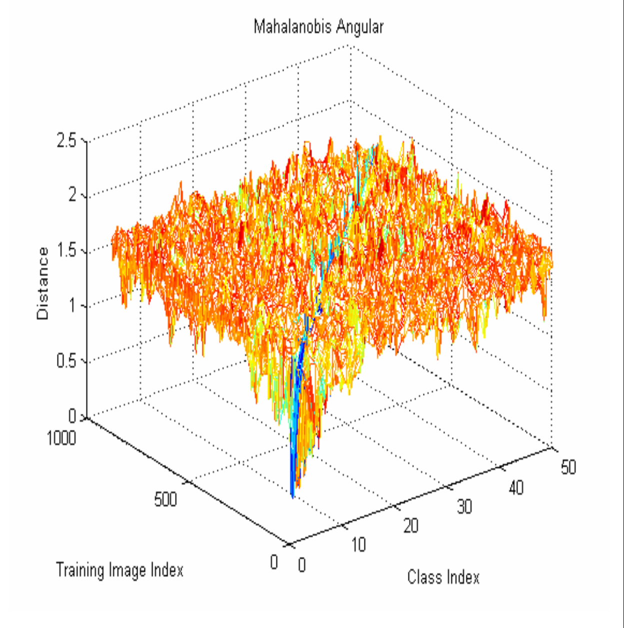
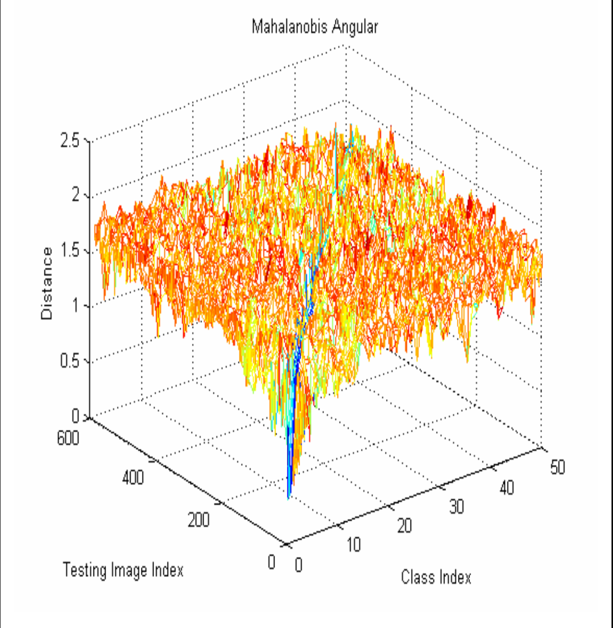
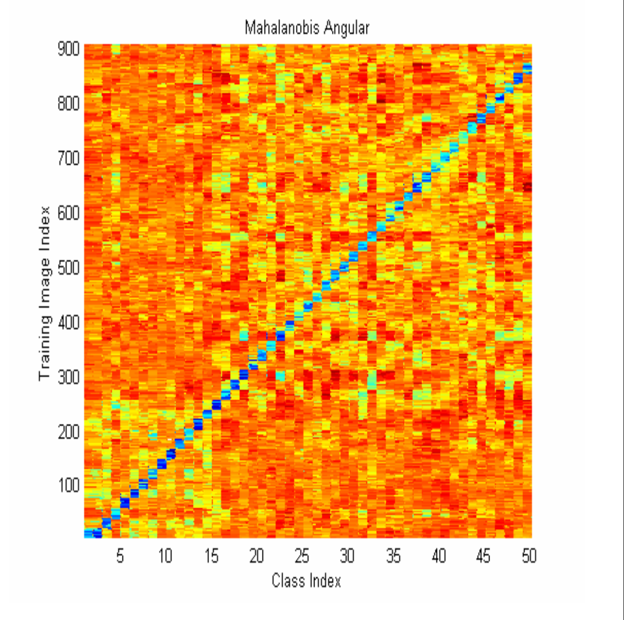
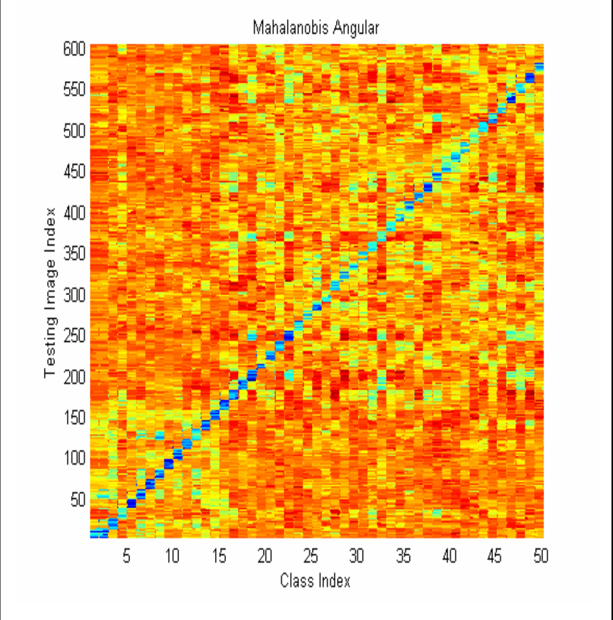
TRAINING IMAGES	TESTING IMAGES
	
	

Figure 4.14. 3–Dimensional plots–contours using Mahalanobis Angular distance and 250 eigenvectors.

5. Rank Score and Overall Class Performance

Cumulative Rank Scores (CRS) and average classification rates are computed to investigate the impact that the various distances, polynomial degree orders and numbers of eigenvectors from the kernel matrix *K* selected in the GDA step have on overall classifier performances. All results provided are obtained with the 40/60 cross–validation variant discussed earlier in Section B.1 and 1000 repetitions (i.e., iterations) to insure results are statistically meaningful. Software implementations computing rank scores and average classification rates are provided in Appendix B.

Figures 4.15 and 4.16 show the rank score and classification performance on a per class basis, obtained for the testing images dataset, for one representative iteration. Additional random iteration rank score plots are included in Appendix C.

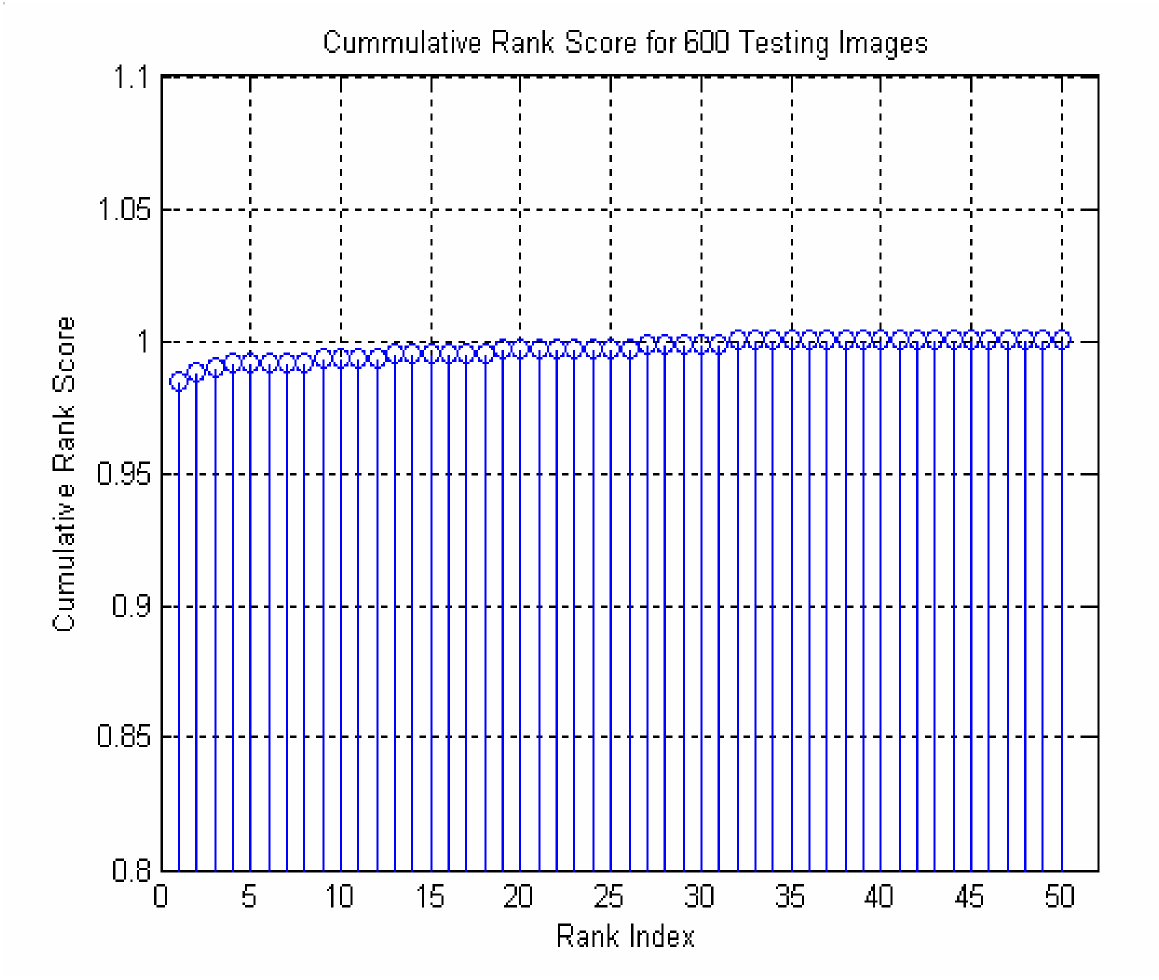


Figure 4.15. Testing data CRS plot; Mahalanobis Angular distance, default number of eigenvectors selected in the GDA step, one iteration.

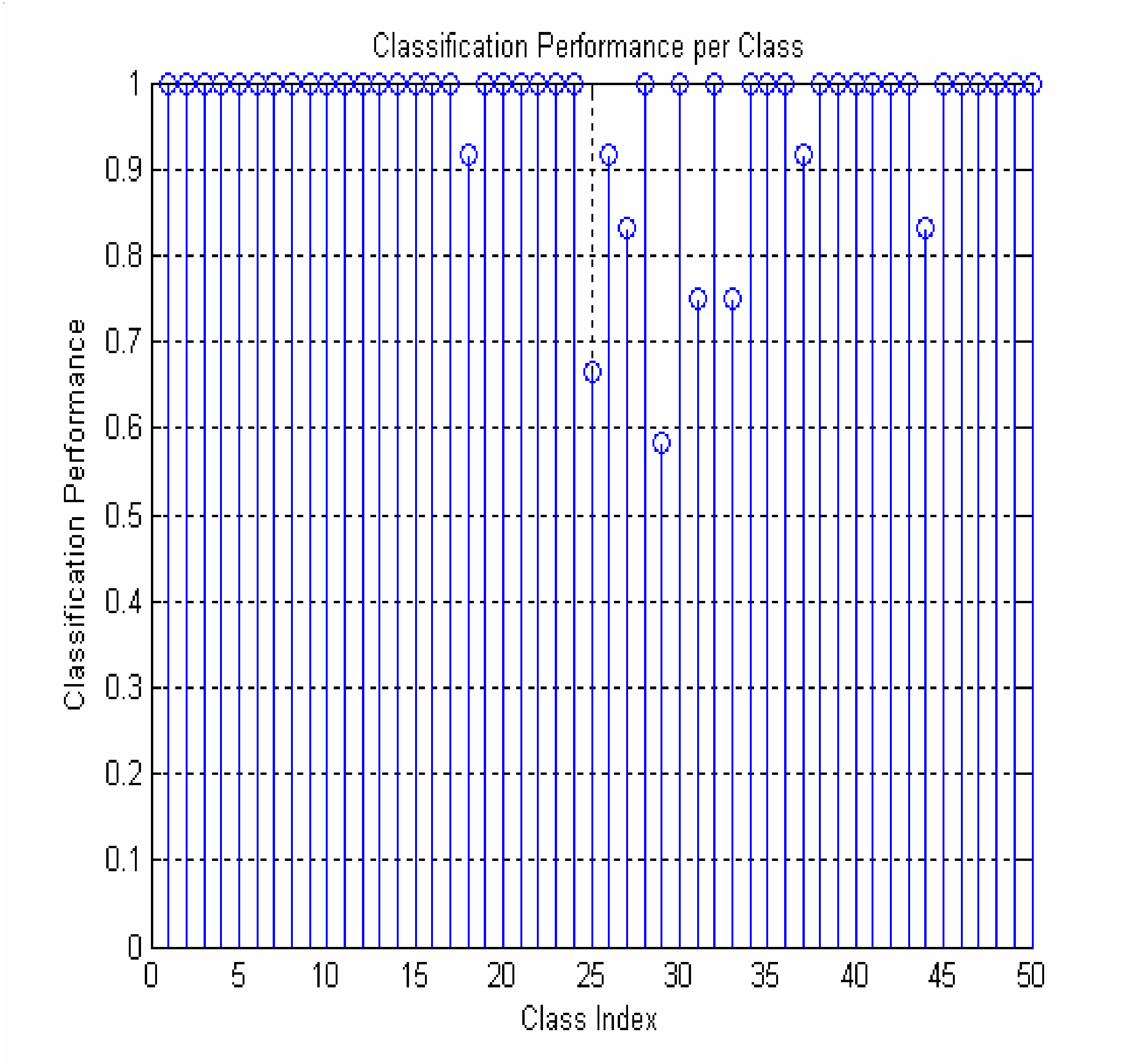


Figure 4.16. Testing data average classification rate; Mahalanobis Angular distance, default number of eigenvectors selected in the GDA step, one iteration.

Table 4.3 presents average classification rates and associated 95% confidence intervals obtained for all scenarios investigated in the study.

The following comments can be made:

- Results obtained for the Fisherface implemented as PCA and GDA with polynomial kernel of degree 1 and default number of eigenvectors selected in the GDA step are similar to those obtained earlier by Lee [Lee, 2004]. It is also noted that the Euclidian distance leads to better classification results (94.59%) than the Mahalanobis Angular distance does (93.70%). Therefore, there is no advantage when using the Fisherface in selecting the Mahalanobis Angular distance.

