Linear Models for Dimensionality Reduction and Statistical Pattern Recognition for Supervised and Unsupervised Learning Tasks

Alok Sharma

BTech, University of the South Pacific, Suva, Fiji

MEng, Griffith University, Brisbane, Australia

March 2006

A Dissertation submitted to the Griffith University in fulfilment of the requirements of the degree of Doctor of Philosophy

Linear Models for Dimensionality Reduction and Statistical Pattern Recognition for Supervised and Unsupervised Learning Tasks

Researcher: Alok Sharma

GUID: S1776645

Principal Supervisor: Prof. Kuldip K. Paliwal, Griffith University, Brisbane, Australia.

Associate Supervisor: Prof. Peter Lisner, Griffith University, Brisbane, Australia.

External Associate Supervisor: Prof. Godfrey C. Onwubolu, University of the South Pacific, Suva, Fiji.

Abstract

In this dissertation a number of novel algorithms for dimension reduction and statistical pattern recognition for both supervised and unsupervised learning tasks have been presented. Several existing pattern classifiers and dimension reduction algorithms are studied. Their limitations and/or weaknesses are considered and accordingly improved techniques are given which overcome several of their shortcomings. In particular, the following research works are carried out:

- Literature survey of basic techniques for pattern classification like Gaussian mixture model (GMM), expectation-maximization (EM) algorithm, minimum distance classifier (MDC), vector quantization (VQ), nearest neighbour (NN) and *k*-nearest neighbour (*k*NN) are conducted.
- Survey of basic dimensional reduction tools viz. principal component analysis (PCA) and linear discriminant analysis (LDA) are conducted. These techniques are also considered for pattern classification purposes.
- Development of Fast PCA technique which finds the desired number of leading eigenvectors with much less computational cost and requires extremely low processing time as compared to the basic PCA model.
- Development of gradient LDA technique which solves the small sample size problem as was not possible by basic LDA technique.
- The rotational LDA technique is developed which efficiently reduces the overlapping of samples between the classes to a large extent as compared to the basic LDA technique.
- A combined classifier using MDC, class-dependent PCA and LDA is designed which improves the performance of the classifier which was not possible by using single classifiers. The application of PCA prior to LDA is conducted in such a way that it avoids small sample size problem (if present).

- The splitting technique initialization is introduced in the local PCA technique. The proposed integration enables easier data processing and more accurate representation of multivariate data.
- A combined technique using VQ and vector quantized principal component analysis (VQPCA) is presented which provides significant improvement in the classifier performance (in terms of accuracy) at very low storage and processing time requirements compared to individual and several other classifiers.
- Survey on unsupervised learning task like independent component analysis (ICA) is conducted.
- A new perspective of subspace ICA (generalized ICA, where all the components need not be independent) is introduced by developing vector kurtosis (an extension of kurtosis) function.

Statement of Originality

This work has not previously been submitted for a degree or diploma in any university. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made in the thesis itself.

Sharma

Alok Sharma

March 2006

Contents

List of Figures

List of Tables

Notation

$\mathbf{x}$	*d*-dimensional feature vector
$\boldsymbol{\mu}$	mean vector
Σ	covariance matrix
$E[\mathbf{x}]$	expectation of feature vector **x**
n	number of samples
$\mathcal{X}$	set of *n* train samples
$\mathcal{Y}$	projected set (usually by PCA) of *n* training samples
$\mathcal{Z}$	projected set (usually by LDA) of *n* training samples
c	number of class
Ω	set of *c* states of nature
ω_i	*i*th class label
$\hat{\mathbf{x}}$	reconstructed feature vector (usually) during PCA transformation
$\mathbf{R}^h$	*h*-dimensional feature space
$\mathbf{R}^d$	d-dimensional feature space (where $h < d$)
$\boldsymbol{\varphi}$	$d \times h$ sized PCA transformation matrix
$\mathbf{W}$	LDA transformation matrix (usually)
S_B	between-class scatter matrix
S_W	within-class scatter matrix
$J(\mathbf{W})$	Fisher's criterion function
$P(\omega_i)$	a priori probability of ω_i
$p(x \mid \omega_i)$	class conditional probability density function
$\boldsymbol{\theta}$	$d \times d$ sized rotational LDA transform
μ_{ref}	reference vector
$\mathbf{A}$	mixing matrix in the ICA technique
kurt	kurtosis function

Acronyms

BSS	Blind source separation
ECG	Electrocardiogram
EM	Expectation-Maximization
EVD	Eigenvalue Decomposition
GMM	Gaussian Mixture Model
IC	Independent Component
ICA	Independent Component Analysis
ISA	Independent Subspace Analysis
KLT	Karhunen-Loéve Transform
*k*NN	*k*-Nearest Neighbour
LCD	Linear Combined Distance
LDA	Linear Discriminant Analysis
MDC	Minimum Distance Classifier
MFCC	Mel Frequency Cepstral Coefficient
MFEAT	Multiple Feature Digit
MICA	Multidimensional Independent Component Analysis
ML	Maximum Likelihood
MSE	Mean Squared Error
NN	Nearest Neighbour
PCA	Principal Component Analysis
TIMIT	Texas Instruments, Massachusetts Institute of Technology
VQ	Vector Quantization
VQPCA	Vector Quantized Principal Component Analysis
VQPCA-sp	VQPCA using Splitting Technique

Acknowledgements

At first I thank the Almighty, *Lord Shiva*, for giving me strength and providing me an opportunity to witness this incredible day when I am writing this final page of my dissertation. Without His blessings nothing could have been possible for me.

Then I would like to convey my sincere gratitude to my principal supervisor Prof. K. Paliwal for giving me specific directions, continuous guidance and encouragement. This has given me enormous support and courage for completion of this paper. In fact I relied on him on many occasions to visit the University and work directly under his supervision as and when so required. His vast knowledge always helped me in exploring the problems and finally implementing the solutions which has given quality productions. I would also like to thank Prof. G. Onwubolu, my external associate supervisor, who has been extremely helpful in regularly encouraging me to do the research work and from time to time has been giving me most needed guidance and inspiration towards the completion of this paper. No doubt, he has been a continuous source of strength to me as a guide and colleague. In fact he has been taking personal interest to see that I am guided properly to achieve the target. In the shortest time available, I have always taken advantage of his wide knowledge in my research and related matters. I would also like to take this opportunity to thank Prof. Peter Lisner, my associate supervisor, who has very kindly supported my research work with whatever materials available when so ever required. He was always available for consultations and friendly guidance that I often required. Certainly, he has been an asset as a University Professor in supporting my research programme. His valuable contribution to my work will always be remembered respectfully. I would also like to thank Prof. Barry Harrison for his kind support over the years. I would also like to sincerely thank Mr. Brett Wildermoth for his kind assistance and guidance, especially at the initial stages of my programme. I would also thank my friend and colleague Dr. R.C. Bansal (the editor IEEE T Energy Conversion) for going through the manuscript and providing constructive comments. I would also like to thank Prof. Erkki Oja (of Helsinki University of Technology) for kind consultation and valuable suggestions. Many thanks to my friend and colleague Mr. Shiven Kumar for his continuous support and suggestions.

I gratefully thank my father Dr. Vivekanand Sharma who has given me enormous moral and spiritual backing, without whom this document would not have been made possible. Many thanks to my youngest sister Vandana Sharma, BSc, MBBS(cont.) who has proof-read several of my papers and has given constructive and crucial comments and suggestions. Also thanks to my other family members.

I would like to note here my special appreciation for the enormous support by the University Research Committee (URC, USP, Fiji), Staff Development Committee (SDC, USP, Fiji) and School of Pure and Applied Sciences Research Committee (SPAS-RC, USP, Fiji) in partially supporting the research work. Thanks to all members of the above mentioned committees, in particular, thanks to Dr. Angela Jokhan (Associate Dean, Faculty of Science & Technology, USP) and Ms. Loraine Tulele (Human Resource Officer).

In concluding I would like to express my grateful thanks to all my colleagues and friends, namely, Kamil Wojicicki, Ben Shannon, Leigh Alsteris, Dr. Stephen So and Nitin Khosla of the Signal Processing Lab and/or the School of Engineering, USP for their direct or indirect assistance towards my research accomplishment.

Linear Models for Dimensionality Reduction and Statistical Pattern Recognition for Supervised and Unsupervised Learning Tasks

Chapter 1

1 Introduction

Humans naturally recognize or classify given objects in their environment. This tendency of recognition or classification is due to some adaptation process in human brain which is a gift of nature to the mankind. We take this adaptation process for granted until we come to feed a machine for the same purpose. A five year old kid can efficiently recognize several objects given to him/her. For the same recognition purpose, even a powerful computer may struggle to give decisions or perform some actions accordingly. This is how we can appreciate the power of human brain which performs several thousand processes in a blink of eye which could be difficult for sophisticated computers or automated systems. Though the performance of machines is still lacking as compared to the human brain, it has several emerging applications in speech recognition, face recognition, forensic science (e.g. fingerprint recognition), banking, multimedia communication, bioinformatics and so on.

Pattern classification plays a major role in everyday life. The evolving computational demand makes this field very challenging and thus open for research. When several multidimensional feature vectors are involved with scarce number, it makes the implementation of the pattern classifier quite impossible. This limitation is usually referred as the *curse of dimensionality*. Efforts are undergoing to reduce the complexity in an efficient manner and at the same time achieve sufficient level of classification accuracy.

Pattern classification can be subdivided into two main categories, namely, supervised learning and unsupervised learning. In supervised learning task the state of the nature of feature vectors are known, which are used to find class models or parameters. These class models (or measured parameters) are stored for the later use in the classification phase for characterization of test features or patterns. Whereas, in unsupervised learning the state of the nature is not known and learning is done by some similarity measurement. Here no training data (known or labelled feature vectors) is given.

The framework of canonical pattern recognizer system (considering the supervised learning case) is illustrated in figure 1.1. The aim of the pattern recognizer is to classify an input data as one of the *c* classes (in a *c*-class problem). This is done in two steps: a feature extraction step and a pattern classification step. In the feature extraction step, the input signal (or data) is analysed and feature vectors are measured where each feature vector has *d* attributes or dimensions. The *d*-dimensional feature vector **x** is processed by parameter measurement block which produced at its output a parameter *T*. This parameter could be a type of vector, matrix, some transformation, or a combination of all. The type of output depends upon the type of pattern classifier used. The measured parameter or class model *T* is stored for the later use for the classification phase. In the classification phase again the raw data or signal is processed through a feature extractor block and provides a test pattern or feature vector at its output of dimension *d*. This test vector **x** is processed by a distance measurement block where the stored class models are utilized to compute distances (or probabilities/likelihood) δ. The measured distance δ is then processed by the comparison block which uses minimization/maximization criterion (which depends on the classifier used) for associating a class label to the test pattern **x**. The class that satisfies the extremum criterion for **x** is considered to be the recognized class.

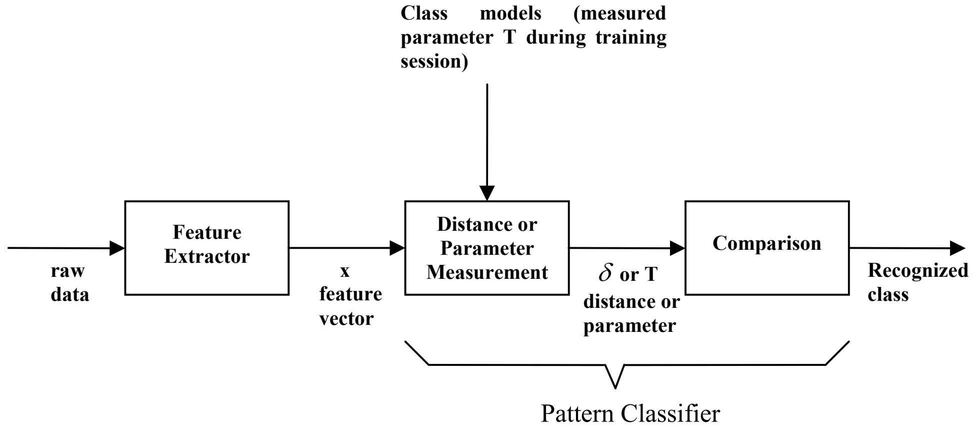

Class models (measured parameter T during training session)
Feature Extractor
raw data
x feature vector
Distance or Parameter Measurement
δ or T distance or parameter
Comparison
Recognized class
Pattern Classifier

Figure 1.1: Framework of canonical pattern recognition system

1.1 Organization of Thesis

In this dissertation the pattern classifier section of the recognition system has been investigated in detail. Several machine learning corpuses (with previously extracted features) and state-of-the-art feature extractors are incorporated in the study. The dissertation concentrates more on the supervised learning task, however, some unsupervised learning tasks like independent component analysis (ICA) have also been explored. The pattern classification methods of several pattern classifiers are analysed and their constraints/limitations, weaknesses and strengths have been discussed. The novel algorithms are developed with improved performances as compared to the existing techniques. Particularly, the following things have been presented in the respective sections:

Chapter 2: This section reviews Gaussian mixture model (GMM) for pattern classification. A description of pattern classification is given for supervised learning tasks. Then expectation-maximization (EM) algorithm is briefly presented. Some of the weaknesses of GMM are also discussed. Then the basic linear classifiers namely minimum distance classifier (MDC), vector quantization (VQ), nearest neighbour (NN) and *k*-nearest neighbour (*k*NN) are discussed. Their training and classification phases are illustrated to understand the functionality of the techniques. Furthermore, two basic dimension reduction and/or pattern classification techniques namely principal component analysis (PCA) and linear discriminant analysis (LDA) are discussed. Their characteristics including drawbacks are briefly illustrated.

Chapter 3: The basic PCA (eigenvalue decomposition based PCA) method is very time consuming when dealing with high dimensional feature vectors. This restricts the practical applications of such technique. In this section an efficient method of computing principal component analysis (PCA) is presented that overcomes the drawbacks of basic PCA method. The algorithm finds the desired number of leading eigenvectors with a computational cost that is much less than that of the eigenvalue decomposition (EVD) based PCA method. The mean squared error generated by the proposed method is very

similar to the EVD based PCA method. This ensures that there is negligible compromise in the quality or resolution of compressed data when compared to the basic PCA method.
Chapter 4: This section presents a novel technique that overcomes the drawbacks introduced by the conventional linear discriminant analysis (LDA) technique. The LDA technique utilizes eigenvalue decomposition (EVD) method which is adversely affected by the small sample size problem. The presented technique is capable in finding the discriminative features especially for small sample size problem. The technique is based on gradient descent algorithm but does not require any learning rate parameter. It also avoids discarding the null space of within-class scatter matrix and between-class scatter matrix which may have discriminative information useful for classification.

Chapter 5: This chapter presents a rotational transform which rotates the individual classes in the original feature space in such a way that the overlapping between the classes in the reduced feature space is further minimized, which is not possible by LDA technique. As a result the classification performance significantly improves which is demonstrated using several corpuses.

Chapter 6: This section presents a technique based on the combination of minimum distance classifier (MDC), class-dependent principal component analysis (PCA) and linear discriminant analysis (LDA) which gives improved performance as compared to other standard techniques when experimented on several machine learning corpuses.

Chapter 7: This section explores a technique that can be easily integrated in the local PCA design and is efficient even when the given statistical distribution is unknown. The initialization using this proposed *splitting technique* not only splits and reproduces the mean vector but also splits the orientation of components in the subspace domain. This would ensure that all clusters are used in the design. The proposed integration with the reconstruction distance local PCA design enables easier data processing and more accurate representation of multivariate data. A comparative approach is undertaken to demonstrate the greater effectiveness of the proposed approach in terms of percentage accuracy.

Chapter 8: This section presents a technique that is a combination of Vector Quantization (VQ) and vector quantized principal component analysis (VQPCA) techniques. The propose linear combined distance (LCD) technique is effective and outperforms all the presented techniques in this section in terms of getting high classification accuracy at very low data storage requirement and processing time. This would allow an object to be accurately classified as quickly as possible using very low data storage capacity.

Chapter 9: This section describes unsupervised learning technique namely independent component analysis (ICA) in solving blind source separation problems. Its basic theories and concepts are discussed.

Chapter 10: This section presents a new perspective of subspace independent component analysis (ICA). The notion of a function of cumulants (kurtosis) is generalized to vector kurtosis. This vector kurtosis is utilized in the subspace ICA algorithm to estimate subspace independent components. One of the main advantages of the presented approach is its computational simplicity. The experiments have shown promising results in estimating subspace independent components.

The work presented in this dissertation resulted in several research articles (Sharma and Paliwal, 2006, 2006b**,** 2006c, 2006d, 2000e, 2000f; Sharma et al., 2005, 2006, 2006b; Sharma, 2004, 2005). The following are the summarised academic outcomes:

1.2 Major and Original Contributions of this Thesis

Several original contributions have resulted from the research reported in this thesis. These contributions can be summarized as follows:

1. A Fast-PCA based method is proposed which computes the desired leading eigenvectors with a significantly much less computational cost than the

traditional PCA technique. The performance degradation by Fast-PCA is negligible.

2. A gradient-LDA based method has been proposed that overcomes the singularity problem of the conventional LDA when the training data is scarce. This method does not discard the null space of within-class scatter matrix and between-class scatter matrix which may have discriminative information useful for classification.
3. A new concept of rotational transform in LDA is proposed which rotates the individual classes in the original feature space in such a way that the overlapping between the neighbouring classes can be minimized. The experimentation shows promising outcomes. This further reduction of overlapping of samples is not possible in the conventional LDA technique.
4. A unified framework of MDC, class-dependent PCA and LDA is proposed. The combined technique shows encouraging classification results when compared with several state-of-the-art techniques.
5. An extension of VQPCA technique is proposed. It not only splits and reproduces the mean vector but also splits the orientation of components in the subspace domain. The proposed technique enables easier data processing and more accurate representation of multivariate data.
6. A combined technique using VQ and VQPCA has been proposed. The proposed technique (called LCD) is effective and outperforms all the presented techniques in chapter 8 in terms of getting high classification accuracy at very low data storage requirement and processing time. This would allow an object to be accurately classified as quickly as possible using very low data storage capacity.
7. A new perspective of subspace independent component analysis (ICA) is proposed. The notion of a function of cumulants (kurtosis) is generalized to vector kurtosis. This vector kurtosis is utilized in the subspace ICA algorithm to estimate subspace independent components. One of the main advantages of the presented approach is its computational simplicity. The experiments have shown promising results in estimating subspace independent components.

1.3 Journal Articles

- **Sharma, A., Paliwal, K.K., Onwubolu, G.C.,** "Class-dependent PCA, LDA and MDC: a combined classifier for pattern classification", *Pattern Recognition*, vol. 39, issue 7, 2006, pp 1215-1229.
- **Sharma, A., Paliwal, K.K.,** "Subspace Independent Component Analysis using Vector Kurtosis", *Pattern Recognition*, vol. 39, issue 11, 2006, pp 2223-2226.
- **Sharma, A., Paliwal, K.K., Onwubolu, G.C.,** "Splitting Technique Initialization in Local PCA", *Journal of Computer Science*, vol. 2, no. 1, 2006, pp 53-58.
- **Sharma, A., Paliwal, K.K.,** "Rotational Linear Discriminant Analysis Using Bayes Rule for Dimensionality Reduction", *Journal of Computer Science*, vol. 2, no. 9, 2006, pp 754-757.
- **Sharma, A., Paliwal, K.K., Onwubolu, G.C.,** "Pattern classification: an improvement using VQ and PCA based techniques", *American Journal of Applied Sciences*, vol. 2, no. 10, 2005, pp 1445-1455.
- **Sharma, A.**, "Signal modeling in speaker recognition", *Botswana Journal of Technology*, vol. 14, no. 1, April 2005, pp 59-66.
- **Sharma, A.**, "Radioactive mineral identification based on FFT Radix-2 algorithm", *IEE Letters*, vol. 40, no. 9, April 2004, pp 536-537.
- **Sharma, A., Paliwal, K.K.,** "Rotational Linear Discriminant Analysis for Dimensionality Reduction", *Machine Learning* (under review).
- **Sharma, A., Paliwal, K.K.,** "Fast Principal Component Analysis using Fixed-Point Algorithm", *Pattern Recognition Letters* (under review).
- **Sharma, A., Paliwal, K.K.,** "A Gradient Linear Discriminant Analysis for Small Sample Sized Problem", *Pattern Analysis and Applications* (under review).
- **Sharma, A., Paliwal, K.K.,** "A New Gradient Linear Discriminant Analysis Technique for Small Sample Size Problem", *Applied Intelligence* (under review).

Chapter 2

Gaussian Mixture Models, Basic Linear Models and Global Models for Pattern Classification and/or Compression

2.1 Abstract

In this chapter we first review Gaussian mixture model (GMM) for pattern classification. The model is discussed for supervised classification tasks. Then expectation-maximization (EM) algorithm is briefly presented. Some weaknesses of GMM are also discussed. Next we present a survey on some of the basic linear models which are used in pattern classification and/or data compression. The models discussed are minimum distance classifier (MDC), vector quantization (VQ), nearest neighbour (NN) and *k*-nearest neighbour (*k*NN). Furthermore, two most commonly used techniques in dimension reduction and/or pattern classification namely principal component analysis (PCA) and linear discriminant analysis (LDA) are illustrated. Their characteristics including drawbacks are briefly discussed. The purpose of this chapter is to give an understanding of the functionalities of some of the basic classifiers.

2.2 Introduction

The pattern classification task can be subdivided into two main categories namely supervised learning and unsupervised learning. In supervised learning a known set of feature vectors or data is given which is used to train the classifier. Training a classifier means to find some informative parameters that characterize a given set of features in some meaningful manner or sense. Thereafter, these parameters are used in finding information about unknown test feature vectors. Whereas in unsupervised learning no such known feature vectors are given.

Thus a supervised learning can be subdivided into a training phase and a classification phase or testing phase. In this chapter we firstly present Gaussian mixture model (GMM) for pattern classification and discuss some of its properties including its weaknesses. Then expectation-maximization (EM) algorithm is also discussed. EM algorithm can be used to estimate parameters in cases where not all the information about the feature vectors is given.

Next we looked at the four basic types of classifiers namely minimum distance classifier (MDC), vector quantization (VQ), nearest neighbour (NN) and *k*-nearest neighbour (*k*NN). MDC is the most simple and inexpensive classifier among them. It is single prototype classifier i.e. it provides only one parameter (vector or scalar) for each of the presented class. The natural generalization of MDC is VQ where each class is represented by multiple prototypes. In the NN classifier all the trained dataset is considered to be the prototypes or parameters, which will be used during the classification phase. The class labelling of a feature vector $\mathbf{x}$ is associated with the class label of the nearest parameter (usually Euclidean distance measure is used). The extension of NN is *k*NN where feature vector $\mathbf{x}$ is associated to the class label of the *k*-nearest parameters.

Furthermore, we considered some dimensionality reduction techniques. Dimension reduction is an important aspect in several applications like image recognition, data transmission etc. The features that are very difficult to handle in high dimensional space are reduced (usually) to a more manageable lower dimensional space. Principal component analysis (PCA) and linear discriminant analysis (LDA) are perhaps the most commonly used techniques in dimension reduction. These techniques are also used in pattern classification and/or recognition. The purpose of PCA is to give a lower dimensional feature space that best represents the original features. On the other hand, the purpose of LDA is to give a lower dimensional feature space that best discriminate the classes. The discriminative power of LDA makes it popular in pattern classification applications. Both the techniques give global linear transforms.

This chapter is subdivided into three main section namely A) Gaussian mixture models for pattern classification B) basic linear models for classification and compression, and C) global models for dimensionality reduction and pattern classification. These sections are illustrated consecutively as follows:

A) Gaussian Mixture Models for Pattern Classification

In this section we elaborate basic theories and concepts of Gaussian mixture model for pattern classification purposes.

2.3 Pattern Recognition and Bayesian Decision Theorem

This section presents Bayesian decision theory to the problem of pattern recognition or classification in supervised learning tasks. We discussed the class labelling or allocation of the membership of an unlabelled feature vector to one of the classes or categories previously trained. The supervised classification procedure can be subdivided into two main phases namely the training phase and the testing or classification phase. In the training phase the classifier is made to learn by known categories of patterns or feature vectors and in the classification phase unknown feature vectors which were not part of the training dataset are assigned class label of the nearest category of trained feature vectors. For example, one typical classification problem could be in the following form.

‘Given an observation of fruits, you are asked to identify which fruit is it? apple or mango.’

Humans can very easily answer such a question by watching the fruits. Now let us say the classification process is carried out in a closed room and the observer is not able to see the fruits. In this case one can only *guess* based on some *a priori* knowledge of the fruits.

Imagine that the existence of apple is higher than mango in the nature. Then the answer will be biased towards apples.

To give a mathematical representation for the above problem, let us denote two classes $C_1 = apple$ and $C_2 = mango$; and suppose *a priori* probabilities are of these classes are $P(C_1)$ and $P(C_2)$ respectively. If the decision has been made purely based on *a priori* probability where $P(C_1) > P(C_2)$ then the selected class will be C_1.

To further extend our discussion, let us assume some measurement x which represents some characteristic of a given class (e.g. x is weight of fruits). Now we slightly modify the question as:

'Some measurement x is given, investigate the likelihood of x corresponding to the given classes C_1 and C_2.'

Intuitively this will generate two probabilities defined as follows:

a)	$p(x \mid C_1)$	probability of x given class C_1 (weight of apple)
b)	$p(x \mid C_2)$	probability of x given class C_2 (weight of mango)

These probabilities are known as *class conditional probability density function* or in short *pdf*. The *pdf* for two classes are depicted in figure 2.1, where apple is uni-model and mango is bi-model i.e. a class having two varieties of mangoes. After solving the *a priori* probability and *pdf* the probability theorem can be yielded as follows.

2.3.1 Probability Theorem

Given the observation x, what is the probability that the observed event was due to class C_j ($j = 1,2$)? In other words, we seek the conditional probability $P(C_j \mid x)$. These probabilities are frequently called *a posteriori* probability.

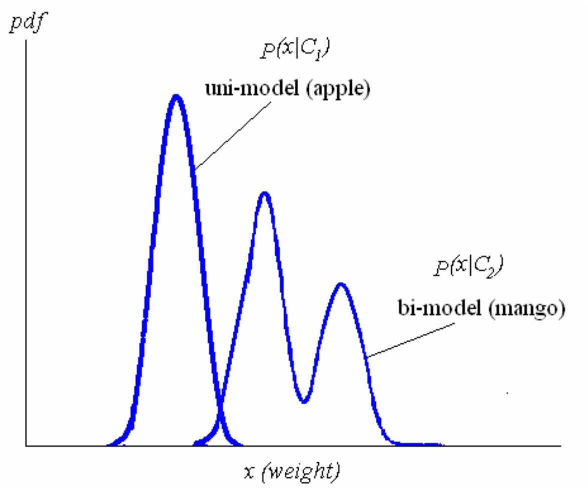


Figure 2.1: Probability density function of two-class model

The decision of class selection will be based on *a posteriori* probability. The following Bayes rule will help us to evaluate the *a posteriori* probability:

$$P(C_j \mid x) = \frac{p(x \mid C_j)P(C_j)}{p(x)} \qquad 2.1$$

where $p(x)$ is the probability of occurrence of x and can be expressed as:

$$p(x) = \sum_{all\ j} p(x \mid C_j)P(C_j) \qquad 2.2$$

It can be observed from equation 2.1 that if $P(C_1 \mid x) > P(C_2 \mid x)$ then x belongs to C_1 or else to C_2. In other words, one can maximize the *a posteriori* probability which will give the best or optimal recognition performance. The following theorem elaborates optimality criteria.

2.3.2 Optimality Theorem

Given an observation x, maximizing *a posteriori* probability $P(C_j \mid x)$ will lead to optimal recognition performance.

Now let us generalize the problem for *K* classes and define the following terms:

Class set or state of nature $S = \{C_j : j = 1,2,...K\}$ and *d*-component (dimension) feature vector also known as pattern $\mathbf{x} = (x_1, x_2, ..., x_d)^{\mathrm{T}}$.

The membership of unlabelled feature vector **x** to one of the *K* classes can be found by using optimality theorem as follows:

$$k = \arg \max_{j=1,2,...,K} P(C_j \mid \mathbf{x}) \qquad 2.3$$

where *k* is the argument for which $P(C_j \mid x)$ is maximum. Equation 2.3 can be simplified by using Bayes rule (equation 2.1), i.e.

$$k = \arg \max_{j=1,2,...,K} \frac{p(\mathbf{x} \mid C_j) P(C_j)}{p(\mathbf{x})} \qquad 2.4$$

$$= \arg \max_{j=1,2,...,K} p(\mathbf{x} \mid C_j) P(C_j) \qquad 2.5$$

since $p(\mathbf{x})$ does not depend on the variable *j* it can be removed. If all the classes are equally likely (i.e. $P(C_j) = 1/K$) then equation 2.5 turns to be

$$k = \arg \max_{j=1,2,...,K} p(\mathbf{x} \mid C_j) \qquad 2.6$$

Equation 2.5 is the criteria function for maximum likelihood decision.

The first step in a classifier design is the training phase or encoding session, where the classifier is trained with respect to its models. In this respect, two problems can be addressed:

(i) How to find *a priori* probability $P(C_j)$?

(ii) How to find *pdf* $p(\mathbf{x} \mid C_j)$?

The first problem of finding $P(C_j)$ can be solved by collecting all the objects available to the classifier, and for the second problem, $p(\mathbf{x} \mid C_j)$ can be computed using ‘histograms’. The *pdf* computation based on histograms is generally referred as non-parametric method. Whereas, the *pdf* computation based on some known distribution (e.g. Gaussian) is known as parametric method. The next section deals with Gaussian mixture model (GMM) for a classifier design.

2.4 Gaussian Mixture Models

The parametric form of Gaussian models is a function of mean and covariance of input feature vectors **x**. The parametric form can be denoted as:

$$p(\mathbf{x} \mid C_j) = N(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) \qquad 2.7$$

where the operator $N(\bullet)$ is the distribution; $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are mean and covariance of the observed feature vectors respectively. For scalar patterns, covariance in equation 2.7 becomes variance and is represented (usually) by σ^2.

Suppose we have a given *d*-dimensional training data $\xi = \{\mathbf{x}_i : i = 1,2,...M\}$ and we model the distribution by Gaussian model then equation 2.7 will be:

$$p(\mathbf{x} \mid \lambda) = \frac{1}{(2\pi)^{d/2} \mid \Sigma \mid^{1/2}} \mathbf{exp}[-\tfrac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})] \qquad 2.8$$

where $\mathbf{x} \in \xi$ and model parameter λ represents mean and covariance i.e. $\lambda = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$. The next thing is to train the data.

2.4.1 Training Problem

Given a training data ξ, how to compute the parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$.

Assuming the feature vectors of the training dataset ξ are independent then equation 2.8 can be written as:

$$p(\xi \mid \lambda) = p(\mathbf{x}_1 \mid \lambda) p(\mathbf{x}_2 \mid \lambda) \ldots p(\mathbf{x}_M \mid \lambda) \qquad 2.9$$

Taking **log** both the sides of equation 2.9 which gives likelihood of ξ:

$$\mathbf{log}\, p(\xi \mid \lambda) = \sum_{i=1}^{M} p(\mathbf{x}_i \mid \lambda) \qquad 2.10$$

By substituting equation 2.8 in equation 2.10 we have:

$$\mathbf{log}\, p(\xi \mid \lambda) = \sum_{i=1}^{M} [-\tfrac{d}{2} \mathbf{log}\, 2\pi - \tfrac{1}{2} \mathbf{log} \mid \boldsymbol{\Sigma} \mid - \tfrac{1}{2} (\mathbf{x}_i - \boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\mathbf{x}_i - \boldsymbol{\mu})] \qquad 2.11$$

We then use equation 2.11 to determine the model that provides maximum likelihood i.e.

$$\nabla_{\lambda} \mathbf{log}\, p(\xi \mid \lambda) = 0 \qquad 2.12$$

where ∇_{λ} is the gradient (derivative) of likelihood with respect to the model $\lambda = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$.

Assuming $\boldsymbol{\Sigma}$ is invariant with respect to $\boldsymbol{\mu}$, then equation 2.11 and 2.12 can be solved as follows:

$$\nabla_\mu \log p(\xi \mid \lambda) = \sum_{i=1}^{M} [-\tfrac{1}{2} 2 \mathbf{\Sigma}^{-1} (\mathbf{x}_i - \boldsymbol{\mu})(-1)] = 0 \qquad 2.13^{1}$$

$$= \sum_{i=1}^{M} \mathbf{\Sigma}^{-1} (\mathbf{x}_i - \boldsymbol{\mu}) = 0$$

$$= \mathbf{\Sigma}^{-1} \sum_{i=1}^{M} (\mathbf{x}_i - \boldsymbol{\mu}) = 0$$

$$= \sum_{i=1}^{M} (\mathbf{x}_i - \boldsymbol{\mu}) = 0, \text{since } \mathbf{\Sigma}^{-1} \neq 0$$

$$= \sum_{i=1}^{M} \mathbf{x}_i - M \boldsymbol{\mu} = 0$$

$$\boldsymbol{\mu} = \tfrac{1}{M} \sum_{i=1}^{M} \mathbf{x}_i \qquad 2.14$$

For $\mathbf{\Sigma}$, assume $\boldsymbol{\mu}$ is invariant wrt $\mathbf{\Sigma}$, then:

Let $A = \mathbf{\Sigma}^{-1}$

$$\nabla_\Sigma \log p(\xi \mid \lambda) = \sum_{i=1}^{M} [-\tfrac{d}{2} \log 2\pi + \tfrac{1}{2} \log |A| - \tfrac{1}{2} (\mathbf{x}_i - \boldsymbol{\mu})^{\mathrm{T}} A (\mathbf{x}_i - \boldsymbol{\mu})] = 0$$

$$\nabla_A \log p(\xi \mid \lambda) = \sum_{i=1}^{M} [\tfrac{1}{2} A^{-1} - \tfrac{1}{2} (\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^{\mathrm{T}}] = 0 \qquad 2.15^{2}$$

$$= \sum_{i=1}^{M} \left[\mathbf{\Sigma} - (\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^{\mathrm{T}} \right] = 0$$

$$= M \mathbf{\Sigma} - \sum_{i=1}^{M} (\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^{\mathrm{T}}$$

$$\mathbf{\Sigma} = \tfrac{1}{M} \sum_{i=1}^{M} (\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^{\mathrm{T}} \qquad 2.16$$

Equations 2.14 and 2.16 will give maximum likelihood of the model. GMM can be used to estimate parameters where not all the information of parameters is present (Bilmes, 1997; Dempster et. al., 1977; Ghahramani, 1994; Ghaharamani and Jordan 1994; Ormoneit and Tresp, 1995).