- Results show that when selecting a polynomial kernel of order 2 in the GDA step and keeping eigenvectors as specified in Baudat [Baudat, October 2000] (shown in row 3 of Table 4.3), the best average classification rates are obtained with the Mahalanobis distance (98.44%), closely followed by the Mahalanobis Angular distance (98.41%). Recall that the Mahalanobis Angular distance is less computationally intensive as it does not require computing the data covariance matrix inverse. As a result, the Mahalanobis Angular distance was selected for all follow–on experiments, and all other results are based on this metric, unless specified otherwise.
- Figures 4.11 and 4.14 show that using only 250 kernel matrix *K* eigenvectors and the Mahalanobis Angular distance appears to lead to better separation between intra–class and inter–class distance values, and more consistent inter–class distance values, than observed with the Mahalanobis distance.
- Results show that reducing the number of kernel matrix *K* eigenvectors selected in the GDA step usually results in classification performance degradations. Specifically, the investigation was conducted by selecting the top 100, 150, 200, 250, 500, and the default number, as defined by Baudat, for a given distance metric. Results show that classification performances degrade as the number of eigenvectors decreases, with the exception of the slightly better performance observed with 500 eigenvectors (98.55%) than with the default number (around 700) (98.41%), which it is not viewed as statistically significant. Overall, eigenvectors associated with very small eigenvalues are expected to have small impact on the resulting classification performances.
- Additionally the GDA with polynomial kernel of degree 1 (i.e., LDA) was investigated directly and resulted in lower classification performances than those obtained with the Fisherface implementation.
- Results show a very slight increase in classification performances by increasing the polynomial degree to 3 (98.45%) from 2 (98.41%), when selecting the default number of kernel matrix *K* eigenvectors.

Overall Classification rates: Mean and 95% confidence interval

Algorithm selected (polynomial deg. order, number of eigenvectors selected in the GDA step)	DISTANCE SELECTED				
	Norm–2	Mahalanobis	Mahalanobis Norm–2	Mahalanobis Norm–1	Mahalanobis Angular
GDA (1, default)	0.9273 [0.9266, 0.9280]				0.9282 [0.9275, 0.9289]
Fisherface (PCA+GDA (1, default))	0.9459 [0.9453, 0.9466]				0.9370 [0.9263, 0.9376]
GDA (2, default)	0.9826 [0.9822, 0.9831]	0.9844 [0.9840, 0.9848]	0.9826 [0.9822, 0.9831]	0.9799 [0.9795, 0.9804]	0.9841 [0.9837, 0.9845]
GDA (2, 500)					0.9855 [0.9851, 0.9859]
GDA (2, 250)					0.9774 [0.9769, 0.9778]
GDA (2, 200)					0.9670 [0.9665, 0.9676]
GDA (2, 150)					0.9448 [0.9442, 0.9455]
GDA (2, 100)					0.8944 [0.8935, 0.8953]
GDA (3, default)					0.9845 [0.9841, 0.9849]

Table 4.3. Classification rates for testing data; 40%/60%–cross–validation variant, 1000 iterations; means and 95% confidence intervals as a function of the kernel function polynomial degree, distance type, and number of eigenvectors selected for the GDA step.

6. Mean Performance Confidence Intervals

Figure 4.17 plots the 95% confidence intervals obtained for rank–1 to rank–5 scores mean classification rates with the five distances investigated in our study and kernel function polynomial of degree 2. Results show that best rank–1 to rank –5 scores are consistently obtained with the Mahalanobis distance (as noted earlier for rank–1 scores in Table 4.3). Results also show that the Mahalanobis Angular distance rank–1 to –5 scores are consistently very close to the Mahalanobis scores, which further validates the selection of the Mahalanobis Angular distance as the metric of choice in this classifier implementation. The Mahalanobis Angular distance has better performance with fewer eigenvectors selected than the Mahalanobis distance.

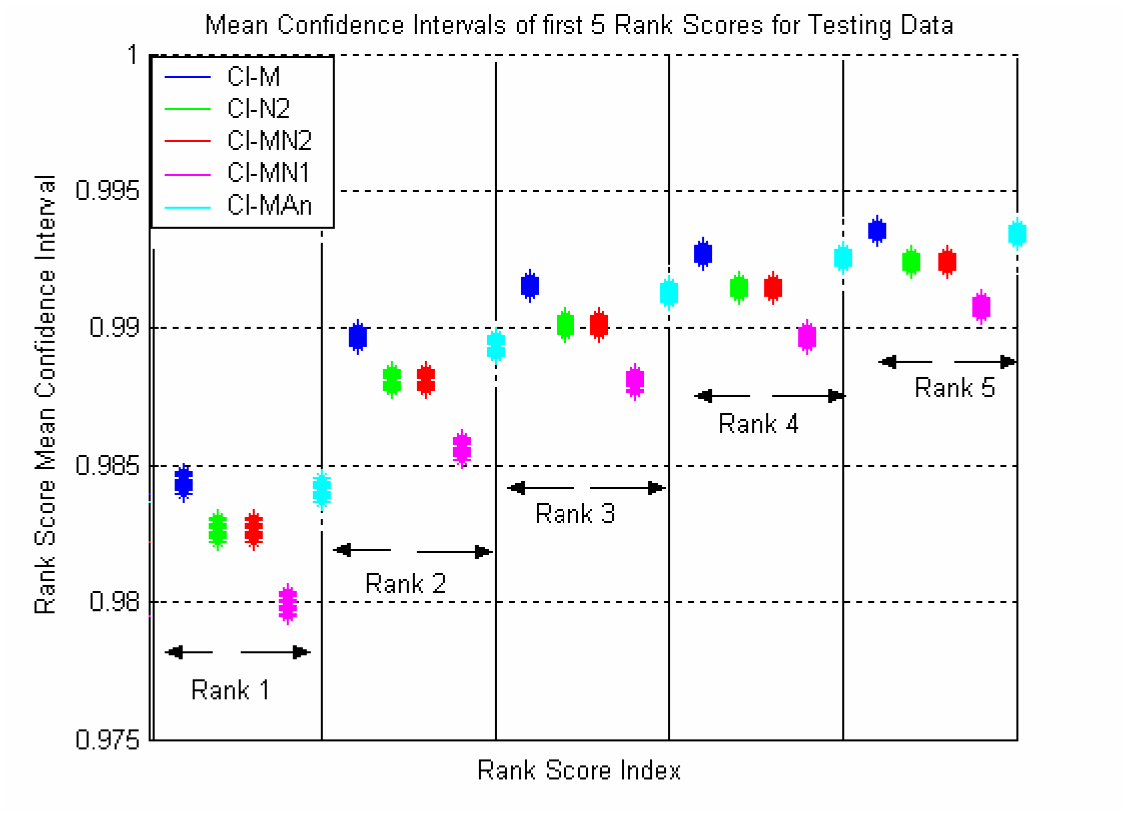


Figure 4.17. 95% Confidence interval of the means of ranks 1, 2, 3, 4, and 5 of the five distances listed on the legend.

7. Confidence Score

The confidence score provides the user with a measure to evaluate the confidence level with which each image belongs to a class. This parameter is bound between 0 and 1 and is useful when evaluating the classifier decision confidence, and/or when fusing various classifier decisions on the same data. Figures 4.18 and 4.19 plot representative 3–dimensional confidence score maps obtained for one iteration. The dimension of the matrix plotted is again equal to $(X \times Y)$, where X corresponds to the number of testing images (600), and Y represents the number of classes (equal to 50 in our study). Note that a perfect classifier should lead to a confidence score map with diagonal elements equal to 1 and off–diagonal elements equal to 0. Figures 4.18 (3–dimensional plot) and 4.19 (associated contour) show a confidence score map that has diagonal elements significantly higher than off–diagonal elements with average values equal to 0.5 and 0.02, respectively. Thus, the results show good average performances on that specific iteration. The software implementation for this computation is included in Appendix B.

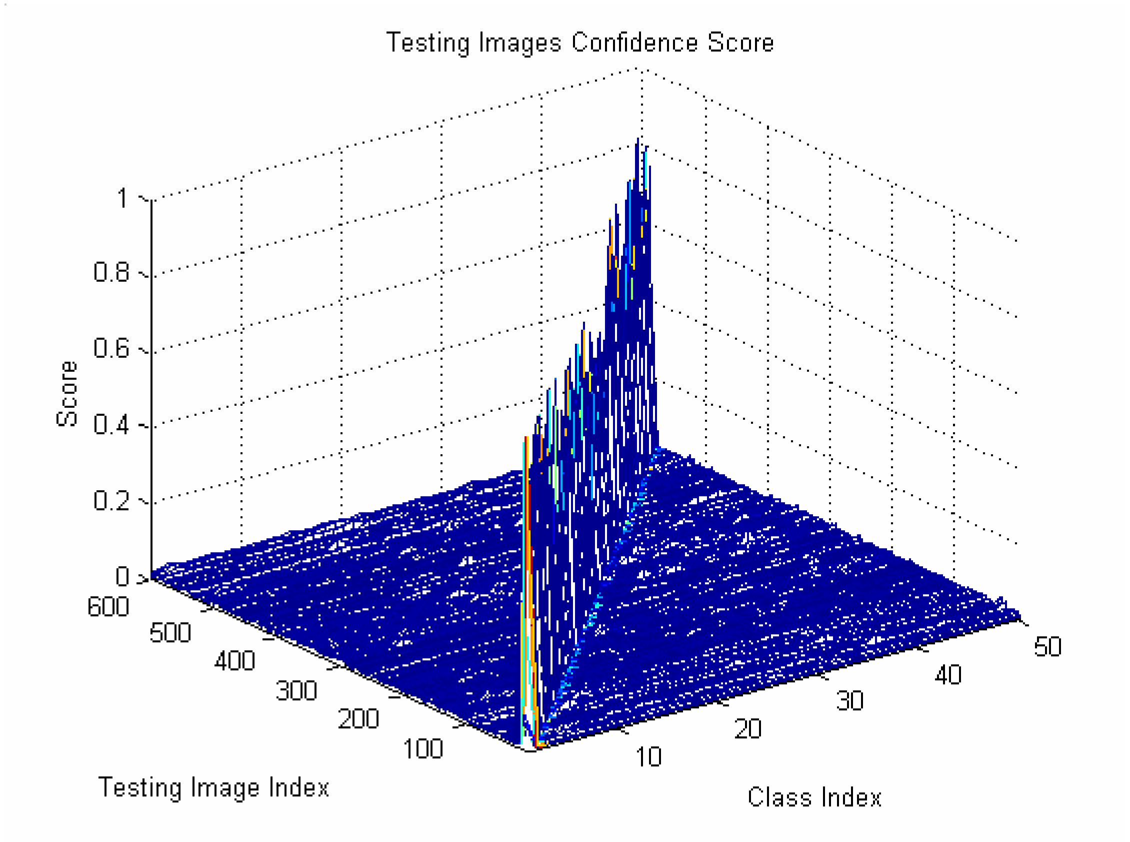


Figure 4.18. Testing set 3–dimensional confidence score map for one representative iteration.

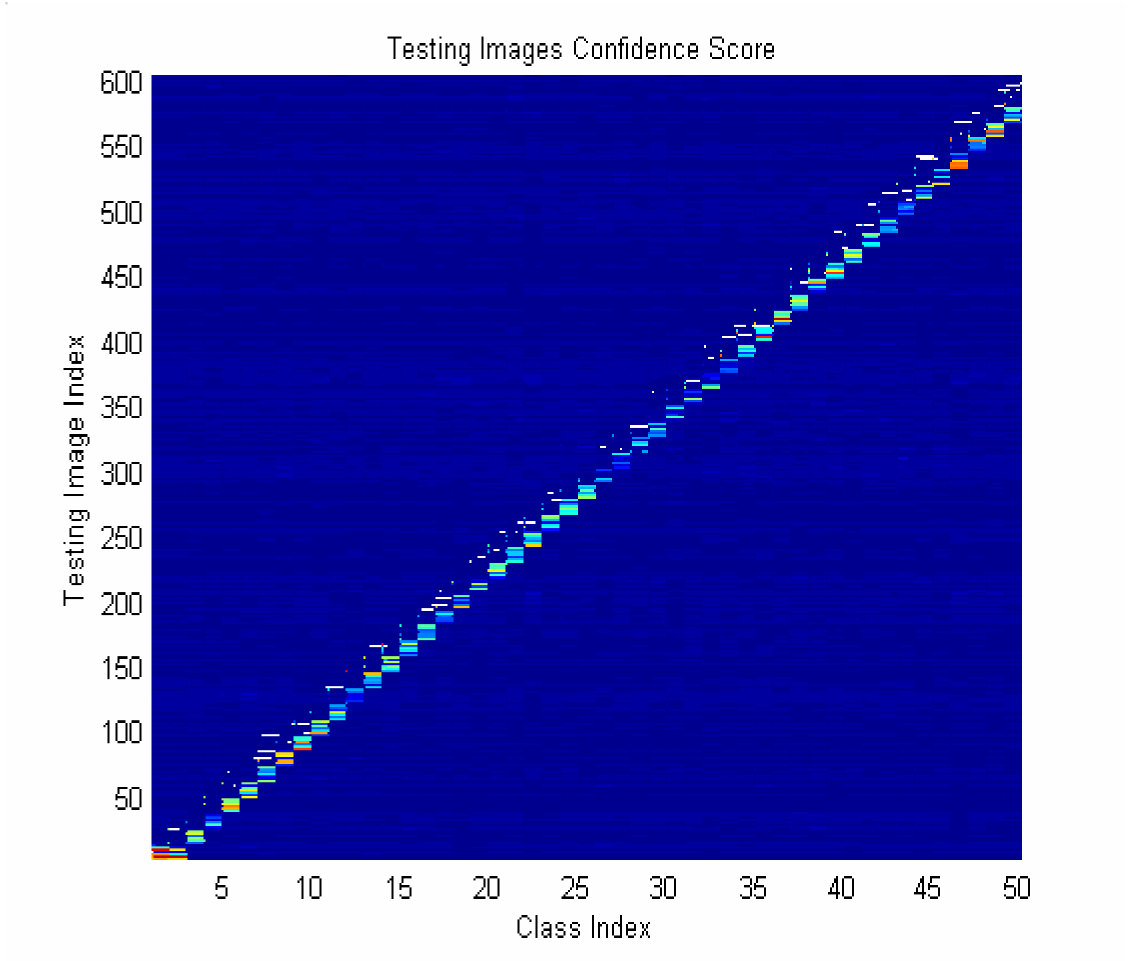


Figure 4.19. Testing set confidence score map for one representative iteration.

C. CONCLUSIONS

This chapter presented the various parameters used to evaluate the GDA–based classification schemes considered in this study. We defined the concept of distance maps and showed how they can be used to visually evaluate classifier performances. We also discussed the concept of average classification rates and 95% confidence intervals, and confidence scores. The concepts of rank scores were reviewed and applied to compare the classifiers investigated in this study. We considered five different types of distances and investigated the impact that the specific number of eigenvectors extracted from the GDA kernel matrix K and selected in the GDA projection step has on resulting classifier performances. Results showed that the optimum performance was recorded for the Mahalanobis distance, used with polynomial kernel function of degree 2, and by selecting the top 500 eigenvectors of the kernel matrix K eigen–decomposition. Results also show that the classifier set–up with the Mahalanobis Angular distance has performances closely approximating those obtained with the Mahalanobis distance, at a much reduced computational cost. Therefore, the Mahalanobis Angular distance was selected as the metric of choice in this study.

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSIONS

This study extended earlier work by Pereira [Pereira, 2002] and Lee [Lee, 2004] who investigated the recognition of infrared face images collected using a low resolution uncooled IR camera under controlled indoor conditions. This study considered a nonlinear kernel–based classifier, the Generalized Discriminant Analysis (GDA) approach proposed earlier by Baudat [Baudat, 2000]. First, two basic linear classification schemes were reviewed, the PCA and the Fisherface approach, which have been widely used in classification applications. Next, the basic idea behind the GDA was presented. It was shown that the GDA can be viewed as a two–step process. First, the data is implicitly nonlinearly projected in a high–dimensional feature space where discrimination becomes easier to be performed. Second, the LDA is implicitly applied to the transformed data. Also the "kernel trick" concept was discussed, which is the fundamental concept behind the GDA. Finally, we proposed a vectorization step to the GDA software implementation proposed by Baudat, which allowed us to speed up computations by a factor of six.

We investigated five different types of distances and selected the best suited for the database under consideration via the evaluation of 3–dimensional distance maps. The concepts of distance classifiers, rank score, confidence interval and confidence score were reviewed as performance measurement tools. A cross–validation variant was implemented to insure that the obtained results were statistically significant. Results showed that the GDA approach leads to better classification performances (98.55%) than those obtained with the linear classifiers considered in the two earlier studies, and increases the average classification performance by 5% over the Fisherface approach. Results also showed that a combination of the Mahalanobis Angular distance, a polynomial kernel of degree 2, and the top 500 eigenvectors lead to the best average classification performances for the GDA implementation for the database under consideration.

In conclusion, this study has shown that a low–cost, low–resolution IR camera combined with an efficient classifier can play an effective role in security related applications.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. MATHEMATICAL BACKGROUND

This appendix presents the main mathematical concepts used in the derivations of the PCA and Fisherface approaches discussed in Chapter II. First, the concept of Rayleigh quotient is reviewed. Next, linear algebra concepts dealing with eigenvalues, eigenvectors, and rank related issues are discussed.

A. ANALYSIS

1. Rayleigh Quotient Analysis

This section shows that computing the vector W, which maximizes the Rayleigh quotient to obtain the Fisher Discriminant vectors and given in Eq. (2.12), Chapter II.B.2:

$$W_{opt} = \arg\left(\max_W \frac{|W^T S_B W|}{|W^T S_W W|} \right), \tag{A.1}$$

is equivalent to solving the generalized eigenvalue problem shown in Eq. (2.13), Chapter II. B.2:

$$S_B w = \lambda S_w w. \tag{A.2}$$

Define the Rayleigh quotient expression $J(w)$ as:

$$J(W) = \frac{|W^T S_B W|}{|W^T S_W W|}. \tag{A.3}$$

The vector W which maximizes $J\left(W\right)$ may be obtained by computing the first derivative of $J\left(W\right)$ and making it equal to zero. The derivation closely follows that presented in [Gutierrez, 2004]:

$$\begin{aligned} &\frac{d[J(W)]}{dW} = \frac{d\left[\dfrac{W^T S_B W}{W^T S_W W}\right]}{dW} = 0 \Rightarrow \\ &[W^T S_W W]\frac{d[W^T S_B W]}{dW} - [W^T S_B W]\frac{d[W^T S_W W]}{dW} = 0 \\ &\Rightarrow [W^T S_W W]2S_B W - [W^T S_B W]2S_W W = 0. \end{aligned} \tag{A.4}$$

Dividing the above equation by the scalar expression $W^T S_W W$ leads to:

$$
\begin{aligned}
&\frac{|W^T S_W W|}{|W^T S_W W|} S_B W - \frac{|W^T S_B W|}{|W^T S_W W|} S_W W = 0 \\
&\Rightarrow S_B W - J S_W W = 0 \\
&\Rightarrow S_B W = J S_W W.
\end{aligned}
\tag{A.5}
$$

Note that the above equation is a generalized eigenvalue problem, which may be rewritten as:

$$S_W^{-1} S_B W - JW = 0, \tag{A.6}$$

which leads to:

$$S_W^{-1} S_B W = JW, \tag{A.7}$$

when S_W is a non singular matrix. Alternate derivations of the Fisher Discriminant vectors when the matrix S_W is singular have been presented by [Cheng, 1992], who proposed to replace the singular matrix by its rank decomposition. However, this specific approach does not appear to have been applied in the face recognition field and is not considered further in our study.

2. Computing the Number of Nonzero Eigenvalues of the Generalized Eigenvalue Eigenvector Decomposition Problem

We show in this section that the following generalized eigen–problem,

$$S_B w = \lambda S_w w \tag{A.8}$$

where S_B and S_W are the Between–Class–Scatter (BCS) and Within–Class–Scatter (WCS) matrices, defined below, exhibits at most $C-1$non–zero generalized eigenvalues, when S_W is not singular, where *C* is the number of the classes.

Recall that:

- The *N*–dimensional matrix S_B is the Between–Class–Scatter–Matrix, is defined as:

$$S_B = \sum_{i=1}^{C} n_i (m_i - m)(m_i - m)^T \tag{A.9}$$

where *m* is the mean of all the element vectors, n_i and m_i, are the number of element vectors, and the mean vector of the class *i*, respectively.

- The *N*–dimensional S_W is the Within–Class–Scatter Matrix, is defined as:

$$S_W = \sum_{i=1}^{C} \sum_{j=1}^{n_i} (x_j - m_i)(x_j - m_i)^T \tag{A.10}$$

where m_i and n_i are the mean vector and the number of elements vectors of the class *i*, respectively.

Note that, when SW is not singular, Eq. (A.8) may be rewritten as:

$$S_W^{-1} S_B W - JW = 0, \tag{A.11}$$

which leads to

$$S_W^{-1} S_B W = JW. \tag{A.12}$$

Next, recall that the rank of a product of two matrices is less or equal to the minimum of the two matrix ranks [Strang, 2003], i.e.,

$$\text{Rank}(AB) \le \min(\text{Rank}(A), \text{Rank}(B)) . \tag{A.13}$$

As a result, the rank of the matrix product $S_W^{-1} S_B$ ($\text{Rank}(S_W^{-1} S_B)$) is less or equal to the smallest rank of the two matrices S_W^{-1} and S_B.

Further, recall that the rank of *S_B* is equal to $C-1$ and the rank of S_W is equal to $P-C$, where *P* is the total number of images, as will be discussed in Appendix A.3. Usually in image classification applications, $C-1 << P-C$, as the number of classes is much smaller than the total number of images. Consequently, we have:

$$\text{Rank}(S_W^{-1} S_B) \le \min\left(C-1, P-C\right) = C-1, \tag{A.14}$$

in practical image classification applications, provided that S_W is not singular. Thus, the maximum number of nonzero eigenvectors extracted from Eq. (A.8) is equal to $C-1$.

3. Computing the Maximum Rank of the Within–Class–Scatter Matrix S_W and of the Between–Class–Scatter Matrix S_B

a. Maximum Rank of the Within–Class–Scatter Matrix

In this section, we show that the *N*–dimensional within–class–scatter (WCS) matrix S_W has maximum rank equal to *P–C*, where *P* is the number of element vectors in the training set, and *C* is the number of classes.

According to Eqs. (2.9) and (2.10) from Chapter II.B.2, the *N*–dimensional WCS matrix S_W of the training data, $\{x_i\}_{i=1,\dots P}$, is defined as:

$$S_W = \sum_{i=1}^{C}\sum_{j=1}^{n_i}(x_j - m_i)(x_j - m_i)^T, \tag{A.15}$$

where n_i and m_i are defined as the number of training vectors and the mean vector for class *i,* respectively. Note that the number of elements per class may vary. As a result, the total number of element vectors is equal to *P,* i.e.,

$$\sum_{i=1}^{C} n_i = P. \tag{A.16}$$

Our derivation is based on the following three matrix rank properties:

1. The rank of the $(N \times N)$ dimensional matrix xx^T, where *x* is a column vector of dimension N, is equal to 1 [Strang, 2003].
2. The maximum possible rank of the matrix *S*, defined as:

$$S = \sum_{i=1}^{P} x_i x_i^T, \tag{A.17}$$

where $\{x_i\}_{i=1,\dots P}$ are a set of column vectors, is equal to *P*.

3. The *N*–dimensional covariance matrix defined from the set of element vectors, $\{x_i\}_{i=1,\dots P}$, and given by:

$$Cov(X) = \sum_{i=1}^{P}(x_i - m)(x_i - m)^T, \tag{A.18}$$

where the mean column vector m is defined as:

$$m = \frac{1}{P}\sum_{i=1}^{P} x_i, \tag{A.19}$$

has possible maximum rank equal to $P-1$, due to the fact that one degree of freedom is lost by the subtraction of the mean element vector in the covariance matrix definition.

Note that the WCS matrix defined in Eq. (A.15) can be viewed as the summation of *C* class–specific covariance matrices, as defined in Eq. (A.18). Using properties 2 and 3 listed above, each i^{th} class–specific covariance matrix can be shown to have a possible maximum rank equal to $(n_i - 1)$. As a result, the overall WCS matrix, derived from the summation of these *C* class–specific covariance matrices has a maximum rank equal to $P - C$, because:

$$\sum_{i=1}^{C} (n_i - 1) = P - C. \tag{A.20}$$

Note that the matrix S_W is *N*–dimensional. Thus, it is nonsingular only when $(P - C) \geq N$, [Strang, 2003]. Further, recall that *P* and *N* represent the number of images and the dimension of the reshaped images as column vectors, respectively. Usually the number of images is smaller than the image dimensions, due to limitations in image collection, resulting in a singular S_W matrix.

b. Maximum Rank of the Between–Class–Scatter Matrix

Next, we show that the maximum rank of the Between–Class–Scatter–Matrix (BCS) S_B is equal to $C - 1$.

Recall that the N–dimensional BCS matrix S_B is defined as:

$$S_B = \sum_{i=1}^{C} n_i (m_i - m)(m_i - m)^T, \tag{A.21}$$

where *m* is the mean of all element vectors, n_i is the number of element–vectors, and m_i is the mean vector of class *i*, respectively.

Note that the matrix S_B is of the same type as the covariance matrix expression defined in Eq. (A.18). Thus, it has a maximum possible rank equal to $\left(C - 1\right)$, due to the fact that one degree of freedom is lost by the subtraction of the mean element vector in the covariance matrix definition.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. MATLAB SOURCE CODE

This appendix lists all Matlab programs used in the study. These programs have been developed or borrowed from other references and modified according to the requirements of the current study. The Matlab programs–functions used, but not developed or modified within the current study, are only referenced without their Matlab code listing being provided.

Below are listed the five categories of experiments being conducted in the subject study. The first program in each category is the one to be executed, whereas the following programs/functions are being called by it.

A. DESCRIPTION

1. PCA–LDA Comparison

- *PCAvsLDA.m*: Creates two 2–dimensional classes, applies PCA and LDA projection on them and compares the two projections.
- *pca.m*: Applies the PCA method on the data, [Lee, 2004].
- *fld.m*: Applies the LDA method on the data, [Lee, 2004].
- *sortem.m*: Sorts eigenvectors by decreasing eigenvalue magnitude [Lee, 2004].

2. Distance Selection –3–Dimensional Plots Creation

- *DistanceSelection.m*: Applies the five distances analyzed in Chapter IV.A.2, to the training/testing images and the classes' centroids to produce the respective 3–dimensional and associated contour plots. Additionally, creates the confidence score 3–dimensional maps when selecting the Mahalanobis Angular distance.
- *dataST.m*: Normalizes input data by setting mean to 0 and standard deviation to 1. (From [Baudat, October 2000].).
- *buildGDA_Opt.m*: Applies the GDA algorithm to the input data and creates the GDA data, which will be used for the input data projection (from [Baudat, October 2000]). This function is modified as commented, in order to vectorize the steps involved in the computations of the GDA projection vectors.
- *eigensystem.m*: Performs eigenvalue–eigenvector decomposition of the input matrix and sorts the eigenvectors by decreasing eigenvalue magnitude (From [Baudat, October 2000].).

- *spreadGDA_Opt.m*: Projects the input data onto the GDA eigenvectors (from [Baudat, October 2000]). This function is modified to vectorize the GDA projection step for higher computational efficiency.
- *KernelFunctiont.m*: It is called only when the Gaussian kernel function is selected by setting the parameter "degree" equal to 0 (From [Baudat, October 2000].).

3. GDA Classification Performance Measurement

- *ClassificationPerformance.m*: Computes and saves the rank–(1–50) scores and the classification performance per class.
- *dataST.m*: As above.
- buildGDA_Opt.m: As above.
- eigensystem.m: As above.
- spreadGDA_Opt.m: As above.
- KernelFunctiont.m: As above.

4. Rank Score 1 Mean Confidence Interval Creation

- *ConfidenceInterval.m*: Derives and plots the 95% confidence intervals obtained for rank–1 to –5 scores for average classification performances.

5. LDA versus Fisherface Comparison, Using GDA with Polynomial Kernel Function of Degree 1 as LDA

- *PCAGDA1.m*: Computes classification performances for the LDA method, (implemented using the GDA with polynomial kernel function of degree 1), and for the Fisherface method (implemented using the PCA and GDA with polynomial kernel of degree 1).
- *pca.m*: As above.
- *sortem.m*: As above.
- buildGDA_Opt.m: As above.
- eigensystem.m: As above.
- *dataST.m*: As above.
- spreadGDA_Opt.m: As above.