---

[1] For derivation the gradient matrix property $\nabla_b (b^{\mathrm{T}} A b) = 2Ab$ is used, where b is a vector and A is a matrix.

[2] The gradient matrix properties $\nabla_A (b^{\mathrm{T}} A b) = bb^{\mathrm{T}}$ and $\nabla_A |A| = |A| A^{-1}$ are used.

2.5 Expectation Maximization (EM) Algorithm

In case not all the information is given then EM algorithm could be used to estimate the remaining parameters. It has two steps. First step is the E-step where expected value of the parameters based on the observed data is found. Second step is the M-step where maximum likelihood from the observed parameters is found. This is an iterative procedure and both the E and M steps are repeated until the convergence is obtained. To illustrate EM-algorithm assume X is the observed data and *s* is the missing data, then the algorithm can be defined as:

i) *E-Step*

$$Q(\lambda \mid \lambda^{(k)}) = E[\log p(\mathrm{X}, s \mid \lambda) \mid \mathrm{X}, \lambda^{(k)}] \tag{2.17}$$

where Q is an auxiliary function and $\lambda^{(k)}$ is an initial estimate.

ii) *M-Step*

Maximizing the auxiliary function

$$\lambda^{(k+1)} = \arg\max_{\lambda} Q(\lambda \mid \lambda^{(k)}) \tag{2.18}$$

Maximizing the auxiliary function will maximize the likelihood. Let the predefined threshold for convergence be ε and initial model parameter be λ^0 then procedure of EM algorithm is illustrated in table 2.1.

TABLE 2.1: The Expectation-Maximization Algorithm

```
1. initialize $\lambda^0, \varepsilon, k \leftarrow 0$
2.     do $k \leftarrow k+1$
3.         E-step: compute $Q(\lambda \mid \lambda^{(k)})$
4.         M-step: $\lambda^{(k+1)} \leftarrow \arg\max_{\lambda} Q(\lambda \mid \lambda^{(k)})$
5.     until $Q(\lambda^{(k+1)} \mid \lambda^{(k)}) - Q(\lambda^{(k)} \mid \lambda^{(k-1)}) \le \varepsilon$
6. return $\hat{\lambda} \leftarrow \lambda^{(k+1)}$
```

2.6 Drawbacks of GMM

1) Robust estimation by this approach is difficult if very little training data is presented.
2) The parameters estimation from GMM through EM algorithm has a disadvantage that it could end the iterative process at a local optimum value; therefore the initial setting is very important.
3) GMM method is computationally expensive.
4) As for the speech, GMM does not take into account the acoustic diversity of different phonetic events encountered during recognition (Park and Hazen, 2003).
5) The identification accuracy drops considerably when presented with noisy speech (Park and Hazen, 2003).
6) In GMM, acoustic features are converted from a source speaker to a target speaker by minimizing mean square error. Its drawback is that the converted features are overly smooth and this makes the reconstructed speech unclear (Chen et. al., 2003).

B) Basic Linear Models for Classification and Compression

In this section we illustrate some of the basic linear classifiers used for classification and/or compression purposes.

2.7 Conventional Classifiers

This section briefly describes the four types of classifiers namely Vector Quantization (VQ), Minimum Distance Classifier (MDC), Nearest Neighbour (NN) and *k*-Nearest Neighbour (kNN). In all the discussions ω_i denotes the state of nature or class label of i^{th} class in a c-class problem, χ denotes the set of n train samples, $\Omega = \{\omega_i : i = 1,2,...,c\}$ be the finite set of c states of nature and let θ' be the class label of train pattern or prototype such that $\theta' \in \Omega$ .The set $\mathcal{X}$ can be separated by class into c subsets $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_c$ with the samples in $\mathcal{X}_i$ belonging to ω_i:

$\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ where $x_j \in \mathbf{R}^d$ (d-dimensional hyperplane)

$\mathcal{X}_i \subset \mathcal{X}$ and $\mathcal{X}_1 \cup \mathcal{X}_2 \cup \ldots \cup \mathcal{X}_c = \mathcal{X}$

Let n_i denote the number of samples in the subset $\mathcal{X}_i$, therefore $\sum_{i=1}^{c} n_i = n$. Figure 2.2 illustrates the class labelling of a test pattern and the relationship between the label of prototype (θ') and the label of class (ω). The prototype could be a train pattern, a centroid, a transform depending upon the type of classifier is used. In figure 2.2 two-class problem is considered where each class consists of 3 prototypes. Each of the class is assigned a unique label namely ω_p and ω_q such that $(\omega_p, \omega_q) \in \Omega$. The class labels of the prototypes are $\theta'_i, \theta'_j, \theta'_k$ and $\theta'_l, \theta'_m, \theta'_n$ such that:

$$\omega_p = \theta'_i = \theta'_j = \theta'_k \qquad \text{and}$$
$$\omega_q = \theta'_l = \theta'_m = \theta'_n$$

The class label of prototype is assigned to a test pattern **x** which is the closest to the prototype based on some distance measurements or conditional probabilities. Therefore if $L(\mathbf{x})$ denotes the class label of a test pattern **x** then from the figure $L(\mathbf{x}) = \theta'_l = \omega_q$.

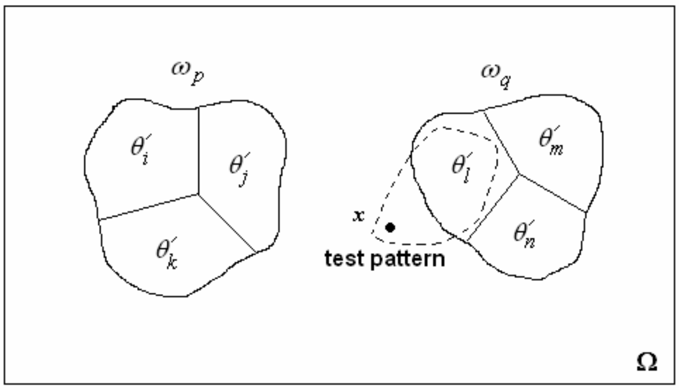


Figure 2.2: Class labelling of a test pattern in a two-class problem

2.8 Vector Quantization

Vector Quantization is a conventional technique for data compression (Gray, 1984) and also applied in pattern classification/recognition applications. For compression, a set of continuous or discrete vectors is transformed into a stream of low rate digital sequence suitable for communication or storage via digital link. A vector quantizer estimates a feature vector extracted from audio, video or any statistical data by one of the encoded set of centroid vector of a disjoint region known as codewords or reference vectors of the quantizer. The set of codewords is known as codebook of the system. A distortion measure is associated with each of the vector quantizer. The most common distortion measures used are Euclidean distance, Holder norm, Minkowski norm, weighted-square distortion and general quadratic distortion (Linde et. al., 1980). The other distortion measure which arises in speech compression system has a hybrid form of distance measure, consisting of a positive semi-definite symmetric matrix (Itakura and Saito, 1986; Chaffee, 1975).

A VQ technique is applied in several areas of pattern compression and recognition, which include speech coding or speech compression (Makhoul et. al., 1985), in speaker recognition (Soong et. al., 1987) and in image coding or image compression (Akansu and Kadur, 1990; Antonini et. al., 1991; Aravind and Gersho 1986; Barba and Hanen, 1993; Cosman et. al., 1996). In pattern recognition/classification the VQ technique partition each class into several disjoint regions where these regions are estimated by its own centroids.

An iterate process for updating codewords of VQ is called generalized Lloyd (Gersho and Gray, 1992) algorithm also known as LBG algorithm after Linde, Buzo and Gray who presented an illustrative design of vector quantizer (1980). VQ technique can also be referred as local classifiers since they partition each class into several local regions and estimate each region by a prototype. The aim of VQ is to find the codebook that minimizes the expected distortion between feature vector **x** and the centroid of the disjoint region.

To explain a LBG algorithm for compression purpose (assuming one class only), suppose a *d*-dimensional ($\mathbf{R}^d$) set of *n* feature vectors $\mathcal{X}$ is given, where VQ performs lossy data compression by mapping a vector set $\mathcal{X}$ into *N* centroid or codewords ($A = \{\boldsymbol{\mu}_j : j = 1,2,...,N\}$) which is the best estimate of their corresponding disjoint regions or Voronoi regions $S = \{C_j : j = 1,2...,N\}$, for $N < n$. For *N* level quantizer, the quantity $R = (\log_2 N)/d$ is defined as a rate of the quantizer in bits per sample. Here the distortion measure $d(\mathbf{x},\boldsymbol{\mu}_j)$ is a *d*-dimensional Euclidean distance between the feature vector **x** and the centroid $\boldsymbol{\mu}_j$. The performance of VQ is the measure of minimum expected distortion of feature vector **x** and the centroid of a disjoint region

$$D = E[\min_{j=1,...N} d(\mathbf{x},\boldsymbol{\mu}_j)] \qquad 2.19$$

The operator $E[\bullet]$ of equation 2.19 represents expectation with respect to **x**. By considering all the mentioned terms above, the LBG algorithm[3] can be yielded as follows:

2.8.1 LBG Algorithm

Step 0. Choose the desired level *N*. Define threshold error $\varepsilon > 0$, initial average distortion $D_1 \to \infty$ and initial centroid $\boldsymbol{\mu}_1 = \frac{1}{n}\sum_{all\,\mathbf{x}} \mathbf{x}$. Set the variable level $M \leftarrow 1$ and $j = 1$.

Step 1. Split reproductive vectors as $\boldsymbol{\mu}_j = [\boldsymbol{\mu}_j + \varepsilon, \boldsymbol{\mu}_j - \varepsilon]$. Update level $M \leftarrow 2M$.

Step 2. Compute distortion $d(\mathbf{x},\boldsymbol{\mu}_j) = \| \mathbf{x} - \boldsymbol{\mu}_j \|^2$ for $j = 1,2,...,M$ and obtain minimum distance $\delta_{\min} = \min_{j=1,2,...,M} [d(\mathbf{x},\boldsymbol{\mu}_j)]$. Find all feature vectors $\mathbf{x} \in C_k$ for $k = \arg[\delta_{\min}]$. With

[3] Three methods for designing a quantizer are described in the classic paper by Linde et. al. (1980), namely (i) quantizer algorithm for known distribution, (ii) quantizer algorithm for unknown distribution by defining initial codebook, and (iii) quantizer algorithm using splitting technique. In this report only method (iii) is considered for implementation.

the obtained new regions update centroid $\boldsymbol{\mu}_j = \frac{1}{n_j} \sum_{\mathbf{x} \in C_j} \mathbf{x}$ where n_j denotes number of feature vectors in the Voronoi region C_j. Iterate this process until $(D_{f-1} - D_f)/D_f \leq \varepsilon$ where D_f is from equation 2.19 and f is some iterative number. Follow next step with the improved reproductive vectors $A = \{\mu_j\}$.

Step 3. Iterate step 1 and step 2 until *M* equalizes the value of *N*.

It can be observed that the distortion D_f reduces along with the number of iterations and converges to some finite value after which the difference between the two successive D_f is very small. The iterative process is conducted for the entire set of training vectors and could be very time consuming if the training feature vectors are very large.

Note the above sequence is given only to train the quantizer. Once the optimum codewords or quantized levels $A = \{\mu_j\}$ are obtained, any vector outside the training set **x** with optimum nearest-neighbour rule is applied for the estimation.

Figure 2.3 depicts LBG algorithm on a small dataset of 12 vectors in a 2-dimensional space (represented as 'o' mark in the figure). Firstly, the centre of dataset $\boldsymbol{\mu}_0$ is computed, and an initial distortion is defined. The centroid ($\boldsymbol{\mu}_0$) is perturbed by a small predefined quantity ε to give a reasonable variation. This will generate two levels by partitioning the given space using distance measure $d(\mathbf{x}, \boldsymbol{\mu}_j)$. The process of partitioning and finding optimum reproductive vectors are carried out iteratively until the desired level (4 in this case) with acceptable distortion is achieved. The dotted lines in figure 2.3 show the approximate boundaries of the obtained regions and the centroid of each region is denoted by $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\mu}_3$ and $\boldsymbol{\mu}_4$.

Unlike the uniform or linear quantizers (Proakis and Manolakis, 1996), vector quantizers can also produce non-uniform quantizer step size, depending upon the statistical distribution of input data. Figure 2.4 illustrates a reconstruction of speech signal by VQ at

3bit/sample. Figure 2.4a is the original input signal and figure 2.4b is the reconstructed or decoded signal. It can be observed that VQ produces non-uniform quantization step i.e. the difference between any two successive levels need not to be equal ($|L1-L2| \neq |L2-L3|$).

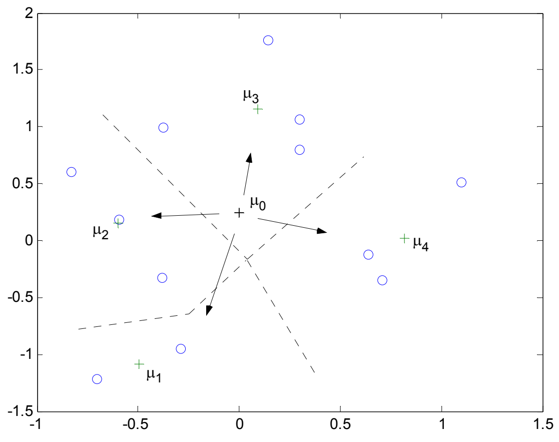


Figure 2.3: Two-dimensional space partition

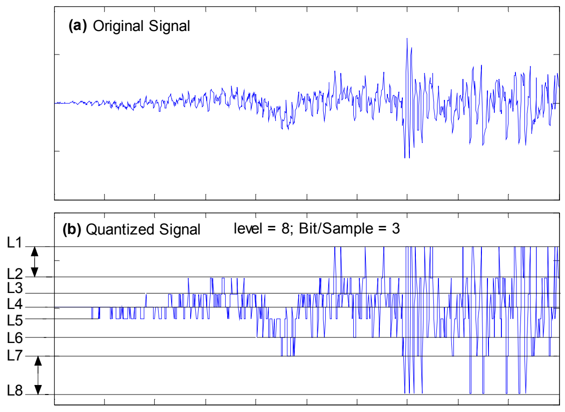


Figure 2.4: Reconstructed speech signal using the VQ technique.

VQ is also employed in pattern recognition/classification applications where each class is estimated by several prototypes. The training procedure for classification is similar to compression procedure. The VQ technique here is applied separately for each of the class for partitioning the class into disjoint regions. In the testing or classification phase the unlabelled feature vector **x** is associated to the class label of the closest centroid of the trained dataset.

2.8.2 Drawbacks of Conventional VQ Algorithms

The following describes drawbacks associated with the classic VQ algorithm:

i) For high dimensional vectors, the computational complexity of searching for the codewords increases exponentially and this severely augments the processing time.
ii) The performance of VQ algorithm is strongly dependent on the choice of the initial conditions and the configuration parameters.
iii) In some cases, codewords could be left alone having no samples associated to them, since some of the codewords are nearer to the other cell.

Several extensions have been developed to minimize the drawbacks associated with VQ algorithms. For the problems related to (i) stochastic algorithms (Ahalt et. al., 1990) are proposed that can be faster than conventional VQ algorithms. A fast search algorithm under the assumption that the distortion is measured by the Euclidean distance is presented by Lee and Chen (1995). The solution to problems related with (ii) and (iii) have been also developed (Karayiannis, 1997; Fritzke 1997; Hofmann and Buhmann 1997; Patané and Russo, 2001).

2.9 Minimum Distance Classifier

MDC is a special case of VQ where each class χ_i is represented by single prototype, which is usually the centroid of the class in the feature space. The goal of MDC is to correctly label as many patterns as possible. It provides minimal total parameter requirement and computational demand. The MDC method finds centroid of classes and measures distances between these centroids and the test pattern. In this method, the test pattern belongs to that class whose centroid is the closest distance to the test pattern. MDC is used in many pattern classification applications (Di Maio and Marciano, 2003; Paclik and Duin, 2003; Sahin, 2000; Datta and Kibler, 1997; Griguolo, 1994) including disease diagnostics (Lewenstein and Chojnacki, 2004), classification of digit mamographic images (Lambrou et al., 2002) and optical media inspection (Toth et al., 2002). The training and classification procedures of MDC classifier can be briefly illustrated as follows:

Training

Step1. Find and store centroid $\boldsymbol{\mu}_j$ for each of the c class:

$$\boldsymbol{\mu}_j = \tfrac{1}{n_j} \sum_{x \in \chi_j} \mathbf{x} \text{ for } j = 1,2,\ldots,c$$

Classification

Step1. For a test pattern $\mathbf{x}$ find the Euclidean distance from $\mathbf{x}$ to the centroid $\boldsymbol{\mu}_j$:

$$\delta_j = \| \mathbf{x} - \boldsymbol{\mu}_j \| \text{ for } j = 1,2,..c$$

Step2. Find the nearest centroid to $\mathbf{x}$:

$$r = \mathbf{arg}\,\mathbf{min}_{j=1}^{c} \delta_j$$

Step3. Assign class label ω_r to the test pattern $\mathbf{x}$.

2.10 Nearest Neighbour Classifier

Nearest Neighbour (NN) classifier (Fukunaga, 1990) is the most simple classifier found up till now. In NN classifier no special procedure is required to do training. All the available data (as maximum as possible) is stored to perform classification, where each test pattern is compared for similarity with all the available training data (pattern). The test pattern is assigned the class label of that training pattern, which is the closest to the test pattern. A major drawback of NN approach is its large total parameter requirement to perform classification task. For example, a dataset with 10 classes, having 5000 vectors or patterns in each class with 64 attributes or dimensions would require total parameters as follows:

$$\textit{total parameters} = \textit{class} \times \textit{NoOfVec} \times \textit{dimension} = 10 \times 5000 \times 64 = 3.2 \times 10^6$$

If the dimension is very high (e.g. in image), then the total parameter requirement for NN approach will be even more severe which would restrict the practical application of such approach. The procedure for NN classifier can be subdivided into two main phases namely, training phase and testing or classification phase. In the training phase all the available patterns $\mathcal{X}$ with their corresponding class label information are stored for classification purpose. In the testing phase a test pattern $\mathbf{x}$ is assigned the class label associated to the nearest train pattern $\mathbf{x}' \in \mathcal{X}$. To illustrate this, let δ_j denote the Euclidean distance between a test pattern $\mathbf{x}$ and a train pattern $\mathbf{x}_j \in \mathcal{X}$, then the classification procedure can be given as follows:

Step1. Compute the distance $\delta_j = \| \mathbf{x} - \mathbf{x}_j \|$

Step2. Find argument *k* that satisfies $k = \mathbf{arg}\,\mathbf{min}_{j=1}^{n} \delta_j$

Step3. Assign class label $\omega_r = \theta'_k$ of the nearest pattern $\mathbf{x}'$ to the test pattern $\mathbf{x}$.

2.11 *k*-Nearest Neighbour Classifier

The classification accuracy of NN approach can be improved by making the decision of a test pattern for class labelling based on *k* nearest patterns. This method is known as *k*-Nearest Neighbour (*k*NN) (Fukunaga, 1990). The computational demand is severely expensive for this approach.

*k*NN classifier is a generalized form of NN classifier. In this approach the test pattern is assigned the class label which has the majority of *k* collected train patterns. The training phase of the kNN classifier is similar to NN classifier where all the training patterns together with their class label information are stored for the later use. The total parameter requirement is also same as NN approach. The classification or testing procedure can be given as:

Step1. Find k train patterns nearest to $\mathbf{x}$, $\mathbf{x}'_1, \mathbf{x}'_2, ..., \mathbf{x}'_k$ where $\mathbf{x}'_j \in \mathcal{X}$.

Step2. Collect the associated labels $\theta'_1, \theta'_2, ..., \theta'_k$, where $\theta'_j \in \Omega$.

Step3. Assign class label ω_r to the test pattern $\mathbf{x}$ that has the majority of representatives in $\theta'_1, \theta'_2, ..., \theta'_k$.

Processing speed of kNN classifier is slower than NN classifier due to the searching of k nearest patterns for each of the test pattern. The classification accuracy may improve with the increase in the value k. This improvement is usually observed when the test patterns and the train patterns are closely matched. However, in some cases when the test patterns and the train patterns do not match the classification accuracy is poor. In this case increasing the value k may not improve the classification accuracy of the system.

C) Global Models for Dimensionality Reduction and Pattern Classification

In this section we describe two global models for dimensionality reduction and/or pattern classification namely principal component analysis (PCA) and linear discriminant analysis (LDA)..

2.12 Principal Component Analysis

PCA finds a linear transformation $\boldsymbol{\varphi}$ which reduces d-dimensional feature vectors to h-dimensional feature vectors (where $h < d$) in such a way that the information is maximally preserved in minimum mean squared error sense. This linear transformation is known as PCA transform or Karhunen-Loéve transform (KLT) (Fukunaga, 1990). It can also reconstruct *h*-dimensional feature vectors back to the *d*-dimensional feature vectors with some finite error known as reconstruction error. The PCA is mostly used in compression and reconstruction of high dimensional feature vectors. Since the transformation is from *d*-dimensional feature space to *h*-dimensional feature space and vice versa the size of $\boldsymbol{\varphi}$ is $d \times h$. The h column vectors are the basis vectors. The first basis vector is in the direction of maximum variance of the given feature vectors. The remaining basis vectors are mutually orthogonal and, in order, maximize the remaining variances subject to the orthogonal condition. Each basis vector represents a principal axis. These principal axes are those orthonormal axes onto which the remaining variances under projection are maximum. These orthonormal axes are given by the dominant/leading eigenvectors (i.e. those with the largest associated eigenvalues) of the measured covariance matrix. In PCA, original feature space is characterized by these basis vectors and the number of basis vectors used for characterization is usually less than the dimensionality *d* of the feature space.

2.12.1 The PCA Encoding

The PCA transform can be found by minimizing mean squared error. To see this, let the feature vector be $\mathbf{x} \in \mathbf{R}^d$ (*d*-dimensional space), reduced dimensional feature vector be $\mathbf{y} \in \mathbf{R}^h$ and reconstructed feature vector be $\hat{\mathbf{x}} \in \mathbf{R}^d$. Then the mean squared error can be represented as

$$\text{MSE} = E[\|\mathbf{x} - \hat{\mathbf{x}}\|^2] \qquad 2.20$$

where $E[\bullet]$ is the expectation operation with respect to $\mathbf{x}$ and $\|\bullet\|^2$ is the norm squared value. We know that PCA transformation $\boldsymbol{\varphi}$ is of size $d \times h$ and it is used to do dimensionality reduction from *d*-dimensional space to *h*-dimensional feature space, i.e. $\boldsymbol{\varphi} : \mathbf{x} \to \mathbf{y}$ or $\mathbf{y} = \boldsymbol{\varphi}^{\mathrm{T}}\mathbf{x}$. If there is no dimension reduction then PCA transformation

$\boldsymbol{\varphi}$ would be a square matrix of size $d \times d$. For no dimension reduction vector $\mathbf{x}$ can be written in terms of vector $\mathbf{y}$ and $\boldsymbol{\varphi}$ as:

$$\mathbf{x} = \boldsymbol{\varphi}\mathbf{y} \tag{2.21}$$

or
$$\mathbf{x} = \sum_{i=1}^{d} y_i \phi_i \tag{2.22}$$

where ϕ_i are the column vectors of $\boldsymbol{\varphi}$ and y_i are the components of vector $\mathbf{y}$.

Suppose we choose only h ($< d$) of $\boldsymbol{\varphi}$ and we still want to approximate $\mathbf{x}$ well, then:

$$\hat{\mathbf{x}} = \sum_{i=1}^{h} y_i \phi_i + \sum_{i=h+1}^{d} b_i \phi_i \tag{2.23}$$

where b_i is any constant. Error in resulting representation would be

$$\begin{aligned}\Delta\mathbf{x} = \mathbf{x} - \hat{\mathbf{x}} &= \mathbf{x} - \sum_{i=1}^{h} y_i \phi_i - \sum_{i=h+1}^{d} b_i \phi_i \\ &= \sum_{i=1}^{d} y_i \phi_i - \sum_{i=1}^{h} y_i \phi_i - \sum_{i=h+1}^{d} b_i \varphi_i \\ &= \sum_{i=h+1}^{d} (y_i - b_i)\phi_i \end{aligned} \tag{2.24}$$

We want to estimate $\hat{\mathbf{x}}$ such that the mean square error (MSE) is minimum, i.e.

$$\text{MSE} = E[\|\Delta\mathbf{x}\|^2] \tag{2.25}$$

From equations 2.24 and 2.25

$$\text{MSE} = E\left[\left(\sum_{i=h+1}^{d} (y \;\; - b_i)\phi_i\right)^t \left(\sum_{j=h+1}^{d} (y_j - b_j)\phi_j\right)\right]$$

$$= E\left[\sum_{i=h+1}^{d}\sum_{j=h+1}^{d}(y_i - b_i)(y_j - b_j)\phi_i^t\phi_j\right]$$

from the orthonormality condition $\phi_i^t\phi_j = 0$ for $i \neq j$.

$$\therefore \qquad = E\left[\sum_{i=h+1}^{d}(y_i - b_i)^2\right]$$

or $$\text{MSE} = \sum_{i=h+1}^{d} E[(y_i - b_i)^2] \qquad 2.26$$

The optimum choice for b_i is obtained by minimizing MSE with respect to b_i. Therefore differentiating equation 2.26 with respect to b_i we have

$$\frac{\partial}{\partial b_i}\text{MSE} = \sum_{i=h+1}^{d}\frac{\partial}{\partial b_i}E[(y_i - b_i)^2] = 0$$

or $$-2E[y_i - b_i] = 0$$

or $$E[y_i] = b_i$$

$\because$ $\mathbf{y} = \boldsymbol{\varphi}^{\mathrm{T}}\mathbf{x}$ therefore $y_i = \phi_i^{\mathrm{T}}\mathbf{x}$, so we get

$$b_i = E[y_i] = E[\phi_i^{\mathrm{T}}\mathbf{x}] = \phi_i^{\mathrm{T}}E[\mathbf{x}] \qquad 2.27$$

Now from equation 2.26, we have

$$\text{MSE} = \sum_{i=h+1}^{d} E[(y_i - b_i)^2]$$

$$= \sum_{i=h+1}^{d} E[(y_i - \phi_i^{\mathrm{T}}E[\mathbf{x}])^2] \qquad \text{(from equation 2.27)}$$

using the property $z^2 = zz^t$ we obtain

$$= \sum_{i=h+1}^{d} E[(\phi_i^{\mathrm{T}}\mathbf{x} - \phi_i^{\mathrm{T}}E[\mathbf{x}])(\phi_i^{\mathrm{T}}\mathbf{x} - \phi_i^{\mathrm{T}}E[\mathbf{x}])^{\mathrm{T}}]$$

$$= \sum_{i=h+1}^{d} E[\phi_i^{\mathrm{T}}(\mathbf{x}-E[\mathbf{x}])(\mathbf{x}^{\mathrm{T}}-E[\mathbf{x}]^{\mathrm{T}})\phi_i]$$

$$= \sum_{i=h+1}^{d} \phi_i^{\mathrm{T}} E[(\mathbf{x}-E[\mathbf{x}])(\mathbf{x}-E[\mathbf{x}])^{\mathrm{T}}]\phi_i$$

$$\text{MSE} = \sum_{i=h+1}^{d} \phi_i^{\mathrm{T}} \Sigma_X \phi_i \qquad 2.28$$

where $\Sigma_X = E[(\mathbf{x}-E[\mathbf{x}])(\mathbf{x}-E[\mathbf{x}])^{\mathrm{T}}]$ is a covariance matrix and $E[\mathbf{x}] = \boldsymbol{\mu}$ is the mean of the given feature vectors.

The optimization problem of MSE can now be solved by finding ϕ_i that occur at constrained relative-extremum of equation 2.28 under the constrained curve $\phi_i^{\mathrm{T}}\phi_i - 1 = 0$. The method of Lagrange multipliers can be used in this case as follows:

$$\nabla f(\phi) = \lambda \nabla g(\phi) \text{ where } \lambda \neq 0 \qquad 2.29$$

or $$\sum_{i=h+1}^{d} \nabla_{\phi_i} \phi_i^{\mathrm{T}} \Sigma_X \phi_i = \sum_{i=h+1}^{d} \lambda_i \nabla_{\phi_i} (\phi_i^{\mathrm{T}}\phi_i - 1)$$

$$= \sum_{i=h+1}^{d} 2\Sigma_X \phi_i - \sum_{i=h+1}^{d} \lambda_i (2\phi_i) = 0 \qquad 2.30[4]$$

$$= \sum_{i=h+1}^{d} (\Sigma_X \phi_i - \lambda_i \phi_i) = 0$$

or $$\Sigma_X \phi_i = \lambda_i \phi_i \qquad 2.31$$

Equation 2.31 is a generalized eigenvalue problem having eigenvalue λ_i and the corresponding eigenvector ϕ_i.

Furthermore, substituting equation 2.31 in equation 2.28, we get

---

[4] Note for a vector b and a matrix A the following property holds true: $\nabla_b b^t A b = 2Ab$ and $\nabla_b b^t b = \nabla_b b^t I b = 2Ib = 2b$.

$$\begin{aligned} \text{MSE} &= \sum_{i=h+1}^{d} \phi_i^{\mathrm{T}} \Sigma_X \phi_i = \sum_{i=h+1}^{d} \phi_i^{\mathrm{T}} \lambda_i \phi_i \\ &= \sum_{i=h+1}^{d} \lambda_i \phi_i^{\mathrm{T}} \phi_i \qquad \because \phi_i^{\mathrm{T}} \phi_i = 1 \\ &= \sum_{i=h+1}^{d} \lambda_i \end{aligned} \tag{2.32}$$

It can be observed from equation 2.32 that the MSE is the sum of the least $d-h$ eigenvalues i.e. leading h eigenvectors corresponding to leading eigenvalues should be taken for dimension reduction and/or classification purposes for minimum MSE. Eigenvector matrix $\boldsymbol{\varphi}$ is adjusted such that its corresponding eigenvalues are in descending order i.e. $\lambda_1 > \lambda_2 > \ldots > \lambda_h$. In solving this transformation, **x** is assumed to be zero mean, if mean is not zero then **y** can be represented as

$$\mathbf{y} = \boldsymbol{\varphi}^{\mathrm{T}}(\mathbf{x} - \boldsymbol{\mu}) \tag{2.33}$$

where $\boldsymbol{\mu} = E[\mathbf{x}]$.

2.12.2 The PCA Reconstruction

Given **y** (from equation 2.33) one can simply transform it back to the original feature space with some finite reconstruction error. Applying back transformation we get reconstructed vector $\hat{\mathbf{x}}$ as

$$\hat{\mathbf{x}} = \boldsymbol{\varphi}\mathbf{y} + \boldsymbol{\mu} \tag{2.34}$$

Substituting equation 2.33 in equation 2.34 we get

$$\hat{\mathbf{x}} = \boldsymbol{\varphi}\boldsymbol{\varphi}^{\mathrm{T}}(\mathbf{x} - \boldsymbol{\mu}) + \boldsymbol{\mu} \tag{2.35}$$

If there is no dimension reduction then $\boldsymbol{\varphi}\boldsymbol{\varphi}^{\mathrm{T}} = \mathrm{I}_{d \times d}$ and the reconstructed vector $\hat{\mathbf{x}}$ is same as $\mathbf{x}$ i.e. $\hat{\mathbf{x}} = \mathbf{x}$. The following example shows PCA implementation to determine principal axes from a set of given 2-dimensional data.

Example 1

Figure 2.5 shows the projection of feature vectors from 2-dimensional space to 1-dimensional space by applying PCA. The following steps elaborate the PCA implementation procedure:

Step0. Generate a set of 2-dimensional data.

Step1. Find mean $\boldsymbol{\mu}$ and covariance Σ_X of the data.

Step2. Evaluate eigenvector matrix $\boldsymbol{\varphi}$ from the covariance matrix and arrange the obtained eigenvectors corresponding to its eigenvalues which is set in descending order i.e. the first vector ϕ_1 of $\boldsymbol{\varphi}$ should correspond to the maximum eigenvalue. The direction of first axis ϕ_1 is the direction of first principal axes which accounts for the maximum amount of variation. The second axis contains the maximum amount of variation orthogonal to the first.

Step4. Compute reconstruction vector $\hat{\mathbf{x}}$ using equation 2.35, which is projected onto the first principal axis (assume dimension reduction is from 2-dimensional space to 1-dimensional space).

Example 2

This example shows the reconstruction of speech data by PCA (figure 2.6).

Step0. Given a speech signal $\mathbf{x}$.

Step1. Obtain eigenvector set $\boldsymbol{\varphi}$ as described in example 1 (for mathematical simplicity we have constructed 2 dimensional space for speech signal, however, one can take any number of points to represent a vector).

Step2. Evaluate transformed feature vectors $\mathbf{y} = [y_1, y_2]^{\mathrm{T}}$.

Step3. Allocate appropriate number of bits to each of the $\mathbf{y}$ component, such that the signal-to-noise ratio (SNR) is maximum (for 8 bit, $SNR = 16.3$ when 5 bits are allocated to y_1 and 3 bits to y_2). For further details about bit allocation see Gersho and Gray (1992).

Step4. Obtain reconstructed speech signal $\hat{\mathbf{x}}$ using equation 2.35.

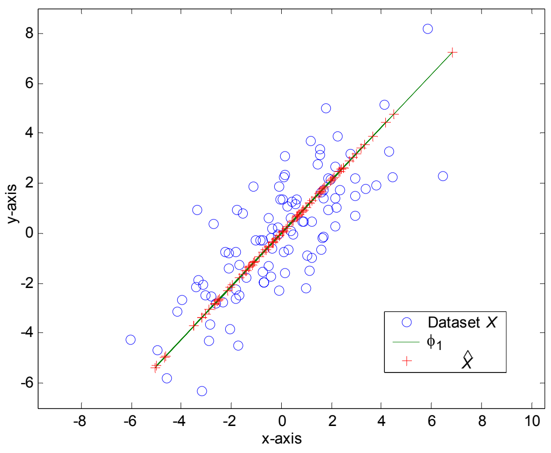


Figure 2.5: Application of PCA on 2-dimensional patterns

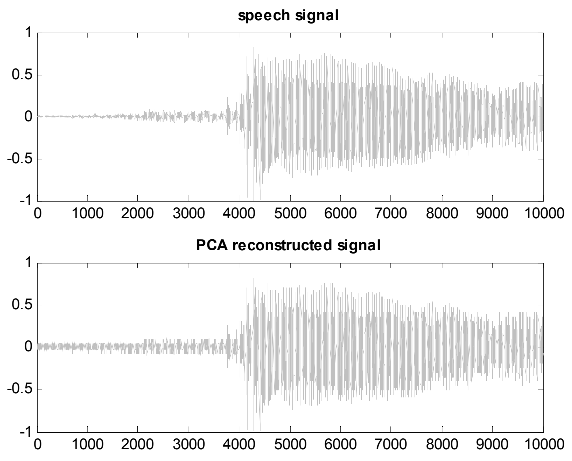


Figure 2.6: Reconstruction on speech signal using PCA

2.12.3 Drawbacks of PCA

1) One of the major drawbacks of PCA is that it requires standard eigenvalue decomposition for solving the transformation $\boldsymbol{\varphi}$ which is expensive in terms of processing time and storage (discussed later in chapter 3).
2) PCA is sensitive with respect to the units of measurement. If there are large differences between the variances of the attributes (dimensions) of feature vector $\mathbf{x}$, then those variables whose variances are largest will tend to dominate the first few principal components. This may be appropriate if all the attributes are measured in same units. However in practical cases, it often occurs that each attribute of feature vector is measured using different units, and therefore the projection obtained by PCA could be a biased estimate.
3) PCA transform is mathematically simple, but only globally linear.
4) PCA significantly distorts the original data topology.
5) PCA methods are based on least squares estimation techniques and thus fail to account for 'outliers' which are common in realistic training sets (De la Torre and Black, 2001).