B. MATLAB CODE LISTING

- ***PCAvsLDA.m:***

```
%% ****************************************************************
%% Filename:          PCAvsLDA.m
%% Thesis Advisor:   Prof.M.P.Fargues   Naval Postgraduate School
%% Author:           Captain Dimitrios I. Domboulas
%%                   Hellenic Air Force
%% Date:             May 2004
%% Description:      1.Creates random two classes of 2–dimensional input vectors
%%                     and apply PCA and LDA.
%%                   2.Plots the projections'
%%                     of the two schemes and compares their separability.
%% Input:            N/A
%% Outputs:          2–Dimensional projection plot
%% Functions used:   pca.m, fld.m, sortem.m
%% ************************************************************

clc;
clear;

m1=[2;3];  %% mean for class 1
sx1=2.8;  %% standard deviation  class 1
sy1=0.4;
m2=[2;4];  %% mean for class 2
sx2=2.7;  %% standard deviation  class 2
sy2=0.4;

% create 2–D data matrices
for k=1:5;
   x1(:,k)=m1+[sx1;sy1].*rand(2,1);  % create  class 1
   x2(:,k)=m2+[sx2;sy2].*rand(2,1);  % create class 2
end

%% use of PCA
x12=[x1,x2];
[W1,m1,Amean1,EVA1]=pca(x12,k);

%% center each class==>subtract the mean vector of each class
[m,n]=size(x1);
mx1=x1–mean(x1,2)*ones(1,n);
mx2=x2–mean(x2,2)*ones(1,n);

%% 1 D Projection
px11=W1(:,1)'*x1;
px12=W1(:,1)'*x2;

figure
whitebg('w');
plot([–10,10]*W1(1,1),[–10,10]*W1(2,1));grid on;hold on;
scatter( x1(1,:) , x1(2,:) ); grid on;
% axis([–10,10,–10,10]);
```

```
axis([–2,6,–4,6]);
hold on;
scatter( x2(1,:) , x2(2,:), '*' );
hold on;
% plot the 1–D projections
scatter(px11*W1(1,1),px11*W1(2,1));
scatter(px12*W1(1,1),px12*W1(2,1),'*');

%% use of LDA
C=[1,1,1,1,1,2,2,2,2,2];
[W2,D]=fld(x12,C);

%% find projected data (LDA eigenspace)
px21=W2'*x1;
px22=W2'*x2;

% plot the LDA projection line
plot([–10,10]*W2(1),[–10,10]*W2(2),'g—');grid on;hold on;
hold on;
% plot the 1–D projections
scatter(px21*W2(1),px21*W2(2),'g');
scatter(px22*W2(1),px22*W2(2),'g*');
legend('PCA','LDA' );
title('PCA vs LDA Projections');xlabel('x');ylabel('y');
```

- ***DistanceSelection.m:***

```
%% ****************************************************************
%% Filename:          DistanceSelection.m
%% Thesis Advisor:   Prof.M.P.Fargues   Naval Postgraduate School
%% Author:            Captain Dimitrios I. Domboulas
%%                    Hellenic Air Force
%% Date:              July 2004
%% Description:       Produces 3–dimensional distance map plots and associated con–
%%                    tours for various types of distances between
%%                    training/testing images and class centroids.
%%                    Also produces the 3–dimensional confidence score plots/contours.
%% Parameters:        1. Kernel function type (Gaussian, Polynomial)
%%                        (degree of polynomial kernel)
%%                    2. Number of K matrix eigenvectors kept
%%                    3. Distance type to be used
%%                    4. Number of iterations
%% Outputs:           1. Respective 3–dimensional plots
%%                    2. Respective contours.
%% Functions used:   buildGDA_Opt.m,  eigensystem.m,  dataST.m,
%%                    spreadGDA_Opt.m, KernelFunction.m
%% ************************************************************

clc;
clear;
load A_all;
clear  img A Db Dme x ans img_name N_class j k  m  n  s  s1  sqrs1  tem1 tem2 time;
clear  IndSamples  N_objects  Section  im_num1  im_num2 Person T  v  w;
Atemp = double(A_all);
clear  A_all;

%% CONSTANTS ' DECLARATION
Train_Class_Size = 18; %% Number of Training Images per Class
Test_Class_Size = 12;  %% Number of Testing Images per Class
Num_Train_Imag = 900; %% Total Number of Training Images
Num_Test_Imag = 600;  %% Total Number of Testing Images
Num_Class = 50; %% Total Number of Classes
cs = [18,18,18,18,18,18,18,18,18,18];
Class_Sizes = [cs, cs, cs, cs, cs]; %%(1 X 50) vector, with class sizes per class
degree = 2;  %%% select Polynomial kernel degree, USE 0 ==> for GAUSSIAN kernel
ONLY
Num_ev = 0; %%% input Number of K matrix eigenvectors  (0 => Default)
select_dist = 2;  %%%  select distance
Num_Iter = 1;   %%% Number of MAIN LOOP ITERATIONS

for loop = 1 : Num_Iter;  %%%%%  MAIN ITERATION LOOP  %%%%%%%%%
tic %% start timing loops

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% Random Permutation Section Borrowed from [Lee, 2004] and modified by
%%%% [DOMBOULAS, 2004]
    A       =   Atemp; %% A contains the Training Images (2700x900)
    B       =   [];    %% B contains the testing images  (2700x600)
    Delete  =   [];
    h = waitbar(0, 'Random Permutation');  %%%%  WAITBAR
  for i=1 : 150  % i goes from 1 to the total number of pictures
    waitbar( i/150 );                    %%%%  WAITBAR
    rp = randperm(10);
    L1 = (i–1) * 10 + 1;
    L2 = i*10;
    A( : , L1 : L2 ) = A( : , (i–1) * 10 + rp ); %% permute images
    IndSamples = [L1 L1+1 L1+2 L1+3];
    B = [B A( : , IndSamples )];
    Delete = [Delete IndSamples];
 end
    close(h);                        %%%%  WAITBAR
 A( : , Delete ) = [];
```

```
%%%% Section Borrowed from [Lee, 2004] and modified by [Domboulas, 2004]
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

A = DataSt( A ); %% Mean =0, Standard Deviation = 1
B = DataSt( B ); %% Mean =0, Standard Deviation = 1
Le = A'; %%% Learning Data (900 Images are rows)
Te = B'; %% 600 Testing images of our Data Base (Images are rows)
[dataGDA,centeredkM,kM] = buildGDA_Opt( Le , Class_Sizes , degree, Num_ev );
%%run GDA
Pgda_A = spreadGDA_Opt( Le, Le, dataGDA, degree ); %% Projected Training images
in GDA
Pgda_B = spreadGDA_Opt( Te, Le, dataGDA, degree ); %% Projected Testing images in
GDA

%%% FIND VARIANCE of each column of the 900X49 Projected Train Data Matrix.
%%% Covariance is used for other Image Distance types
for k = 1 : size( Pgda_A , 2 );
Pr_Tr_Dat_Cov_Col(k) = cov( Pgda_A(:,k) );  %% Row Vector 1X49 with variances
%%%  of each of the 49 columns of the Projected Training Data Matrix 900X49
end;
Pr_Tr_Dat_Cov_Mat = cov(Pgda_A); %% 49X49 covariance matrix of Proj Train Data
Matrix
%%%%%% It is Diagonal!!Invertible & Each Dimension 's features are
%%%%%% independent from each othert ==> It is the Ideal case
Inv_Cov = inv( Pr_Tr_Dat_Cov_Mat );  %%% Inverse Covar Matr
%%% For the Approximated Mahalanobis Distance
for k = 1 : size( Inv_Cov,2 );
   s_inv(k) = Inv_Cov(k,k); %% ROW VECTOR 1 x 49
end;
sr_s_inv = sqrt(s_inv(k));  %% Row Vector
sr_inv_diag = diag( sr_s_inv ); %%Diagonal Matrix with diag elem the sq roots of
%%% the inverse diag elem of cov matrix  49 x 49
%%%%%%%%%%%%   END VARIANCE SECTION  %%%%%%%%%%%%

%% Find Centroids per Each Class Projected Training Images
for k= 1 : Num_Class
ProjTrainCentr( k , : ) = mean( Pgda_A( ((k–1)*Train_Class_Size + 1) : ( k *
Train_Class_Size ) , : ) );
end;

%%% create cell array
DistName = {'Norm–2             ';'Mahalanobis        ';
   'Mahalanobis Norm–2 ';'Mahalanobis Norm–1 ';'Mahalanobis Angular'};
%%  Find Various Distances of Each Training Image from all the Classes ' Centroids,
%% (TrainDist 900 X 50)
for k = 1 : Num_Train_Imag ;
```

```
   for m = 1: Num_Class;

     if select_dist == 1;  %% NORM–2 Distance
     TrainDist(k,m) = norm( Pgda_A(k,:) – ProjTrainCentr(m,:),2  );

     elseif select_dist == 2; %% Mahalanobis Distance
     TrainDist(k,m) = sqrt( (Pgda_A(k,:) – ProjTrainCentr(m,:)) * Inv_Cov *
(Pgda_A(k,:) – ProjTrainCentr(m,:))' );

     elseif select_dist == 3; %% Mahalanobis NORM–2 Distance
     temp = ( sr_s_inv .* ( Pgda_A(k,:) – ProjTrainCentr(m,:) ) ) * ( sr_s_inv .* (
Pgda_A(k,:) – ProjTrainCentr(m,:) ) )' ;
     TrainDist(k,m) = sqrt( sum( temp ) ); %% Approxim Mahalan Dist

     elseif select_dist == 4;  %% Mahalanobis NORM–1 Distance
     TrainDist(k,m)=sr_s_inv .* norm( Pgda_A(k,:) – ProjTrainCentr(m,:),1 );

     else       %%%% MAHALANOBIS ANGULAR DISTANCE
     TrainDist(k,m) = acos( ( ( Pgda_A(k,:) * sr_inv_diag ) * ( sr_inv_diag * ProjTrain-
Centr(m,:)' ) ) / ( norm( Pgda_A(k,:).*sr_s_inv ,2 ) * norm( ProjTrain-
Centr(m,:).*sr_s_inv ,2 ) ) );

     end;  %%% end if
   end;
end;

%%%  Plot in 3–D the 900 Projected Training Images ' Distances from the
%%%  Classes '  Centroids
for n=1:Num_Train_Imag  %% Create Training Images ' Indeces for the “mesh” function
   Pgda_AIndex(n)=n;
end;
for d=1:Num_Class  %% Create Classes ' Indeces for the “mesh” function
   Class_Index(d) = d;
end;
figure;
mesh(Class_Index,Pgda_AIndex,TrainDist);
xlabel('Class Index');ylabel('Training Image Index');zlabel('Distance');
title( DistName(select_dist) ); %%% Num_Eigv(select_file)
figure;
mesh(Class_Index,Pgda_AIndex,TrainDist);
xlabel('Class Index');ylabel('Training Image Index');zlabel('Distance');
title( DistName(select_dist) ); %%% Num_Eigv(select_file)
view(0,90);  %% view 3–D plot from top
axis([1,50,1,900]);

%%% Find Various Distances for Testing Images
```

```
for k = 1 : Num_Test_Imag ;
   for m = 1: Num_Class;

   if select_dist == 1;    %% NORM–2 Distance
   TestDist(k,m) = norm( Pgda_B(k,:) – ProjTrainCentr(m,:),2  );

   elseif select_dist == 2;  %% MAHALANOBIS DISTANCE
   TestDist(k,m) = sqrt( (Pgda_B(k,:) – ProjTrainCentr(m,:)) * Inv_Cov * (Pgda_B(k,:) –
ProjTrainCentr(m,:))' ); %% Mahalanobis Distance

   elseif select_dist == 3; %% Mahalanobis NORM–2 Distance
   temp = ( sr_s_inv .* ( Pgda_B(k,:) – ProjTrainCentr(m,:) ) ) * ( sr_s_inv .* (
Pgda_B(k,:) – ProjTrainCentr(m,:) ) )' ;
   TestDist(k,m) = sqrt( sum( temp ) ); %% Approxim Mahalan Dist

   elseif select_dist == 4; %% Mahalanobis NORM–1 Distance
   TestDist(k,m)=sr_s_inv .* norm( Pgda_B(k,:) – ProjTrainCentr(m,:),1 );

   else  %%%% MAHALANOBIS ANGULAR DISTANCE
   TestDist(k,m) = acos( ( ( Pgda_B(k,:) * sr_inv_diag ) * ( sr_inv_diag * ProjTrain-
Centr(m,:)' ) ) / ( norm( Pgda_B(k,:).*sr_s_inv ,2 ) * norm( ProjTrain-
Centr(m,:).*sr_s_inv ,2 ) ) );

   end;  %%% end if

  end
end

%%%  Plot in 3–D the 600 Projected Testing Images ' Distances from the
%%%  Classes '  Centroids
for n=1:Num_Test_Imag  %%Create Testing Images' Indeces for "mesh" function
   Pgda_BIndex(n)=n;
end;
for d=1:Num_Class  %% Create Classes ' Indeces for the "mesh" function
   Class_Index(d) = d;
end;
figure;
mesh(Class_Index,Pgda_BIndex,TestDist);
xlabel('Class Index');ylabel('Testing Image Index');zlabel('Distance');
title( DistName(select_dist) ); %%% Num_Eigv(select_file)
figure;
mesh(Class_Index,Pgda_BIndex,TestDist);
xlabel('Class Index');ylabel('Testing Image Index');zlabel('Distance');
title( DistName(select_dist) ); %%% Num_Eigv(select_file)
view(0,90);  %% view 3–D plot from top
axis([1,50,1,600]);
```

```
%%% Assign 1–50 Class Indeces for each of the 600 test Images
T_B = [1,1,1,1,1,1,1,1,1,1,1,1];
for m = 1 : Num_Class;
T_B( ( Test_Class_Size * (m–1) + 1 ) : ( Test_Class_Size * m )  ) = m * T_B(
1:Test_Class_Size) ;
end; %%% T_B contains the class number of each testing image (1X600)

%%% Sort Distances per Testing  Image – RANK
for k = 1 : Num_Test_Imag ;
temp1 = sort( TestDist(k,:) );
I(k) = find(  (temp1 == TestDist(k,T_B(k))*ones(1,Num_Class)) );
end;

%%% Performance Measurement Per Class
for  k = 1 : Num_Class;
   temp2 = find( I( ((k–1)*Test_Class_Size + 1) : ( k * Test_Class_Size ) ) == ones(
1,Test_Class_Size ) );
   Perf_Class(k) = size( temp2 , 2 ) / Test_Class_Size;
end;
figure;
stem(Perf_Class);grid on;
title('Classification Performance per Class');
xlabel('Class Index');ylabel('Classification Performance');

%%% Find the probability of each of the 50 Ranks assigned to all the testing images
for k =1 : Num_Class ;
   temp3 = find( I == k * ones( 1,Num_Test_Imag ) );
   Prob(k) = size( temp3 , 2 ) / Num_Test_Imag;
end;

%%%%  RANK SCORE Evaluation – Cummulative Probability of Ranks – Incresing
Order
Rank(1) = Prob(1);
for k = 2 : Num_Class ;
   Rank( k ) = Rank( k–1 ) + Prob( k );  %%Rank is 1X50 with accumulated Probability
end;
figure;
stem(Rank); grid on; axis([0,52,0.8,1.1]);
title('Cummulative Rank Score for 600 Testing Images');
xlabel('Rank Index');ylabel('Cumulative Rank Score');

if select_dist == 5; %%% Confidence Score ONLY for Mahal Angular distance
%%% Find Confidence Score for each Test Image. This will be 1X50 vector per
%%% image, which will assign the confidence assignment of the test image to each class.
mm=2; %%% parameter for confidense calculation
```

```
for j =  1 : Num_Test_Imag ;
   for k = 1 : Num_Class ;
      %% d has the distances of proj test imag j from all the proj class centroids
% %    d(k,j) = norm( Pgda_B(j,:) – ProjTrainCentr(k,:) , 2 ); %%%% ONLY for
NORM–2 distance
      %%%% ONLY for Mahalanobis Angular Distance
      d(j,k) = acos( ( ( Pgda_B(j,:) * sr_inv_diag ) * ( sr_inv_diag * ProjTrainCentr(k,:)' ) )
/ ( norm( Pgda_B(j,:).*sr_s_inv ,2 ) * norm( ProjTrainCentr(k,:).*sr_s_inv ,2 ) ) );
   end
   dd=d'; %%% ONLY For Mahalanobis Angular
   for i = 1 : Num_Class ;
      %% u has the confidence of assigning test imag j to class k
      u(i,j) = 1 / sum( ( dd(i,j) * ones(Num_Class,1) ./ dd(:,j) ) .^(2/(mm–1)) );   %%%%
NORM–2  CASE
   end
end
uu=u';  %%% ONLY For Mahalanobis Angular

%%%  Plot in 3–D the 600 Projected Testing Images ' Distances from the
%%%  Classes '  Centroids
for n = 1 : Num_Test_Imag; %%Create Testing Images' Indeces for the “mesh” function
   Pgda_BIndex(n) = n;
end;
for d = 1 : Num_Class; %% Create Classes ' Indeces for the “mesh” function
   Class_Index(d) = d;
end;
figure;
mesh(Class_Index,Pgda_BIndex,uu);
xlabel('Class Index');ylabel('Testing Image Index');zlabel('Score');
title( 'Testing Images Confidence Score' ); %%% Num_Eigv(select_file)
axis([1,50,1,600,0,1]);
figure;
mesh(Class_Index,Pgda_BIndex,uu);
xlabel('Class Index');ylabel('Testing Image Index');zlabel('Score');
title( 'Testing Images Confidence Score' ); %%% Num_Eigv(select_file)
axis([1,50,1,600,0,1]);
view(0,90);  %% view 3–D plot from top
end; %%% end if

Rank_718(loop,:) = Rank;
Perf_Class_718(loop,:) = Perf_Class;

clear  A  TA  B  TB  f  L1  L2  Le  Te  dataGDA  centeredkM  kM;
clear  Pgda_A  Pgda_B  k  j  Pr_Tr_Dat_Cov_Col  Pr_Tr_Dat_Cov_Mat;
clear  TrainDist  TestDist temp  Pgda_Aindex  Class_Index;
clear  Inv_Cov  s_inv  sr_inv_diag   ProjTrainCentr  DistName;
clear  Pgda_BIndex  Class_Index  T_B  temp1  I  temp2  temp3;
clear  Prob  Delete  d  dim1  dim2  h  i  m  n sr_s_inv;
clear  Rank  Perf_Class  Pgda_AIndex TestDist cs rp;

time(loop) = toc; %% Stop timing each Main Iteration Loop
end;  %%%%%%  END OF MAIN ITERATION LOOP  %%%%%%%%%%%%%
```