2.13 Linear Discriminant Analysis

Linear discriminant analysis (LDA) is a well known technique for dimensionality reduction. It was first proposed by Fisher (1936) and later generalized by Rao (1948) to multi-class problems. Procedures that are analytically or computationally manageable in low-dimensional spaces can become completely impractical in a space of 50 or 100 dimensions (Duda and Hart, 1973). Therefore, dimension reduction becomes an essential and integral part of pattern classification problems. Dimensional reduction techniques are used in several areas like data compression, image processing, signal/speech processing etc. The LDA technique finds an orientation $\mathbf{W}$ that reduces a high dimensional feature vectors belonging to different classes to a lower dimensional feature space such that the projected feature vectors of a class on this lower dimensional space are well separated from the feature vectors of other classes. If the dimensionality reduction is from d-dimensional ($\mathbf{R}^d$) space to h-dimensional ($\mathbf{R}^h$) space (where $h < d$) then the size of the orientation matrix **W** would be $d \times h$. Therefore **W** has h column vectors known as the basis vectors. The orientation **W** is evaluated so that the Fisher's criterion function $J(\mathbf{W})$ is maximum. The criterion function depends on three factors: orientation **W**, within-class scatter matrix (S_W) and between-class scatter matrix (S_B). For a c-class problem the value of h will be $c-1$ or less, a constraint due to S_B. In the basic or conventional LDA technique, the orientation **W** is computed by using eigenvalue decomposition (EVD) method. LDA is illustrated here for 2-class problem and for multi-class problem. Next section depicts LDA for 2-class problem.

2.13.1 Two-class Linear Discriminant Analysis

In a two class problem dimension reduction is from d-dimensional space to 1-dimensional plane. To elaborate this suppose in a two-class problem a set of n feature vectors $\mathcal{X} = \{\mathbf{x}_j : j = 1,2,\ldots n\}$ are given, where $\mathbf{x}_j \in \mathbf{R}^d$ (d-dimensional space). Assume n_1 vectors of set $\mathcal{X}_1$ (where $\mathcal{X}_1 \subset \mathcal{X}$) belongs to class ω_1 and the remaining vectors ($n_2 = n - n_1$) from $\mathcal{X}_2$ (where $\mathcal{X}_2 \subset \mathcal{X}$) belongs to the second class ω_2. Let $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$ be the d-dimensional sample mean of the two class set represented as:

$$\boldsymbol{\mu}_i = \frac{1}{n_i} \sum_{\mathbf{x} \in \mathcal{X}_i} \mathbf{x} \qquad \text{for } i = 1,2 \qquad 2.36$$

Between-class scatter matrix S_B and within-class scatter matrix S_W are defined as:

$$S_B = (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^{\mathrm{T}} \qquad 2.37$$

$$S_W = \sum_{\mathbf{x} \in \mathcal{X}_i} (\mathbf{x} - \boldsymbol{\mu}_i)(\mathbf{x} - \boldsymbol{\mu}_i)^{\mathrm{T}} \qquad \text{for } i = 1,2 \qquad 2.38$$

The orientation vector w (of size $d \times 1$) is taken so that the criteria $J(w)$ is maximum, where $J(w)$ can be represented in terms of S_B , S_W and w as:

$$J(w) = \frac{w^{\mathrm{T}} S_B w}{w^{\mathrm{T}} S_W w} \qquad 2.39$$

The value of orientation vector w could be investigated by evaluating the optimum value of criteria function $J(w)$. The following theorem and corollary explore the value of w.

Theorem 1

Let A and B be two symmetric matrices. Suppose that B is semidefinite. Then the maximum (minimum) of $w^{\mathrm{T}} Aw$ given

$$w^{\mathrm{T}} Bw = 1 \qquad 2.40$$

is attained when x is the eigenvector of $B^{-1}A$ corresponding to the largest (smallest) eigenvalue of $B^{-1}A$. Thus if λ and δ are the largest (smallest) eigenvalues of $B^{-1}A$, then, subject to constraint (equation 2.40),

$$\max_x [w^{\mathrm{T}} Aw] = \lambda, \qquad \min_x [w^{\mathrm{T}} Aw] = \delta$$

Proof of theorem 1 can be referred in Mardia et. al. (1979). This theorem is useful in proving the following corollary.

Corollary 1

Let S_B and S_W be two symmetric matrices. Suppose that S_W is semidefinite. If $J(w) = w^{\mathrm{T}} S_B w / w^{\mathrm{T}} S_W w$ then, for $w \neq 0$, $\mathbf{max}[J(w)] = \lambda$ satisfies $S_B w = \lambda S_W w$.

Proof: Since $J(w)$ is invariant under changes of scale of w, we can regard the problem as maximizing $w^{\mathrm{T}} S_B w$ given $w^{\mathrm{T}} S_W w = C$ (generalize form of equation 2.40), where $C > 0$.

Let $g(w) = w^{\mathrm{T}} S_W w - C = 0$ and $f(w) = w^{\mathrm{T}} S_B w$, then to maximize $f(w)$ given condition $g(w) = 0$, we use Langrange's multiplier and could write the following differential equality:

$$\nabla_w f(w) = \lambda \nabla_w g(w)$$

or $$\nabla_w (w^{\mathrm{T}} S_B w) = \lambda \nabla_w (w^{\mathrm{T}} S_W w - C)$$

i.e. $$2 S_B w = 2 \lambda S_W w$$

if S_W is non-singular then

$$S_W^{-1} S_B w = \lambda w \qquad 2.41$$

Equation 2.41 becomes a conventional eigenvalue problem, where matrix $S_W^{-1} S_B$ has λ eigenvalue. From the obtained eigenvector *w*, the projected samples of the corresponding feature vector $\mathbf{x}$ can be expressed as:

$$y = w^{\mathrm{T}} \mathbf{x} \qquad 2.42$$

The mapping is from *d*-dimensional space to 1-dimensional space ($w : \mathbf{R}^d \to \mathbf{R}^1$). This mapping is many-to-one and therefore it is not possible to reduce minimum achievable error.

2.13.2 Multi-class Linear Discriminant Analysis

To explicitly define S_B and S_W for the Fisher's criterion function, in a *c*-class (assuming $c > 2$) problem let $\mathcal{X}$ denotes *d*-dimensional set of *n* feature vectors,

$\Omega = \{\omega_i : i = 1,2,...,c\}$ be the finite set of c states of nature or class labels where ω_i denotes the i^{th} class label. The set $\mathcal{X}$ can be subdivided into c subsets $\mathcal{X}_1, \mathcal{X}_2,\ldots, \mathcal{X}_c$ where each subset $\mathcal{X}_i$ belongs to ω_i and consists of n_i number of samples such that:

$$n = \sum_{i=1}^{c} n_i \qquad 2.43$$

The samples or patterns of set $\mathcal{X}$ can be written as:

$$\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\} \text{ where } \mathbf{x}_j \in \mathbf{R}^d$$

$$\mathcal{X}_i \subset \mathcal{X} \text{ and } \mathcal{X}_1 \cup \mathcal{X}_2 \cup \ldots \cup \mathcal{X}_c = \mathcal{X}.$$

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

$$S_B = \sum_{j=1}^{c} n_j (\boldsymbol{\mu}_j - \boldsymbol{\mu})(\boldsymbol{\mu}_j - \boldsymbol{\mu})^{\mathrm{T}} \qquad 2.44$$

It can be observed from equation 2.44 that S_B is the sum of c matrices of rank one or less, and because only $c-1$ of these are independent, S_B is of rank $c-1$ or less (Duda and Hart, 1973).

The within-class scatter matrix which is the sum of c scatter matrices is defined as

$$S_W = \sum_{i=1}^{c} S_i \qquad 2.45$$

where $S_i = \sum_{\mathbf{x} \in \mathcal{X}_i} (\mathbf{x} - \boldsymbol{\mu}_j)(\mathbf{x} - \boldsymbol{\mu}_j)^{\mathrm{T}}$ 2.46

It can be observed from equations 2.45 and 2.46 that S_W is the sum of c scatter matrices and each of the scatter matrices is the sum of n_i matrices, and because only $c(n_{avg}-1)$ or less are independent (where $n_{avg} = \frac{1}{c}\sum_{j=1}^{c} n_j = n/c$), the rank of S_W (for $n \geq c$) is

$$rank(S_W) \leq c(n_{avg}-1) = n-c \tag{2.47}$$

If $n-c \geq d$ then S_W is full rank matrix i.e. non-singular and its inversion is possible. Now given scatter matrices S_B and S_W we can define Fisher's criterion as a function of $\mathbf{W}$ as (Duda and Hart, 1973)

$$J(\mathbf{W}) = \frac{|\mathbf{W}^{\mathrm{T}} S_B \mathbf{W}|}{|\mathbf{W}^{\mathrm{T}} S_W \mathbf{W}|} \tag{2.48}$$

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

$$S_W^{-1} S_B \mathbf{w}_i = \lambda_i \mathbf{w}_i \tag{2.49}$$

where $\mathbf{w}_i$ are the column vectors of $\mathbf{W}$ that correspond to the largest eigenvalues (λ_i) in equation 2.49. It is evident from equation 2.49 that the explicit solution of the orientation can be found when S_W is non-singular. If S_W is singular (i.e. $n-c<d$) then it is not possible to obtain the orientation $\mathbf{W}$ by using equation 2.49.

Figure 2.7 depicts LDA and PCA transformation on 2-dimensional Gaussian data from 2D-space to 1D-space. It could be observed from the figure that the transformed vectors of LDA in lower dimensional space provide best discrimination between the given classes. On the other hand, PCA projects data on lower dimensional space that best describes the representation of the data. This means PCA investigates the direction that is useful in representation and LDA searches for the direction that is optimum in discrimination among the given classes.

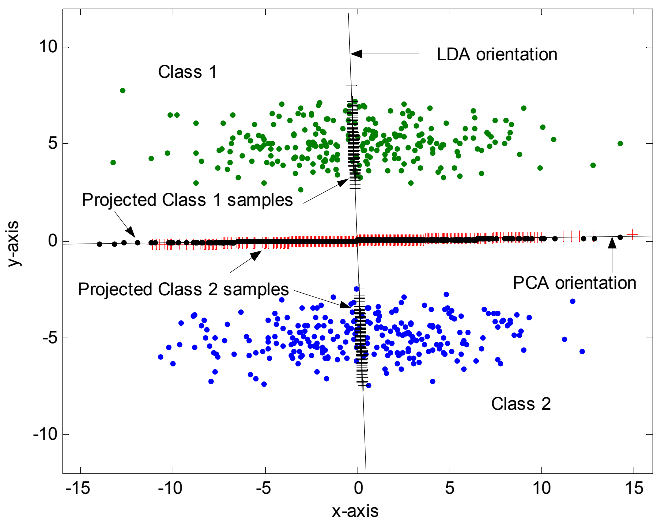


Figure 2.7: LDA and PCA projections

2.13.3 Drawbacks of LDA

1) A major drawback of LDA is the problem of singularity of within-class scatter matrix S_W due to the small sample size. This problem arises whenever the number of samples is smaller than the dimensionality of samples. For example, a 32×32 image data in a face recognition system has 1024 dimensions, which requires more than 1024 training samples to ensure that S_W is non-singular. So, LDA is not a stable method in practice when the training data are scarce.
2) LDA cannot extract more features than the number of classes minus one. In the two-class case this means that only a reduction to one dimension is possible.
3) LDA provides only one transformation matrix over whole data, it is not sufficient to discriminate the complex data consisting of many classes like human faces.
4) In LDA, the training just accounts to estimate the mean and covariance matrix, regardless of how many samples are available.

5) For many classes LDA projection onto lower dimensional space usually causes overlap of neighbouring classes, which is certainly suboptimal.

2.13.4 Pattern Classification Using Linear Discriminant Analysis

LDA gives features that are most discriminative which is ideal for pattern classification applications. Firstly, the parameters (e.g. orientation **W**) from the known training dataset can be obtained simply by applying the LDA technique. Then in the classification or testing phase an unknown class label feature vector **x** is associated to the class label of the nearest class using some distance measure. Usually, nearest neighbour (NN) technique is applied to associate the class label of **x**. To elaborate this in mathematical terms suppose unlabelled feature vector $\mathbf{x} \in \mathbf{R}^d$ and projected feature vector (using orientation **W**) is $\mathbf{y} \in \mathbf{R}^h$. Let $\boldsymbol{\mu}_j$ be the centroid of jth class in d-dimensional space and $\hat{\boldsymbol{\mu}}_j$ be the corresponding centroid in h-dimensional space (where $h < d$). The parameters ($\hat{\boldsymbol{\mu}}_j$ and **W**) are stored during the training phase which will be used in the classification phase. The NN technique is applied in the following manner to decide the membership of the feature vector **x**:

$$\mathbf{y} = \mathbf{W}^{\mathrm{T}}\mathbf{x} \qquad 2.50$$

$$k = \mathbf{arg}\min_{j=1}^{c} \| \mathbf{y} - \hat{\boldsymbol{\mu}}_j \| \qquad 2.51$$

where $\| \bullet \|$ is the normalized value. Associate the class label ω_k to the feature vector **x**.

2.13.5 GMM plus LDA for Pattern Classification

The performance of LDA classifier in terms of classification accuracy can be improved by using gaussian mixture model (GMM) in place of NN in the classification phase. If GMM is used then covariance of each class in h-dimensional should also be evaluated in the training phase. This covariance matrix is used in the classification phase. For brevity we call this method as GMM plus LDA. The following procedure is applied for GMM plus LDA in the classification phase:

$$\mathbf{y} = \mathbf{W}^{\mathrm{T}}\mathbf{x}$$

$$p_j(\mathbf{y}) = \frac{1}{(2\pi)^{h/2}\,|\Sigma_{\mathbf{y}_j}|^{1/2}}\mathbf{exp}[-\tfrac{1}{2}(\mathbf{y}-\hat{\boldsymbol{\mu}}_j)^{\mathrm{T}}\Sigma_{\mathbf{y}_j}^{-1}(\mathbf{y}-\hat{\boldsymbol{\mu}}_j)] \quad \text{for } j=1\ldots c \qquad 2.52$$

$$k = \mathbf{arg}\max_{j=1}^{c} p_j(\mathbf{y}) \qquad 2.53$$

Finally, associate the class label ω_k to the feature vector **x**. The improvement of GMM plus LDA over the basic LDA is illustrated on SatImage database (figure 2.8) and multiple feature digit (using Zernike moments) database (figure 2.9). The detailed information about the databases used is described in section 5.3.10. In both the figures, *x*-axes represent the dimension and y-axes represent the classification accuracy. It can be observed from both the figures that the GMM plus LDA is performing better than the basic LDA technique. As the dimension is increased the improvement in terms of classification accuracy is evident from the figures.

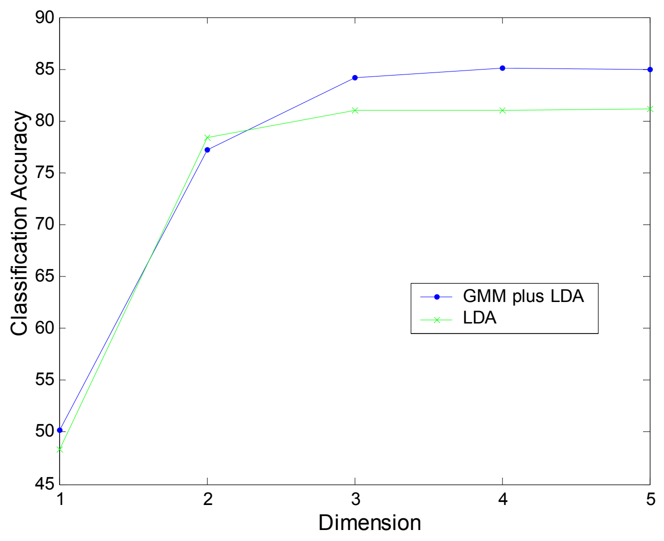


Figure 2.8: A comparison between LDA and GMM plus LDA on SatImage database.

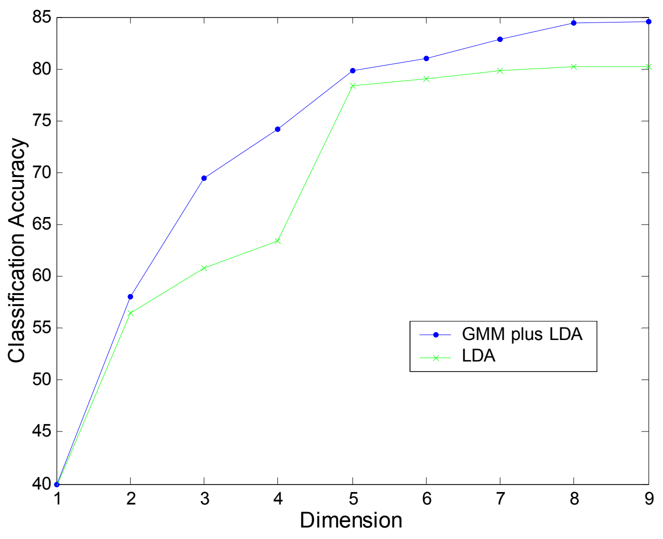


Figure 2.9: A comparison between LDA and GMM plus LDA on multiple feature digit – Zernike moments database.

2.14 Summary

The Gaussian mixture model (GMM) for pattern classification has been reviewed. Its basic concepts and theories are illustrated. The model is discussed for the supervised learning tasks. We have also briefly discussed the expectation-maximization (EM) algorithm. Some weaknesses of GMM are also described. We have also surveyed some of the conventional linear classifiers used today. Their properties and functionalities are described for understanding the behaviour of the classifiers. For each of the models, training and classification phases are elaborated. Next we looked at the two most common techniques for dimension reduction and/or pattern classification: principal component analysis (PCA) and linear discriminant analysis (LDA). Their characteristics, functionalities and drawbacks are discussed. A combined technique using GMM and LDA is presented which is giving improved performance in terms of classification accuracy as compared to the basic LDA technique.

Chapter 3

Fast Principal Component Analysis using Fixed-Point Algorithm

3.1 Abstract

In this chapter we present an efficient way of computing principal component analysis (PCA). The algorithm finds the desired number of leading eigenvectors with a computational cost that is much less than that from the eigenvalue decomposition (EVD) based PCA method. The mean squared error (MSE) generated by the proposed method is very similar to the EVD based PCA method that is the proposed algorithm can be used with negligibly scarifying the performance (in MSE sense).

3.2 Introduction

Principal component analysis (PCA) finds a linear transformation $\boldsymbol{\varphi}$ which reduces *d*-dimensional feature vectors to *h*-dimensional feature vectors (where $h < d$) in such a way that the information is maximally preserved in minimum mean squared error sense. This linear transformation is known as PCA transform or Karhunen-Loéve transform (KLT) (Fukunaga, 1990). The size of transformation $\boldsymbol{\varphi}$ is $d \times h$. See section 2.12 for details about PCA.

The computation of PCA requires eigenvalue decomposition (EVD) of the covariance matrix of the feature vectors. One well-known EVD method is the Cyclic Jacobi's method (Golub, 1996). The Jacobi's method which diagonalizes a symmetric matrix requires around $O(d^3 + d^2 n)$ computations (Golub, 1996) (where n is the number of feature vectors or samples used). This computation in many applications (e.g. fixed-point implementation) is undesirable. A number of methods have been proposed in the literature for computing the PCA transform with reduced computational complexity. Reddy and Herron (Reddy and Herron, 2001) proposed modification to Jacobi's method which favours fixed-point implementations. Their computational complexity is, however, still of order $O(d^3)$ for each symmetric rotation (assuming symmetric matrix of size $d \times d$ is previously computed). Basically the improvement in computational complexity is negligible. Roweis (1997) proposed expectation maximizing (EM) algorithm for PCA, which is computationally effective than EVD method for PCA. But this technique uses EM algorithm which could be expensive in time. It also requires matrix inverse[1] computation in both the E-step and M-step for each of the iteration, which is an expensive exercise. Furthermore, EM algorithm does not converge to a global maximum; it achieves only a local maximum and thus the choice of the initial guess used in the algorithm becomes crucial. The power method (Schilling and Harris, 2000) is also used to find leading eigenvector which is less expensive method but can compute only one most leading eigenvector. Another method is snap-shot algorithm (Sirovich, 1987) which does not explicitly compute sample covariance matrix, however, requires matrix inversion and is based on the assumption that the eigenvectors being searched are linear combinations of feature vectors.

In this chapter we present a computationally-fast technique for finding the desired number of leading eigenvectors without diagonalizing any symmetric matrix. Thus it avoids Cyclic Jacobi's method for eigendecomposition. We have used the fixed-point algorithm (Hyvärinen and Oja, 1997) to find all the h leading eigenvectors which converges in just a few iterations without the need of any initial setting. Furthermore, it is free from matrix inverse computations. As a result, the presented algorithm is computationally efficient, consumes very small amount of computation time and is very easy to implement. For brevity we call this method as Fast PCA. This Fast PCA generates approximately the same amount of mean squared error as that generated by EVD based PCA method.

[1] Note that the matrix inversion scales as the cube of the matrix size.

3.3 PCA Revisited

The PCA transform can be found by minimizing mean squared error. To see this, let the feature vector be $\mathbf{x} \in \mathbf{R}^d$ (*d*-dimensional space), reduced dimensional feature vector be $\mathbf{y} \in \mathbf{R}^h$ and reconstructed feature vector be $\hat{\mathbf{x}} \in \mathbf{R}^d$. Then the mean squared error can be represented as

$$\text{MSE} = E[\| \mathbf{x} - \hat{\mathbf{x}} \|^2] \qquad 3.1$$

where $E[\bullet]$ is the expectation operation with respect to $\mathbf{x}$ and $\| \bullet \|^2$ is the norm squared value. We know that PCA transformation $\boldsymbol{\varphi}$ is of size $d \times h$ and it is used to do dimensionality reduction from *d*-dimensional space to *h*-dimensional feature space, i.e. $\boldsymbol{\varphi} : \mathbf{x} \rightarrow \mathbf{y}$ or $\mathbf{y} = \boldsymbol{\varphi}^{\mathrm{T}} \mathbf{x}$. It should be noted that in this transformation, **x** is assumed to be zero mean, if mean is not zero then **y** can be represented as

$$\mathbf{y} = \boldsymbol{\varphi}^{\mathrm{T}} (\mathbf{x} - \boldsymbol{\mu}) \qquad 3.2$$

where $\boldsymbol{\mu} = E[\mathbf{x}]$**.** Given **y** (from equation 3.2) one can simply transform it back to the original feature space with some finite reconstruction error. Applying back transformation we get reconstructed vector $\hat{\mathbf{x}}$ as

$$\hat{\mathbf{x}} = \boldsymbol{\varphi} \mathbf{y} + \boldsymbol{\mu} \qquad 3.3$$

Substituting equation 3.2 in equation 3.3 we get

$$\hat{\mathbf{x}} = \boldsymbol{\varphi} \boldsymbol{\varphi}^{\mathrm{T}} (\mathbf{x} - \boldsymbol{\mu}) + \boldsymbol{\mu} \qquad 3.4$$

Equation 3.1 can be rewritten by utilizing equation 3.4 as

$$\text{MSE} = E[\| (\mathrm{I}_{d\times d} - \boldsymbol{\varphi}\boldsymbol{\varphi}^{\mathrm{T}})(\mathbf{x}-\boldsymbol{\mu}) \|^2] \qquad 3.5$$

The desired h basis vectors of $\boldsymbol{\varphi}$ span the d-dimensional subspace and are mutually orthonormal, i.e. $\boldsymbol{\varphi}^{\mathrm{T}}\boldsymbol{\varphi} = \mathrm{I}_{h\times h}$. Using the orthonormality condition equation 3.5 can be further simplified as

$$\text{MSE} = E[g(\boldsymbol{\varphi}, \mathbf{x})] \qquad 3.6$$

where $g(\boldsymbol{\varphi}, \mathbf{x})$ is a scalar function and is given by

$$g(\boldsymbol{\varphi}, \mathbf{x}) = (\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}(\mathrm{I}_{d\times d} - \boldsymbol{\varphi}\boldsymbol{\varphi}^{\mathrm{T}})(\mathbf{x}-\boldsymbol{\mu}) \qquad 3.7$$

From appendix 3.1 we can write the derivative of $E[g(\boldsymbol{\varphi}, \mathbf{x})]$ with respect to $\boldsymbol{\varphi}$ as

$$\frac{\partial}{\partial \boldsymbol{\varphi}} E[g(\boldsymbol{\varphi}, \mathbf{x})] = -2E[(\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\varphi}] \qquad 3.8$$

Given equation 3.8 and the orthonormality condition of $\boldsymbol{\varphi}$ we can apply fixed-point algorithm (Hyvärinen and Oja, 1997) to solve for the values of $\boldsymbol{\varphi}$ i.e.

$$\boldsymbol{\varphi} \leftarrow E[(\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}]\boldsymbol{\varphi} \qquad 3.9$$
$$\boldsymbol{\varphi} \leftarrow \textit{orthonormalize}\,(\boldsymbol{\varphi}) \qquad 3.10$$

Note that the negative sign is removed from equation 3.9 since $\boldsymbol{\varphi}$ and $-\boldsymbol{\varphi}$ define the same direction. Moreover, since the expectation is with respect to $\mathbf{x}$, transformation $\boldsymbol{\varphi}$ is taken out of it. The algorithm can also be represented as

$$\boldsymbol{\varphi} \leftarrow \Sigma_{\mathbf{x}}\,\boldsymbol{\varphi} \qquad 3.10$$
$$\boldsymbol{\varphi} \leftarrow \textit{orthonormalize}\,(\boldsymbol{\varphi}) \qquad 3.11$$

where $\Sigma_{\mathbf{x}} = E[(\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}]$ is the covariance of $\mathbf{x}$. The orthonormalization of $\boldsymbol{\varphi}$ can be done by using Gram-Schmidt orthonormalization procedure. Thus the computation of h eigenvectors does not require eigenvalue decomposition procedure. This means that the Cyclic Jacobi's method can be altogether avoided for eigendecomposition. This would certainly reduce the computation complexity in finding eigenvectors for PCA. Furthermore, use of fixed-point algorithm converges the algorithm very fast in just a few iterations without the need of any prior knowledge of the learning rate/step size or initial settings as per required by the gradient descent based and/or EM based methods. The next section depicts the iterative algorithm for computing eigenvectors for PCA.

3.4 Fast PCA Algorithm

One way to compute all the h orthonormal basis vectors is to use Gram-Schmidt method. The most dominating/leading eigenvector or principal axis will be measured first. Similarly, all the remaining $h-1$ basis vectors (orthonormal to the previously measured basis vectors) will be measured one by one in a reducing order of dominance. The previously measured $(p-1)^{\text{th}}$ basis vectors will be utilized for finding the p^{th} basis vector. The algorithm for p^{th} basis vector will converge when the new and old values $\boldsymbol{\varphi}_p$ point in the same direction i.e. $\boldsymbol{\varphi}_p^{+\mathrm{T}}\boldsymbol{\varphi}_p \approx 1$ (where $\boldsymbol{\varphi}_p^{+}$ is the new value of $\boldsymbol{\varphi}_p$). It is usually economical to use a finite tolerance error to satisfy convergence criteria

$$abs(\boldsymbol{\varphi}_p^{+\mathrm{T}}\boldsymbol{\varphi}_p - 1) < \varepsilon \qquad 3.12$$

where ε is a predefined tolerance or threshold and $abs(\bullet)$ is the absolute value. The Fast PCA algorithm is illustrated in table 3.1.

3.4.1 Computational Complexity of the Algorithm

Let L be the number of iterations used in converging the algorithm for $\boldsymbol{\varphi}_p$ and n be the number of samples used. Then the estimated computational complexity is given in table 3.2.

The value of L is quite small (usually 2 ~ 5) and therefore the computational complexity can be estimated to be $O(d^2h + d^2n)$. If dimension d is large compared to h and n then computational complexity can be estimated to be $O(d^2)$.

TABLE 3.1: Fast PCA algorithm for computing leading eigenvectors

```
1. Choose $h$, the number of principal axes or eigenvectors required to estimate.
   Compute covariance $\Sigma_{\mathbf{x}}$ and set $p \leftarrow 1$.
2. Initialize eigenvector $\boldsymbol{\varphi}_p$ of size $d \times 1$ e.g. randomly.
3. Update $\boldsymbol{\varphi}_p$ as $\boldsymbol{\varphi}_p \leftarrow \Sigma_{\mathbf{x}} \boldsymbol{\varphi}_p$.
4. Do the Gram-Schmidt orthogonalization process
       $\boldsymbol{\varphi}_p \leftarrow \boldsymbol{\varphi}_p - \sum_{j=1}^{p-1} (\boldsymbol{\varphi}_p^{\mathrm{T}} \boldsymbol{\varphi}_j) \boldsymbol{\varphi}_j$
5. Normalize $\boldsymbol{\varphi}_p$ by dividing it by its norm: $\boldsymbol{\varphi}_p \leftarrow \boldsymbol{\varphi}_p / \| \boldsymbol{\varphi}_p \|$.
6. If $\boldsymbol{\varphi}_p$ has not converged, go back to step 3.
7. Increment counter $p \leftarrow p + 1$ and go to step 2 until $p$ equals $h$.
```

TABLE 3.2: Computational complexity of the algorithm

Major processing steps involved in the algorithm	**Computational complexity**
Covariance Σ_x (step 1)	$O(d^2 n)$
Gram-Schmidt orthogonalization for $\boldsymbol{\varphi}_p$ (p^{th} basis vector)	$O(dpL)$
Gram-Schmidt orthogonalization for all $p = 1 \ldots h$ basis vectors	$O(dh^2 L)$
Updating process for all $p = 1 \ldots h$ basis vectors (step 3)	$O(d^2 hL)$
Total estimated	$O(d^2 hL + d^2 n) \approx O(d^2 h + d^2 n)$

3.4.2 An Illustration

To verify the performance of the Fast PCA method in terms of processing time and mean squared error we have compared it with the performance of EVD based method for PCA. For the processing time, the cputime[2] is evaluated while increasing the data dimensionality. Similarly, mean squared error is evaluated while increasing the data dimensionality. We have generated uniformly distributed random vectors of dimensions starting from 100 and going up to 4000. The number of samples or feature vectors is fixed and is 100. The dimension is reduced to $h = 10$ in all the cases and the tolerance (equation 3.12) is set to 0.01 in the verification of the algorithm.

Figure 3.1 is illustrating mean squared error for both the methods as a function of data dimensionality. In the figure '+' sign indicates PCA using EVD method and '.' sign indicates Fast PCA method. It can be observed that the errors generated by both the methods are very close to each other for all the data dimensions. From this it can be inferred that the difference in performance of these methods in terms of the mean squared error is negligible.

[2] cputime is a MATLAB keyword that returns the CPU time in seconds for any process.

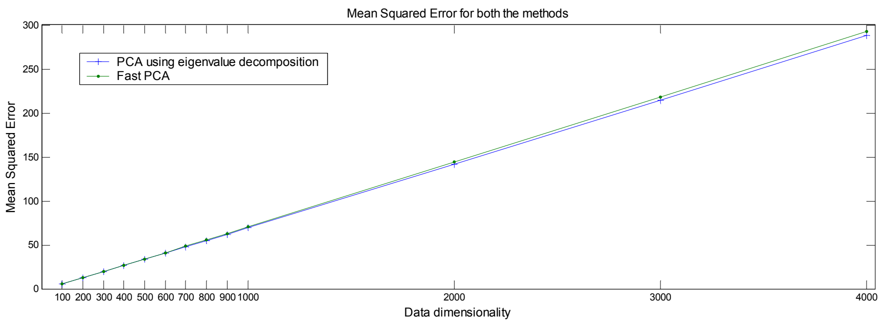


Figure 3.1: Mean squared error for PCA using EVD and Fast PCA as a function of data dimensionality.

Figure 3.2 depicts the cputime consumed in finding the *h* leading eigenvectors for both the methods as a function of data dimensionality. The data dimensions from 100 to 1000 are illustrated in figure 3.2 and dimensions from 2000 to 4000 are illustrated in table 3.3. It is evident from figure 3.2 that cputime curve for EVD based PCA increases exponentially as the data dimensionality is increased. Moreover, from table 3.3 cputime for dimensions above 2000 are very expensive for EVD based PCA method and may restrict practical applications which involve such high dimensions. For dimension 4000 the EVD based PCA method is consuming around 1413 cputime in seconds to find 10 leading eigenvectors. On the other hand, it can be observed from figure 3.2 as well as from table 3.3 that the cputime for Fast PCA method is very economical for all the dimensions. Even for data dimensionality 4000, cputime is only 7.53 seconds, which is extremely low as compared to EVD based PCA method. Thus Fast PCA can also be efficiently used for high dimensional applications.

It can be concluded from figure 3.2 and table 3.3 that the Fast PCA method is highly efficient in finding the leading eigenvectors in terms of time due to its reduced computational complexity. We have also tested (not shown in this paper) the Fast PCA algorithm for $h = 100$ and we found that it is still very economical than EVD based PCA method.

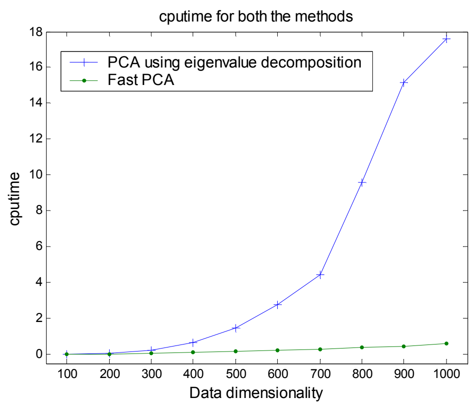


Figure 3.2: cputime for PCA using EVD method and Fast PCA method as a function of data dimensionality from 100 to 1000.

TABLE 3.3: cputime for PCA using EVD method and Fast PCA method as a function of data dimensionality from 2000 to 4000.

Data dimensionality	EVD based PCA method (cputime in seconds)	Fast PCA method (cputime in seconds)
2000	153.26	2.28
3000	531.56	5.30
4000	1413.72	7.53

3.5 Summary

In this chapter we have presented a method (called Fast PCA) for computing the PCA transformation with reduced computational cost. This Fast PCA method utilizes fixed-point algorithm. The proposed method is compared with the EVD based PCA method. It was seen that the Fast PCA method is computationally effective and economical in finding small number of leading eigenvectors or orthonormal axes for PCA as compared to the EVD based PCA method. The mean squared error provided by proposed method is also very close to the MSE provided by EVD based PCA method.

Appendix 3.1

Lemma 1: Let the scalar function $g(\boldsymbol{\varphi}, \mathbf{v}) = \mathbf{v}^{\mathrm{T}}(\mathrm{I}_{d\times d} - \boldsymbol{\varphi}\boldsymbol{\varphi}^{\mathrm{T}})\,\mathbf{v}$ be a differentiable function of a $d \times h$ rectangular matrix $\boldsymbol{\varphi}$ such that $h < d$. Suppose $\mathbf{v}$ is any vector of size $d \times 1$. Then the gradient of $g(\boldsymbol{\varphi}, \mathbf{v})$ is defined as $\nabla_{\boldsymbol{\varphi}} g(\boldsymbol{\varphi}, \mathbf{v}) = -2\,\mathbf{v}\mathbf{v}^{\mathrm{T}}\boldsymbol{\varphi}$.

Proof I: The scalar function $g(\boldsymbol{\varphi}, \mathbf{v})$ can be simplified as $g(\boldsymbol{\varphi}, \mathbf{v}) = \mathbf{v}^{\mathrm{T}}\mathbf{v} - \mathbf{v}^{\mathrm{T}}\boldsymbol{\varphi}\boldsymbol{\varphi}^{\mathrm{T}}\mathbf{v}$. Its derivative with respect to $\boldsymbol{\varphi}$ is given as

$$\begin{aligned}\partial g(\boldsymbol{\varphi}, \mathbf{v}) &= \mathit{trace}[\partial(\mathbf{v}^{\mathrm{T}}\mathbf{v})] - \mathit{trace}[\partial(\mathbf{v}^{\mathrm{T}}\boldsymbol{\varphi}\boldsymbol{\varphi}^{\mathrm{T}}\mathbf{v})] \\ &= -\{\mathit{trace}[\mathbf{v}^{\mathrm{T}}\,\partial\,\boldsymbol{\varphi}\boldsymbol{\varphi}^{\mathrm{T}}\mathbf{v}] + \mathit{trace}[\mathbf{v}^{\mathrm{T}}\boldsymbol{\varphi}\,\partial\,\boldsymbol{\varphi}^{\mathrm{T}}\mathbf{v}]\} \\ &= -2\mathit{trace}[\mathbf{v}\mathbf{v}^{\mathrm{T}}\boldsymbol{\varphi}\,\partial\,\boldsymbol{\varphi}^{\mathrm{T}}] \;\{\because \mathit{tr}(\mathrm{A}^{\mathrm{T}}) = \mathit{tr}(\mathrm{A}) \text{ and } \mathit{tr}(\mathrm{AB}) = \mathit{tr}(\mathrm{BA})\end{aligned}$$

or $$\nabla_{\boldsymbol{\varphi}} g(\boldsymbol{\varphi}, \mathbf{v}) = -2\,\mathbf{v}\mathbf{v}^{\mathrm{T}}\boldsymbol{\varphi}$$

Chapter 4

A Gradient Linear Discriminant Analysis for Small Sample Sized Problem

4.1 Abstract

The purpose of conventional linear discriminant analysis (LDA) is to find an orientation which projects high dimensional feature vectors of different classes to a more manageable low dimensional space in the most discriminative way for classification. The LDA technique utilizes an eigenvalue decomposition (EVD) method to find such an orientation. This computation is usually adversely affected by the small sample size problem. In this chapter we have presented a new direct LDA method (called gradient LDA) for computing the orientation especially for small sample size problem. The gradient descent based method is used for this purpose, however, the technique does not require any learning rate parameter. It also avoids discarding the null space of within-class scatter matrix and between-class scatter matrix which may have discriminative information useful for classification.

4.2 Introduction

Linear discriminant analysis (LDA) is a well known technique for dimensionality reduction. It finds an orientation **W** that reduces high dimensional feature vectors belonging to different classes to a lower dimensional feature space such that the projected feature vectors of a class on this lower dimensional space are well separated from the feature vectors of other classes. If the dimensionality reduction is from *d*-dimensional ($\mathbf{R}^d$) space to *h*-dimensional ($\mathbf{R}^h$) space (where $h < d$) then the size of the orientation matrix **W** would be $d \times h$. Therefore **W** has *h* column vectors known as the basis vectors. The orientation **W** is evaluated so that the Fisher's criterion function $J(\mathbf{W})$ is maximum. The criterion function depends on three factors:

orientation **W**, within-class scatter matrix (S_W) and between-class scatter matrix (S_B). For a c-class problem the value of h will be $c-1$ or less, a constraint due to S_B. In the basic or conventional LDA technique, the orientation **W** is computed by using eigenvalue decomposition (EVD) method where scatter matrix S_W is arranged in such a way that it restricts the computation of **W** if it is being singular or reduced rank matrix. This limitation (quite often arises in human face recognition problem) is due to the high dimensionality of original feature vectors in comparison with the low number of feature vectors available. This drawback of LDA is known as small sample size problem (Fukunaga, 1990). To overcome this problem, several authors (Swets and Weng, 1996; Belhumeur et al. 1997, Zhao et al.,1998, 1999, Sharma et al. 2006) have used intermediate techniques like principal component analysis (PCA) prior to the application of LDA. The PCA technique is used in such a way that the projected feature vectors on h-dimensional space give a full rank S_W matrix. Thereby the computation of the inverse of S_W is feasible and thus orientation **W** can then be found by the basic LDA method. The application of intermediate techniques would, however, sacrifice some classification performance. There are some techniques recently developed to solve small sample size problems. Chen et al. (2000) have proposed a new LDA-based method. Their new LDA is based on the modified Fisher's criterion and involves discarding the null space of S_W, which contains the most discriminative information useful for classification (Chen et al., 2000; Yu and Yang, 2001). Yu and Yang (2001) presented a direct LDA method which discards the null space of S_B, however, prevents discarding the null space of S_W. Lu et al. (2003) presented an approach based on the combination of direct LDA and fractional-step LDA (Lotlikar and Kothari, 2000) methods that overcomes shortcomings and limitations of individual methods used in the combination.

In this paper we do not extend any techniques presented in (Chen et al., 2000; Yu and Yang, 2001; Lotlikar and Kothari, 2000; Lu et al. 2003). However, we have presented a new way of computing the orientation **W** which is derived directly from the conventional LDA technique. We used gradient descent method to solve for the orientation **W** in such a way that the initial setting or learning rate parameter is not required. This can be made possible by substituting learning rate parameter to be unity and by using $J(\mathbf{W})$ adaptively in the iterative process. This makes the convergence fast and reliable which is empirically presented. For brevity we call the proposed technique as gradient LDA technique. The gradient LDA technique can compute orientation $\mathbf{W}$ for both singular and non-singular S_W. This technique does not discard any null spaces of S_W and S_B thereby preserving discriminative information that may be useful for classification.

4.3 LDA Revisited

For a c-class (assuming $c > 2$) problem the Fisher's criterion function is given as (for details see section 2.13.2)

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

The optimum value of $\mathbf{W}$ is the solution of the following conventional eigenvalue problem:

$$S_W^{-1} S_B \mathbf{w}_i = \lambda_i \mathbf{w}_i \qquad 4.1$$

where $\mathbf{w}_i$ are the column vectors of $\mathbf{W}$ that correspond to the largest eigenvalues (λ_i). From section 2.13.2 we have seen that rank of S_W is $n - c$, where n is the number of d-dimensional feature vectors available. If $n - c \geq d$ then S_W is full rank matrix i.e. non-singular and its inversion is possible. It is evident from the equation 4.1 that the explicit solution of the orientation can be found when S_W is non-singular. If S_W is singular (i.e. $n - c < d$) then it is not possible to obtain the orientation $\mathbf{W}$ by using equation 4.1. To overcome this singularity problem, we have presented the gradient LDA method which is described in the next section.

4.4 Gradient LDA for Reduced Rank Within-class Scatter Matrix

It is possible to find the desired leading h eigenvectors of the orientation $\mathbf{W}$ for reduced rank S_W matrix provided $rank(S_W) \geq h$ and $rank(S_B) \geq h$. A direct computation of $\mathbf{W}$ can be achieved by applying gradient descent method on the Fisher's criterion function. To derive the gradient LDA method we first find the derivative of $J(\mathbf{W})$ then update $\mathbf{W}$ using gradient descent method while normalizing the column vectors of $\mathbf{W}$ for each of the iterations. The derivative of $J(\mathbf{W})$ can be given from appendix 4.1 as

$$\frac{\partial J(\mathbf{W})}{\partial \mathbf{W}} = 2\, J(\mathbf{W})[S_W\, \mathbf{W}(\mathbf{W}^{\mathrm{T}}\, S_W\, \mathbf{W})^{-1} - S_B\, \mathbf{W}(\mathbf{W}^{\mathrm{T}}\, S_B\, \mathbf{W})^{-1}] \qquad 4.2$$

It can be observed from equation 4.2 that the inverse of S_W (a $d \times d$ sized matrix) is not computed in the equation as has been done in equation 4.1. However, inverse of $(\mathbf{W}^{\mathrm{T}}\, S_W\, \mathbf{W})$ and $(\mathbf{W}^{\mathrm{T}}\, S_B\, \mathbf{W})$ are computed to find the derivative of $J(\mathbf{W})$ which are full rank $h \times h$ sized matrices. Equation 4.2 can be utilized in the gradient descent algorithm to solve for the values of $\mathbf{W}$

$$\mathbf{W} \leftarrow \mathbf{W} - \alpha \frac{\partial J(\mathbf{W})}{\partial \mathbf{W}} \qquad 4.3$$

$\mathbf{W} \leftarrow$ *Normalize each of the column vectors of* $\mathbf{W}$ *separately* 4.4

where α is a learning rate parameter. It can also be observed from substituting equation 4.2 in equation 4.3 that $J(\mathbf{W})$ is updated for each of the iterations. The gradient LDA algorithm is illustrated in table 4.1. It will be empirically seen in the next section that the algorithm converges fast when unity value of α is taken and when $J(\mathbf{W})$ is utilized adaptively in the algorithm. This makes the algorithm fast converging and independent of initial settings for the iteration process.

The convergence relation proof of the gradient LDA technique can be easily shown.

Since it is a typical gradient descent based algorithm, the convergence proof will be similar to that of the LMS (least-mean-squared).

4.4.1 An Illustration

In this section we first compare the performance of the proposed gradient LDA technique with that of the basic LDA technique using the Fisher's criterion value as a prototype. Since the basic LDA can be applied only for full ranked S_W matrix we have taken the dataset accordingly. For this purpose Sat-Image dataset from UCI repository (Blake and Merz, 1998) is used. The Sat-Image dataset consists of 6 distinct classes

TABLE 4.1: Gradient LDA algorithm for computing the orientation **W**

```
1. Choose h, the number of leading eigenvectors required to estimate.
2. Initialize the orientation W of size $d \times h$ e.g. randomly or using identity
   matrix[1]
3. while (true)
4.      Compute $J(\mathbf{W}) = |\mathbf{W}^{\mathrm{T}} S_B \mathbf{W}| / |\mathbf{W}^{\mathrm{T}} S_W \mathbf{W}|$
5.           $\mathbf{W} \leftarrow \mathbf{W} - \alpha 2 J(\mathbf{W})[S_W \mathbf{W}(\mathbf{W}^{\mathrm{T}} S_W \mathbf{W})^{-1} - S_B \mathbf{W}(\mathbf{W}^{\mathrm{T}} S_B \mathbf{W})^{-1}]$
6.      Normalize column vectors of W
            for $j = 1$ to $h$
                $\mathbf{W}(:, j) \leftarrow \mathbf{W}(:, j) / \|\mathbf{W}(:, j)\|$[2]
            end
7. end
```

with 36 dimensions or attributes. It has 4435 feature vectors for training purpose and 2000 feature vectors for testing purpose. However, for this comparison we have taken the features from the first three classes of the training set. The coordinates of dataset taken are given as follows:

[1] In a $d \times h$ identity matrix $\mathrm{I}_{d \times h}$, the first h rows and columns is an identity matrix $\mathrm{I}_{h \times h}$ and the last $d-h$ rows are zero elements i.e. $\mathrm{I}_{d \times h} = [\mathrm{I}_{h \times h}\ 0_{h \times d-h}]^{\mathrm{T}}$.

[2] In $\mathbf{W}(:, j)$, '$:, j$' indicates elements of all the rows of jth column (i.e. jth column vector) and $\|\bullet\|$ denotes the norm value of this column vector.

	Class 1	Class 2	Class3
Number of training feature vectors per class (nvec):	1072	479	961

The dimension is reduced from 36-dimensional space to 2-dimensional plane. The total number of feature vectors minus the number of classes ($n-c$) is 2509 which is greater than the original dimension ($d=36$). Therefore S_W is a full rank matrix of size 36×36. Figure 4.1 illustrates the comparison between both the techniques for this dataset. The *x*-axis represents the number of iterations used for gradient LDA method and *y*-axis represents Fisher's criterion in logarithmic scale ($\mathbf{log}\,J(\mathbf{W})$) for both the techniques. Five different values (0.1, 0.5, 1, 2 and 5) of α are taken for the gradient LDA algorithm. The α values[3] 2 and 5 do not converge and provide negative $J(\mathbf{W})$ values which cannot be plotted on the figure (since $\mathbf{log}\,J(\mathbf{W})$ will yield a complex value). It can be observed from the figure that gradient LDA algorithm converges fast for $\alpha=1$. Substituting this unity value for α (in equation 4.3) means that the convergence becomes independent of any learning rate parameter or initial settings. One of the reasons for this fast convergence is the adaptive use of parameter $J(\mathbf{W})$ in the algorithm (table 4.1) i.e. $J(\mathbf{W})$ is updated for each of the iterations or for every single change in the value of $\mathbf{W}$. This adaptation makes the process fast and reliable. The value of $J(\mathbf{W})$ for gradient LDA is very close to the value of $J(\mathbf{W})$ of the basic LDA method. This test indicates that the orientation $\mathbf{W}$ obtained by both the techniques will discriminate different classes of feature vectors in a similar fashion.

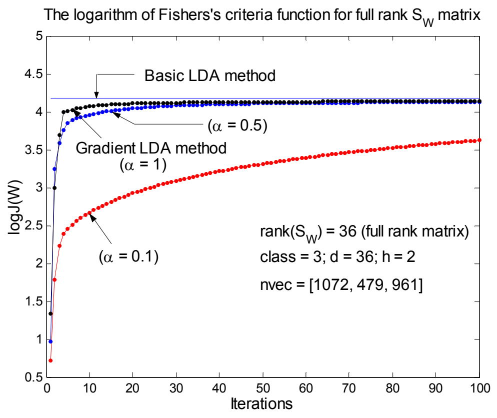


Figure 4.1: A comparison between basic LDA method and Gradient LDA method using Fisher's criterion value as a prototype.

[3] The α values above 1 usually do not provide very stable $J(\mathbf{W})$ values i.e. convergence is not guaranteed.

Next, we have taken feature vectors such that S_W is no longer full rank matrix to demonstrate its use in solving the small sample size problem. The same Sat-Image dataset is used where only 4 vectors from each of the three classes are taken i.e.

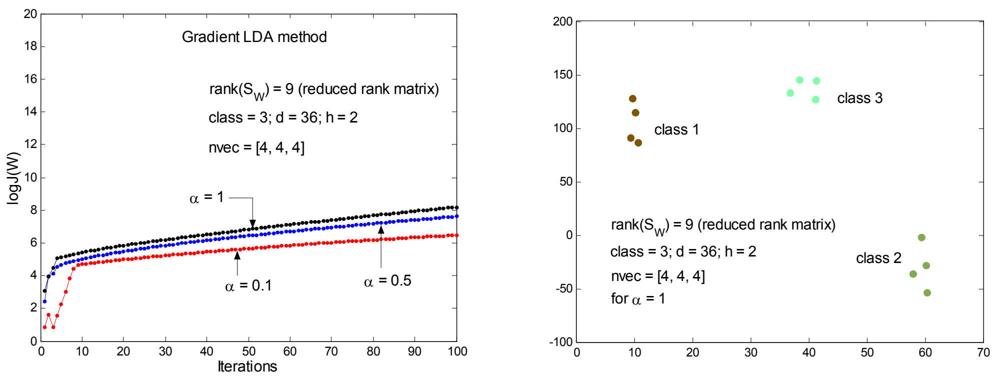
	Class 1	Class 2	Class3
Number of training feature vectors per class (nvec):	4	4	4

The rest of the parameters are not altered (i.e. $d = 36$ and $h = 2$). The size of S_W matrix is still 36×36. However, its rank is now $n - c = 9$. The basic LDA method cannot be applied here since S_W is singular thereby its inverse is not possible. The gradient LDA method is applied in this case for the same five values of α. The Fisher's criterion in logarithmic scale is depicted in figure 4.2a and projected samples ($\mathbf{y} = \mathbf{W}^{\mathrm{T}}\mathbf{x}$) on 2-dimensional plane is depicted in figure 4.2b (for $\alpha = 1$).

Here also the α values greater than unity (2 and 5) diverge and give complex $\mathbf{log}\,J(\mathbf{W})$ values which cannot be plotted in figure 2a. The convergence using other values of α is depicted in the figure. It can be observed from the figure that the algorithm achieves stable Fisher's criterion value somewhere before the tenth iteration. The orientation **W** at this iteration is adequate for providing the discrimination between different classes of feature vectors in the reduced dimensional space. In this case well, the unity value of α is giving better results than the other presented values. This means that $\alpha = 1$ is a suitable choice for the convergence of the algorithm. This selection makes the algorithm independent of initial settings.

Figure 4.2b illustrates projection of 36-dimensional feature vectors onto 2-dimensional plane using orientation **W** which is obtained by the gradient LDA method (for $\alpha = 1$). It is evident from the figure that different classes of feature vectors are well separated.

It can be concluded from the experiments that gradient LDA method is an efficient substitute of basic LDA method especially for reduced rank within-class scatter matrix (small sample size problem).



a) The logarithm of Fisher's criterion function for reduced rank S_W matrix

b) Projection of feature vectors onto 2-dimensional plane using Gradient LDA method for $\alpha = 1$

Figure 4.2: Gradient LDA application for small sample size problem.

4.5 Summary

We have presented a new way of computing the orientation **W** in LDA which addresses small sample size problem. The proposed method (called gradient LDA) is based on gradient descent method but it does not require any learning rate parameter which makes the convergence fast and reliable. The gradient LDA method does not discard any null spaces of S_W and S_B matrices and thus preserves discriminative information which is useful for classification.

Appendix 4.1

Lemma 1: Let the scalar function $J(\mathbf{W}) = |\mathbf{W}^{\mathrm{T}} S_B \mathbf{W}| / |\mathbf{W}^{\mathrm{T}} S_W \mathbf{W}|$ be a differentiable function of a $d \times h$ rectangular matrix $\mathbf{W}$ such that $h < d$. The size of both the symmetric matrices S_B and S_W is $d \times d$ and the rank for both is greater or equal to *h*. Then the derivative of $J(\mathbf{W})$ is defined as $\partial J(\mathbf{W}) / \partial \mathbf{W} = 2J(\mathbf{W})[S_W \mathbf{W}(\mathbf{W}^{\mathrm{T}} S_W \mathbf{W})^{-1} - S_B \mathbf{W}(\mathbf{W}^{\mathrm{T}} S_B \mathbf{W})^{-1}]$.

Proof 1: Using the quotient rule of differentiation we can differentiate $J(\mathbf{W})$ with respect to $\mathbf{W}$ as

$$\frac{\partial J(\mathbf{W})}{\partial \mathbf{W}} = [|\mathbf{W}^{\mathrm{T}} S_B \mathbf{W}| \frac{\partial}{\partial \mathbf{W}}(|\mathbf{W}^{\mathrm{T}} S_W \mathbf{W}|) - \frac{\partial}{\partial \mathbf{W}}(|\mathbf{W}^{\mathrm{T}} S_B \mathbf{W}|) |\mathbf{W}^{\mathrm{T}} S_W \mathbf{W}|] / |\mathbf{W}^{\mathrm{T}} S_W \mathbf{W}|^2 \qquad \text{A1}$$

from appendix II we can write equation A1 as

$$= 2 |\mathbf{W}^{\mathrm{T}} S_B \mathbf{W}| |\mathbf{W}^{\mathrm{T}} S_W \mathbf{W}| [S_W \mathbf{W}(\mathbf{W}^{\mathrm{T}} S_W \mathbf{W})^{-1} - S_B \mathbf{W}(\mathbf{W}^{\mathrm{T}} S_B \mathbf{W})^{-1}] / |\mathbf{W}^{\mathrm{T}} S_W \mathbf{W}|^2$$

Therefore

$$\partial J(\mathbf{W}) / \partial \mathbf{W} = 2\, J(\mathbf{W})[S_W \mathbf{W}(\mathbf{W}^{\mathrm{T}} S_W \mathbf{W})^{-1} - S_B \mathbf{W}(\mathbf{W}^{\mathrm{T}} S_B \mathbf{W})^{-1}]$$

Appendix 4.2

Lemma 2: Let the scalar function $g(\mathbf{W}) = |\mathbf{W}^T \mathbf{S} \mathbf{W}|$ be a differentiable function of a $d \times h$ rectangular matrix $\mathbf{W}$ such that $h < d$. The size of symmetric matrix $\mathbf{S}$ is $d \times d$ and $rank(\mathbf{S}) \geq h$. Then derivative of $g(\mathbf{W})$ with respect to $\mathbf{W}$ is defined as $\partial g(\mathbf{W}) / \partial \mathbf{W} = 2|\mathbf{W}^{\mathrm{T}} \mathbf{S} \mathbf{W}| \mathbf{S} \mathbf{W}(\mathbf{W}^{\mathrm{T}} \mathbf{S} \mathbf{W})^{-1}$.

Proof 2: The derivative of any determinant $\mathbf{X}$ is given by [Magnus-Neudecker]

$$\partial |\mathbf{X}| / \partial \mathbf{X} = |\mathbf{X}| (\mathbf{X}^{\mathrm{T}})^{-1} \qquad \text{A2}$$

equation A2 can also be written in the *trace* format as

$$\partial|\mathbf{X}| = |\mathbf{X}|\,trace[(\mathbf{X}^{\mathrm{T}})^{-1}\partial\mathbf{X}^{\mathrm{T}}] \qquad \text{A3}$$

from equation A3 the derivative of $g(\mathbf{W})$ is

$$\partial g(\mathbf{W}) = |\mathbf{W}^{\mathrm{T}}\mathbf{SW}|\,trace[(\mathbf{W}^{\mathrm{T}}\mathbf{SW})^{\mathrm{T}^{-1}}\partial(\mathbf{W}^{\mathrm{T}}\mathbf{SW})^{\mathrm{T}}]$$

$$= |\mathbf{W}^{\mathrm{T}}\mathbf{SW}|\{trace[(\mathbf{W}^{\mathrm{T}}\mathbf{SW})^{\mathrm{T}^{-1}}\partial\mathbf{W}^{\mathrm{T}}\mathbf{S}^{\mathrm{T}}\mathbf{W}] + trace[(\mathbf{W}^{\mathrm{T}}\mathbf{SW})^{\mathrm{T}^{-1}}\mathbf{W}^{\mathrm{T}}\mathbf{S}^{\mathrm{T}}\partial\mathbf{W}]\}$$

$$= |\mathbf{W}^{\mathrm{T}}\mathbf{SW}|\{trace[\mathbf{S}^{\mathrm{T}}\mathbf{W}(\mathbf{W}^{\mathrm{T}}\mathbf{SW})^{\mathrm{T}^{-1}}\partial\mathbf{W}^{\mathrm{T}}] + trace[\mathbf{SW}(\mathbf{W}^{\mathrm{T}}\mathbf{SW})^{-1}\partial\mathbf{W}^{\mathrm{T}}]\}$$

$\{\because trace(\mathrm{A}^{\mathrm{T}}) = trace(\mathrm{A})$ and $trace(\mathrm{AB}) = trace(\mathrm{BA})\}$

$$= 2|\mathbf{W}^{\mathrm{T}}\mathbf{SW}|\,trace[\mathbf{SW}(\mathbf{W}^{\mathrm{T}}\mathbf{SW})^{-1}\partial\mathbf{W}^{\mathrm{T}}]$$

$\{\because$ **S** is a symmetric matrix therefore $(\mathbf{W}^{\mathrm{T}}\mathbf{SW})$ is symmetric too$\}$

$$\therefore\ \partial g(\mathbf{W})/\partial\mathbf{W} = 2|\mathbf{W}^{\mathrm{T}}\mathbf{SW}|\,\mathbf{SW}(\mathbf{W}^{\mathrm{T}}\mathbf{SW})^{-1}$$

Chapter 5

Rotational Linear Discriminant Analysis Technique for Dimensionality Reduction

5.1 Abstract

The linear discriminant analysis (LDA) technique finds a linear transformation such that the overlapping between the classes is minimum for the projected samples in the reduced feature space. This overlapping, if present, adversely affects the classification performance. In this chapter we present a rotational transform that rotates the individual classes in the original feature space in such a way that the overlapping between the classes in the reduced feature space is further minimized. As a result the classification performance significantly improves which is demonstrated using several corpuses.

5.2 Introduction

In a typical pattern recognition application, some characteristic properties (or features) of an object are measured and the resulting feature vector is classified into one of the finite number of classes. When the number of features is relatively large, it becomes difficult to train a classifier using a finite amount of training dataset. In addition, the complexity of a classifier increases with the number of features used. In such situations, it becomes important to reduce the dimensionality of feature space. There are a number of techniques proposed in the literature for dimensionality reduction (Fukunaga, 1990); the linear discriminant analysis (LDA) technique is perhaps the most popular among them. The LDA technique uses a linear transformation to project the measured feature vectors to a subspace in such a way that the overlapping between the classes is minimized in the reduced feature space.

In order to provide an illustration, consider a 2-dimensional feature space with 3 classes as shown in figure 5.1a. When we use LDA to reduce the dimensionality to one, we get the orientation **W** along which the overlap between the classes is minimum. Note that the overlap between classes (though minimum) is still finite and as a result we get a finite amount of classification error. In order to reduce this classification error, we propose in this chapter to use a (rotational) transform $\boldsymbol{\theta}$ prior to LDA. The transform $\boldsymbol{\theta}$ rotates the distribution of each class around its own mean; it is chosen in such a way that the overlap between the classes in the resulting LDA orientation is minimum (see figure 5.1b). It can be observed that classification error in figure 5.1b is less than that seen in figure 5.1a. Since we are using here the rotational transform with LDA, we call it the Rotational LDA technique.

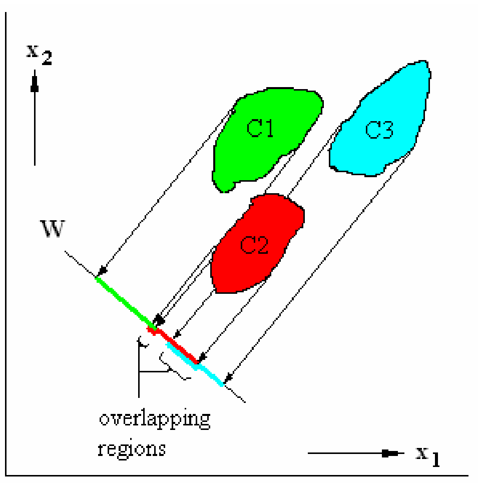


Figure 5.1a

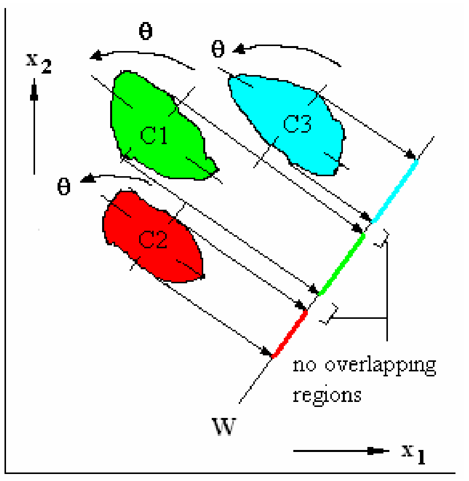


Figure 5.1b

Figure 5.1: A comparison between basic LDA and Rotational LDA techniques.

5.3 Rotational Linear Discriminant Analysis

This section indulges on the mathematical details and proofs of the Rotational LDA. Let $\mathcal{X}$ denote the d-dimensional set of n training samples (feature vectors) in a c-class problem, $\Omega = \{\omega_i : i = 1,2,...,c\}$ be the finite set of c states of nature or class labels where ω_i denotes the i^{th} class label. The set $\mathcal{X}$ can be subdivided into c subsets $\mathcal{X}_1$,

$\mathcal{X}_2,\ldots,\mathcal{X}_c$ where each subset $\mathcal{X}_i$ belongs to ω_i and consists of n_i number of samples such that:

$$n = \sum_{i=1}^{c} n_i$$

The samples or patterns of set $\mathcal{X}$ can be written as:

$\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2,\ldots, \mathbf{x}_n\}$ where $\mathbf{x}_j \in \mathbf{R}^d$ (d-dimensional hyperplane)

$\mathcal{X}_i \subset \mathcal{X}$ and $\mathcal{X}_1 \cup \mathcal{X}_2 \cup \ldots \cup \mathcal{X}_c = \mathcal{X}$

Let $\mathcal{Y}_j$ be h-dimensional transformed samples from $\mathcal{X}_j \in \omega_j$ using LDA technique where $h < d$, then the samples of reduced dimensional set or transformed sample set $\mathcal{Y}$ can be depicted as:

$\mathcal{Y} = \{\mathbf{y}_1, \mathbf{y}_2,\ldots, \mathbf{y}_n\}$ where $\mathbf{y}_j \in \mathbf{R}^h$ (h-dimensional hyperplane)

$\mathcal{Y}_j \subset \mathcal{Y}$ and $\mathcal{Y}_1 \cup \mathcal{Y}_2 \cup \ldots \cup \mathcal{Y}_c = \mathcal{Y}$ where $\mathcal{Y}_j$ is derived from $\mathcal{X}_j$

For illustration, a two-class problem ($c = 2$) is depicted in figure 5.2, where $\mathcal{X}_1$ and $\mathcal{X}_2$ are the two subsets of feature vectors in the original 2-dimensional plane ($\mathbf{x} \in \mathbf{R}^2$). The class labels of these subsets are represented as ω_1 and ω_2 respectively. This original space is transformed to a lower 1-dimensional plane ($y \in \mathbf{R}^1$), producing transformed sample sets $\mathcal{Y}_1$ and $\mathcal{Y}_2$ which belong to the class labels ω_1 and ω_2 respectively. The transformation is conducted using a LDA directional vector/matrix or orientation $\mathbf{W}$ of size $d \times h$ (in this illustration $h = 1$) i.e. $\mathbf{W}$: $\mathbf{x} \rightarrow y$, or $y_j = \mathbf{W}^{\mathrm{T}}\mathbf{x}_j$ for $j = 1,\ldots,n$. The respective probability distributions of $\mathcal{Y}_1$ and $\mathcal{Y}_2$ are also shown. Suppose that a classifier has divided the 1-dimensional plane into two regions $\mathcal{R}_1$ and $\mathcal{R}_2$. There are two possibilities in which a classification error could occur; either observation y ($\mathbf{W}$: $\mathbf{x} \rightarrow y$) falls in the region $\mathcal{R}_1$ and the true state of nature is ω_2, or y falls in the region $\mathcal{R}_2$ and the true state of nature is ω_1. Since these events are mutually exclusive and exhaustive (Duda and Hart, 1973), we can define probability of error as:

$$\begin{aligned}\text{Perror} &= P(y \in \omega_2, \mathcal{R}_1) + P(y \in \omega_1, \mathcal{R}_2) \\ &= P(y \in \mathcal{R}_1 \mid \omega_2) P(\omega_2) + P(y \in \mathcal{R}_2 \mid \omega_1) P(\omega_1) \\ &= \int\limits_{y \in \mathcal{R}_1} p(y \mid \omega_2) P(\omega_2)\, dy + \int\limits_{y \in \mathcal{R}_2} p(y \mid \omega_1) P(\omega_1)\, dy\end{aligned}$$

where $P(\omega_j)$ is the *a priori* probability of $\mathcal{Y}_j$. In a multiclass case, it would be easier to find the probability of being correct (Duda and Hart, 1973). Therefore

$$\text{Pcorrect} = \sum_{j=1}^{c} \int\limits_{y \in \mathcal{R}_j} p(y \mid \omega_j) P(\omega_j)\, dy$$

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

$$\text{Ptotal} = \sum_{j=1}^{c} \int\limits_{y \in \mathcal{Y}_j} p(y \mid \omega_j) P(\omega_j)\, dy\,.$$

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

$$\text{Ptotal} = \text{Pcorrect} + \text{Perror}$$

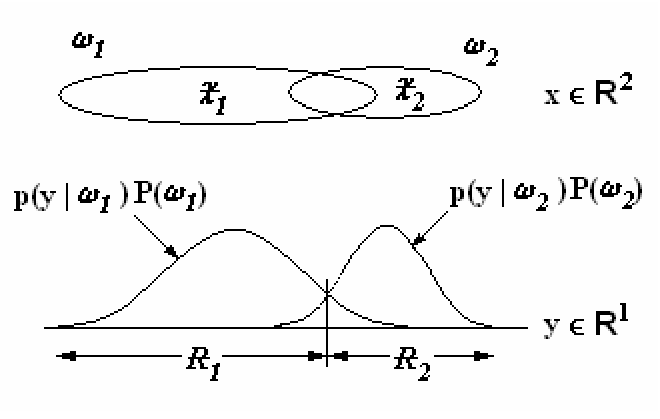


Figure 5.2: Probability error for two-class problem

Therefore the probability error function can be written as

$$\begin{aligned}\text{Perror} &= b - \text{Pcorrect} \\ &= b - \sum_{j=1}^{c} \int_{y\in R_j} p(y\,|\,\omega_j)P(\omega_j)\end{aligned} \qquad 5.1$$

where b is a constant and is equal to Ptotal. In practice the number of samples is limited to a finite value, thus the integration in equation 5.1 is approximated by the summation using the following estimation (Anton, 1995)

$$A = \int f(x)dx = \lim_{n\to\infty} \sum_{k=1}^{n} f(x_k)\Delta x \approx \sum_{k=1}^{n} f(x_k)\Delta x \qquad 5.2$$

Using this estimation (equation 5.2), equation 5.1 can be rewritten as

$$\text{Perror} = b - \sum_{j=1}^{c} \sum_{y\in R_j} p(y\,|\,\omega_j)P(\omega_j)\Delta V \qquad 5.3$$

where ΔV is a volume of tiny hypercube. This ΔV is a scalar quantity and depends upon the transformed sample set $\mathcal{Y}$. Equation 5.3 is a probability error function for a scalar y which can be extended to vector $\mathbf{y}$ simply by replacing a vector in place of scalar y.