- ***buildGDA_Opt.m:***

```
%% ***************************************************************
%% Filename:          buildGDA_Opt.m
%%                    Thesis Advisor:   Prof.M.P.Fargues   Naval Postgraduate School
%% Author:            G. Baudat, F. Anouar (“buildGDA.m”–21 October 2000)
%% Modified by:       Captain Dimitrios I. Domboulas
%%                    Hellenic Air Force
%% Date modified:     July 2004
%% Description:       Creates the GDA data from the input training
%%                    data.
%% Inputs:            1. Normalized training data matrix (m=0,stdev=1)
%%                    2. Vector with number of element vectors per class
%%                    3. Kernel Function type and degree
%%                    4. Number of Kernel matrix K eigenvectors kept
%% Outputs:           1. GDA data
%%                    2. Centered K matrix
%%                    3. Uncentered K matrix
%% Functions used:    eigensystem.m, KernelFunction.m
%% ***************************************************************

%%% Modified 041115
function [dataGDA,centeredkM,kM]=BuildGDA_Opt(L,S,degree,Num_ev)

n=length(S);
m=length(L(:,1));
mM=ones(m)/m;

%%% Create kernel matrix (uncentered)
if degree == 0; %%% ADDED by D. Domboulas
%%%% Added waitbars by D. Domboulas for code optimization
kM=zeros(m);
h=waitbar(0,'Create Matrix K');               %%%%%% WAITBAR
for i=1:m
  waitbar(i/m,h);                             %%%%%% WAITBAR
   for j=1:m
     kM(i,j)=KernelFunction(L(i,:),L(j,:));
  end
end
```

```
close(h);                                    %%%%%% WAITBAR

%%% ADDED by D.DOMBOULAS FOR POLYNOMIAL KERNEL FUNCTIONS
ONLY
else
kM = ( L * L' ).^degree; %%%Matrix Vectorization
end;  %%% end if
%%% ADDED by D.DOMBOULAS FOR POLYNOMIAL KERNEL FUNCTIONS
ONLY

sumK = mM * kM;

% center the kernel matrix
centeredkM=kM–sumK'–sumK+sumK*mM;
clear mM;

% decomposition of the centered kernel for beta eigenvectors
[vecCkM,valCkM]=eigensystem(centeredkM);

%%%%%%%%%%% Plot K matrix Eigenvalues ==> ADDED by D.DOMBOULAS
for k=1:size(valCkM)
  ValK(k)=valCkM(k,k);
end
ValK(1:50);
figure;plot(ValK);grid on;
xlabel('Eigenvalue Index');ylabel('Eigenvalue Magnitude');
title('GDA K Matrix Eigenvalues');
%%%%%%%%%%%  Plot K matrix Eigenvalues ==> ADDED by D.DOMBOULAS

minVal=valCkM(1,1)/1000;  %define the lowest eigen value used

if Num_ev == 0
% % rankValCkM=250;  %%% Test Reliability of Results
rankValCkM=m; %%% Default numbre of eigvectors kept.

%%%%%%  ADDED by D. Domboulas
else
rankValCkM = Num_ev; %%% keep this number of eigvectors
end
%%%%%%% ADDED by D. Domboulas

h=waitbar(0,'Center K Matrix with Highest Eigvalues'); %%%% WAITBAR
for i=1:m
   waitbar(i/m,h);                           %%%% WAITBAR
   if valCkM(i,i)<minVal;
     valCkM(i,i)=0;
```

```
    rankValCkM=rankValCkM–1;
  end
end
close(h);                                         %%%% WAITBAR

valCkM=valCkM(1:rankValCkM,1:rankValCkM);
vecCkM=vecCkM(:,1:rankValCkM);
%%%centered K matrix with only the highest eigen values/vectors
centeredkM=vecCkM*valCkM*vecCkM';

%w matrix of the weights, bloc diagonal
wM=zeros(m);
stBloc=1;
endBloc=0;
h=waitbar(0,'Create Matrix W');                   %%%% WAITBAR
for i=1:n
   waitbar(i/n,h);                                %%%% WAITBAR
   endBloc=endBloc+S(i);
  for j=stBloc:endBloc
    for k=stBloc:endBloc
      wM(j,k)=1/S(i);
    end
  end
  stBloc=stBloc+S(i);
end
close(h);                                         %%%% WAITBAR

%compute alpha normalized vectors
nbAxes=min([rankValCkM],[n–1]);
[vecBeta,valBeta]=eigensystem(vecCkM'*wM*vecCkM);
tmp=zeros(1,nbAxes);

h=waitbar(0,'Create TEMP Matrix');                %%%% WAITBAR
for i=1:nbAxes ;
   waitbar(i/nbAxes,h);                           %%%% WAITBAR
   tmp(i)=valBeta(i,i);
end
close(h);                                         %%%% WAITBAR

% fprintf('Inertia: %f\n',tmp);
tmp=vecCkM*inv(valCkM)*vecBeta;
norAlpha=zeros(m,nbAxes);

h=waitbar(0,'Create norAlpha matrix');            %%%% WAITBAR
for i=1:nbAxes ;
   waitbar(i/nbAxes,h);                           %%%% WAITBAR
   norAlpha(:,i)=tmp(:,i)/sqrt(tmp(:,i)'*centeredkM*tmp(:,i));
end
close(h);                                       %%%% WAITBAR

sumK=sumK(1,:);
dataGDA=struct('norAlpha',norAlpha,'sumK',sumK);
```

- ***spreadGDA_Opt.m:***

```
%% ***************************************************************
%% Filename:           spreadGDA_Opt.m
%% Thesis Advisor:   Prof.M.P.Fargues   Naval Postgraduate School
%% Author:              G. Baudat, F. Anouar ("buildGDA.m"–21 October 2000)
%% Modified by:        Captain Dimitrios I. Domboulas
%%                          Hellenic Air Force
%% Date modified:     July 2004
%% Description:         Projects the input GDA data on to Kernel matrix
%%                          K eigenvectors.
%% Inputs:               1. Normalized testing data (m=0, stdev=1)
%%                              to be projected on the GDA data eigvectors
%%                          2. Normalized training data (m=0, stdev=1),
%%                              which is used to build the GDA data
%%                          3. GDA data
%%                          4. Kernel Function type and degree
%% Outputs:              Projected data on the Kernel matr K eigvectors
%% Functions used:     KernelFunction.m
%% ***************************************************************

function projectedVectors=SpreadGDA_Opt(T,L,dataGDA, degree);
n=length(T(:,1));        %size of the test set
m=length(L(:,1));       %size of the learning set
KernelEva=zeros(n,m);

%%% Waitbars added by D. Domboulas for code optimization
if degree == 0;  %%% ADDED by D. Domboulas
h=waitbar(0,'Center K Matrix for SPREADGDA'); %%%% WAITBAR
for i=1:n
waitbar(i/n,h);                         %%%% WAITBAR
   tmp=zeros(1,m);
  for j=1:m
     tmp(j)=KernelFunction(T(i,:),L(j,:));
  end
  KernelEva(i,:)=tmp–sum(tmp)/m;
end
close(h);                             %%%% WAITBAR
```

```
%%%%%%%%%%% ADDED by D. DOMBOULAS for Matrix Vectorization
%%%%%%%%%%% ONLY for POLYNOMIAL KERNELS
else
temp1 = (T*L').^degree;  %%% Matrix Vectorization
KernelEva = temp1 – mean(temp1,2) * ones(1,m); %% Remove Mean
end; %%% end if
%%%%%%%%%%% ADDED by D. DOMBOULAS for Matrix Vectorization
%%%%%%%%%%% ONLY for POLYNOMIAL KERNELS

inter=dataGDA.sumK–sum(dataGDA.sumK)/m;
h=waitbar(0,'  PROJECTION SPREADGDA');        %%%% WAITBAR
for i = 1 : n;
waitbar(i/n,h);                              %%%% WAITBAR
   KernelEva(i,:)=KernelEva(i,:)–inter;
end
projectedVectors=KernelEva*dataGDA.norAlpha;
close(h);                                    %%%% WAITBAR
```