Constant ΔV is independent of any class and therefore is taken outside from the summations of equation 5.3. The value of a *priori* probability $P(\omega_j) = n_j / n$ is substituted in equation 5.3, this yields Perror as

$$\text{Perror} = b - k\sum_{j=1}^{c} n_j \sum_{\mathbf{y}\in R_j} p(\mathbf{y}\,|\,\omega_j) \qquad 5.4$$

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

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

$$\mathbf{y} = \mathbf{W}^{\mathrm{T}}[\boldsymbol{\theta}^{\mathrm{T}}(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}_j}) + \boldsymbol{\mu}_{\mathbf{x}_j}] \tag{5.5}$$

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

At the beginning when no rotation has taken place then the transformation $\boldsymbol{\theta}$ would be $d \times d$ identity matrix and equation 5.5 would reduce to the basic LDA transformation. To find the optimum rotation of $\boldsymbol{\theta}$ it is required to differentiate scalar function (equation 5.4) with respect to the transformation matrix $\boldsymbol{\theta}$. The value of $\boldsymbol{\theta}$ that corresponds to the minima of Perror would be the optimum rotation of $\boldsymbol{\theta}$. Therefore from equation 5.4 we get

$$\frac{\partial}{\partial \boldsymbol{\theta}}\mathrm{Perror} = -k\sum_{j=1}^{c} n_j \sum_{\mathbf{y}\in R_j} \frac{\partial}{\partial \boldsymbol{\theta}} p(\mathbf{y} \mid \omega_j) \tag{5.6}$$

where $\mathbf{y}$ is from equation 5.5. The next thing is to find the probability distribution $p(\mathbf{y} \mid \omega_j)$ before differentiating it with respect to $\boldsymbol{\theta}$. One way to estimate $p(\mathbf{y} \mid \omega_j)$ is to use parametric techniques where we assume a functional form of Gaussian distribution characteried by a few parameters, for example, assuming $\mathbf{y}$ to be multidimensional Gaussian[1]. Therefore $p(\mathbf{y} \mid \omega_j)$ of equation 5.6 turns to be

$$\frac{\partial}{\partial \boldsymbol{\theta}} p(\mathbf{y} \mid \omega_j) = \frac{\partial}{\partial \boldsymbol{\theta}} \left\{ \frac{1}{(2\pi)^{h/2} \mid \boldsymbol{\Sigma}_{\mathbf{y}_j} \mid^{1/2}} \mathbf{exp}\left(-\tfrac{1}{2}(\mathbf{y} - \boldsymbol{\mu}_{\mathbf{y}_j})^{\mathrm{T}} \Sigma_{\mathbf{y}_j}^{-1} (\mathbf{y} - \boldsymbol{\mu}_{\mathbf{y}_j})\right) \right\} \tag{5.7}$$

[1] It should be noted that Gaussian density is used here for simplicity purposes only. Other types of parametric functions or even non-parametric way of computing densities can be used, since we are only interested in developing some sort of measurement for the overlapping of samples belonging to different classes in the reduced feature space. Thus the assumption of Gaussian density $p(\mathbf{y} \mid \omega_j)$ will not much affect the measurement even if the density is non Gaussian.

where $\boldsymbol{\mu}_{\mathbf{y}_j}$ and $\boldsymbol{\Sigma}_{\mathbf{y}_j}$ are mean and covariance of $\mathcal{Y}_j$. Substituting equation 5.5 in equation 5.7 we get

$$\begin{aligned}\frac{\partial}{\partial\theta}p(\mathbf{y}\,|\,\omega_j) &= \frac{\partial}{\partial\theta}p(\mathbf{x},\boldsymbol{\theta},\mathbf{W}\,|\,\omega_j)\\ &= \frac{\partial}{\partial\boldsymbol{\theta}}\left\{\frac{1}{(2\pi)^{h/2}\,|\,\boldsymbol{\Sigma}_{\mathbf{y}_j}|^{1/2}}\mathbf{exp}\left(-\tfrac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_{\mathbf{x}_j})^{\mathrm{T}}\boldsymbol{\theta}\mathbf{W}\boldsymbol{\Sigma}_{\mathbf{y}_j}^{-1}\mathbf{W}^{\mathrm{T}}\boldsymbol{\theta}^{\mathrm{T}}(\mathbf{x}-\boldsymbol{\mu}_{\mathbf{x}_j})\right)\right\}\end{aligned} \qquad 5.8$$

where $\mathbf{x}$ is the corresponding vector of $\mathbf{y}\in\mathcal{R}_j$, that is only those vectors of $\mathbf{x}\in\mathcal{X}_j$ are taken that correspond to $\mathbf{y}\in\mathcal{R}_j$ in the equation. Let us represent this correspondence relation by $\mathbf{x}(\mathbf{y}\in\mathcal{R}_j)$. The following Lemma would help in solving equation 5.8.

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

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

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

$$\therefore\qquad \nabla_{\boldsymbol{\theta}}\,\mathbf{exp}(-\tfrac{1}{2}u)=-\tfrac{1}{2}\mathbf{exp}(-\tfrac{1}{2}u)\boldsymbol{\Lambda}\boldsymbol{\Lambda}^{\mathrm{T}}\boldsymbol{\theta}\mathbf{W}(\mathbf{B}+\mathbf{B}^{\mathrm{T}})\mathbf{W}^{\mathrm{T}}$$

Using Lemma 1 we can rewrite equation 5.8 as

$$\frac{\partial}{\partial \boldsymbol{\theta}} p(\mathbf{y} \mid \omega_j) = -\frac{1}{2(2\pi)^{h/2} |\Sigma_{\mathbf{y}_j}|^{1/2}} \exp(-\tfrac{1}{2}u) \left[(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}_j})(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}_j})^{\mathrm{T}} \boldsymbol{\theta} \mathbf{W} (\boldsymbol{\Sigma}_{\mathbf{y}_j}^{-1} + \boldsymbol{\Sigma}_{\mathbf{y}_j}^{-1\,\mathrm{T}}) \mathbf{W}^{\mathrm{T}} \right] \qquad 5.9$$

where $u = (\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}_j})^{\mathrm{T}} \boldsymbol{\theta} \mathbf{W} \Sigma_{\mathbf{y}_j}^{-1} \mathbf{W}^{\mathrm{T}} \boldsymbol{\theta}^{\mathrm{T}} (\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}_j})$. Substituting equation 5.9 in equation 5.6, we get

$$\frac{\partial}{\partial \boldsymbol{\theta}} \mathrm{Perror} = k' \sum_{j=1}^{c} \frac{n_j}{|\Sigma_{\mathbf{y}_j}|^{1/2}} \sum_{\mathbf{x}(\mathbf{y} \in R_j)} \exp(-\tfrac{1}{2}u) \left[(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}_j})(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}_j})^{\mathrm{T}} \boldsymbol{\theta} \mathbf{W} (\boldsymbol{\Sigma}_{\mathbf{y}_j}^{-1} + \boldsymbol{\Sigma}_{\mathbf{y}_j}^{-1\,\mathrm{T}}) \mathbf{W}^{\mathrm{T}} \right] \qquad 5.10$$

where $k' = k / (2(2\pi)^{h/2})$. Equation 5.10 can also be written in the expectation form i.e.

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

where $\mathrm{F}(\mathbf{x}, \boldsymbol{\theta}, \mathbf{W}, \boldsymbol{\mu}_{\mathbf{x}_j}, \Sigma_{\mathbf{y}_j}) = \exp(-\tfrac{1}{2}u)[(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}_j})(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}_j})^{\mathrm{T}} \boldsymbol{\theta} \mathbf{W} (\boldsymbol{\Sigma}_{\mathbf{y}_j}^{-1} + \boldsymbol{\Sigma}_{\mathbf{y}_j}^{-1\,\mathrm{T}}) \mathbf{W}^{\mathrm{T}}]$ and $E[\bullet]$ is the expectation of $\mathrm{F}(\bullet)$ with respect to $\mathbf{x}$.

Since the topography of the original data should remain unchanged during the rotation $\boldsymbol{\theta}$, the column vectors of $\boldsymbol{\theta}$ should be orthonormal. This means the square matrix $\boldsymbol{\theta}$ is orthonormal i.e. $\boldsymbol{\theta}^{\mathrm{T}} \boldsymbol{\theta} = \mathrm{I}_{d \times d}$, thus we can obtain the following gradient algorithm:

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

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

It is well known that the gradient algorithm is slow in convergence and depends on an initial choice of the learning rate. Thus the convergence is very critical to the initial choice of the learning rate. However, fixed point algorithm (Hyvärinen and Oja 1997) can be applied here which would make the learning drastically faster and more reliable. Thus by using the fixed point algorithm we can obtain the following algorithm:

$$\boldsymbol{\theta} \propto \sum_{j=1}^{c} \frac{n_j^2}{|\Sigma_{\mathbf{y}_j}|^{1/2}} \underset{\mathbf{x}(\mathbf{y}\in R_j)}{E}[\mathrm{F}(\mathbf{x},\boldsymbol{\theta},\mathbf{W},\boldsymbol{\mu}_{\mathbf{x}_j},\Sigma_{\mathbf{y}_j})] \qquad 5.11$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}(\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\theta})^{-1/2} \qquad 5.12$$

The values of $\mathbf{W}$ and $\boldsymbol{\Sigma}_{\mathbf{y}_j}$ will be changing depending upon the rotation of the original feature space, whereas the center of class $\boldsymbol{\mu}_{\mathbf{x}_j}$ will remain invariable for any such rotation since the rotation of the original feature space is always with respect to its center $\boldsymbol{\mu}_{\mathbf{x}_j}$. The matrices $\mathbf{W}$ and $\boldsymbol{\Sigma}_{\mathbf{y}_j}$ should be updated for every iteration of $\boldsymbol{\theta}$ for equation 5.11. The inverse of $\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\theta}$ in equation 5.12 is computed using eigenvalue decomposition. There are iterative methods for orthonormalization that avoid the matrix inverse and eigendecomposition. In that case the rotation matrix $\boldsymbol{\theta}$ can be orthonormalized by using symmetric orthonormalization procedure starting from a nonorthogonal matrix and continuing the iterative process until $\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\theta} \approx \mathrm{I}_{d\times d}$ (Hyvärinen, 1999).

In this algorithm we have used orientation **W** which we have derived from LDA technique. One can also investigate some other criteria of separating and/or classifying the class vectors. For example instead of **W,** Gaussian model or some other useful techniques can be used. Investigation of other appropriate transforms/techniques that could be applied to the rotational method instead of **W** is beyond the scope of this paper.

The value of Perror can be estimated more economically also for each of the iteration of equations 5.11 and 5.12 by applying equation 5.13 instead of applying equation 5.1 as:

$$\text{Perror} = 1 - \frac{\sum_{j=1}^{c} \text{number of samples belongs to } \mathcal{R}_j \text{ given } \omega_j}{\text{total number of samples in } \chi} = 1 - \frac{\sum_{j=1}^{c} (n_j \mid \mathcal{R}_j, \omega_j)}{n} \qquad 5.13$$

To obtain the Perror in percentage simply multiply it by 100. Region $\mathcal{R}_j$ of training samples can be obtained by several methods. We have used minimum distance classification method (taking centroid of a class as a prototype) for finding the regions.

5.3.1 Rotation of Original Feature Space for *m* Iterations

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

$$\text{Iteration 1:} \quad \mathbf{x}_1 = \boldsymbol{\theta}_1^{\mathrm{T}}(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}_j}) + \boldsymbol{\mu}_{\mathbf{x}_j} \qquad 5.14$$

After 2nd iteration the feature vectors would be

$$\text{Iteration 2:} \quad \mathbf{x}_2 = \boldsymbol{\theta}_2^{\mathrm{T}}(\mathbf{x}_1 - \boldsymbol{\mu}_{\mathbf{x}_j}) + \boldsymbol{\mu}_{\mathbf{x}_j} \qquad 5.15$$

Similarly after mth iteration, feature vectors can be given as

$$\text{Iteration } m\text{:} \quad \mathbf{x}_m = \boldsymbol{\theta}_m^{\mathrm{T}}(\mathbf{x}_{m-1} - \boldsymbol{\mu}_{\mathbf{x}_j}) + \boldsymbol{\mu}_{\mathbf{x}_j} \qquad 5.16$$

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

Substituting equation 5.14 in equation 5.15 we get

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

Similarly we can say that

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

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

where $\boldsymbol{\theta} = \boldsymbol{\theta}_1\boldsymbol{\theta}_2 \ldots \boldsymbol{\theta}_m$

The number of iterations *m* is usually a small value (2 ~ 4) unless the given data is very complicated. The next section describes the training session of the algorithm.

5.3.2 Training Session of the Rotational LDA Algorithm

Training session involves finding the optimum rotation $\boldsymbol{\theta}$ of the original feature space such that the overlapping error between the adjacent classes is minimum in the transformed domain. The first step is to find the orientation **W** (assuming $\boldsymbol{\theta}$ to be identity matrix) by applying basic LDA procedure. The obtained orientation **W** will be such that the classes are maximally discriminated in the reduced dimensional space. This transformation may produce overlapping of samples in reduced dimensional space between adjacent classes which cannot be reduced any further by moving the direction (orientation) **W** around the origin in reduced dimensional feature space. However, by applying rotation $\boldsymbol{\theta}$ in original feature space we can get a reduced dimensional feature space with much less overlapping between the adjacent classes. The application of this rotational step improves the recognition performance significantly. The training session is briefly illustrated here under for an introduction:

- Find the optimum orientation **W** using the basic LDA procedure. Let the minimum error be P_1.
- Rotate the original space to get error P_2 such that $P_2 < P_1$.

- Repeat the rotation until the minimum error P_t is found ($P_t < P_{t-1} \dots < P_1$). Let the rotation for this minimum error be $\boldsymbol{\theta}$.

The feature vector $\mathbf{x}$ of each class is separately taken for the rotational transform $\boldsymbol{\theta}$. This transformation is taken with respect to the center of their class. The detailed training session is illustrated in table 5.1. It should be noted (from table 5.1) that in first iteration ($m = 1$) LDA is computed for $\mathbf{W}$ but rotation is not applied to the feature vectors of individual classes for LDA computation. Rotation is applied only after the first iteration (i.e. 2nd iteration onwards) on feature vectors for LDA computation. Some of the advantages and drawbacks of the algorithm are illustrated as follows:

5.3.3 Advantages of Using the Algorithm

1. It gives much less overlapping error between the adjacent classes in the training phase as compared to LDA method by finding the optimum rotation $\boldsymbol{\theta}$ for the original feature space.
2. The optimum rotation helps in improving the recognition performance significantly.
3. It is able to achieve quite low classification error at even very low dimensional space.
4. The algorithm converges very fast due to the use of fixed-point algorithm.

5.3.4 Drawbacks of the Algorithm

1. One of the drawbacks of the algorithm (table 5.1) is to compute basic LDA at each stage of iteration. This means the algorithm is computing within-class scatter matrix ($\mathbf{S}_W$) and between class scatter matrix ($\mathbf{S}_b$) for each of the iteration.
2. It can also be observed from table 5.1 that training feature vectors are used and updated in each stage of the iteration.

3. The use of expectation operator in equation 5.11 (or step 10 in table 5.1) would slow the process.
4. Class labelling for single test vector is not possible (discussed later in section 5.3.8).

We have provided suggestions in section 5.3.6 to minimize some of its drawbacks.

5.3.5 Storage Requirement of the Algorithm

Some parameters are stored during the training phase which will be used in the testing phase. These parameters and their corresponding sizes are depicted in table 5.2. The total parameter requirement for storage is the sum of column 3 of table 5.2.

5.3.6 Suggestions for Optimizing the Training Phase

Some suggestions are given here that would address the drawbacks presented in section 5.3.4. For the first drawback, the LDA procedure can be modified such that it will not compute $\mathbf{S}_W$ and $\mathbf{S}_b$ for each step of the iteration process. We can modify these matrices by investigating how $\mathbf{S}_W$ and $\mathbf{S}_b$ alter during the iteration process. It is known that $\mathbf{S}_W$ is the sum of scatter matrices $\mathbf{S}_j$ (Duda and Hart, 1973) i.e.

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

where $\mathbf{S}_j = \sum_{\mathbf{x}\in\mathcal{X}_j} (\mathbf{x}-\boldsymbol{\mu}_{\mathbf{x}_j})(\mathbf{x}-\boldsymbol{\mu}_{\mathbf{x}_j})^{\mathrm{T}}$

After rotation $\hat{\boldsymbol{\theta}}$, feature vector $\mathbf{x}$ will change according to equation 5.16, this would change the scatter matrix as

$$\hat{\mathbf{S}}_j = \hat{\boldsymbol{\theta}}^{\mathrm{T}}[\sum_{\mathbf{x}\in\mathcal{X}_j} (\mathbf{x}-\boldsymbol{\mu}_{\mathbf{x}_j})(\mathbf{x}-\boldsymbol{\mu}_{\mathbf{x}_j})^{\mathrm{T}}]\hat{\boldsymbol{\theta}} = \hat{\boldsymbol{\theta}}^{\mathrm{T}}\mathbf{S}_j\hat{\boldsymbol{\theta}}$$

TABLE 5.1: Rotational LDA algorithm for estimating orientation $\mathbf{W}$ and rotation $\boldsymbol{\theta}$.

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

TABLE 5.2: List of parameters stored during the training phase which will be required in the testing phase with their corresponding sizes.

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

[2] After careful observation one would be convinced that the orientation $\mathbf{W}$ and mean $\boldsymbol{\mu}_{\mathbf{y}_j}$ should be stored at step 8 instead of step 5 for their use in the testing session. At the occurrence of break point (step 7) the prior value of $\mathbf{W}$ (i.e. $\hat{\mathbf{W}}$) and $\boldsymbol{\mu}_{\mathbf{y}_j}$ should be considered for the testing session.

Therefore modified within-class scatter matrix will become

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

On the other hand, $\mathbf{S}_b$ depends on the class centers and the total mean vector (Duda and Hart, 1973) which would not change during the rotation. Therefore $\mathbf{S}_b$ will remain unchanged with respect to rotation. Thus the new value of orientation **W** can be computed directly from rotation $\hat{\boldsymbol{\theta}}$ and the previous value of $\mathbf{S}_W$ using eigenvalue decomposition i.e.

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

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

The second drawback (section 5.3.4) might be addressed by introducing some kind of weighting coefficients which would update the parameters depending upon the rotation and their previous values. Introducing this type of updating process might overcome this drawback.

For the third drawback, the expectation operation (section 5.3.4) might be omitted by introducing on-line or adaptive version of the algorithm, where parameters may be updated for every feature vector **x** instead of taking the class average of feature vectors.

These suggestions are not investigated in detail any further in this paper. However, interested readers may wish to consider pursuing these suggestions further in minimizing these drawbacks.

5.3.7 An Example of the Training Phase of Rotational LDA

This section describes an example of the training phase of the Rotational LDA algorithm. For this purpose Sat-Image dataset from UCI repository (Blake and Merz, 1998) is used. The Sat-Image dataset consists of 6 distinct classes with 36 dimensions or attributes. It consists of 4435 feature vectors for training purpose and 2000 feature vectors for testing purpose. However, in this example we have taken some of the features from the first three classes and from the first two dimensions. The coordinates of training dataset taken for this example are given as follows:

	Class 1	Class 2	Class3
Number of training feature vectors per class:	600	400	800

Parametric details are given in table 5.3. All the values are given up to 4 decimal places.

TABLE 5.3: List of the values of parameters during an example run

Initial dimension $d = 2$, *reduced dimension* $h = 1$, $class = 3$, $\mathrm{P}_0 = 100\%$								
$\boldsymbol{\mu}_{\mathbf{x}_1} = [63.6283,\ 96.0867]^{\mathrm{T}}$, $\boldsymbol{\mu}_{\mathbf{x}_2} = [50.5850,\ 43.0275]^{\mathrm{T}}$, $\boldsymbol{\mu}_{\mathbf{x}_3} = [86.8150,\ 104.3275]^{\mathrm{T}}$								
Iteration *m*	Perror (P_m)	$\mathbf{W}$	$\boldsymbol{\theta}$	$\hat{\boldsymbol{\theta}}$	$\boldsymbol{\mu}_{\mathbf{y}_1}$	$\boldsymbol{\mu}_{\mathbf{y}_2}$	$\boldsymbol{\mu}_{\mathbf{y}_3}$	Figure
1	21.1667	$\begin{bmatrix} 0.8837 \\ -0.4680 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0.9070 & -0.4211 \\ 0.4211 & 0.9070 \end{bmatrix}$	11.2658	24.5688	27.9005	5.3a
					$\boldsymbol{\Sigma}_{\mathbf{y}_1}$	$\boldsymbol{\Sigma}_{\mathbf{y}_2}$	$\boldsymbol{\Sigma}_{\mathbf{y}_3}$	
					23.4744	8.7760	9.2393	
2	18.7222	$\begin{bmatrix} 0.5107 \\ -0.8598 \end{bmatrix}$	$\begin{bmatrix} 0.9070 & -0.4211 \\ 0.4211 & 0.9070 \end{bmatrix}$	$\begin{bmatrix} 0.8396 & -0.5432 \\ -0.5432 & -0.8396 \end{bmatrix}$	$\boldsymbol{\mu}_{\mathbf{y}_1}$	$\boldsymbol{\mu}_{\mathbf{y}_2}$	$\boldsymbol{\mu}_{\mathbf{y}_3}$	5.3b
					-50.1910	-11.1612	-45.3634	
					$\boldsymbol{\Sigma}_{\mathbf{y}_1}$	$\boldsymbol{\Sigma}_{\mathbf{y}_2}$	$\boldsymbol{\Sigma}_{\mathbf{y}_3}$	
					27.2460	13.7915	8.2005	
3	0.6667	$\begin{bmatrix} 0.9173 \\ 0.3981 \end{bmatrix}$	$\begin{bmatrix} 0.9903 & -0.1391 \\ -0.1391 & -0.9903 \end{bmatrix}$	$\begin{bmatrix} 0.7260 & -0.6877 \\ -0.6877 & -0.7260 \end{bmatrix}$	$\boldsymbol{\mu}_{\mathbf{y}_1}$	$\boldsymbol{\mu}_{\mathbf{y}_2}$	$\boldsymbol{\mu}_{\mathbf{y}_3}$	5.3c
					96.6242	63.5340	121.1749	
					$\boldsymbol{\Sigma}_{\mathbf{y}_1}$	$\boldsymbol{\Sigma}_{\mathbf{y}_2}$	$\boldsymbol{\Sigma}_{\mathbf{y}_3}$	
					24.7971	10.2408	8.3090	

In this example the original feature dimension is 2 and it is reduced to 1-dimensional plane for recognition and/or classification purposes. It can be seen from table 5.3 that the algorithm converged at the third iteration. At first iteration there is no rotation of the original feature space, only basic LDA method is applied to compute the value of orientation **W**. The overlapping error is noted to be 21.17% at the first iteration (without any rotation). When first rotation is applied at second iteration the error reduces to 18.72% and to only 0.67% at the third iteration (on the application of second rotation). This example is also illustrated in figures 5.3a, 5.3b and 5.3c for iterations 1, 2 and 3 respectively. In all the three figures, the projection of feature vectors is illustrated from 2-dimensional space onto 1-dimensional plane for all the three classes. Figure 5.3a depicts the projection when only basic LDA method is applied (i.e. iteration 1). Thereafter, rotation is applied which are depicted in figures 5.3b and 5.3c. It is quite clear from all the three figures that the rotation does help in minimizing the overlapping error significantly in the transformed space which is not possible by LDA method.

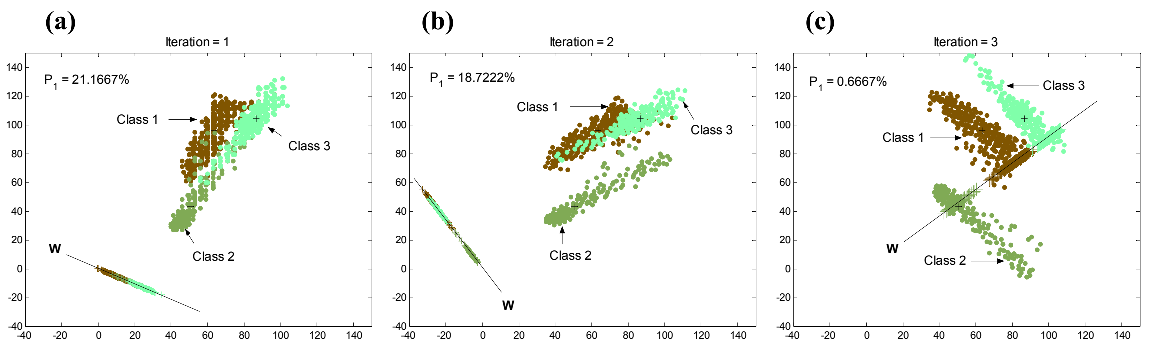


Figure 5.3: Illustration of the transformation from 2-dimensional feature space to 1-dimensional feature plane using Rotational LDA method on a 3-class problem.

The testing or classification is discussed in the next section.

5.3.8 Classification Phase of the Rotational LDA Algorithm

The strategy of classifying or testing a test vector using Rotational LDA algorithm is slightly different from the basic LDA algorithm. It is not possible to classify one unknown feature vector at a time using this algorithm. This is due to the fact that the rotation $\boldsymbol{\theta}$ is applied individually with respect to the center of each given class in the training phase for feature vectors. If an unknown feature vector is tested at a time, then it would be rotated with respect to the center of given classes separately prior to the transformation onto lower dimensional space by orientation **W**. The rotation would change the location of a test vector depending upon the class centers. Therefore, if a test vector is rotated with respect to different class centers then it could fall in a region of different classes. Thereby the application of orientation **W** for the projection onto lower dimensional space would no longer help in classifying the test vector. However, this problem can be overcome quite easily by considering more than one test vectors at a time. Then these test vectors can be rotated around their own center prior to the application of **W**. Therefore, if L test vectors are taken at a time then these L vectors are firstly rotated around their own center using rotation $\boldsymbol{\theta}$ before transforming to lower dimensional space by **W**. Thus, the constraint of this algorithm is to have more than one test vectors which should be of the same class labels at a time. The classification phase is illustrated in table 5.4.

TABLE 5.4: Classification phase of the Rotational LDA algorithm

1. Take L test vectors $\hat{\mathcal{X}}_L = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_L\}$ such that $\hat{\mathcal{X}}_L \in \omega_k$ where k is unknown.
2. Find the rotation of these L test vectors with respect to the center of $\hat{\mathcal{X}}_L$ i.e.

$$\mathbf{x}_{rot} = \boldsymbol{\theta}^{\mathrm{T}}(\mathbf{x} - \boldsymbol{\mu}_L) + \boldsymbol{\mu}_L$$

where $\mathbf{x} \in \hat{\mathcal{X}}_L$ and $\boldsymbol{\mu}_L = \frac{1}{L}\sum_{j=1}^{L}\mathbf{x}_j$.

3. Transform $\mathbf{x}_{rot}$ to lower dimensional space by **W**

$$\mathbf{y} = \mathbf{W}^{\mathrm{T}}\mathbf{x}_{rot}$$

4. Associate the class label of the closest center to **y**

$$k = \mathbf{arg}\min_{j=1}^{c} \| \mathbf{y} - \boldsymbol{\mu}_{\mathbf{y}_j} \|$$

It can be observed from table 5.4 that minimum distance classification method has been applied to label the unknown feature vectors. The classification can easily be extended for Bayesian classifier. In that case step 4 of table 5.4 will be replaced by Bayes decision rule as

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

where $p(\mathbf{y} \mid \omega_j)$ is the probability density function of **y** and $P(\omega_j)$ is the *a priori* probability as described in section 5.3. The next section describes the computational complexity of the algorithm.

5.3.9 Computational Complexity

The computing complexity of the classification phase (table 5.4) is illustrated in table 5.5.

TABLE 5.5: Computational complexity of the classification phase of the Rotational LDA algorithm

No. of step from table 5.4	Computational complexity
step 2	$O(2d^2 + dL + d) \approx O(d^2 + dL)$
step 3	$O(2dh - h)$
step 4	$O(ch + c)$

The computing complexity of step 2 is dominating among other steps. Thus, the total computational complexity of the classification phase is estimated to be $O(d^2 + dL)$. The next section indulges on the experimentation of the algorithm on several corpuses.

5.3.10 Experimentation and Comparison of Rotational LDA Method Using Several Corpuses

This section demonstrates the performance of the proposed algorithm in comparison with LDA, minimum distance classifier (MDC) and Nearest Neighbor (NN) methods. Seven sets of machine learning corpuses have been utilized namely Sat-Image dataset (Blake and Merz, 1998), Waveform dataset (Blake and Merz, 1998), TIMIT[3] dataset (Garofalo et al., 1986), multiple features (Mfeat) dataset for Karhunen-Loéve coefficients, Fourier coefficients, Zenike moments and pixel averages (Jain et al., 2000). The coordinates of these datasets taken in this experiment are illustrated in table 5.6.

TABLE 5.6: Coordinates of the datasets used for the experimentation

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

To conduct the experimentation, original feature vectors are reduced from d-dimensional space to h-dimensional space, where $h = 1,2,\ldots,c-1$ since $d > c$ for all the databases. The 'classification error in percentage' (ε) is reported for all the h dimensions. The lesser the classification error, the better the performance of the algorithms. It should be noted that MDC and NN are not dimension reduction methods. Therefore the classification error tests for these methods are conducted on d-dimensional space, whereas for LDA and Rotational LDA the tests are conducted on h-dimensional space. For Rotational LDA, $L = 10$ vectors are taken at a time for testing and Perror is also reported for each of the iterations until convergence is reached. The results are reported in table 5.7 for all the databases.

[3] From TIMIT corpus a set of 10 distinct vowels are extracted, then each vowel is divided into three segments and each segment is used in getting mel-frequency cepstral coefficients with energy-delta-acceleration (MFCC_E_D_A) feature vectors (Young et al., 2002).

In table 5.7 there are some blank spaces (marked as '-') under iteration III columns, this means that the Rotational LDA algorithm converged prior to iteration III (i.e. at iteration II).

It is evident from all the conducted tests that Rotational LDA is performing far better than LDA technique in terms of reducing the classification error. The appreciation of the algorithm comes when the classification error produced was less than the NN and MDC classifiers for all the tests.

The minimum classification error for Sat-Image dataset produced by LDA is 19.2%, whereas only 1.1% by Rotational LDA algorithm. Similarly for databases Mfeat-Fourier coefficients, Mfeat-Zernike moments, Mfeat-pixel averages, Mfeat-Karhunen-Loéve coefficients, TIMIT and Waveform, minimum classification error produced by LDA are 19.0%, 19.8%, 4.2%, 5.0%, 11.2% and 17.8% respectively, whereas that by Rotational LDA are 3.8%, 12.0%, 0.0%, 0.2%, 5.7% and 4.1% respectively. Thus, minimum classification error rates are better for Rotational LDA for all the datasets used in the chapter. It can also be observed that Rotational LDA is producing better classification error rate at very low dimensional space (1 or 2) for all the datasets except for the TIMIT dataset. Nonetheless, the classification error for TIMIT reduces gradually and becomes better than LDA algorithm when dimension *h* is increased. Overall, it can be concluded that Rotational LDA performs better than the other techniques presented in this chapter. The next section follows with our concluding remarks and suggestions for future work.

5.4 Summary and Future Work

By considering rotational transform in the original feature space, a more general perspective of LDA can be envisioned: rotational LDA. It was seen that rotational LDA plays a major role in reducing the overlapping between the classes of the projected samples in reduced feature space. Consequently, this increased the classification performance which was empirically demonstrated. The proper mathematical derivations were also presented to support the arguments.

The following suggestions could be made for future work:

- Optimize the Rotational LDA algorithm such that the iterative application of original feature vectors can be minimized or removed.
- Apply some other discriminant techniques than LDA for further improvements.
- Develop a criterion for training phase that could approximately identify the number of samples L to be considered during the testing phase of the algorithm, alternatively, design a method that does not require $L > 1$ samples to be considered at a time.

TABLE 5.7: A comparison of algorithms using classification error in percentage (ε) as a prototype.

Sat-Image

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

NN = 9.7, MDC = 23.4.

Mfeat – pixel averages

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

NN = 2.8, MDC = 6.2.

Waveform

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

NN = 23.3, MDC = 20.

Mfeat – Fourier coefficients

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

NN = 17.8, MDC = 20.2.

Mfeat –Karhunen Loéve coefficients

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

NN = 3.2, MDC = 6.4.

Mfeat – Zernike moments

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

NN = 18.0, MDC = 24.6.

TIMIT – vowels

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

NN = 26.0, MDC = 24.7.

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

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

Chapter 6

Class-Dependent PCA, MDC and LDA: A Combined Classifier for Pattern Classification

6.1 Abstract

Several pattern classifiers give high classification accuracy but their storage requirements and processing time are severely expensive. On the other hand some classifiers require very low storage requirement and processing time but their classification accuracy is not satisfactory. In either of the cases the performance of the classifier is poor. In this chapter we have presented a technique based on the combination of minimum distance classifier (MDC), class-dependent principal component analysis (PCA) and linear discriminant analysis (LDA) which gives improved performance as compared with other standard techniques when experimented on several machine learning corpuses.

6.2 Introduction

Humans can easily recognize faces, spoken words, handwritten or printed digits, images and many other things in everyday life. A high school teacher can recognize several of his/her students in a classroom just by looking at their faces, though it would be difficult for him/her to recognize the faces of all the students in his/her school. Thereafter he/she can also speak out their names or act accordingly once the faces are recognized. This recognition becomes feasible due to some adaptation process in human brain which is a gift of nature to mankind. In other words, if there is a limited number of categories or classes (here, student faces) then recognition performance may be improved however the same might not be very efficient if several categories are present.

Over the past several years study on brain has been conducted and as a result, complex mathematical models have been developed with similar functionalities as the brain but at limited extent only. At present, several neural and cognitive systems have evolved which are of immense value in the applications such as banking, multimedia, forensic science, computer vision, remote sensing, image recognition, speech recognition, defect detection in manufacturing, obstacle avoidance in robotics and others. The recognition of objects depend upon several characteristics; for example in face recognition, location of eyes, width and length of nose/mouth, length of eyebrows and complexion etc. are some characters which would give information about a face. The objective of pattern recognition is to identify any given object or pattern and provide some actions or decisions using computers or automated systems.

The pattern recognition problem can be divided into two main categories (i) supervised classification: where the state of nature for each pattern is known and (ii) unsupervised classification: where the state of nature is unknown and learning is based on the similarity of patterns (Jain et al., 2000). In this chapter only supervised pattern classification procedures have been considered. A supervised classification could be subdivided into two main phases namely training phase and testing phase. In the training phase the classifier is made to learn by known categories of patterns and in the classification or the testing phase unknown patterns which were not part of the training dataset are assigned class label of the nearest category of trained patterns.

How well a given pattern classifier/recognizer can classify or make some predefined decisions in the shortest possible time and at the lowest cost would determine its performance. In general the performance of a classifier depends on several factors. Some of them are (Jain et al., 2000):

(i) number of training samples available to the classifier.
(ii) generalization ability i.e. its performance in classifying test patterns which were not used during the training stage.
(iii) classification error – some measured value based on the incorrect decision of the class labelling of any given pattern.
(iv) complexity – in some cases the number of features or attributes (dimensions) are relatively larger than the number of training samples usually referred as

curse of dimensionality.

(v) speed – processing speed of training and/or testing phase(s) and

(vi) storage – total amount of parameters required to store after the training phase for classification (testing) purposes.

For a fixed number of training samples in a given classifier model, the performance mainly depends on the generalization ability/capability (classification accuracy), speed and implementation cost (due to storage of information). The number of parameters stored during the training phase that is required in performing classification task (testing), is referred as ‘total parameters’. For a given classifier we can associate the total parameters to the implementation cost of the classification system and the generalization capability may depend upon the type of parameters (distribution, values etc.) used, which is derived from the training phase of classifiers. The higher the total parameters required for classification task the costlier the system would be. Another important factor is the speed of the classifier. The higher the computational speed the lower the processing time. We therefore want to reduce the total parameters and processing time and at the same time least sacrifice the classification accuracy. In other words, we search for the optimal classification accuracy or least classification error, involving as minimum total parameters and processing time as possible. This would allow the system to accurately classify/recognize an object as quickly as possible at low cost.

Several classifiers are found today in which minimum distance classifier (MDC) is one of the most economical one. In MDC each class is estimated by a single prototype, usually a centroid. It provides classification at minimal total parameter requirement and computational demand but could be at the price of accuracy. The goal of MDC is to correctly label as many patterns as possible. The MDC method finds centroid of classes and measures distances between these centroids and the test pattern. In this method, the test pattern belongs to that class whose centroid is the closest distance to the test pattern. For details about MDC see section 2.9.

An alternate way of performing classification is by utilizing linear subspace classifiers (Oja, 1983; Oja and Parkkinen, 1984). Here each class is represented by its Karhunen-Loéve transform (KLT) (Fukunaga, 1990) or principal component analysis (PCA).

The objective of PCA is to find a linear transform for each class using the training patterns for that class in the feature space. This gives class-dependent basis vectors[1]. The first basis vector is in the direction of maximum variance of the given data. The remaining basis vectors are mutually orthogonal and, in order, maximize the remaining variances subject to the orthogonal condition. The principal axes are those orthonormal axes onto which the remaining variances under projection are maximum. These orthonormal axes are given by the dominant eigenvectors (i.e. those with the largest associated eigenvalues) of the covariance matrix. In this classifier, each class is characterized by class-dependent basis vectors and the number of basis vectors used for characterization has to be less than the dimensionality d of the feature space. For more details see (Dony and Haykin, 1997) and section 2.12.

Linear discriminant analysis (LDA) is a well known technique for dimensionality reduction. In LDA, the dimensional embeddings are reduced in such a way that the orientation of the projected data of classes on an arbitrary line or space is well separated from each other. The transformation vectors w are taken so that the criteria J is maximum, where J is the ratio of between-class scatter matrix (S_B) and within-class scatter matrix (S_W) (Duda and Hart, 1973). In a c-class problem the LDA projects from d-dimensional space to $c-1$ or less dimensional space ($R^d \rightarrow R^{c-1}$). There are some limitations in applying LDA directly viz. matrix S_W can become singular due to high dimensionality of original feature vectors in comparison with low number of training vectors available. To overcome this limitation, a number of authors have proposed the use of class-independent PCA prior to LDA in the feature extraction stage. Swets and Weng (1996) showed two stage PCA plus LDA method where PCA is first used for dimension reduction so as to make S_W non-singular before the application of LDA especially when training samples are scarce. Belhumeur et al. (1997) proposed a projection method which is based on LDA and PCA techniques for face recognition. In their technique class-independent PCA is first reduce the original space to $N-c$ (where N is the number of training samples available), and then LDA is applied to reduce the dimension to $c-1$. Zhao et al. (1998; 1999) demonstrated a technique based on the combination of LDA and PCA. A complete Kernel Fisher discriminant (KFD) was introduced to implement kernel PCA plus LDA strategy by

[1] Note that here we are using the class-dependent PCA for classification. PCA is also used in a class-independent fashion for feature extraction (Fukunaga, 1990).

Yang et al. (2005) after KFD implementation by Mika et al. (2003). Wu et al. (2004) presented a direct LDA method that is applicable to small sample size problems. Jian et al. (2004) suggested subspace algorithm for determining the optimal projection for LDA that addressed two LDA problems viz. 'small sample size' and 'illumination and pose variations'. Xiaogang and Xiaoou (2004a) then presented a unified framework using PCA, LDA and Bayes techniques for face recognition. Ye et al. (2004) showed generalized optimization criteria based on pseudoinverse for discriminant analysis to address undersample (S_W being singular) problems. In this chapter we are using class-dependent PCA prior to LDA for classification purposes as shown in figure 6.1[2].

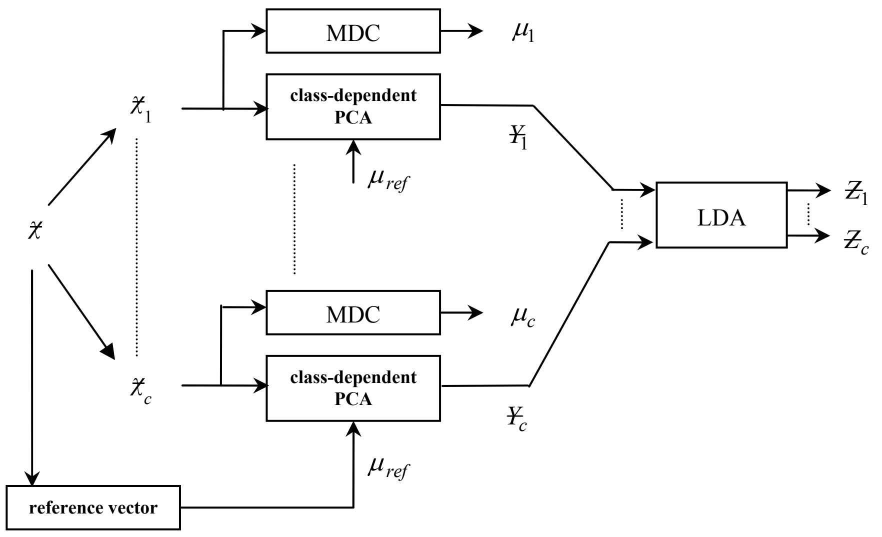


Figure 6.1: Framework of MPL classifier.

LDA was applied in face recognition (Swets and Weng, 1996; Belhumeur et al., 1997; Xiaogang and Xiaoou, 2004a, 2004b; Tae-Kyun et al., 2003), in speech recognition (Haeb-Umbach and Ney, 1992; Siohan, 1995; Lieb and Haeb-Umbach, 2000), in speech reading (Potamianos and Graf, 1998) and in optical character recognition (Chen et al., 2004) and so on.

In this chapter we have presented a unified framework of MDC, class-dependent PCA

[2] Figure 1 has been explained in detail in section 6.7.

and LDA techniques. For brevity we refer to this combination as MPL where 'M', 'P' and 'L' refer to MDC, class-dependent PCA and LDA respectively.

The strategy of combining classifiers has been previously applied by Xu et al. (1992) for handwriting recognition. They have illustrated the combination using some basic classifiers such as Bayesian and kNN, and shown three categories of combination which depend upon the levels of information available from the classifiers. Jacobs et al. (1991) suggested supervised learning procedure for systems composed of many separate expert networks. Ho et al. (1994) used multiple classifier system to recognize degraded machine-printed characters and words from large lexicons. Tresp and Taniguchi (1995) presented modular ways for combining estimators. Woods et al. (1996) and Woods (1997) presented a method for combining classifiers that uses estimates of each individual classifier's local accuracy in small regions of feature space surrounding a test pattern. Zhou and Imai (1996) showed a combination of VQ and multi layer perceptron for Chinese syllables recognition. Alimoglu and Alpaydin (1997) used the combination of two multi layer perceptron neural networks for handwritten digit recognition. Kittler et al. (1996, 1998) developed a common theoretical framework for combining classifiers which uses distinct pattern representations. Breukelen van and Duin (1998) showed the use of combined classifiers for the initialization of neural network. Alexandre et al. (2000) combined classifiers using weighted average after Turner and Gosh (1999). Ueda (2000) presented linearly combined multiple neural network classifiers based on statistical pattern recognition theory. Senior (2001) used combination of classifiers for fingerprint recognition. Lei et al. (2002) demonstrated a combination of multiple classifiers for handwritten Chinese character recognition and Yao et al. (2002) used a combination based on fuzzy integral and Bayes method.

The proposed unification (MPL) could reduce the expected distortion $E[\| x-\mu_j \|]$ (due to MDC), mean squared error $E[\| x-\hat{x} \|^2]$ (due to PCA) and maximize the criteria function J on feature space $\mathbf{R}^h$ after the application of class-dependent PCA, where $\hat{x}$ denotes reconstructed vector of x. All the three individual techniques are linearly combined that could help in reducing the classification error. Each constituent technique in MPL may have its own local regions where it performs the best and this could give better performance than individual techniques. To successfully apply LDA technique in the unification we need to ensure that the scatter matrix S_W does not become singular otherwise it may restrict the direct use of LDA procedure for discriminating features. One way to avoid this situation is to use PCA prior to the application of LDA. We have adopted this two stage PCA and LDA procedure (Swets and Weng, 1996; Belhumeur et al., 1997; Zhao et al., 1999) which is also known as Fisherface method (an LDA based technique) (Belhumeur et al., 1997) and extended the approach by considering class-dependent PCA technique. In Fisherface method, d-dimensional features are firstly reduced to h-dimensional feature space by the application PCA and then LDA is applied to further reduce features to k dimensions. There are several criteria for determining the value of h (Swets and Weng, 1996; Zhao et al., 1999). One way is to select h such that 95% of the variance present in the original feature is retained (Swets and Weng, 1996). Thus MPL could also be applied under the situation where sample size is scarce.

In this chapter we are not proposing any new strategy for combining single classifiers. However, we are using a standard linear combination technique for combining distances from the three classifiers and using the combined distance for classification. The contribution of this chapter is as follows:

1. Modified subspace classifier: In (Oja, 1983; Oja and Parkkinen) as well as in (Dony and Haykin, 1997) PCA is taken with respect to origin. We have used subspace classifier with respect to the class centers.
2. We show that out of the three classifiers (MDC, class-dependent subspace classifier (PCA) and class-dependent PCA+LDA) one may give better classification result than the others depending on the location of the test vector. Therefore, it makes sense to combine the distances from the three techniques to get better results. In this chapter we use linear combination of these three distances.
3. Reference vector (μ_{ref}): In this chapter we have shown that class-dependent PCA would produce overlapping of samples in the reduced dimensional space. If LDA is then applied on the obtained transformed samples it would produce a complex mixture of samples where samples of adjacent class may overlap with each other. This could lead to a poor

performance. This defect should be minimized or removed prior to the application of LDA since if the samples of the adjacent classes are producing overlaps in the transformed space by class-dependent PCA then it would certainly produce overlaps in the transformed space by LDA. We have minimized or removed this problem by introducing a ‘reference vector’ which would prevent the samples of adjacent classes of being overlapped in the transformed space by class-dependent PCA. Then these transformed samples can be further transformed by LDA with fewer errors.

The performance of MPL classifier is compared with MDC, VQ, class-dependent PCA, VQPCA, Fisherface, NN and kNN classifiers and a quantitative analysis of the performance is presented using Sat-Image dataset, TIMIT dataset and Multiple Feature-Digit dataset. The goal of MPL is to label unknown patterns accurately and at the same time maintain total parameter requirement and processing time as low as possible.

The chapter is organized as follows: section 6.3 focuses on the notations and descriptions used in the chapter, section 6.4 briefly describes MDC, class-dependent PCA and LDA techniques, section 6.5 explains a problem in representing class-dependent PCA before the application of LDA technique, section 6.6 deals with the solution of overcoming the problem described in section 6.5, section 6.7 illustrates a general framework of the MPL classifier, section 6.8 deals with the implementation of MPL classifier, section 6.9 presents experimentation on machine learning corpuses followed by concluding remarks in section 6.10.

6.3 Notations and Descriptions

In the remaining discussions $\mathcal{X}$ denotes the d-dimensional set of n training samples in a c-class problem, $\Omega = \{\omega_i : i = 1,2,...,c\}$ be the finite set of c states of nature or class label where ω_i denotes the i^{th} class label. The set $\mathcal{X}$ is partitioned into c subsets $\mathcal{X}_1$, $\mathcal{X}_2,\ldots,\mathcal{X}_c$ where each subset $\mathcal{X}_i$ belongs to ω_i and consists of n_i number of samples such that:

$$n = \sum_{i=1}^{c} n_i$$

The samples or patterns of set $\mathcal{X}$ can be written as:

$\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ where $x_j \in \mathbf{R}^d$ (d-dimensional hyperplane)

$\mathcal{X}_i \subset \mathcal{X}$ and $\mathcal{X}_1 \cup \mathcal{X}_2 \cup \ldots \cup \mathcal{X}_c = \mathcal{X}$

Let $\mathcal{Y}_j$ be h-dimensional transformed samples from $\mathcal{X}_j \in \omega_j$ where $h < d$ then the samples of reduced dimensional set or projected sample set $\mathcal{Y}$ can be depicted as:

$\mathcal{Y} = \{y_1, y_2, \ldots, y_n\}$ where $y_j \in \mathbf{R}^h$ (h-dimensional hyperplane)

$\mathcal{Y}_j \subset \mathcal{Y}$ and $\mathcal{Y}_1 \cup \mathcal{Y}_2 \cup \ldots \cup \mathcal{Y}_c = \mathcal{Y}$

For convenience the notations used in the rest of the chapter are elaborated as follows:

$L(x)$: class label of test pattern x

μ_j : centroid of subset $\mathcal{X}_j \in \omega_j$

Σ_j : covariance of subset $\mathcal{X}_j \in \omega_j$

Φ_j : $d \times h$ transformation matrix of subset $\mathcal{X}_j \in \omega_j$ (during PCA)

ϕ_i : eigenvector that is a subset of Φ_j

$\hat{\delta}_j$: normalized weighted distance

α or α_i : weighting coefficient

$\hat{x}$: reconstructed pattern of $x \in \mathbf{R}^d$

W : $h \times k$ transformation matrix of set $\mathcal{Y}$ (during LDA)

$\mathcal{Z}_j$: k-dimensional transformed samples from $\mathcal{Y}_j \in \omega_j$ where $k < h$.

6.4 A Review of MDC, Class-dependent PCA and LDA Classifiers

This section briefly describes three constituent classifiers that are used in designing

MPL classifier.

6.4.1 MDC

The training procedure of MDC technique is simple and straightforward. This is single prototype classifier i.e. it finds only one feature vector from a given class. MDC estimates a class by its centroid μ_j and store it for later use in the classification task. The centroid can be computed as follows:

$$\mu_j = \tfrac{1}{n_j} \sum_{x \in \mathcal{X}_j} x \quad \text{for } j = 1,2,\ldots,c \qquad 6.1$$

The storage or total parameter requirement is $d \times c$. In the classification phase, an unknown test pattern is assigned a class label of the stored centroids for which the Euclidean distance is minimum.

6.4.2 Class-dependent PCA

In class-dependent PCA each class is separately represented by its KLT. For given training samples $x \in \mathcal{X}_j$ in a *d*-dimensional feature space it will find an orthonormal transformation matrix Φ_j of size $d \times h$ where $h < d$ is a lower dimensional representation of *d*-dimensional feature space. The transformation y can be obtained from the vector x and Φ_j as:

$$y = \Phi_j^t (x - \mu_j) \qquad 6.2$$

where $y \in \mathcal{Y}_j \in \mathbf{R}^h$ and μ_j is from equation 6.1. The transformation matrix Φ_j is obtained by minimizing mean squared error $E[\| x - \hat{x} \|^2]$ which turns out to be a generalized eigenvalue problem i.e.:

$$\Sigma_j \phi_i = \lambda_i \phi_i \qquad 6.3$$

where $\Phi_j = \{\phi_i : i = 1,2,...,h\}$, $\phi_i \in \mathbf{R}^d$, $\Sigma_j = \underset{x \in \mathcal{X}_j}{E}[(x-\mu_j)(x-\mu_j)^{\mathrm{T}}]$ and λ_i denotes eigenvalues corresponding to ϕ_i.

The eigenvectors $(\phi_1,...,\phi_h)$ of Φ_j should be arranged such that their corresponding eigenvalues are in descending order $\lambda_1 > \lambda_2 > ... > \lambda_h$. The direction of first eigenvector ϕ_1 is the direction of maximum variance of given patterns. The second eigenvector (ϕ_2) contains the maximum amount of variance orthogonal to the first one, and so on. The total parameter requirement for class-dependent PCA classifier can be given as:

$$total\ paramaters = centroid\_paramters + eigenvector\_parameters$$
$$total\ paramaters = c \times d + c \times (d \times h) = cd(h+1)$$

In the classification phase a test pattern *x* is assigned the class label for which the reconstruction distance δ_j is minimum. The reconstruction distance can be illustrated as follows:

$$\delta_j = \| x - \hat{x} \| \text{ where } \hat{x} = \mu_j + \Phi_j \Phi_j^{\mathrm{T}}(x-\mu_j)$$
$$\therefore \quad \delta_j = \| x - (\mu_j + \Phi_j \Phi_j^t (x-\mu_j)) \| = \| (I - \Phi_j \Phi_j^t)(x-\mu_j) \| \qquad 6.4$$

6.4.3 LDA

In LDA the projection is from *h*-dimensional feature space to *k*-dimensional feature space where $k < h$ such that the samples or patterns of classes are well-separated. For a *c*-class problem, assuming $c > 2$ the transformation can be given as:

$z = W^{\mathrm{T}} y$ where $z \in \mathcal{Z}_j \in \mathbf{R}^k$ and y is from equation 6.2. 6.5

The transformation matrix *W* is computed by maximizing Fisher's criteria $J(W) = |W^{\mathrm{T}} S_B W| / |W^{\mathrm{T}} S_W W|$, where S_B and S_W can be computed from equations

2.37 and 2.38:

The transformation matrix *W* is given by (Duda and Hart, 1973):

$$S_B w_i = \lambda_i S_W w_i \qquad 6.6$$

where $W = \{w_i : i = 1,2,...,k\}$. The eigenvectors w_i (columns of *W*) correspond to the eigenvalues λ_i. Since the rank of between-class scatter matrix S_B is $c-1$ or less, $k \le c-1$. See Duda and Hart (1973) and section 2.13 for details.

6.5 Representation Problem with Class-dependent PCA

In a *c*-class problem each class $\mathcal{X}_j \in \mathbf{R}^d$ is separately taken for Karhunen-Loéve transformation which yields sample set $\mathcal{Y}_j \in \mathbf{R}^h$. The center of each transformed subset $\mathcal{Y}_j$ (for $j = 1,..,c$) is identical and located at origin in $\mathbf{R}^h$ plane/hyperplane. This may result the samples of $\mathcal{Y}_j$ overlapping with their neighboring classes. If LDA is then applied on the obtained transformed samples it would produce a complex mixture of samples $\mathcal{Z}_j \in \mathbf{R}^k$ where samples of adjacent classes may overlap with each other. This could lead to poor performance. This defect should be removed prior to the application of LDA since if the samples of adjacent classes are producing overlaps in the original feature space $\mathbf{R}^h$ then the possibility to get well-separated samples in $\mathbf{R}^k$ feature space would be slim where $k < h$. If only class-dependent PCA is used for representation or compression purposes then this would not be an issue since compressed features can be easily represented back in the original space with some definite errors known as reconstruction error $\| x - \hat{x} \|$.

The center of sample subset $\mathcal{Y}_j$ is a common vector in $\mathbf{R}^h$ feature space, this can be illustrated as follows:

From equation 6.2, $y = \Phi_j^{\mathrm{T}}(x - \mu_j)$ where $y \in \mathbf{R}^h$ and $x \in \mathbf{R}^d$

center of $\mathcal{Y}_j$ is simply the sum of equation 6.2 i.e.:

$$\text{center of } \mathcal{Y}_j = \tfrac{1}{n_j} \sum_{y \in \mathcal{Y}_j} y = \tfrac{1}{n_j} \sum_{x \in \mathcal{X}_j} \Phi_j^{\mathrm{T}} (x - \mu_j)$$

$$= \tfrac{1}{n_j} \Phi_j^t \left(\sum_{x \in \mathcal{X}_j} x - \sum_{x \in \mathcal{X}_j} \mu_j \right) = \tfrac{1}{n_j} \Phi_j^t (n_j \mu_j - n_j \mu_j)$$

$$= \mathbf{0} \in \mathbf{R}^h \text{ for } j = 1,2,\ldots,c$$

Therefore it would be helpful to select a reference vector in $\mathbf{R}^d$ feature space that could reduce the probability of overlapping of adjacent samples in lower dimensional space $\mathbf{R}^h$. Then LDA can be applied in $\mathbf{R}^h$ space to get well-separated samples in $\mathbf{R}^k$ space. This is described in the next section.

6.6 Reference Vector for Discriminant Analysis

This section introduces reference vector which is used prior to the applications of class-dependent PCA and LDA techniques. The usage of reference vector with class-dependent PCA and LDA can be viewed in section 6.8. The reference vector will be derived from the $\mathbf{R}^d$ sample space which would help in separating features for class-dependent PCA process in $\mathbf{R}^h$ sample space. Therefore it is required to find a vector in $\mathbf{R}^d$ hyperplane such that its direction and displacement from the origin provides maximum separation of samples of classes $\mathcal{Y}_j$ in $\mathbf{R}^h$ feature space. In other words the solution of conventional eigenvalue problem of equation 6.7 will give the required reference vector:

$$S_B v_i = \lambda_i v_i \qquad 6.7$$

where eigenvectors v_i are the column vectors of $d \times (c-1)$ rectangular matrix V and λ_i are the corresponding eigenvalues of v_i. Note S_B in equation 6.7 is computed on the sample set $\mathcal{X} \in \mathbf{R}^d$.

The reference vector can computed from equation 6.7 as:

$$\mu_{ref} = \lambda_m v_m \tag{6.8}$$

where λ_m is the maximum eigenvalue and v_m is the corresponding eigenvector.

Figure 6.2 illustrates the difference between the projections of class-dependent PCA without using the reference vector (figure 6.2a) and class-dependent PCA with reference vector (figure 6.2b) on Sat-Image dataset (Blake and Merz, 1998; Michie et al., 1994; Feng et al., 1993). The Sat-Image dataset consists of 36-dimensional samples of 6 distinct classes. The 36-dimensional patterns are projected on to 2-dimensional space and each pattern is represented by a string in the figures which corresponds to their class number.

It can be seen from figure 6.2a that all the samples of classes are drawn around the common center which is the origin of 2D space. If LDA is applied in this reduced 2D space then it may not be able to well separate the samples of each class. On the other hand the inclusion of reference vector in class-dependent PCA well separates the projected samples of classes in 2D space (figure 6.2b). This would allow the application of LDA with fewer errors.

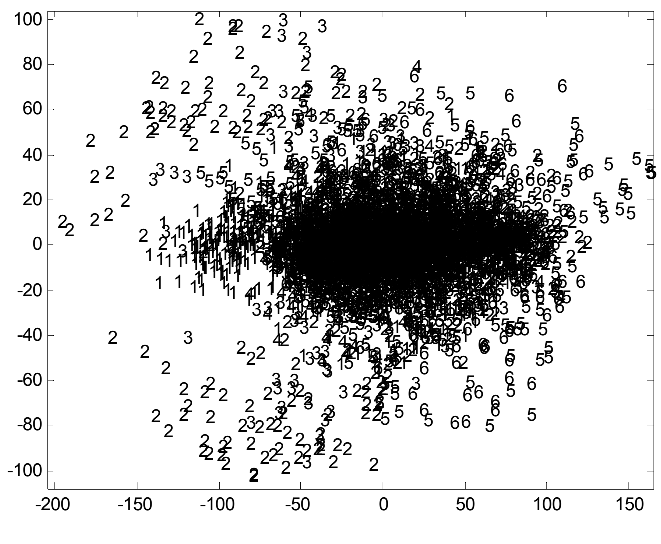


Figure 6.2a: Projection of 36-dimensional features on to a 2-dimensional space using class-dependent PCA.

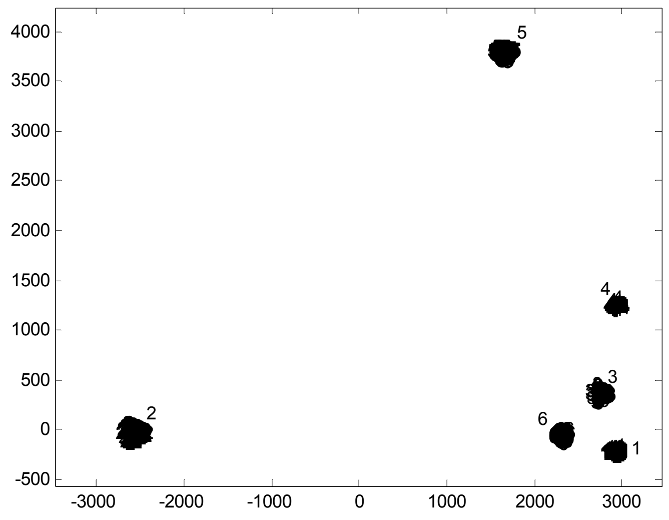


Figure 6.2b: Projection of 36-dimensional features on to a 2-dimensional space using class-dependent PCA including reference vector.

The maximum eigenvalue obtained by solving equation 6.7 for Sat-Image dataset is very large magnitude ($\lambda_m \approx 10^6$) which is reduced to some reasonable value ($\lambda_m^{'} \approx 10^3$) to get a zoomed-in visualization of projected samples in 2D space.

6.7 Framework

This section describes the framework of the overall design of MPL classifier. Figure 6.1 mainly illustrates the training phase of the classifier. The training sample set $\chi \in \mathbf{R}^d$ is first taken for reference vector evaluation procedure. Once the reference vector μ_{ref} is computed it is stored for later use during class-dependent PCA process. The training sample set is a union of the samples of c classes. Therefore sample set χ can be separated into c subsets $\chi_1, \chi_2, \ldots, \chi_c$ where subset χ_i belongs to the class ω_i. All the subsets are taken for MDC and class-dependent PCA procedures. MDC gives centroid μ_j from the subsets and store it for later use in the classification process. All the same subsets are utilized for class-dependent PCA block including reference vector μ_{ref}. This block would give eigenvector matrix Φ_j which is also stored for later use during the classification phase. The eigenvector matrix is used to find transformed sample sets $\mathcal{Y}_1,...,\mathcal{Y}_c$ which belong to classes $\omega_1,...,\omega_c$ respectively. Once all the transformed samples are obtained they are then sent to LDA block. This block finds the orientation (direction matrix W which is stored for later use) such that the new projected sample sets $\mathcal{Z}_1,...,\mathcal{Z}_c$ are well separated.

All the classifiers are linearly combined during the classification phase for decision making. The test pattern is assigned the class label of the closest train pattern for which the combined distance is minimal. Section 6.8.2 depicts the classification phase of MPL classifier in detail.

6.8 MPL

MPL is a unified framework of MDC, class-dependent PCA and LDA techniques respectively. It can be subdivided into training phase and testing or classification phase. The combination may help in reducing the expected errors. All the constituent techniques may have their local regions where they may perform the best. MDC will attempt to reduce the expected distortion $E[\|x-\mu_j\|]$, class-dependent PCA will try to minimize the mean squared error $E[\|x-\hat{x}\|^2]$ and LDA will give the orientation for which Fisher's criteria J is maximum i.e. separating the samples of different classes in lower dimensional space. This approach may improve the generalization capability and at the same time incur low total parameter requirement and low processing time. The application of class-dependent PCA before LDA could also address small sample size problem or singularity problem of matrix S_W. In many cases (e.g. in image recognition or face recognition) the dimension size is very large as compared to the number of feature vectors available. This makes the rank of S_W smaller than the required dimension, making the matrix singular. Several techniques can be found that address small sample size problem e.g. adding a small constant or perturbation to S_W so that it becomes non-singular (Zhao et al., 1999; Zhou and Huang, 2001), PCA application prior to LDA (also known as Fisherface method) for reducing dimension from d to h such that $h \leq$ number of feature vectors (Swets and Weng, 1996; Belhumeur et al., 1997) or the use of pseudoinverse matrix for discriminant analysis

(Ye et al., 2004; Duin, 1995; Skurichina and Duin, 1996, 1999; Raudys and Duin, 1998).

Our procedure can be categorized under Fisherface method where class-dependent PCA is used instead of PCA and a concept of reference vector is introduced and applied. In addition to class-dependent PCA, MDC is also incorporated in the design. Moreover we have concentrated more on getting reasonably well generalization capability using minimal possible storage or total parameter requirement and processing time. The performance of a classifier is reasonably well when it gives high classification accuracy at low memory storage and less processing time. A classifier with high accuracy cannot be categorized as an optimum performance classifier if it is extremely slow in giving any results and the memory requirement is extensively high. Thus we have taken important factors like generalization capability, total parameter requirement and processing time in the consideration while measuring the performance of a classifier. There are several applications of such a classifier for example in obstacle avoidance in robotics where obstacles should be detected at very fast rate and at very low memory storage requirements. In such applications processing time and total parameter requirement are important aspects of classifier performance.

The basic working of the MPL approach can be briefly described by considering a two-class problem. Suppose a 3-dimensional feature space of two distinct subsets $\mathcal{X}_1$ and $\mathcal{X}_2$ of class labels ω_1 and ω_2 are given. The MPL approach first finds the center of the distinct subsets separately, i.e. center or centroid is one of the prototypes of the approach. The reason of using centroid as a prototype for MPL is because it may help in reducing the expected distortion and it is computationally inexpensive procedure with minimal storage requirements. The next thing is to find eigenvectors of subsets by using KLT or PCA on each subset separately (class-dependent). The integration of class-dependent PCA would help in reducing high dimensional space onto a parsimonious data space. The reduction to a lower dimensional space would be such that the error is minimal in mean squared error sense. Moreover, if the number of samples is scarce then it would not be a disadvantage for MPL since the problem of within-class scatter matrix being singular may be resolved by the use of class-dependent PCA prior to the application of LDA. The application of class-dependent PCA will give eigenvectors in the principal direction and in the secondary directions. These eigenvectors are mutually orthogonal to each other and maximize the variances subject to the orthogonality condition. From the eigenvectors and subsets, MPL will transform 3-dimensional feature space to a 2-dimensional (say) feature space using reference vector (see section 6.6 for details about reference vector). Finally in MPL algorithm the LDA step is conducted which would maximize the class separation by eliminating redundant components (i.e. minimizing the overlapping of adjacent or neighbouring classes in lower dimensional plane/hyperplane) that may present in a 2-dimensional feature space. The application of LDA step in MPL approach will bring a 2-dimensional space to a 1-dimensional space (say). The LDA step is integrated since it searches for the directions that are efficient for discrimination.

In the classification phase of MPL, an unknown test pattern is allocated a class label (either ω_1 or ω_2) based on the nearest MPL distance measure. This MPL distance measure is derived from the combination of MDC, class-dependent PCA and LDA techniques.

The reason for combining classifiers is simple. A feature or a feature vector could have diverse characteristics. For example, a feature vector may have different structural primitives, physical properties, variation in numerical values (Xu et al., 1992) etc. All of these diverse characters are then constituted in one vector. This makes it difficult for a single classifier (eg. by MDC or class-dependent PCA or LDA alone) to handle such a diversified feature. If we could combine the different classifiers that could appropriately account for the various forms of a feature then classification performance may improve. To understand this, let us consider the following three cases (figure 6.3) where MPL is performing better than the individual classifiers. In all the three cases the membership of an unknown vector x is to be determined to one of the two regions R_1 and R_2. In case 1 vector x is assigned to R_2 when MDC is used and it is assigned to R_1 when PCA is used. It can be observed that in case 1 MDC is not behaving well than PCA technique since vector x is closer towards R_1 than R_2. On the other hand, in case 2, vector x is assigned to R_1 when using PCA technique which is not performing better than MDC in this particular case. In case 3, a comparison between LDA and MDC is conducted, where vector x is assigned to R$_1$ when LDA technique is used and it is assigned to R$_2$ when MDC technique is used. Here MDC is performing better than LDA for the classification of

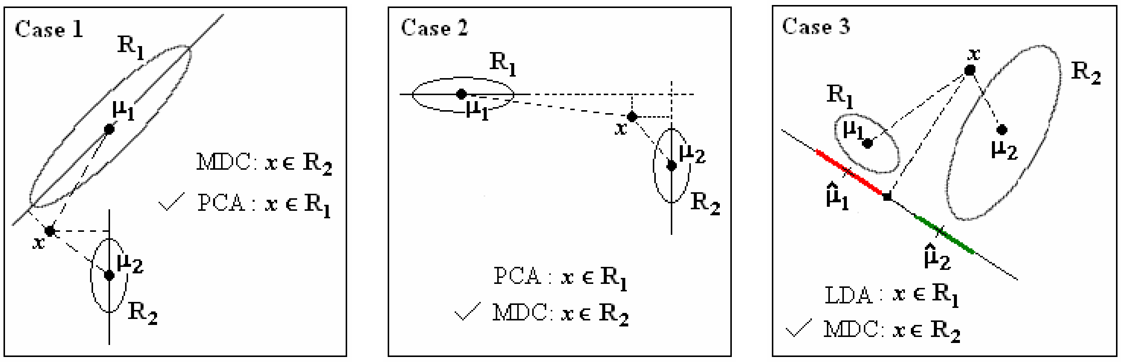


Figure 6.3: Three cases where MPL is performing better than the individual techniques.

vector *x*. Similarly several cases can be demonstrated where single classifiers are not performing well. This means that none of the technique could be stated as the best technique. All the techniques have some regions or local regions where they may perform the best and it is also true that on some regions they may not be able to perform satisfactorily well. However, the combination of single classifiers may improve the classification performance. Now considering the same three cases using MPL technique (details about the classification procedure are given in section 6.8.2), we can observe that in case 1, *x* is assigned to R$_1$ (assuming LDA distance measure is same for both of the regions). Similarly in case 2, *x* is assigned to R$_2$ (assuming LDA distance measure is same) and in case 3, *x* is assigned to R$_2$ (assuming PCA distance measure is same). This means that MPL is performing better than the other techniques. The various distance used in finding the membership of a vector *x* are directly measured from the given figure for all the techniques.