- ***ClassificationPerformance.m:***

```
%% **************************************************************
%% Filename:          ClassificationPerformance.m
%% Thesis Advisor:   Prof.M.P.Fargues   Naval Postgraduate School
%% Author:            Captain Dimitrios I. Domboulas
%%                     Hellenic Air Force
%% Date:              July 2004
%% Description:       Computes classification performance of the GDA
%%                     method, for various kernel functions
%% Parameters:        1. Kernel function type (Polyn degree/Gaussian)
%%                     2. Number of K matrix eigenvectors kept
%%                     3. Distance type to be used
%%                     4. Number of iterations
%% Outputs:           1. Rank–(1–50) Scores (in ".mat" file)
%%                     2. Classification Performance per Class
%%                        (in ".mat" file)
%% Functions used:   buildGDA_Opt.m,  eigensystem.m,  dataST.m,
%%                     spreadGDA_Opt.m, KernelFunction.
%% **************************************************************

clc;
clear;

tic %% start timing 1000 iterations

load A_all;
clear  img A Db Dme x ans img_name N_class j k  m  n  s  s1  sqrs1  tem1 tem2 time;
clear  IndSamples  N_objects  Section  im_num1  im_num2 Person T  v  w;
Atemp = double(A_all);
clear  A_all;

%% CONSTANTS ' DECLARATION
Train_Class_Size = 18; %% Number of Training Images per Class
Test_Class_Size = 12;  %% Number of Testing Images per Class
Num_Train_Imag = 900; %% Total Number of Training Images
Num_Test_Imag = 600;  %% Total Number of Testing Images
Num_Class = 50; %% Total Number of Classes
cs = [18,18,18,18,18,18,18,18,18,18];
Class_Sizes = [cs, cs, cs, cs, cs]; %%(1 X 50) vector, with class sizes per class
degree = 2;  %% select Polynom kernel degree, USE 0 ==> for GAUSSIAN ker ONLY
Num_ev = 0; %%% input Number of K matrix eigenvectors  (0 => Default)
select_dist = 2;  %%%  select distance
Num_Iter = 1;   %%% Number of MAIN LOOP ITERATIONS

h=waitbar(0,'  MAIN ITERATION LOOP');        %%%% WAITBAR

for loop = 1 : Num_Iter;  %%%%%  MAIN ITERATION LOOP
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

waitbar(loop/Num_Iter , h);                            %%%% WAITBAR

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% Random Permutation Section Borrowed from [Lee, 2004] and modified by
%%%% [DOMBOULAS, 2004]
   A       =   Atemp; %% A contains the Training Images (2700x900)
   B       =   [];    %% B contains the testing images  (2700x600)
   Delete  =   [];
% %     h = waitbar(0, 'Random Permutation');  %%%%  WAITBAR
  for i=1 : 150  % i goes from 1 to the total number of pictures
% %     waitbar( i/150 );                    %%%%  WAITBAR
   rp = randperm(10);
   L1 = (i–1) * 10 + 1;
   L2 = i*10;
   A( : , L1 : L2 ) = A( : , (i–1) * 10 + rp ); %% permute images
   Index = [L1 L1+1 L1+2 L1+3];
   B = [B A( : , Index )];
   Delete = [Delete Index];
 end
% %     close(h);                            %%%%  WAITBAR
 A( : , Delete ) = [];
%%%% Section Borrowed from [Lee, 2004] and modified by [Domboulas, 2004]
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
A = DataSt( A ); %% Mean =0, Standard Deviation = 1
B = DataSt( B ); %% Mean =0, Standard Deviation = 1
Le = A'; %%% Learning Data (900 Images are rows)
Te = B'; %% 600 Testing images of our Data Base (Images are rows)
[dataGDA,centeredkM,kM] = buildGDA_Opt( Le , Class_Sizes , degree, Num_ev );
%%run GDA
Pgda_A = spreadGDA_Opt( Le, Le, dataGDA, degree ); %% Projected Training images
in GDA
Pgda_B = spreadGDA_Opt( Te, Le, dataGDA, degree ); %% Projected Testing images in
GDA

%%% FIND VARIANCE of each column of the 900X49 Projected Train Data Matrix.
%%% Covariance is used for other Image Distance types
for k = 1 : size( Pgda_A , 2 );
Pr_Tr_Dat_Cov_Col(k) = cov( Pgda_A(:,k) );  %% Row Vector 1X49 with variances
%%%  of each of the 49 columns of the Projected Training Data Matrix 900X49
end;
Pr_Tr_Dat_Cov_Mat = cov(Pgda_A); %% 49X49 covariance matrix of Proj Train Data
Matrix
%%%%%% It is Diagonal!!Invertible & Each Dimension 's features are
%%%%%% independent from each othert ==> It is the Ideal case
Inv_Cov = inv( Pr_Tr_Dat_Cov_Mat );  %%% Inverse Covar Matr
%%% For the Approximated Mahalanobis Distance
for k = 1 : size( Inv_Cov,2 );
   s_inv(k) = Inv_Cov(k,k); %% ROW VECTOR 1 x 49
end;
sr_s_inv = sqrt(s_inv(k));  %% Row Vector
sr_inv_diag = diag( sr_s_inv ); %%Diagonal Matrix with diag elem the sq roots of
%%% the inverse diag elem of cov matrix  49 x 49
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%END VARIANCE SEC-
TION

%% Find Centroids per Each Class Projected Training Images
for k= 1 : Num_Class
ProjTrainCentr( k , : ) = mean( Pgda_A( ((k–1)*Train_Class_Size + 1) : ( k *
Train_Class_Size ) , : ) );
end;

%%% create cell array
DistName = {'Norm–2              ';'Mahalanobis         ';
   'Mahalanobis Norm–2 ';'Mahalanobis Norm–1 ';'Mahalanobis Angular'};

%%% Find Various Distances for Testing Images
for k = 1 : Num_Test_Imag ;
   for m = 1: Num_Class;
```

```
    if select_dist == 1;    %% NORM–2 Distance
    TestDist(k,m) = norm( Pgda_B(k,:) – ProjTrainCentr(m,:),2  );

    elseif select_dist == 2;  %% MAHALANOBIS DISTANCE
    TestDist(k,m) = sqrt( (Pgda_B(k,:) – ProjTrainCentr(m,:)) * Inv_Cov * (Pgda_B(k,:) –
ProjTrainCentr(m,:))' ); %% Mahalanobis Distance

    elseif select_dist == 3; %% Mahalanobis NORM–2 Distance
    temp = ( sr_s_inv .* ( Pgda_B(k,:) – ProjTrainCentr(m,:) ) ) * ( sr_s_inv .* (
Pgda_B(k,:) – ProjTrainCentr(m,:) ) )' ;
    TestDist(k,m) = sqrt( sum( temp ) ); %% Approxim Mahalan Dist

    elseif select_dist == 4; %% Mahalanobis NORM–1 Distance
    TestDist(k,m)=sr_s_inv .* norm( Pgda_B(k,:) – ProjTrainCentr(m,:),1 );

    else  %%%% MAHALANOBIS ANGULAR DISTANCE
    TestDist(k,m) = acos( ( ( Pgda_B(k,:) * sr_inv_diag ) * ( sr_inv_diag * ProjTrain-
Centr(m,:)' ) ) / ( norm( Pgda_B(k,:).*sr_s_inv ,2 ) * norm( ProjTrain-
Centr(m,:).*sr_s_inv ,2 ) ) );

    end;  %%% end if

  end
end

%%% Assign 1–50 Class Indeces for each of the 600 test Images
T_B = [1,1,1,1,1,1,1,1,1,1,1,1];
for m = 1 : Num_Class;
T_B( ( Test_Class_Size * (m–1) + 1 ) : ( Test_Class_Size * m )  ) = m * T_B(
1:Test_Class_Size) ;
end; %%% T_B contains the class number of each testing image (1X600)

%%% Sort Distances per Testing  Image – RANK
for k = 1 : Num_Test_Imag ;
temp1 = sort( TestDist(k,:) );
I(k) = find(  (temp1 == TestDist(k,T_B(k))*ones(1,Num_Class)) );
end;

%%% Performance Measurement Per Class
for  k = 1 : Num_Class;
    temp2 = find( I( ((k–1)*Test_Class_Size + 1) : ( k * Test_Class_Size ) ) == ones(
1,Test_Class_Size ) );
    Perf_Class(k) = size( temp2 , 2 ) / Test_Class_Size;
end;

%%% Find the probability of each of the 50 Ranks assigned to all the testing images
```

```
for k =1 : Num_Class ;
   temp3 = find( I == k * ones( 1,Num_Test_Imag ) );
   Prob(k) = size( temp3 , 2 ) / Num_Test_Imag;
end;

%%%%  RANK SCORE Evaluation – Cummulative Probability of Ranks – Incresing
Order
Rank(1) = Prob(1);
for k = 2 : Num_Class ;
   Rank( k ) = Rank( k–1 ) + Prob( k );  %% Rank is 1X50 with accumulated Probability
end;

Rank_718(loop,:) = Rank;
Perf_Class_718(loop,:) = Perf_Class;

clear  A  TA  B  TB  f  L1  L2  Le  Te  dataGDA  centeredkM  kM;
clear  Pgda_A  Pgda_B  k  j  Pr_Tr_Dat_Cov_Col  Pr_Tr_Dat_Cov_Mat;
clear  TrainDist  TestDist temp  Pgda_Aindex  Class_Index;
clear  Inv_Cov  s_inv  sr_inv_diag   ProjTrainCentr  DistName;
clear  Pgda_BIndex  Class_Index  T_B  temp1  I  temp2  temp3;
clear  Prob  Delete  d  dim1  dim2   i  m  n sr_s_inv  Index;
clear  Rank  Perf_Class  Pgda_AIndex TestDist cs rp;

% % save  Mah_Ang_Rank_PerfClas_GDA_0_7_040926   Rank_718  Perf_Class_718 ;
save  Mah_Ang_Rank_PerfClas_GDA_2_041116   Rank_718  Perf_Class_718 ;

end;  %%%%%%  END OF MAIN ITERATION LOOP %%%%%%%%%%%%%%
close(h);                              %%%%%%%%%%% WAITBAR
time(loop) = toc; %% Stop timing each Main Iteration Loop
```

- ***ConfidenceInterval.m:***

```
%% ***************************************************************
%% Filename:           ConfidenceInterval.m
%% Thesis Advisor:   Prof.M.P.Fargues   Naval Postgraduate School
%% Author:             Captain Dimitrios I. Domboulas
%%                        Hellenic Air Force
%% Date:                August 2004
%% Description:       Plots the Confidence Interval of the
%%                        rank–1 to rank–5 score means of 5 different
%%                        distances used for classification.
%%                        Also, computes the mean rank–1 score for all
%%                        ".mat" files created by "ClassificationPerformance.m"
%% Input:               ".mat" files created by "ClassificationPerformance.m"
%% Outputs:            1. Confidence Interval plot
%%                        2. Rank–1 score mean for each file.
%% Functions used:   N/A
%% ****************************************************************

clc;
clear;

%%%%%  Execute  GDA Polynomial Deg 2  %%%%%%%%%%%%%%
load Mahalanobis_Rank_040818;
Mah = Rank_718;
av_Mah = mean( Mah(:,1) )

load Norm2_Rank_040817;
N2 = Rank_718;
av_N2 = mean( N2(:,1) )

load Mahalanobis_N2_Rank_040820;
Mah_N2 = Rank_718;
av_Mah_N2 = mean( Mah_N2(:,1) )

load Mahalanobis_N1_Rank_040821;
Mah_N1 = Rank_718;
av_Mah_N1 = mean( Mah_N1(:,1) )

load Mahalanobis_Ang_Rank_040822;
Mah_Ang_1 = Rank_718;
av_Mah_Ang_1 = mean( Mah_Ang_1(:,1) )

%%% It has both Rank & Perf per Class
load Mahalanobis_Ang_Rank_040826;
Mah_Ang_2 = Rank_718;
av_Mah_Ang_2 = mean( Mah_Ang_2(:,1) )
%%%%%%%  GDA  Polynom Deg 2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%    GDA  Polynom  Deg  1
%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Only GDA Pol Deg 1==> LDA
%%% It has both Rank & Perf per Class
load Norm2_Rank_PerfClas_LDA_040827;
N2_GDA1 = Rank_718;
av_N2_GDA1 = mean( N2_GDA1(:,1) )
%%% Only GDA Pol Deg 1==> LDA
%%  It has only Rank
load Mahalanobis_Ang_Rank__LDA_040824;
Mah_Ang_GDA1 = Rank_718;
av_Mah_Ang_GDA1 = mean( Mah_Ang_GDA1(:,1) )
```

```
%%%%  First PCA then GDA Pol deg 1 <==> PCA + LDA
%%% It has both Rank & Perf per Class
load N2_Rank_PerfClas_PCAGDA1_040829;
N2_PCAGDA1 = Rank_718;
av_N2_PCAGDA1 = mean( N2_PCAGDA1(:,1) )

%%%%  First PCA then GDA Pol deg 1 <==> PCA + LDA
%%% It has both Rank & Perf per Class
load Mah_Ang_Rank_PerfClas_PCAGDA1_040901;
Mah_Ang_PCAGDA1 = Rank_718;
av_Mah_Ang_PCAGDA1 = mean( Mah_Ang_PCAGDA1(:,1) )
%%%%%%%%%%%    GDA  Polynom Deg 1
%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%% GDA  Polynom Deg 2  VARIOUS NUMBER EIGENVEC-
TORS
%%%%  GDA Pol deg 2 ==> 100 EigVec
%%% It has both Rank & Perf per Class
load Mah_Ang_Rank_PerfClas_GDA2_100EigVec_040911;
Mah_Ang_GDA2_100EV = Rank_718;
av_Mah_Ang_GDA2_100EV = mean( Mah_Ang_GDA2_100EV(:,1) )

%%%%  GDA Pol deg 2 ==> 150 EigVec
%%% It has both Rank & Perf per Class
load Mah_Ang_Rank_PerfClas_GDA_2_150EigVec_041027;
Mah_Ang_GDA2_150EV = Rank_718;
av_Mah_Ang_GDA2_150EV = mean( Mah_Ang_GDA2_150EV(:,1) )

%%%%  GDA Pol deg 2 ==> 200 EigVec
%%% It has both Rank & Perf per Class
load Mah_Ang_Rank_PerfClas_GDA_2_200EigVec_041026;
Mah_Ang_GDA2_200EV = Rank_718;
av_Mah_Ang_GDA2_200EV = mean( Mah_Ang_GDA2_200EV(:,1) )

%%%%  GDA Pol deg 2 ==> 250 EigVec
%%% It has both Rank & Perf per Class
load Mah_Ang_Rank_PerfClas_GDA_2_250EigVec_041025;
Mah_Ang_GDA2_250EV = Rank_718;
av_Mah_Ang_GDA2_250EV = mean( Mah_Ang_GDA2_250EV(:,1) )

%%%%  GDA Pol deg 2 ==> 500 EigVec
%%% It has both Rank & Perf per Class
load Mah_Ang_Rank_PerfClas_GDA2_500EigVec_040908;
Mah_Ang_GDA2_500EV = Rank_718;
av_Mah_Ang_GDA2_500EV = mean( Mah_Ang_GDA2_500EV(:,1) )
```

```
%%%%%%%%%%%%%%GDA  Polynom Deg 2  VARIOUS NUMBER EIGENVEC-
TORS

%%%%  GDA Pol deg 3
%%% It has both Rank & Perf per Class
load Mah_Ang_Rank_PerfClas_GDA_3_041003;
Mah_Ang_GDA3 = Rank_718;
av_Mah_Ang_GDA3 = mean( Mah_Ang_GDA3(:,1) )
%%%%  GDA Pol deg 3

%%% Find Confidence Intervals of the Rank Means–
%%% Assume GAUSSIAN distr, default CI=95%
%%% of the values of the Ranks
[muhat_M,sigmahat,muci_M,sigmaci] = normfit( Mah );
[muhat_N2,sigmahat,muci_N2,sigmaci] = normfit( N2 );
[muhat_MN2,sigmahat,muci_MN2,sigmaci] = normfit( Mah_N2 );
[muhat_MN1,sigmahat,muci_MN1,sigmaci] = normfit( Mah_N1 );
[muhat_MAn,sigmahat,muci_MAn,sigmaci] = normfit( Mah_Ang_1 );

resol = 10 ;
rank = 10;
for k = 1 : rank;
   CI_M(:,k) = ( linspace( muci_M(1,k) , muci_M(2,k) , resol ) )';
   CI_N2(:,k) = ( linspace( muci_N2(1,k) , muci_N2(2,k) , resol ) )';
   CI_MN2(:,k) = ( linspace( muci_MN2(1,k) , muci_MN2(2,k) , resol ) )';
   CI_MN1(:,k) = ( linspace( muci_MN1(1,k) , muci_MN1(2,k) , resol ) )';
   CI_MAn(:,k) = ( linspace( muci_MAn(1,k) , muci_MAn(2,k) , resol ) )';
end

figure;
% % plot( muhat_MAn , 'r' );
axis([0,25,0.975,1]); grid on; hold on;
title('Confidence Intervals of Means of first 5 Rank Scores');
xlabel('Rank Score Index');ylabel('Rank Score Mean');

%%%%% Create resolution for 95% Confidence Intervals
for k = 1 : resol;
   x1(k)=1; x2(k)=2; x3(k)=3; x4(k)=4; x5(k)=5; x6(k)=6; x7(k)=7; x8(k)=8; x9(k)=9;
x10(k)=10; x11(k)=11;
   x12(k)=12; x13(k)=13; x14(k)=14; x15(k)=15; x16(k)=16; x17(k)=17; x18(k)=18;
x19(k)=19;
   x20(k)=20; x21(k)=21; x22(k)=22; x23(k)=23; x24(k)=24; x25(k)=25;
end;

%%%% Plot 95% Confidence Intervals of first 5 Rank Scores
```

```
plot( 0 , CI_M(1,1), 'b' , 0 , CI_N2(1,1), 'g' , 0 , CI_MN2(1,1), 'r' , 0 , CI_MN1(1,1), 'm' ,
0 , CI_MAn(1,1), 'c'  );
plot( x1 , CI_M(:,1), 'b*' , x6 , CI_M(:,2), 'b*' , x11 , CI_M(:,3), 'b*' , x16 , CI_M(:,4),
'b*' , x21 , CI_M(:,5), 'b*'  );
plot( x2 , CI_N2(:,1), 'g*' , x7 , CI_N2(:,2), 'g*' , x12 , CI_N2(:,3), 'g*' , x17 , CI_N2(:,4),
'g*' , x22 , CI_N2(:,5), 'g*'  );
plot( x3 , CI_MN2(:,1), 'r*' , x8 , CI_MN2(:,2), 'r*' , x13 , CI_MN2(:,3), 'r*' , x18 ,
CI_MN2(:,4), 'r*' , x23 , CI_MN2(:,5), 'r*'  );
plot( x4 , CI_MN1(:,1), 'm*' , x9 , CI_MN1(:,2), 'm*' , x14 , CI_MN1(:,3), 'm*' , x19 ,
CI_MN1(:,4), 'm*' , x24 , CI_MN1(:,5), 'm*'  );
plot( x5 , CI_MAn(:,1), 'c*' , x10 , CI_MAn(:,2), 'c*' , x15 , CI_MAn(:,3), 'c*' , x20 ,
CI_MAn(:,4), 'c*' , x25 , CI_MAn(:,5), 'c*'  );
legend('CI–M', 'CI–N2', 'CI–MN2', 'CI–MN1', 'CI–MAn' )
```