The next subsection describes the training phase of MPL classifier.

6.8.1 Training of MPL

The training procedure can be illustrated as follows:

Evaluation of Reference Vector and MDC

Step1. Find between-class scatter matrix and centroid of each class from the training sample set $\mathcal{X}$ (from equation 4.25):

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

where μ_j can be computed from equation 6.1 and μ can be computed as:

$$\mu = \tfrac{1}{n} \sum_{x \in \mathcal{X}} x$$

Step2. Solve generalized eigenvalue problem and find reference vector (from equations 6.7 and 6.8):

$$S_B v_i = \lambda_i v_i$$

$$\mu_{ref} = \lambda_m v_m$$

λ_m is the maximum eigenvalue and v_m is the corresponding eigenvector.

Class-dependent PCA Step

Step1. Find transformation matrix Φ_j from equation 6.3.

Step2. Project the samples on lower dimensional space $\mathbf{R}^h$:

$$y = \Phi_j^{\mathrm{T}}(x - \mu_{ref}) \qquad \text{where } y \in \mathcal{Y}_j \in \mathbf{R}^h \text{ and } x \in \mathcal{X}_j \in \mathbf{R}^d \text{ for } j = 1,\ldots,c$$

LDA Step

Step1. Find transformation W from the generalized eigenvalue problem of equation 6.6:

$$S_B w_i = \lambda_i S_W w_i$$

where $W = \{w_i : i = 1,2,\ldots,k\}$, S_B and S_W are computed on sample set $\mathcal{Y}$.

Step2. Project the samples on $\mathbf{R}^k$ feature space (equation 6.5):

$z = W^t y$ where $z \in \mathcal{Z}_j \in \mathbf{R}^k$ and $y \in \mathcal{Y}_j \in \mathbf{R}^h$ (it is assumed that $k < h$) for $j = 1,\ldots,c$

Step3. Find centroid $\hat{\mu}_j$ of the projected samples $\mathcal{Z}_j$:

$$\hat{\mu}_j = \tfrac{1}{n_j} \sum_{z \in \mathcal{Z}_j} z \qquad 6.9$$

Some parameters are stored during the training phase which will be used in the classification phase. These parameters and their corresponding size are depicted in table 6.1.

TABLE 6.1: List of parameters stored during the training phase which will be required in classification phase with their corresponding size

Parameters	Unit Size	Total size for c classes
μ_{ref}	$d \times 1$	d
μ_j	$d \times 1$	dc
Φ_j	$d \times h$	dhc
W	$h \times k$	hk
$\hat{\mu}_j$	$k \times 1$	kc

Thus the total parameter requirement for MPL classifier is the sum of column 3 of table 6.1. The next subsection illustrates the classification phase of MPL classifier.

6.8.2 Classification of MPL

The classification procedure of class labelling of an unknown test pattern $x \in \mathbf{R}^d$ is given as follows:

Step1. Compute the distance δ_j^1 between a test pattern x and the centroid μ_j of class $\mathcal{X}_j \in \omega_j$:

$$\delta_j^1 = \| x - \mu_j \| \text{ for } j = 1,\ldots,c$$

Step2. Since the addition of reference vector in the classifier design would not affect the orientation of the components of $\mathcal{Y}$ derived by KLT on $\mathcal{X}$, the reconstruction distance can be given by:

$$\begin{aligned}\delta_j^2 &= \| x - \hat{x} \| \\ &= \| (I - \Phi_j \Phi_j^{\mathrm{T}})(x - \mu_j) \| \quad \text{(from equation 6.4)}\end{aligned}$$

Step3. Find the projected sample of x due to the reference vector μ_{ref} in h-dimensional space:

$$y_j = \Phi_j^{\mathrm{T}}(x - \mu_{ref}) \;\; y_j \in \mathbf{R}^h \qquad 6.10$$

The projected pattern on to h-dimensional space would further reduce to k-dimensional space (from equation 6.10):

$$z_j = W^{\mathrm{T}} y_j \qquad \text{where } z_j \in \mathbf{R}^k \tag{6.11}$$

Compute the distance between the transformed pattern z_j and transformed centroid $\hat{\mu}_j$ (from equations 6.9 and 6.11):

$$\delta_j^3 = \| z_j - \hat{\mu}_j \|$$

Step4. Normalize distances δ_j^1, δ_j^2 and δ_j^3 to eliminate the difference in their amplitudes to allow them to contribute equally in decision making.

$$\hat{\delta}_j^1 = \delta_j^1 / \mathop{\mathbf{max}}_{j=1}^{c}(\delta_j^1)\ ,\ \hat{\delta}_j^2 = \delta_j^2 / \mathop{\mathbf{max}}_{j=1}^{c}(\delta_j^2) \text{ and } \hat{\delta}_j^3 = \delta_j^3 / \mathop{\mathbf{max}}_{j=1}^{c}(\delta_j^3)$$

Step5. Add distances $\hat{\delta}_j^1$, $\hat{\delta}_j^2$ and $\hat{\delta}_j^3$:

$$\hat{\delta}_j = \alpha_1 \hat{\delta}_j^1 + \alpha_2 \hat{\delta}_j^2 + \alpha_3 \hat{\delta}_j^3$$

where α_1, α_2 and α_3 are weighting constant in the range [0,1] such that $\sum_{i=1}^{3} \alpha_i = 1$.

Step6. Find the argument for which the combined distance $\hat{\delta}_j$ is minimum:

$$k = \mathop{\mathbf{arg\,min}}_{j=1}^{c} \hat{\delta}_j$$

Assign the class label ω_k to the test pattern x.

For simplicity we have taken unbiased weighting constant i.e. $\alpha_1 = \alpha_2 = \alpha_3 = 1/3$ (all equal combination) for the classifier design. However, in figure 6.4 we have shown the classification accuracy for 37 different combinations of α's (including all equal combination) using 0.1 weighting interval. The combinations of α are given as follows:

$$(\alpha_1, \alpha_2, \alpha_3) = \underbrace{(0.1,0.1,0.8)}_{1}, \underbrace{(0.1,0.2,0.7)}_{2}, \underbrace{(0.1,0.3,0.6)}_{3}, \ldots, \underbrace{(0.8,0.1,0.1)}_{36} \text{ and}$$

$$\underbrace{(0.33,0.33,0.33)}_{37} \tag{6.12}$$

Three machine learning corpuses have been utilized namely Sat-Image (figure 6.4a)

(Blake and Merz, 1998; Michie et al., 1994; Feng et al., 1993), TIMIT (figure 6.4b) (Garofalo et al., 1986) and Multiple Feature-Digit (figure 6.4c) with Zernike-Moments (Jain et al., 2000; Blake and Merz, 1998) in exhibiting classification accuracy vs. dimension-set for all the combination of α (equation 6.12). The dimension-set refers to $\{h,k\}$ where h denotes reduced dimension obtained by applying class-dependent PCA on initial d-dimensional feature vector and k denotes reduced dimension obtained by applying LDA technique on reduced h-dimensional feature vector. The complete information about all the datasets is given in section 6.9.

The bold line in figure 6.4 denotes the all equal weighting combination and slim lines denote the remaining 36 combinations of α as per depicted in equation 6.12. From figure 6.4 it could be observed that there are many weighting combinations for which the classification accuracy is better than the all equal combination and similarly for many combinations the classification accuracy is poor. The classification accuracy obtained by all equal combination is close to optimum in all the three datasets. Empirically, we can take a weighting combination for which optimal performance may be achieved. But since the theoretical framework for the selection of weighting constants has not been developed in this chapter, we opted to consider all equal weighting combination for the classifier design. However, the presented figure 6.4 gives an idea that there could be some weighting combinations for which optimal results may be achieved. The interested readers may wish to develop a framework or criteria for the selection of weighting combinations that may depend upon the distribution of a given training dataset and the algorithms that are taken into consideration for the combined classifier design.

6.8.3 Computational Complexity

We would be concentrating on the total computing complexity of a classification session, since the training session can be conducted offline. The computing complexity of each step of section 6.8.2 is illustrated in table 6.2.

TABLE 6.2: Computational complexity of the classification phase of MPL algorithm

No. of step from section 6.8.2	Computational complexity
Step 1	$O(dc)$
Step 2	$O(d^2ch)$
Step 3	$O(dch) + O(chk) + O(ck) \approx O(dch)$
Step 4	$O(c)$
Step 5	$O(5)$
Step 6	$O(c)$

The computing complexity of step 2 is dominating among other steps. Thus, the total computing complexity of the classification phase is estimated to be $O(d^2ch)$.

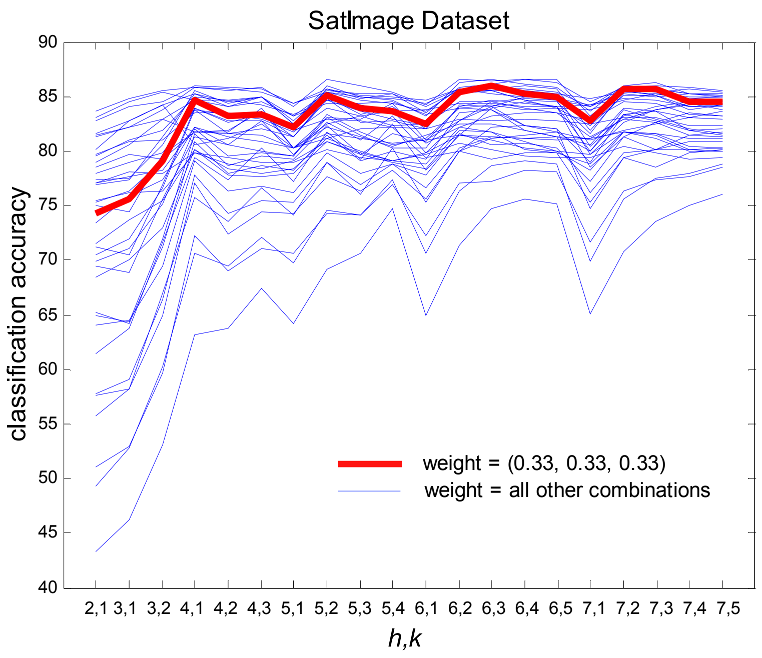


Figure 6.4a: Classification accuracy vs. dimension-set (h,k) for 37 weighting combinations on Sat-Image dataset.

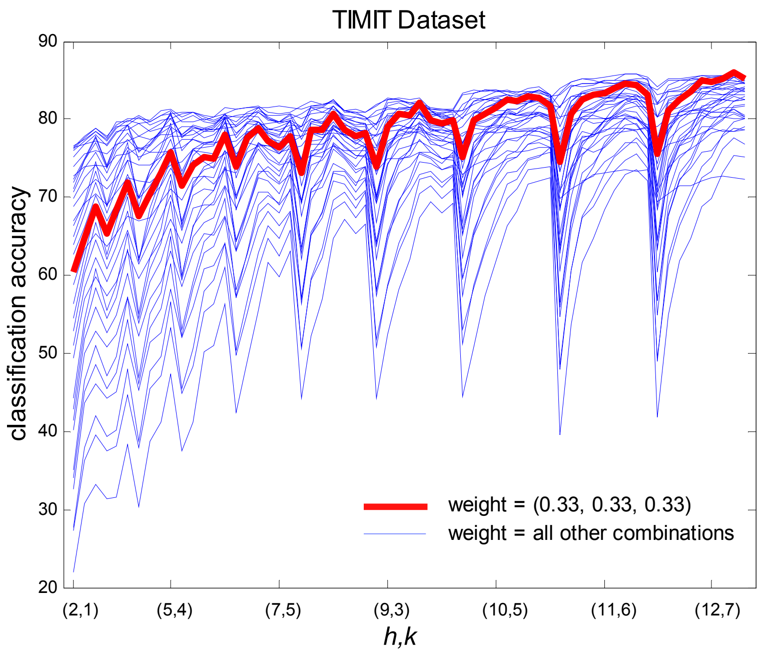


Figure 6.4b: Classification accuracy vs. dimension-set (*h*,*k*) for 37 weighting combinations on TIMIT dataset.

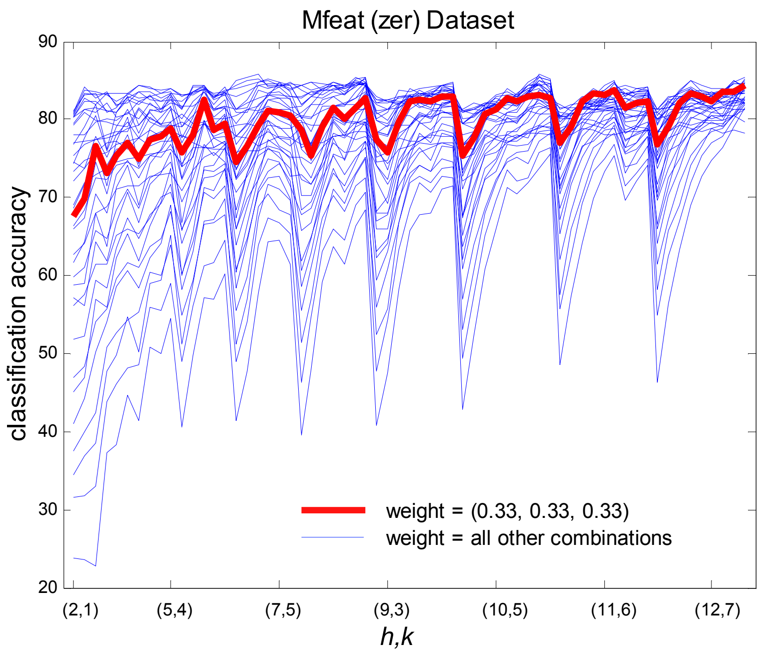


Figure 6.4c: Classification accuracy vs. dimension-set (*h*,*k*) for 37 weighting combinations on Multiple Feature-Digit (Zernike moments) dataset.

6.9 Experimentation

This section demonstrates the performance of the proposed classifier in comparison with MDC, PCA, VQPCA (Euclidean) (Sharma et al., 2005; Kambhatla, 1996), Fisherface, Nearest-Neighbour (NN) and *k* Nearest-Neighbour (kNN) (Fukunaga, 1990) techniques. Three sets of machine learning corpuses are utilized namely Sat-Image dataset, TIMIT dataset and Multiple-Feature-Digit dataset to analyse the classifier performance.

The Sat-Image dataset consists of 6 distinct classes with 36 dimensions or attributes. A sum of 4435 feature vectors is used to train the classifier and a separate set of 2000 vectors is used for verifying the performance of the classifier. From TIMIT corpus a set of 10 distinct vowels are extracted, then each vowel is divided into three segments and each segment is used in getting mel-frequency cepstral coefficients with energy-delta-acceleration (MFCC_E_D_A) feature vectors (Young et al., 2002). A total of 9357 MFCC_E_D_A vectors of dimension 39 for training session and a separate set of 3222 vectors for classification are utilized. The third corpus used is Multiple-Feature-Digit dataset with Zernike moments which consists of 10 distinct classes. A sum of 1500 vectors is used for training the classifier and a separate set of 500 vectors is used for classification. Each vector has 47 attributes or dimensions.

The performance of the techniques is a measure of classification accuracy as a function of total parameter requirement and processing time. Two plots for each corpus are drawn to exhibit the performance of classifiers. The first plot for each corpus (figures 6.5.1a, 6.5.2a and 6.5.3a) shows classification accuracy versus total parameters in logarithmic scale and the second plot (figures 6.5.1b, 6.5.2b and 6.5.3b) shows classification accuracy versus processing time.

The following plotting schemes have been adopted for various classifiers:

- MPL, class-dependent PCA and VQPCA: The first value of classification accuracy curve is plotted for which the total parameter requirement is minimum (for e.g. in MPL the starting point of the graph would be when $h = 2$ and $k = 1$ or in class-dependent PCA when the dimension is reduced to

1). The next value plotted is only that which provided improvement in classification accuracy compared to the previous value. The curve ends at the maximum achievable classification accuracy obtained by a given classifier and thereafter the increase in the total parameters does not improve the classification accuracy any further. This strategy may answer the following question:

"What is the minimal total parameter requirement and its corresponding processing time to achieve a certain range of classification accuracy?"

For VQPCA we have opted not to exceed the levels[3] beyond 16 since the classification accuracy does not significantly increase. Moreover, the total parameter requirement and processing time become severely expensive.

- VQ: all four levels 2, 4, 8 and 16 are presented.
- MDC and NN: one unique result is presented.
- kNN: classification accuracy is presented for five values of K i.e. $K = 3,5,7,9$ and 11.
- Fisherface: The LDA dimension is taken as $k = 1 \ldots p-1$ where $p = c$ if $c < h$ or $p = h$ if $h < c$. The value of h is determined by using a criteria presented by Swets and Weng (1996).

Figures 6.5.1a and 6.5.1b illustrate the performance of classifiers on Sat-Image dataset. It can be observed from figure 6.5.1a that at the beginning, MPL gives classification accuracy of 74.3% at $10^{2.840}$ total parameter requirement and at 3.74 cpu-time in seconds (sec). Increasing the total parameter up to $10^{3.200}$ increases the classification accuracy up to 86.1% at 3.77 sec. It can be seen that no other presented techniques are producing this classification accuracy at specified total parameters and processing time. It can also be seen that NN and kNN techniques are providing better classification accuracy but their total parameter requirement and processing time are severely expensive i.e. their performance is not very encouraging. It can be observed from figure 6.5.1a that MDC gives minimal total parameter requirement (figure

[3] Level is the number of disjoint regions in a given class or the number of sub-classes.

6.5.1b) but the classification accuracy is quite poor (76.6%). For Fisherface method when using $h = 6$ and $k = 1...5$, the total parameter requirement and processing time are reasonably well though the classification accuracy is not very encouraging.

Next we conducted experiments on TIMIT dataset (figures 6.5.2a and 6.5.2b). It is evident from figures 6.5.2a and 6.5.2b that MPL technique is performing better than all the other techniques including NN and kNN in terms of achieving high classification accuracy at low total parameter requirement and low processing time. The classification accuracy of NN technique is even poorer than MDC, class-dependent PCA, VQ, Fisherface ($h = 10$ and $k = 1\ldots9$) and VQPCA techniques; this means that increasing total parameters does not always help in improving the classification accuracy. The maximum classification accuracy for MPL technique is 86.1% at $10^{3.723}$ using 13.28 sec of processing time, whereas the nearest technique to MPL in terms of accuracy is kNN which is giving 78.3% (for $K = 11$) at $10^{5.562}$ using 794.08 sec of processing time.

Figure 5.3a and figure 5.3b illustrate several classifiers performance on Multiple-Feature-Digit dataset using Zernike moments. It can be observed from the figures that class-dependent PCA is giving reasonable level of classification accuracy followed by MPL and other techniques. The maximum classification accuracy by class-dependent PCA is 84% at $10^{3.575}$ total parameters and at 0.65 sec whereas the maximum classification accuracy by MPL is 84.4% at $10^{3.803}$ total parameters and at 1.38 sec. Fisherface ($h = 13$ and $k = 1...9$) is consuming very less processing time but does not provide sufficient classification accuracy. It can also be seen from figure 4c that MPL may perform even better than all the presented techniques if a different combination of α is used.

The running time for each step of the MPL technique was also conducted on TIMIT dataset during the training session. It is given here just for an overview of the processing time of each step of the algorithm. The results are illustrated in table 6.3. The running time is computed using all the ten classes and 9357 feature vectors.

In the table, columns 1 and 2 denote dimension reduced by class-dependent PCA (h)

and dimension reduced by LDA (k) respectively. Dimension h is shown up to 6 and dimension k is shown up to $h-1$. Columns 3, 4 and 5 illustrate processing time of each step of MPL during the training session and column 6 shows the total average time in running the algorithm. It can be observed from the table that as h and k increase the running time of the algorithm increases progressively. It is also evident that class-dependent PCA step consumes the maximum running time and increases with h.

It can be concluded from the experiments on several machine learning corpuses that MPL technique produces promising results in terms of getting classification accuracy in reasonably accepted range and at the same time maintaining minimal total parameter requirement and processing time. This would enable the user to classify a given object accurately and quickly with minimal storage requirements.

We would also like to state here that although MPL is exhibiting better performance overall, it cannot be guaranteed that this technique would produce better performance for all type of data distributions as was seen in the case of Multiple Feature-Digit dataset.

6.10 Summary and Future Work

The chapter presented MPL technique which is a linear combination of MDC, class-dependent PCA and LDA techniques. The performance of the classifiers was measured in terms of classification accuracy as a function of storage and processing time. It was observed that the proposed combination of classifiers provided improved performance over all the other presented techniques. MPL technique on Sat-Image dataset produced maximum classification accuracy of 86.1% at $10^{3.2}$ total parameter requirement and at 3.77 sec. NN and kNN techniques also produced higher classification accuracy but the total parameter requirement and processing time were severely expensive. Similarly on TIMIT database MPL produced the best performance with 86.1% classification accuracy at $10^{3.723}$ total parameter requirement and at 13.28 sec of processing time. On the other hand, in Multiple Feature Digit class-dependent PCA produced best performance among the presented techniques followed by MPL. However, it was noted that MPL could produce better performance if a different combination of α was utilized.

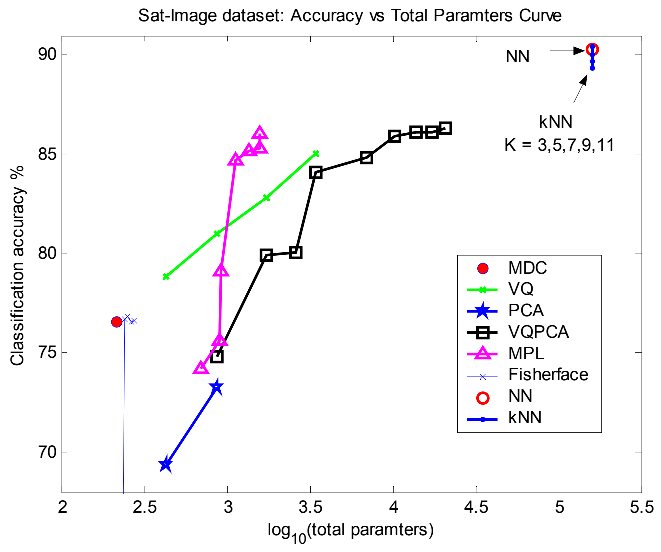
The following questions could be addressed for future work:

- What is the best method of obtaining the subspace dimensions h and k for MPL technique?
- For optimal performance what should be the combination of weighting coefficient α ?
- What is the best criterion of combining individual classifiers for MPL?
- Is there any other classifier that can be combined with MPL for further improvement?
- Theoretically on what type of data distributions the algorithm should achieve better performance?

TABLE 6.3: Processing time of each step of MPL during the training session on TIMIT dataset

h	k	Evaluation of reference vector and MDC step (sec)	Class-dependent PCA step (sec)	LDA step (sec)	Total average running time (sec)
2	1	0.10	3.48	0.03	3.61
3	1	0.08	3.51	0.02	3.62
	2	0.08	3.52	0.02	
4	1	0.07	3.64	0.03	3.70
	2	0.08	3.62	0.02	
	3	0.07	3.54	0.02	
5	1	0.07	3.63	0.03	3.71
	2	0.08	3.59	0.03	
	3	0.09	3.56	0.03	
	4	0.10	3.58	0.03	
6	1	0.07	3.63	0.03	3.72
	2	0.07	3.66	0.03	
	3	0.06	3.56	0.04	
	4	0.05	3.61	0.04	
	5	0.12	3.59	0.03	

Figure 6.5.1a **Figure 6.5.1b**



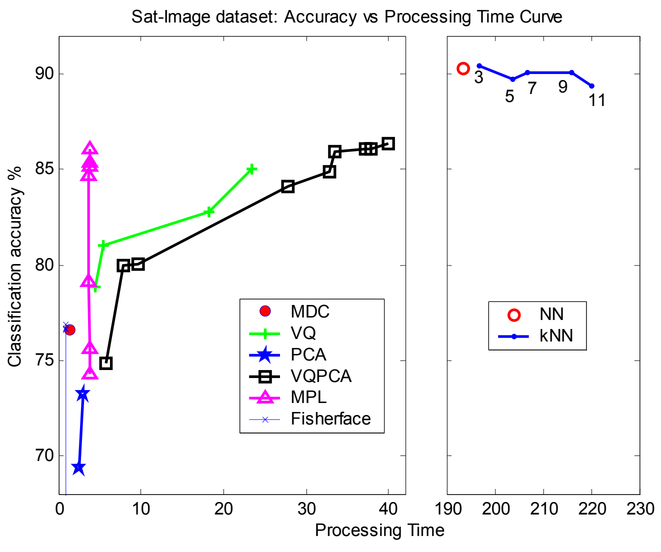
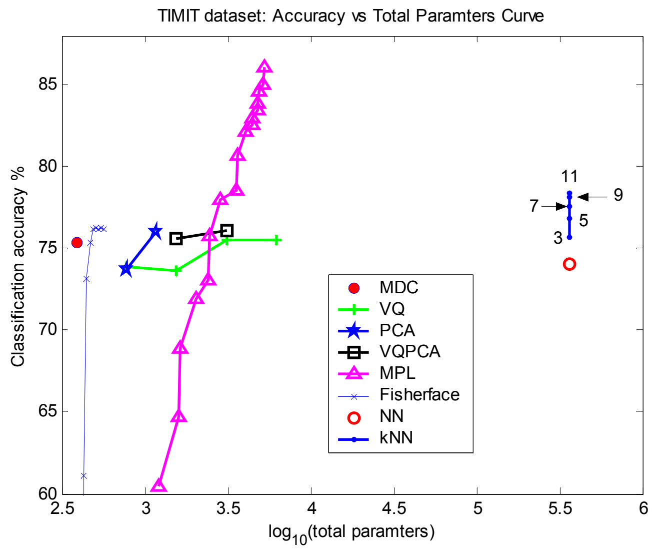


Figure 6.5.2a **Figure 6.5.2b**



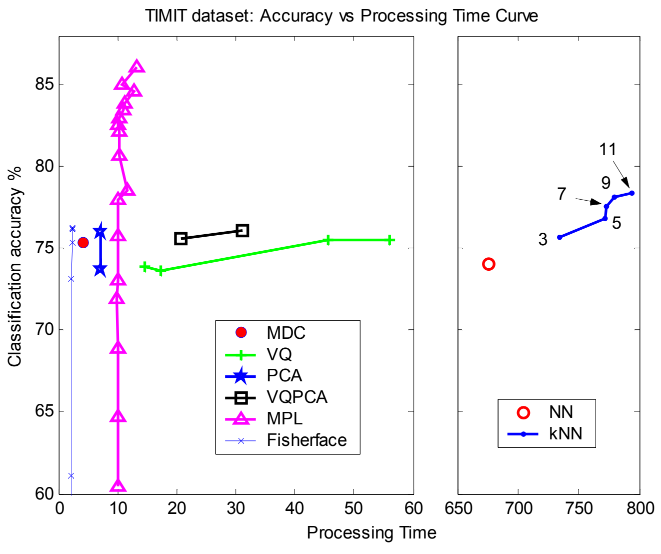
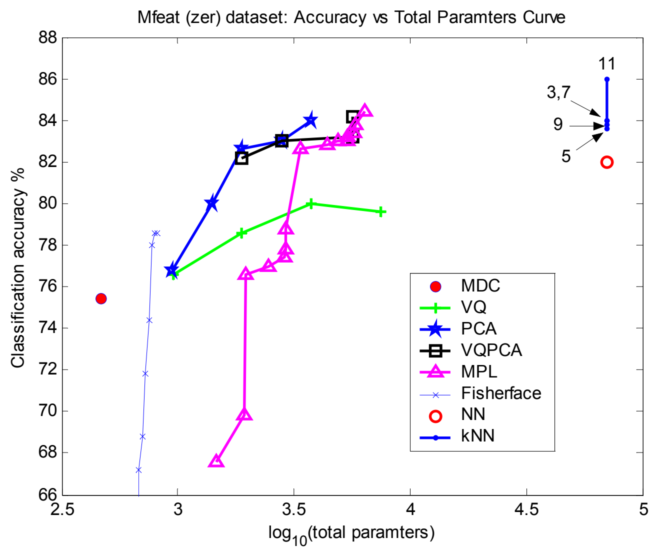


Figure 6.5.3a **Figure 6.5.3b**



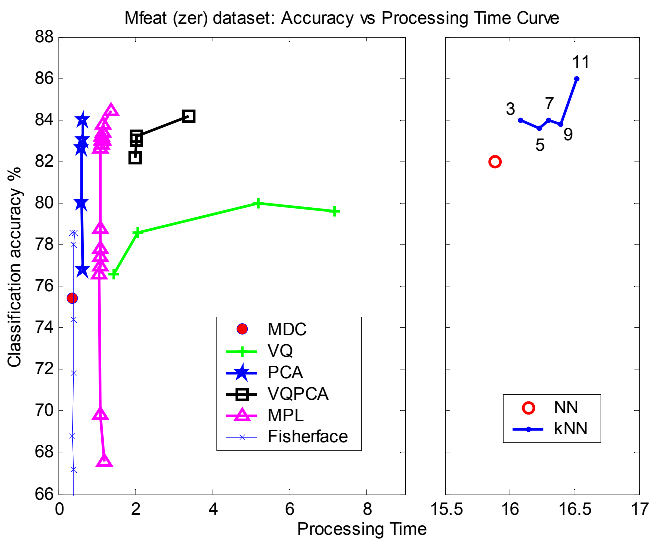


Figure 6.5.1a: Classification accuracy vs. total parameter requirement on Sat-Image dataset.

Figure 6.5.1b: Classification accuracy vs. processing time on Sat-Image dataset.

Figure 6.5.2a: Classification accuracy vs. total parameter requirement on TIMIT dataset.

Figure 6.5.2b: Classification accuracy vs. processing time on TIMIT dataset.

Figure 6.5.3a: Classification accuracy vs. total parameter requirement on Multiple Feature-Digit dataset.

Figure 6.5.3b: Classification accuracy vs. processing time on Multiple Feature-Digit dataset.

Chapter 7

Splitting Technique Initialization in Local PCA

7.1 Abstract

The local Principal Component Analysis (PCA) reduces high dimensional space to a low dimensional space of lesser linearly redundant components. It deploys an *initial guess technique* which can be utilized when the distribution of a given multivariate data is known to the user. The problem in initialization arises when the distribution is not known. This chapter presents a technique that can be easily integrated in the local PCA design and is efficient even when the given statistical distribution is unknown. The initialization using this proposed *splitting technique* splits and reproduces not only the mean vector but also the orientation of components in the subspace domain. This would ensure that all clusters are used in the design. The proposed integration with the reconstruction distance local PCA design enables easier data processing and more accurate representation of multivariate data. A comparative approach is undertaken to demonstrate the greater effectiveness of the proposed approach in terms of percentage accuracy.

7.2 Introduction

Dimension reduction methods are employed in statistical pattern classification problem to represent higher dimensional embeddings in a lower dimensional space such that the information loss in lower dimensional representation is minimum. The interpretation of multivariate data or feature vectors becomes quite unmanageable when the dimension size is high. This severely increases the memory/storage requirements and augments the problems in pattern classification. It then becomes essential to express and understand a given high dimensional vector onto a parsimonious data space that best describes the feature vectors.

The conventional technique for dimension reduction is PCA also known as Karhunen-Loéve technique (KLT) (Fukunaga, 1990). The objective of PCA is to find a global linear transform of a given data in the feature space. See section 2.12 for details about PCA.

Perceiving the constraints involved in PCA, researchers have extended the basic PCA model. Oja (1982) introduced a simple linear neuron model for PCA with constrained Hebbian-type modification and derived unconstrained learning rules and showed how the neuron model extracts the one dimensional principal components. Several other neural network algorithms for PCA have also been developed (Baldi and Hornik, 1995; Peper and Noda, 1995; Oliveira and Romero, 1996; Diamantaras, 1998; Wang et al., 1998; Tao et al., 2003; Chin-Teng, 2004). Hastie (1984) introduced principal curves and surfaces as the estimates of non-linear generalizations of linear one dimensional PCA technique and Tibshirani (1992) presented an alternative definition of principal curves based on a mixture model. Tipping and Bishop (1998) demonstrated how principal axes of a set of observed patterns are determined through maximum likelihood estimation. Xu (1995), De la Torre and Black (2001), and Koren and Carnel (2004) suggested robust PCA model which can perform well under the presence of outliers. A non-linear form of PCA (1998), local linear PCA (Kambhatla, 1997; Dony and Haykin, 1995; 1997b), and mixture of local PCA (Dony and Haykin, 1997) have also been developed. In the local linear PCA approach a class is partitioned into several disjoint regions by vector quantization, and then performs local PCA about each cluster. This local PCA approach is further extended by utilizing hybrid distance (explained in section 7.3) measure also referred to as reconstruction distance (Kambhatla, 1997), which has been proved to be one of the optimal techniques in terms of producing low reconstruction error. Local PCA based on hybrid distance is a replacement of Euclidean distance, which has been proved to be a better distance measurement tool in local linear approach for the cluster separability (Kambhatla, 1997). This distance criteria is derived from the mean squared error (MSE) of the system. The hybrid-distance local PCA also known as vector quantization principal component analysis (VQPCA) for reconstruction distance deploys an *initial guess technique* (Linde et al., 1980), which is utilized when the distribution, of a given multivariate data is known to the user[1]. This means that for a known statistical distribution the reproductive vectors or codewords are initially defined by manually placing them in the vicinity of the data. The number of codewords previously defined will not change during the process except the location of codewords, which will be updated until the best region or cluster is found. In most of the practical cases, statistical distribution is not known to the user which augments the problem of initialization of codewords. Furthermore, in VQPCA some of the codewords if not very carefully selected, end up being isolated having no samples associated to it. This restricts the performance of VQPCA, and thus the model is strongly dependent on the selection of initial codewords.