- ***PCAGDA1.m:***

```
%% *************************************************************
%% Filename:           PCAGDA1.m
%% Thesis Advisor:     Prof.M.P.Fargues   Naval Postgraduate School
%% Author:             Captain Dimitrios I. Domboulas
%%                     Hellenic Air Force
%% Date:               August 2004
%% Description:        Computes classification performance for the LDA
%%                     method (i.e. GDA with polynomial kernel deg–1)
%%                     or of the Fisherface method (PCA + GDA–1)
%% Parameters:         1. Method to be used (GDA–1 or Fisherface)
%%                     2. Number of K matrix eigenvectors kept
%%                     3. Distance type to be used
%%                     4. Number of iterations
%% Outputs:            1. Rank–(1–50) Scores
%%                     2. Classification Performance per Class
%% Functions used:     buildGDA_Opt.m,  eigensystem.m,  dataST.m,
%%                     spreadGDA_Opt.m, pca.m,  sortem.m
%% *************************************************************

clc;
clear;

tic %% start timing 1000 iterations

load A_all;
clear  img A Db Dme x ans img_name N_class j k  m  n  s  s1  sqrs1  tem1 tem2;
clear  IndSamples N_objects Section im_num1 im_num2 Person T time v  w;
Atemp = double(A_all);
clear  A_all;
```

```
%% CONSTANTS ' DECLARATION
Train_Class_Size = 18; %% Number of Training Images per Class
Test_Class_Size = 12;  %% Number of Testing Images per Class
Num_Train_Imag = 900; %% Total Number of Training Images
Num_Test_Imag = 600;  %% Total Number of Testing Images
Num_Class = 50; %% Total Number of Classes
cs = [18,18,18,18,18,18,18,18,18,18];
Class_Sizes = [cs, cs, cs, cs, cs]; %%(1X50) vector, with class sizes per class
sel_method = 1; %%%Use 0 for GDA–1 and anything else for Fisherface
degree = 1; %%% ALWAYS Polynomial kernel deg 1 FOR GDA–1==>LDA
Num_ev = 0; %%% input Number of K matrix eigenvectors  (0 => Default)
select_dist = 1;  %%%  select distance
Num_Iter = 1;   %%% Number of MAIN LOOP ITERATIONS

h=waitbar(0,'  MAIN ITERATION LOOP');        %%%% WAITBAR
for loop = 1 : Num_Iter;  %%%%%  MAIN ITERATION LOOP  %%%%%%%%%%
waitbar(loop/Num_Iter , h);                  %%%% WAITBAR

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% Random Permutation Section Borrowed from [Lee, 2004] and modified by
%%%% [DOMBOULAS, 2004].
    A       =   Atemp; %% A contains the Training Images (2700x900)
    B       =   [];    %% B contains the testing images  (2700x600)
    Delete  =   [];
  for i=1 : 150  % i goes from 1 to the total number of pictures
    rp = randperm(10);
    L1 = (i–1) * 10 + 1;
    L2 = i*10;
    A( : , L1 : L2 ) = A( : , (i–1) * 10 + rp ); %% permute images
    Index = [L1 L1+1 L1+2 L1+3];
    B = [B A( : , Index )];
    Delete = [Delete Index];
 end
 A( : , Delete ) = [];
%%%% Section Borrowed from [Lee, 2004] and modified by [DOMBOULAS, 2004].
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%% Run GDA–1 OR PCA first, then GDA–1 (Fisherface) %%%%%%%%%%%
%%%%% Run GDA–1   %%%%%%%
if sel_method == 0;
A = DataSt( A ); %% Normalize Data—> Mean =0, Standard Deviation = 1
B = DataSt( B ); %% Normalize Data—> Mean =0, Standard Deviation = 1
Le = A'; %%% Learning Data (900 Images are rows)
Te = B'; %% 600 Testing images of our Data Base (Images are rows)
[dataGDA,centeredkM,kM] = buildGDA_Opt( Le , Class_Sizes , degree, Num_ev );
%%run GDA
```

```
Pgda_A = spreadGDA_Opt( Le, Le, dataGDA, degree ); %% Projected Training images
in GDA
Pgda_B = spreadGDA_Opt( Te, Le, dataGDA, degree ); %% Projected Testing images in
GDA

else  %%%% Run Fisherface
%%%%%%%%  PCA
[Wpca,m,Amean,Ad] = pca(A);   %%%Apply PCA to IrisData
PApca = Wpca' * (Ad); %% Compute the PCA projection matrix of IrisData
PBpca = Wpca' * (B – Amean*ones(1,Num_Test_Imag));

%%%%% GDA Pol Deg 1 ==> LDA
Le = PApca'; %%% Learning Data (900 Images are rows) (DO NOT NORMALIZE
DATA)
Te = PBpca'; %%% 600 Testing images of our Data Base (Images are rows) (DO NOT
NORMALIZE DATA)
[dataGDA,centeredkM,kM] = buildGDA_Opt( Le , Class_Sizes , degree, Num_ev );
%%run GDA, Polyn Deg 1
Pgda_A = spreadGDA_Opt( Le, Le, dataGDA , degree ); %% Projected Train images in
GDA, Polyn Deg 1
Pgda_B = spreadGDA_Opt( Te, Le, dataGDA , degree ); %% Projected Test images in
GDA, Polyn Deg 1
end; %%% end if
%%%%%%%%%%%%%%%%%%%%%%  PCA LDA   %%%%%%%%%%%%%

%%% FIND VARIANCE of each column of the 900X49 Projected Train Data Matrix.
%%% Covariance is used for other Image Distance types
for k = 1 : size( Pgda_A , 2 );
Pr_Tr_Dat_Cov_Col(k) = cov( Pgda_A(:,k) );  %% Row Vector 1X49 with variances
%%%  of each of the 49 columns of the Projected Training Data Matrix 900X49
end;
%%%% 49X49 covariance matrix of Proj Train Data Matrix
Pr_Tr_Dat_Cov_Mat = cov(Pgda_A);
%%%%%% It is Diagonal!!Invertible & Each Dimension 's features are
%%%%%% independent from each othert ==> It is the Ideal case
Inv_Cov = inv( Pr_Tr_Dat_Cov_Mat );  %%% Inverse Covar Matr
%%% For the Approximated Mahalanobis Distance
for k = 1 : size( Inv_Cov,2 );
   s_inv(k) = Inv_Cov(k,k); %% ROW VECTOR 1 x 49
end;
sr_s_inv = sqrt(s_inv(k));  %% Row Vector
sr_inv_diag = diag( sr_s_inv ); %%Diagonal Matrix with diag elem the
%%% square roots of the inverse diag elem of cov matrix  49 x 49
%%%%%%%%%%%%%   END VARIANCE SECTION   %%%%%%%%%%%

%% Find Centroids per Each Class Projected Training Images
```

```
for k= 1 : Num_Class
ProjTrainCentr( k , : ) = mean( Pgda_A( ((k–1)*Train_Class_Size + 1) : ( k *
Train_Class_Size ) , : ) );
end;

%%% create cell array
DistName = {'Norm–2              ';'Mahalanobis         ';
   'Mahalanobis Norm–2 ';'Mahalanobis Norm–1 ';'Mahalanobis Angular'};

%%% Find Various Distances for Testing Images
for k = 1 : Num_Test_Imag ;
   for m = 1: Num_Class;

   if select_dist == 1;    %% NORM–2 Distance
   TestDist(k,m) = norm( Pgda_B(k,:) – ProjTrainCentr(m,:),2  );

   elseif select_dist == 2;  %% MAHALANOBIS DISTANCE
   TestDist(k,m) = sqrt( (Pgda_B(k,:) – ProjTrainCentr(m,:)) * Inv_Cov * (Pgda_B(k,:) –
ProjTrainCentr(m,:))' ); %% Mahalanobis Distance

   elseif select_dist == 3; %% Mahalanobis NORM–2 Distance
   temp = ( sr_s_inv .* ( Pgda_B(k,:) – ProjTrainCentr(m,:) ) ) * ( sr_s_inv .* (
Pgda_B(k,:) – ProjTrainCentr(m,:) ) )' ;
   TestDist(k,m) = sqrt( sum( temp ) ); %% Approxim Mahalan Dist

   elseif select_dist == 4; %% Mahalanobis NORM–1 Distance
   TestDist(k,m)=sr_s_inv .* norm( Pgda_B(k,:) – ProjTrainCentr(m,:),1 );

   else  %%%% MAHALANOBIS ANGULAR DISTANCE
   TestDist(k,m) = acos( ( ( Pgda_B(k,:) * sr_inv_diag ) * ( sr_inv_diag * ProjTrain-
Centr(m,:)' ) ) / ( norm( Pgda_B(k,:).*sr_s_inv ,2 ) * norm( ProjTrain-
Centr(m,:).*sr_s_inv ,2 ) ) );

   end;  %%% end if

  end
end

%%% Assign 1–50 Class Indeces for each of the 600 test Images
T_B = [1,1,1,1,1,1,1,1,1,1,1,1];
for m = 1 : Num_Class;
T_B( ( Test_Class_Size * (m–1) + 1 ) : ( Test_Class_Size * m )  ) = m * T_B(
1:Test_Class_Size) ;
end; %%% T_B contains the class number of each testing image (1X600)

%%% Sort Distances per Testing  Image – RANK
```

```
for k = 1 : Num_Test_Imag ;
temp1 = sort( TestDist(k,:) );
I(k) = find(  (temp1 == TestDist(k,T_B(k))*ones(1,Num_Class)) );
end;

%%% Performance Measurement Per Class
for  k = 1 : Num_Class;
   temp2 = find( I( ((k–1)*Test_Class_Size + 1) : ( k * Test_Class_Size ) ) == ones(
1,Test_Class_Size ) );
   Perf_Class(k) = size( temp2 , 2 ) / Test_Class_Size;
end;

%%% Find the probability of each of the 50 Ranks assigned to all the testing images
for k =1 : Num_Class ;
   temp3 = find( I == k * ones( 1,Num_Test_Imag ) );
   Prob(k) = size( temp3 , 2 ) / Num_Test_Imag;
end;

%%%%  RANK SCORE Evaluation – Cummulative Probability of Ranks – Increasing
Order
Rank(1) = Prob(1);
for k = 2 : Num_Class ;
   Rank( k ) = Rank( k–1 ) + Prob( k ); %% Rank is 1X50 with accumulated Probability
end;

Rank_718(loop,:) = Rank;
Perf_Class_718(loop,:) = Perf_Class;

clear  A  TA  B  TB  f  L1  L2  Le  Te  dataGDA  centeredkM  kM;
clear  Pgda_A  Pgda_B  k  j  Pr_Tr_Dat_Cov_Col  Pr_Tr_Dat_Cov_Mat;
clear  TrainDist  TestDist temp  Pgda_Aindex  Class_Index;
clear  Inv_Cov  s_inv  sr_inv_diag   ProjTrainCentr  DistName;
clear  Pgda_BIndex  Class_Index  T_B  temp1  I  temp2  temp3;
clear  Prob  Delete  d  dim1  dim2  i  m  n sr_s_inv;
clear  Rank  Perf_Class   Pgda_AIndex TestDist cs rp;

% % save  Mah_Ang_Rank_PerfClas_GDA_0_7_040926   Rank_718  Perf_Class_718 ;
save  Mah_Ang_Rank_PerfClas_GDA_1_041116   Rank_718  Perf_Class_718 ;

end;  %%%%%%  END OF MAIN ITERATION LOOP  %%%%%%%%%%%%
close(h);                    %%%%%%%%%%%%%    WAITBAR
time(loop) = toc; %% Stop timing each Main Iteration Loop
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C. RANK SCORE PLOTS

In this appendix, the following typical random rank score plots are provided. Each was derived from one random iteration of the program "DistanceSelection.m", listed in Appendix B.

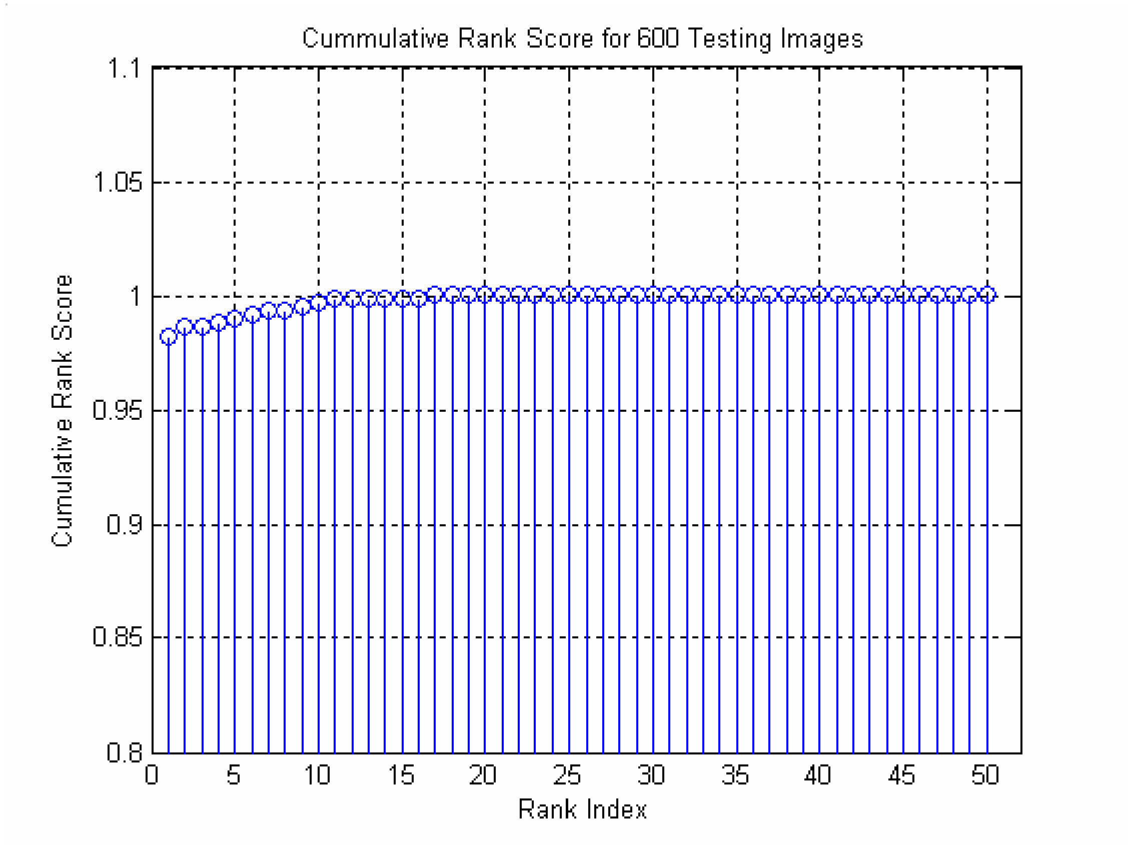


Figure C.1. Rank score plot example 1.

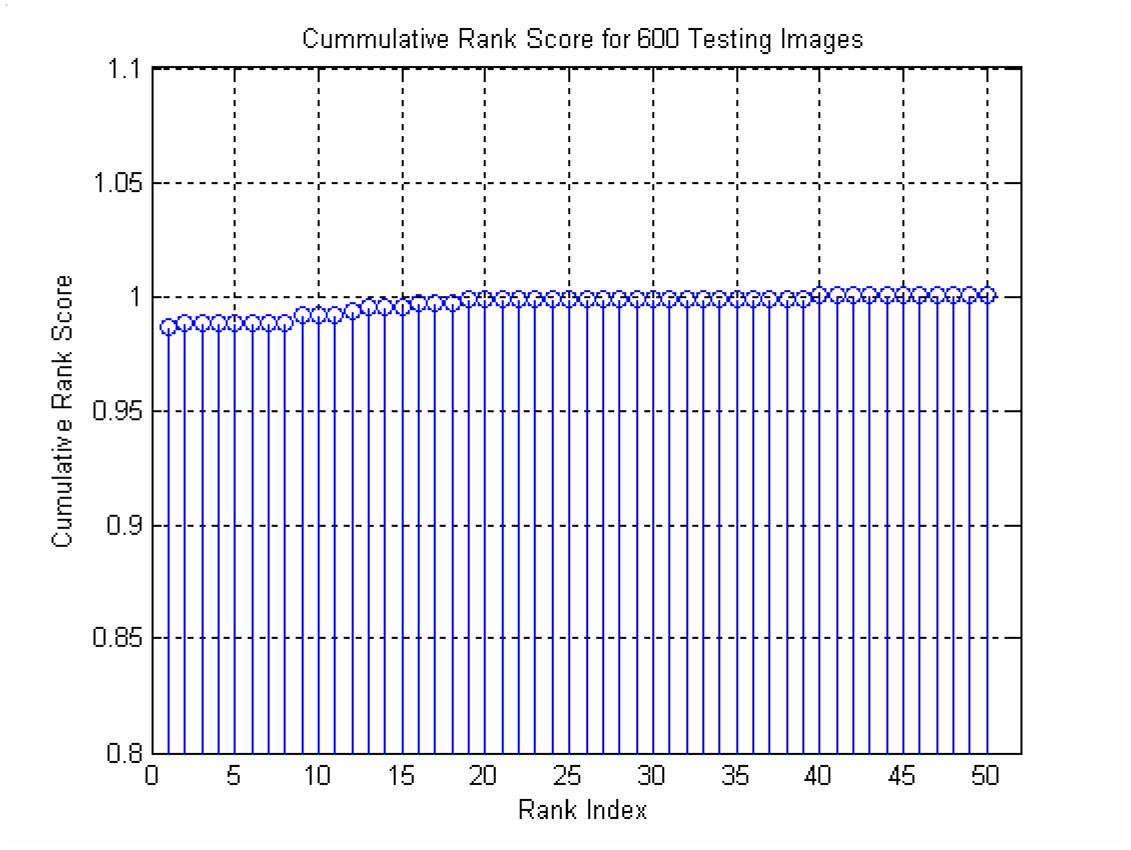


Figure C.2. Rank score plot example 2.

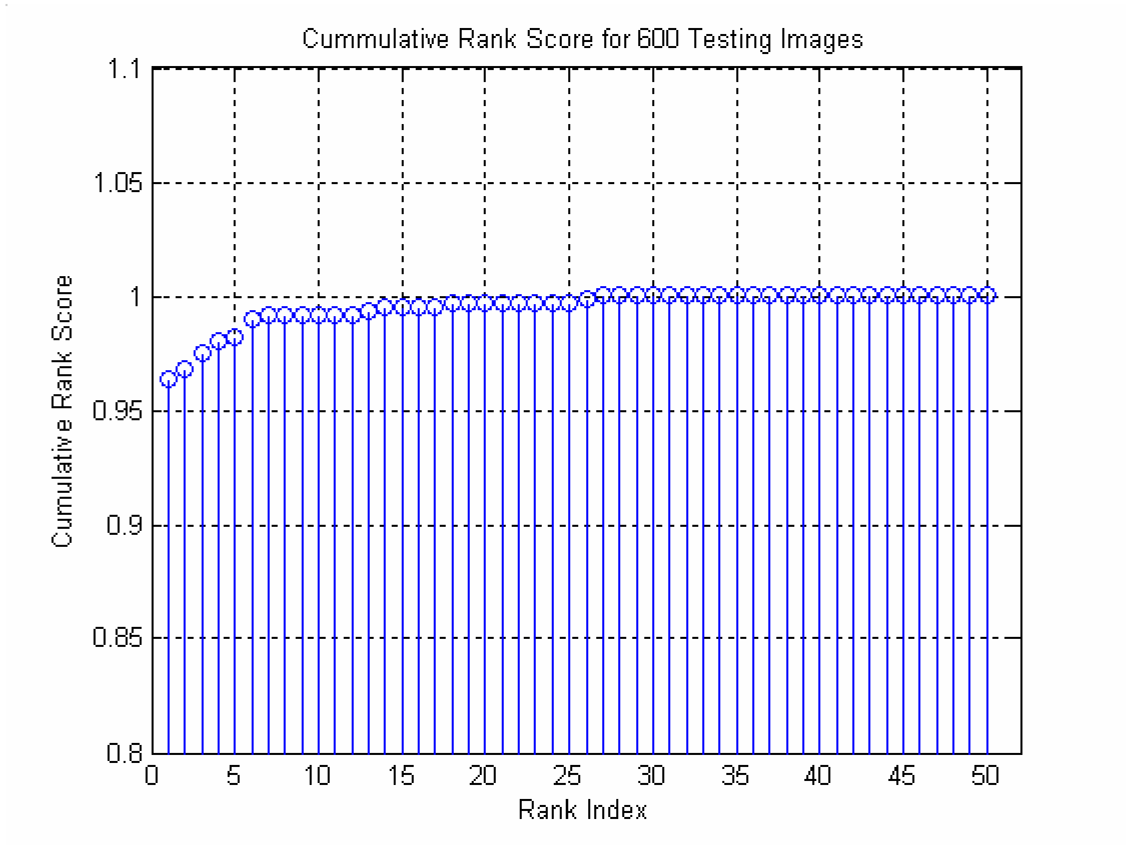


Figure C.3. Rank score plot example 3.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Adini, Y., Moses, Y., and Ulman, S., "Face recognition: the problem of compensating for changes in illumination direction," *European Conference Computer Vision*, Vol. 19, No. 7, pp. 721–732, July 1997.

Aizerman, M. A., Braverman, E. M., and Rosonoer, L. I., "Theoretical foundations of the potential function method in pattern recognition learning," *Automation and Remote Control*, Vol. 25, pp. 821–837, 1964.

Baudat, G., and Anouar, F., "Feature vector selection and projection using kernels," *Neurocomputing*, Vol. 55, No. 1–2, pp. 21–38, 2003.

Baudat, G., and Anouar, F., "Generalized discriminant analysis software," 21 October 2000, [http://www.kernel–machines.org/code/gda.zip], last accessed July 2004.

Baudat, G., and Anouar, F., "Generalized discriminant analysis using a kernel approach," *Neural Computation*, Vol. 12, pp. 2385–2404, 2000.

Belhumeur, P. N., Hespanha, P. J., and Kriegman, D. J., "Eigenfaces vs. Fisherfaces: recognition using class specific linear projection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, No. 7, pp. 711–720, July 1997.

Bishop, C. M., *Neural Networks for Pattern Recognition*, Clarendon Press, Oxford, 1995.

Brown, M. P. S., Grundy, W. N., Lin, D., Cristianini, N., Sugnet, C., Furey, T. S., Ares, M., and Haussler, D., "Knowledge–based analysis of microarray gene expression data using support vector machines," *Proceedings of National Academy of Sciences,* Vol. 97, No. 1, pp. 262–267, January 2000.

Chen, X., Flynn, P. J., and Bowyer, K. W., "PCA–based face recognition in infrared imagery: Baseline and comparative studies," *IEEE International Workshop on Analysis and Modeling of Faces and Gestures*, No. 17, pp. 127–134, October 2003.

Cheng, Y. Q., Zhuang, Y. M., and Yang, J. Y., "Optimal Fisher discriminant analysis using the rank decomposition," *Pattern Recognition*, Vol. 25, No. 1, pp. 101–111, 1992.

Dereniak, E. L., and Boreman, G. D., *Infrared Detectors and Systems*, 1st Edition, Wiley Interscience, New York, 1996.

Duda, R. O., Hart, P. E., and Stork, D. G., *Pattern Classification*, 2nd Edition, Wiley Interscience, New York, 2001.

Fargues, M. P., *Investigation of Feature Dimension Reduction Schemes for Classification Applications*, NPS Technical Report, NPS–EC–01–005, Monterey, California, USA, June 1, 2001.

Gutierrez–Osunak, R., "Introduction to pattern analysis," Texas A&M University, Kingsville, Texas, [http://research.cs.tamu.edu/prism/lectures/pr/pr_l10.pdf], last accessed May 2004.

Hearst, M. A., Dumais, S. T., Osman, E., Platt, J., and Scholkopf, B., "Support vector machines," *IEEE Intelligent Systems*, Vol. 13, No. 4, pp. 18–28, July–August 1998.

Infrared Solutions Inc., "IR–160 thermal imager," [http://www.infraredsolutions.com/images/downloads/ImagerDS.pdf], last accessed November 2004.

Jain, R., *The Art of Computer Systems Performance Analysis*, Wiley Interscience, New York, 1991.

Jang, J. S. R., Sun, C. T., and Mizutani, E., *Neuro–Fuzzy and Soft Computing*, Prentice Hall, Upper Saddle River, New Jersey, 1997.

Joachims, T., "Text categorization with support vector machines: Learning with many relevant features," *Proceedings of the European Conference on Machine Learning*, pp. 137–142, Berlin, 1998.

Jones, B. F., "A reappraisal on the use of infrared thermal image analysis in medicine," *IEEE Transactions on Medical Imaging*, Vol. 17, No. 6, pp. 1019–1027, December 1998.

Kwang, I. K., Keechul, J., and Hang, J. K., "Face recognition using kernel principal component analysis," *IEEE Signal Processing Letters*, Vol. 9, No. 2, pp. 40–42, February 2002.

Lee, C. K., *Infrared Face Recognition*, M.S. Electrical Engineering Thesis, Naval Postgraduate School, Monterey California, June 2004.

Liu, C., "Kernel Fisher linear discriminant with fractional power polynomial models for face recognition," SPIE *Defense and Security Symposium, Conference on Biometric Technology for Human Identification*, Orlando, Florida, April 12–16, 2004.

Liu, C., "Gabor–based kernel PCA with fractional power polynomial models for face recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 26, No. 5, pp. 572–581, May 2004.

Lu, J., Plataniotis, K. N., and Venetsanopoulos, A. N., "Face recognition using kernel direct discriminant analysis algorithms," *IEEE Transactions on Neural Networks*, Vol. 14, No. 1, pp. 117–126, January 2003.

Martinez, A. M., and Kak, A. C., "PCA vs LDA," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 23, No. 2, pp. 228–233, February 2001.

Muller, K. R., Mika, S., Ratsch, G., Tsuda, K., and Scholkopf, B., “An introduction to kernel based learning algorithms,” *IEEE Transactions on Neural Networks*, Vol. 12, No. 2, pp. 181–201, March 2001.

Pereira, D.C., *Face Recognition Using Infrared Imaging*, Electrical Engineer Thesis, Naval Postgraduate School, Monterey, California, December 2002.

Philips, P. J., Rauss, P. J., and Der, S. Z., *FERET (Face Recognition Technology) Recognition Algorithm Development and Test Results*, Technical Report ARL–TR–995, Adelphi, Maryland, October 1996.

Roobaert, D., and Van Hulle, M. M., “View–based 3D object recognition with support vector machines,” *IEEE Proceedings on Neural Networks for Signal Processing Workshop1999*, Madison WI, pp. 77–84, August 1999.

Scholkopf, B., *Statistical Learning and Kernel Methods*, Technical Report MSR–TR–2000–23, February 2000, [ftp://ftp.research.microsoft.com/pub/tr/tr–2000–23.pdf], last accessed October 2004.

Scholkpof, B., Smola, A., and Muller, K. R., “Nonlinear component analysis as a kernel eigenvalue problem,” *Neural Computation*, Vol. 10, pp. 1299–1319, 1998.

Sierra Pacific Corp., “Thermal IR imaging, military FLIR,” [http://www.x20.org/library/thermal/desert_fox.htm], last accessed November 2004.

Socolinsky, D. A., Selinger, A., and Neuheisel, J., “Face recognition with visible and thermal infrared imagery,” *Computer Vision and Image Understanding*, Vol. 91, pp. 72–114, July–August 2003.

Strang, G., *Introduction to Linear Algebra*, 3rd Edition, Wellesley–Cambridge Press, Wellesley Massachusetts, 2003.

Therrien, C. W., *Discrete Random Signals and Statistical Signal Processing*, Prentice Hall, Upper Saddle River, New Jersey, 1992.

Yambor, W. S., *Analysis of PCA Based and Fisher Discriminant – Based Image Recognition Algorithms*, M.S. Computer Science Thesis, Colorado State University, July 2000.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Professor John P. Powers
   Chairman, Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, California

4. Professor Daphne Kapolka
   Department of Physics
   Naval Postgraduate School
   Monterey, California

5. Professor Monique P. Fargues
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, California

6. Professor Roberto Cristi
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, California

7. Professor Gamani Karunasiri
   Department of Physics
   Naval Postgraduate School
   Monterey, California

8. Dimitrios I. Domboulas
   27 Kyvelis Str.
   Larisa 41335
   Greece