For VQ algorithms alone, several extensions have been developed (Gresho and Gray, 1992; Paliwal and Atal, 1993; Karayiannis, 1997; Hofmann and Buhmann, 1997; Fritzke, 1997; Patané and Russo, 2001) to improve the performance and overcome drawbacks. Whereas in VQPCA direct implementation of splitting Linde-Buzo-Gray (LBG) technique (1980) cannot be integrated with the hybrid-distance local PCA design since it does not accounts for updating and reproducing eigenvectors (orientation) with the corresponding covariance matrix of regions. To overcome this type of problems associated with the hybrid-distance local PCA technique, we have presented a *splitting* initialization approach that can be easily integrated in local PCA design and is efficient even when the given statistical distribution is unknown. The introduced approach updates not only the centroid (mean) of a cluster but also the orientation of components in subspace domain with the corresponding covariance matrix, through splitting and reproducing codewords. Overall one can view this proposed approach as an improved hybrid-distance local PCA which can efficiently accommodate processing and clustering of unknown statistical distributed data. For brevity we refer to this approach as VQPCA-sp in this paper, where the suffix *sp* denotes initialization of VQPCA using *splitting technique*. The chapter is organized as follows. Section 7.3 and section 7.4 illustrate splitting technique and the implementation scheme respectively. Section 7.5 covers the experimentation and section 7.6 presents our concluding remarks.

[1] For unknown distribution as well the *initial guess technique* can be deployed but it may not provide optimal partitions within class i.e. there could be a possibility of some codewords being remained isolated without associated with any disjoint regions.

7.3 Design Model

This section elaborates the VQPCA-sp approach using hybrid-distance as a distance measurement tool. To explain hybrid-distance, suppose feature vector $\mathbf{x}$ is in d-dimensional hyperplane, $\boldsymbol{\mu}_{im}$ denotes the mean vector of m^{th} partition in i^{th} class and, ϕ_k^{im} denotes k^{th} eigenvector of m^{th} partition which is in i^{th} class, then hybrid-distance is defined as:

$$dist(\mathbf{x},\boldsymbol{\mu},\mathbf{P}) = (\mathbf{x}-\boldsymbol{\mu}_{im})^{\mathrm{T}}\,\mathbf{P}_{im}(\mathbf{x}-\boldsymbol{\mu}_{im}) \qquad 7.1$$

where $\mathbf{P}_{im}$ is a local projection matrix which projects the feature vectors onto a subspace orthogonal to the local h-dimensional PCA hyperplane (Kambhatla and Leen, 1997) i.e.

$$\mathbf{P}_{im} = \sum_{k=h+1}^{d} (\phi_k^{im})(\phi_k^{im})^{\mathrm{T}} \text{ where } h < d \qquad 7.2$$

It is evident from the expression of the hybrid-distance (equations 7.1 and 7.2) that it depends upon eigenvector and mean of local clusters. Thereby these two vectors should be taken under consideration for the reproduction.

In VQPCA-sp approach firstly, the set of feature vectors is separated into disjoint regions by applying vector quantization technique for each given class, and then local PCA is performed using *splitting* technique about each of the cluster center using hybrid-distance. In other words this approach segregates data by class and then performs VQPCA on each class using *splitting* technique. If Euclidean-distance is used in place of hybrid-distance then LBG algorithm could simply be integrated for local PCA implementation. In this case, the regions or clusters are partitioned independently without considering the orientation of PCA and thus produces suboptimal results. On the other hand, the hybrid-distance as given in equation 7.2 depends not only upon the mean of the clusters but also upon the eigenvectors of the covariance matrix of the clusters. In this case a direct splitting LBG algorithm or Enhance LBG algorithm (Patané and Russo, 2001) cannot be integrated in the local

PCA design since an iterative process is required to reproduce eigenvectors as well by updating covariance matrix of the clusters. An *initial guess technique* was utilized in (Kambhatla and Leen, 1997) for hybrid-distance local PCA design where codewords are initialized to random input vectors from the training dataset. If poorly initialized this initialization approach may lead some regions not to be used. If a user has some prior knowledge of statistical distribution of a given feature vectors then the performance (in terms of percentage accuracy or reconstruction error) could improve further. However, this *a priori* information is not always present when the feature vectors are given for processing. In many pragmatic cases, *initial guess technique* is not appropriate and thus it requires some technique that does not require prior knowledge of data distribution. On the other hand, *splitting* technique approach does not require any *a priori* information because the initialized codeword is the center of the data. Thereafter it splits and searches for the region for which the expected error is minimum. Figure 7.1 depicts *splitting* approach on a given two dimensional data presented in an elliptical form. Firstly the initial mean $\boldsymbol{\mu}^0$ and initial eigenvector, ϕ_1^0 as primary component and ϕ_2^0 as secondary component are computed respectively. These reproductive vectors are perturbed using small predefined quantity to get a slight variation in the values. In the figure two new mean vectors after splitting are $\boldsymbol{\mu}^+$ and $\boldsymbol{\mu}^-$ and their corresponding local eigenvectors are defined as $\boldsymbol{\varphi}^+ = (\phi_1^+, \phi_2^+)$ and $\boldsymbol{\varphi}^- = (\phi_1^-, \phi_2^-)$ respectively.

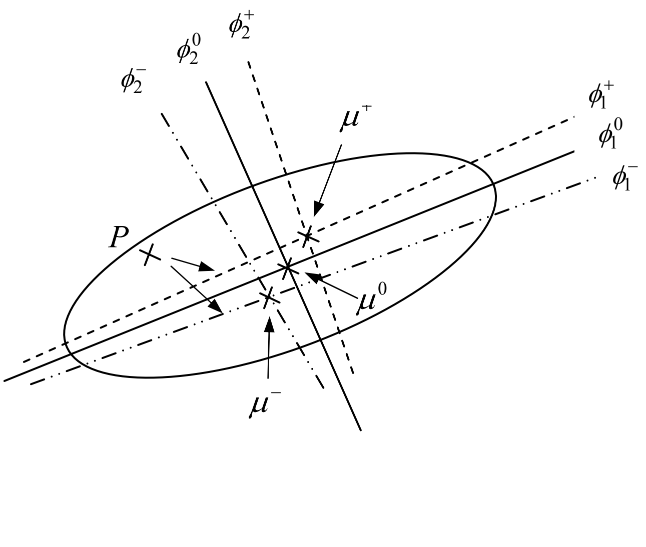


Figure 7.1 *Splitting initialization* process

Any arbitrary vector P of the feature vectors (refer to figure 7.1) is taken into account to determine the membership of the vector to one of the two separated regions (either $\boldsymbol{\varphi}^{+}$ or $\boldsymbol{\varphi}^{-}$) as defined by their reproductive vectors. This will form two new regions and the mean and eigenvector will be updated. The reproduction of the two set of vectors will be carried iteratively until the distortion in reconstruction is smaller than some threshold error (predefined value). Once the satisfactory distortion level is achieved, the reproductive vectors split again and carry out the same above iterative process, until the desired number of clusters is obtained. The determination of membership of the arbitrary point is done by using hybrid-distance. The VQPCA-sp accommodates both of the vectors by updating and reproducing them using iterative process which will be discussed in the next section. The improved hybrid-distance local PCA could be of two types (i) where splitting occurs at random without following any particular direction as illustrated in Figure 7.1 and (ii) where split follows the direction of dominating or principal component. The principal component refers to the eigenvector for which the corresponding eigenvalue is maximum.

7.4 Implementation Scheme

This section deals with the implementation scheme of VQPCA-sp using hybrid-distance as prototype. Suppose in a c-class (assuming $c > 2$) problem let $\mathcal{X}$ denotes d-dimensional set of n feature vectors, $\Omega = \{\omega_i : i = 1,2,...,c\}$ be the finite set of c states of nature or class labels where ω_i denotes the ith class label. The set $\mathcal{X}$ can be subdivided into c subsets $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_c$ where each subset $\mathcal{X}_i$ belongs to ω_i. Each class is subdivided into a set of regions defined by $\xi^i = \{C_k^i ; k = 1,2,...,N\}$, where $N > 2^p$ denotes the total number of desired regions (levels) that is required for each given class and p is any real integer greater than or equal to zero; C_k^i denotes k^{th} regions in $\mathcal{X}_i$. Let d-dimensional training data in $\mathcal{X}_i$ be defined as $\mathbf{x} = \{\mathbf{x}_j^{(i)} ; j = 1,2,...,n_i\}$ where n_i is the number of samples per given class, $\mathbf{x}_j^{(i)}$ denotes any arbitrary feature vector. The transformation $\boldsymbol{\varphi} : \mathbf{R}^d \rightarrow \mathbf{R}^h$ is from d-dimensional hyperplane to h-dimensional hyperplane/plane such that $h < d$. By considering all the mentioned terms above, the VQPCA-sp algorithm can be given as follows:

Step 0. Define threshold error $\varepsilon > 0$, initial average distortion $D_1 \to \infty$ for class ω_1.

Step 1. Initialize mean $\boldsymbol{\mu}$ and covariance Σ as

$$\underline{\mu}_{i1} = \frac{1}{n_i} \sum_{\mathbf{x} \in \chi_i} \mathbf{x};$$

$$\Sigma_{i1} = \frac{1}{n_i} \sum_{x \in \chi_i} (\mathbf{x} - \boldsymbol{\mu}_{i1})(\mathbf{x} - \boldsymbol{\mu}_{i1})^{\mathrm{T}}$$

where $i = 1,2,\ldots,c$. Set the variable level $M \leftarrow 1$.

Step 2. Apply KLT to obtain $d \times d$ eigenvector set $\boldsymbol{\varphi}_{im} = \{\underline{\phi}_l^{im}; l = 1,2,\ldots,d\}$ arranged according to their corresponding eigenvalue set which is placed in descending order, where $\underline{\phi}_l^{im}$ is any arbitrary eigenvector of ω_i class and C_m^i level or partition, for $m = 1,2,\ldots,M$. Split the reproductive vectors as $\boldsymbol{\mu}_{im} = [\boldsymbol{\mu}_{im} + \varepsilon, \boldsymbol{\mu}_{im} - \varepsilon]$ and $\boldsymbol{\varphi}_{im} = [\boldsymbol{\varphi}_{im} + \varepsilon, \boldsymbol{\varphi}_{im} - \varepsilon]$ if the direction of splitting is allowed to be random, otherwise $\boldsymbol{\mu}_{im} = [\boldsymbol{\mu}_{im} + \varepsilon\underline{\phi}_1^{im}, \boldsymbol{\mu}_{im} - \varepsilon\underline{\phi}_1^{im}]$ is used when the splitting is following the direction of principal component $\underline{\phi}_1^{im}$ for which the corresponding eigenvalue is maximum. Update level $M \leftarrow 2M$.

Step 3. Given the sum of $(d-h)$ trailing eigenvectors

$$\mathbf{P}_{im} = \sum_{k=h+1}^{d} (\phi_k^{im})(\phi_k^{im})^{\mathrm{T}}$$

compute the reconstruction distance $d(\mathbf{x}, \boldsymbol{\mu}_{im}, \mathbf{P}_{im}) = (\mathbf{x} - \boldsymbol{\mu}_{im})^{\mathrm{T}} \mathbf{P}_{im} (\mathbf{x} - \boldsymbol{\mu}_{im})$ to get the minimum distance $\delta_{\mathbf{min}} = \min\limits_{m=1,2,\ldots,M} [d(\mathbf{x}, \boldsymbol{\mu}_{im}, \mathbf{P}_{im})]$. Find all feature vectors $\mathbf{x} \in C_k^i$ where $k = \mathbf{arg}[\delta_{\mathbf{min}}]$. For updating $\boldsymbol{\mu}$ and Σ of the new partitioned Voronoi regions, equations

$$\boldsymbol{\mu}_{im} = \frac{1}{n_{i_m}} \sum_{\mathbf{x}\in C_m^i} \mathbf{x}$$

and $$\Sigma_{im} = \frac{1}{n_{i_m}} \sum_{\mathbf{x}\in C_m^i} (\mathbf{x}-\boldsymbol{\mu}_{im})(\mathbf{x}-\boldsymbol{\mu}_{im})^{\mathrm{T}}$$

are used, where n_{i_m} represent number of samples in the new C_m^i Voronoi region. Iterate the process until $(D_{f-1}-D_f)/D_f \le \varepsilon$ where $D_f = \frac{1}{n_i.d}\sum_{\mathbf{x}\in\chi_i}\delta_{\mathbf{min}}$ and f is some iterative number. Follow next step with the revised values of $\boldsymbol{\mu}_{im}$ and ϕ_l^{im}.

Step 4. Iterate step 2 and step 3 until M equalizes the value of N.

Step 5. Follow the same procedure (step 1 – step 4) for all the remaining classes $\{\omega_i; i = 2,3,...,c\}$.

For the reconstruction of vector, expression

$$\hat{\mathbf{x}} = \left(\sum_{l=1}^{h} \phi_l^{im}(\phi_l^{im})^{\mathrm{T}}\right)(\mathbf{x}-\boldsymbol{\mu}_{im}) + \boldsymbol{\mu}_{im}$$

is used where $i = 1,2,...,c$ and $m = 1,2,...,N$. The normalized difference between the vector $\mathbf{x}$ and the reconstructed vector $\hat{\mathbf{x}}$ is the reconstruction error. The next section deals with the experimental results on machine learning corpuses.

7.5 Experimental Results

This section concentrates on the simulation of the discussed models. Several machine learning corpuses have been employed for estimating the accuracy of the proposed model and the existing model. Figure 7.2 depicts percentage error as a function of dimension reduction at levels 2, 4, 8 and 16. The percentage error obtained by VQPCA and VQPCA-sp are represented as small circles (‘o’) and asterisk sign (‘$+$’)

respectively. It is observed that at some points no results were obtained; this is due to the fact that codewords are left out alone, having no feature vectors associated to them, which is an erroneous situation and has not been considered in the testing session. The reconstruction or hybrid distance has been used as a prototype for distance measurement in all the designs.

Figure 7.2a exhibits a design using Multi-Feature Digit Dataset (Jain et al., 2000; Blake and Merz, 1998) of 64 dimensional Karhunen-Loéve coefficients. A total of 1500 vectors of 10 distinct classes were utilized for training session and a total of 500 vectors were used for testing the system. For VQPCA model codewords are initialized close to the centre of samples and updated iteratively until the best cluster in terms of minimum mean square error is achieved. Dimension is reduced from 64 to 1, 2 and 3. It can be observed that VQPCA-sp model demonstrates better performance in all the selected levels in terms of producing lesser error and greatly overcoming the problem of codewords being isolated. For example at level-16 no results are obtained for VQPCA model, this depicts the strong dependence of VQPCA model on the initial selection of codewords, whereas VQPCA-sp produces 2.4% error.

Figure 7.2b illustrates a classifier design using 10 distinct vowels from TIMIT database (Garofolo, 1986). A total of 6000 mel-frequency cepstral coefficients with energy-delta-acceleration (MFCC_E_D_A) (Young et al., 2002) vectors of dimension 39 in training session and 2000 MFCC_E_D_A vectors in testing session were used. Dimension reduction is from 39 to 1, 2 and 3. Here again VQPCA-sp produces much better performance than VQPCA, producing up to 24.3% whereas minimum error obtained by VQPCA is 28.2%.

In Figure 7.2c a classification design using Multi-Feature Digit Dataset (Jain et al., 2000; Blake and Merz, 1998) of 76 dimensional Fourier coefficients was undertaken. A sum of 1500 vectors of 10 distinct classes was utilized to train the classifier. Then a separate set of 500 vectors was used for validation. Dimension is reduced from 76 to 1, 2 and 3. Here also VQPCA-sp exhibits better performance than VQPCA at almost all the levels. The lowest error noted by VQPCA-sp is 15.4% and that by VQPCA is 16.2%. At some points (level-4 dimension 3, level-8 dimension 3, level-16 dimension

1-3) VQPCA is not able to produce any result. However, VQPCA-sp produces result at all the points.

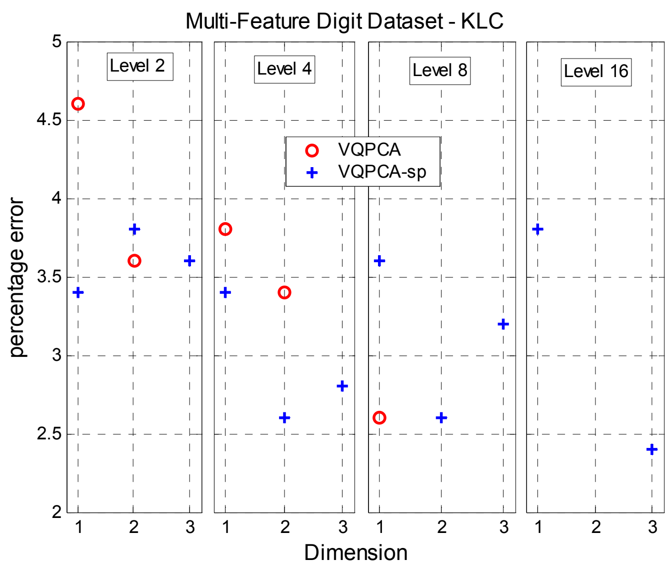
It could be observed from the experiments that VQPCA-sp method produces better representation of multivariate data and able to overcome up to the greater extent the problem related to codewords being isolated with no samples associated to it.

7.6 Summary

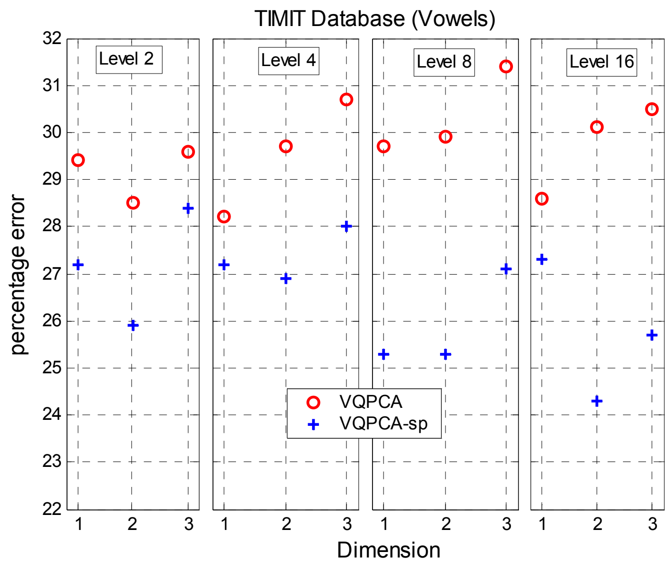
This chapter has described a new splitting technique on local PCA approach (VQPCA) utilizing hybrid-distance as a distance measure tool for cluster separation. It was observed from the experiments on several machine learning corpuses that the proposed approach produces more accurate representation of multivariate data in reduced dimensional space. This VQPCA-sp approach is efficient even when the given distribution of statistical data is unknown. It was also experienced that VQPCA-sp was much easier in initializing codewords and the probability of codewords being left alone was much less as compared to VQPCA. The percentage accuracy obtained by VQPCA-sp model is independent of initial codeword selection since the codewords are selected involuntarily starting from the center of the considered data. This method splits not only the mean but also the orientation of the components on a regular iterative basis which was not accommodated on VQPCA alone.

Figure 7.2 Percentage error as a function of dimension reduction at levels 2, 4, 8 and 16 on machine learning corpuses.

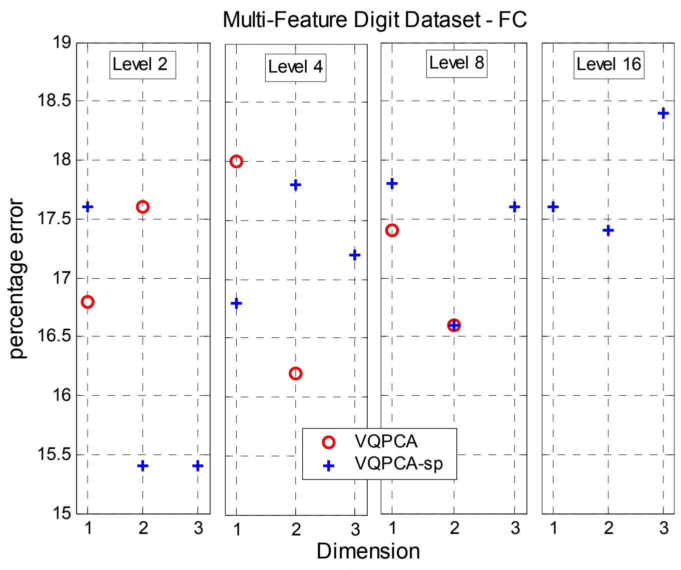
(a)



(b)



(c)



Chapter 8

Pattern Classification: An Improvement Using Combination of VQ and PCA Based Techniques

8.1 Abstract

This chapter firstly presents a brief description on basic classifiers in terms of total parameter requirement and processing time. The classifiers discussed are minimum distance classifier (MDC), vector quantization (VQ), principal component analysis (PCA), nearest neighbour (NN) and *k*-nearest neighbour (*k*NN). Then vector quantized principal component analysis (VQPCA) which is generally used for representation purposes is considered for performing classification task. Some classifiers achieve high classification accuracy but their data storage requirement and processing time are severely expensive. On the other hand, some methods for which storage and processing time are economical do not provide sufficient level of classification accuracy. In both the cases the performance is poor. By considering the limitations involved in the classifiers we have developed linear combined distance (LCD) classifier which is the combination of VQ and VQPCA techniques. The proposed technique is effective and outperforms all the other techniques in terms of getting high classification accuracy at very low data storage requirement and processing time. This would allow an object to be accurately classified as quickly as possible using very low data storage capacity.

8.2 Introduction

Pattern classification/recognition is an area where we learn how to best familiarize the objects to the machine and get actions or decisions based on the observed categories of the pattern. A pattern could be human face, sampled speech, handwritten or printed digits, any letter, gesture, spoken word, financial data, biometric data or any statistical data. Humans naturally classify/recognize patterns from the environment in everyday life. A five year old kid can adapt to different type of objects or patterns and react accordingly. This adaptation is taken for granted until we come to teach a machine to classify/recognize and provide actions or decisions on the same patterns.

The more the patterns available, the better the decision would be. This gives hope to design a classifier system. For the last five decades research is going on in this field to provide an optimum classifier/recognizer. But the classifier performance is still far behind the perception of a human brain. However, pattern classification/recognition plays a crucial role in the areas like banking, multimedia communication, data synthesis, speech or image processing, forensic sciences, computer vision and remote sensing, data mining, robotics and artificial intelligence. It emerged as an essential and integral part of daily life. The evolving computational demand in pattern classification makes this field very challenging and thus open for research. For example in image recognition, several thousands of multidimensional patterns are required for processing which makes the implementation of the classifier system quite impossible.

In this chapter only supervised pattern classification procedures have been considered. A supervised classification could be subdivided into two main phases namely training phase and testing phase. In the training phase the classifier is learned by known categories (classes) of patterns and in the classification or the testing phase unknown patterns which were out of the training dataset are assigned class labels of train patterns for which the distance from the test pattern to the prototype(s) is minimum.

The performance of a classifier depends upon several factors. Some of the main factors have been previously discussed in section 6.2. For a given classifier we can associate the total parameters to the implementation cost of the classification system and the generalization capability may depend upon the type of parameters (distribution, values etc.) used. The higher the total parameters required for classification task the costlier the system would be. Another important factor in classifier design is the speed or the processing time required to do the task. It is possible in a classifier that at two different instances the total parameter requirement is same but the processing time differs. We therefore want to reduce the total parameters and processing time but at the same time least sacrifice the classification accuracy. In other words, we search for the optimal classification accuracy or least classification error, involving as minimum total parameters and processing time as possible. This would allow the system to classify/recognize an object as quickly as possible at minimum cost.

In Nearest Neighbour (NN) classifier (Fukunaga, 1990) all the available data (as maximum as possible) is stored to perform classification, where each test pattern is compared for similarity with all the available training data (pattern). The test pattern is assigned the class label of that training pattern, which is the closest to the test pattern. A major drawback of NN approach is its large total parameter requirement to perform classification task. For example, a dataset with 10 classes, having 5000 vectors or patterns in each class with 64 attributes or dimensions would require total parameters as follows:

$$total\ parameters = class \times NoOfVec \times dimension = 10 \times 5000 \times 64 = 3.2 \times 10^{6}$$

The next method is *k*-Nearest Neighbour (kNN) (Fukunaga, 1990) technique in which the total parameter requirement is same as that of NN approach except for the computational demand, which is severe in the former approach.

The implementation cost of the classification system could be reduced by estimating each class by a single prototype, usually a centroid. This would help in decreasing the total parameter requirement for the classification task but could be at the price of classification accuracy. This type of classifier is known as minimum distance classifier (MDC). It provides minimal total parameter requirement and computational demand. Taking the same above example of 10 classes, the total parameter requirement for MDC would be just 640, which is about 1/5000 as compared to NN approach. Usually classification accuracy is sacrificed to get this advantage of extremely low processing time and total parameter requirement.

The natural extension of single prototype is multi prototype, where each class is estimated by several prototypes like in vector quantization (VQ) (see section 2.8). So the training procedure is to find the codebook and store it for classification task. Increasing the number of codewords per class would increase the performance up to some extent but it would also augment the total parameter requirement and processing time.

In principal component analysis (PCA) finds a global linear transform of given patterns in the feature space and produce class-independent basis vectors which is not helpful for classification. Class-independent PCA cannot be used for classification purposes since all the classes are scattered over the feature space with different centroid values or mean and variances for each class making impossible to preserve the individual class information by a single KLT for the entire train samples. Therefore dominant eigenvectors are taken for each class separately (class-dependent).

It has been seen that the subspace classification is further improved by its local linear extension (Kambhatla and Leen, 1997). Here the performance depends upon the subspace dimension and the number of local regions. Kambhatla and Leen (1997) and Kambhatla (1995) have shown local linear PCA or VQPCA for representation purposes. The goal of VQPCA is to minimize the mean squared reconstruction error $E[\| x-\hat{x} \|^2]$ where $\hat{x}$ is the reconstructed pattern of *x*. Kambhatla (1995) showed VQPCA using Euclidean distance (VQPCA-Euc) and VQPCA using reconstruction distance (VQPCA-rec). VQPCA-rec is a better technique than VQPCA-Euc for representation purposes in terms of achieving lesser reconstruction error, but this achievement comes with the expense of higher total parameter requirement and computational demand. For example, taking the same 10 class problem, where each class is subdivided into 4 disjoint regions (local regions), this would require storage of $d \times d$ (64×64) eigenvector set for each disjoint region together with other parameters (centroid of disjoint region) i.e.

$$total\ parameters = parameters\ due\ to\ eigenvectors\ +\ parameters\ due\ to\ centroid$$
$$total\ parameters\,(VQPCA\text{-}rec) = (d \times d) * class * level + (d \times 1) * class * level$$
$$total\ parameters\,(VQPCA\text{-}Euc) = (d \times h) * class * level + (d \times 1) * class * level$$

where the term *level* is the number of disjoint regions or local regions per class and $h < d$. This yields total parameters requirement for VQPCA-rec 1.66×10^5 (for $d = 64$), whereas 7680 (for $h = 2$) for VQPCA-Euc which is $1/(\frac{d+1}{h+1})$ compared to VQPCA-rec. Although the VQPCA-rec model exhibits slight improvement over VQPCA-Euc model, it severely increases the total parameter requirement and computational demand. This would increase the implementation cost and processing time of the classification system. Considering the implementation cost and computational demand we opted for an economical model (VQPCA-Euc) to train the system. Hereafter VQPCA-Euc model will be referred as VQPCA model. Some modification is required in VQPCA model prior to use as a classifier. The current VQPCA model first partitions the data space into disjoint regions and then performs local PCA about each cluster[1] centre. This is ideal for representation purposes but for the classification task a minor change in distance measurement is required which should reflect the distance of a test pattern from the centroid and dominant eigenvectors of each disjoint region concurrently. The VQPCA model as a classifier does not exhibit very encouraging results but still can be used to perform classification task. Nonetheless it can be shown that VQPCA model as a classifier behaves satisfactorily in terms of obtaining reasonably well percentage accuracy at low total parameter requirements and processing time. Section 8.4 deals with the VQPCA model as a classifier.

The performance of VQPCA as a classifier could be significantly improved by combining the linear distances of VQ and VQPCA. The normalized reconstruction distance measure $\| x - \hat{x} \|$, and the normalized distance between the test pattern and the center of disjoint region $\| x - \mu_j \|$, are combined linearly to form a new distance measure for the classification. This distance measure would minimize the combination of the mean squared reconstruction error (MSE) $E[\| x - \hat{x} \|^2]$ and the expected distortion $E[\| x - \mu_j \|]$. Each distance added together may have its own local regions in the feature space where it performs the best. We have introduced this linear combination of distance (LCD) technique and shown in this chapter that it is a better classifier with no extra total parameter requirement than VQPCA. Classification

[1] Cluster is referred as a disjoint region of a class.

results obtained by LCD exhibit significant improvement over MCD, VQ, VQPCA, NN and kNN classifiers in terms of achieving higher percentage accuracy or lower classification error and at the same time maintaining the total parameters requirement and processing time as minimum as possible. Consequently, this would allow classification or recognition of the objects as quickly as possible at minimum cost. Section 8.5 deals with LCD approach and Section 8.6 indulges on the experimentation using Sat-Image dataset (Blake and Merz, 1998; Michie et al. 1994) and TIMIT dataset (Garofalo, 1986), and Section 8.7 presents our concluding remarks.

8.3 Conventional Classifiers

This section briefly describes the six types of classifiers namely NN, kNN, MDC, VQ, PCA and VQPCA. The style of notations is adopted from Duda and Hart (1973). In all the discussions ω_i denotes the state of nature or class label of i^{th} class in a c-class problem, $\mathcal{X}$ denotes the set of n train samples, $\Omega = \{\omega_i : i = 1,2,\dots,c\}$ be the finite set of c states of nature and let θ' be the class label of train pattern or prototype such that $\theta' \in \Omega$. The set $\mathcal{X}$ can be separated by class into c subsets $\mathcal{X}_1, \mathcal{X}_2, \dots, \mathcal{X}_c$, with the samples in $\mathcal{X}_i$ belonging to ω_i:

$$\mathcal{X} = \{x_1, x_2, \dots, x_n\} \text{ where } x_j \in \mathbf{R}^d \text{ (d-dimensional hyperplane)}$$

$$\mathcal{X}_i \subset \mathcal{X} \text{ and } \mathcal{X}_1 \cup \mathcal{X}_2 \cup \dots \cup \mathcal{X}_c = \mathcal{X}$$

Let n_i denote the number of samples in the subset $\mathcal{X}_i$, therefore $\sum_{i=1}^{c} n_i = n$.

NN Classifier

The total parameter requirement for NN approach is given by:

$$\textit{total parameters} = d \times \sum_{i=1}^{c} n_i = d \times n \qquad 8.1$$

It can be seen from equation 8.1 that total parameters depend upon the attribute or dimension *d*, number of class and number of train patterns. In many practical applications the values of *d* and *n* are very large which severely affects the storage requirements and processing time, increasing the cost and reducing the speed of the classifier system.

kNN Classifier

The total parameter requirement for *k*NN is same as NN approach. The processing speed is slower than NN classifier due to the searching of *k* nearest patterns for each of the test pattern. The classification accuracy may improve with the increase in the value *k*. This improvement is usually observed when the test patterns and the train patterns are closely matched. However, in some cases when the test patterns and the train patterns do not match the classification accuracy is poor. In this case increasing the value *k* may not improve the classification accuracy of the system.

MDC Classifier

MDC requires minimal total parameter requirement and least computational demand. The total parameter requirement for MDC is:

$$total\ parameters = d \times c$$

which is $c / \sum_{i=1}^{c} n_i$ as compared to NN or kNN classifier. This advantage of low total parameter requirement and fast computation may achieve by sacrificing some classification accuracy.

VQ Classifier

For VQ classifier the total parameter requirement is $d \times (Q \times c)$ where *Q* is the level of classifier i.e. number of disjoint regions or codewords for each of the class.

PCA Classifier

Class dependent PCA is considered for classification where each class is represented by its KLT. The total parameter requirement for PCA classifier is:

$$total\ paramaters = centroid\_paramters + eigenvector\_parameters$$

$$total\ paramaters = c \times d + c \times (d \times h) = cd(h+1)$$

where $h < d$ is the number of eigenvectors used.

8.4 VQPCA as a Classifier

This section elaborates the VQPCA approach using Euclidean distance for classification purpose. In this approach, firstly, the set of train patterns are partitioned into disjoint regions by applying VQ technique for each class separately and then KLT is performed about each of the disjoint region or local region center (Kambhatla and Leen, 1997). The aim of VQPCA is to minimize MSE $E[\|x-\hat{x}\|^2]$ in the local regions, where $E[\bullet]$ is expectation operator. To illustrate training and classification procedures let Q be the number of disjoint regions or levels per class. Details of the training procedure are given in Kambhatla (1995). Section 8.4.1 gives a brief description of the training procedure for VQPCA classifier. VQPCA can also be trained using splitting technique (sharma et al., 2006b).

8.4.1 Training

Step1. Take train patterns $\chi_i \subset \chi$ of class label ω_i at a time for consideration, where $i = 1,2,...,c$.

Step2. Apply VQ technique and partition χ_i into Q disjoint regions; for all $i = 1,2,...,c$.

Step3. For each disjoint region compute centroid μ_j and covariance matrix Σ_j where $j = 1,2,...,(c \times Q)$.

Step4. Evaluate $d \times h$ rectangular matrix of eigenvectors $W_j = \{w_l : l = 1,2,...,h\}$ for each disjoint region where $h < d$ and w_i is from equation 2.31 (where w_i and ϕ_i represents the same eigenvectors); arrange the obtained eigenvectors such that its corresponding eigenvalues are in descending order. Let the class label of eigenvector set W_j be $\theta'_j \in \Omega$.

Step5. Store W_j and μ_j with their corresponding class information for classification.

The total parameter requirement for VQPCA can be given by:

$$total\ paramters = parameters\_centroids + paramters\_eigenvectors$$
$$total\ paramters = Q \times d \times c + Q \times (d \times h) \times c = Qdc(h+1)$$

which is Q times the total parameter requirement of PCA classifier.

If VQPCA is used for representation purposes then in the decoding step (here classification) firstly the closest disjoint region to a test pattern x is computed. Once the closest region is obtained, next step is to use its corresponding eigenvector and centroid information to compute reconstructed pattern $\hat{x}$. For classification VQPCA procedure would provide no better performance than VQ technique since the decision would lie only on the closest disjoint region to the test pattern x and the computation of KLT for disjoint regions may become redundant. Therefore a procedure for decision making of a test pattern should be adopted that uses both the centroid and direction (eigenvector) information in parallel. Section 8.4.2 illustrates the classification procedure for VQPCA approach. This procedure would give better classification accuracy than VQ technique for small value of dimensions h which is elaborated in the experiment section 8.6.

8.4.2 Classification

Step1. Compute reconstruction distance δ_j between a test pattern x and its reconstructed pattern $\hat{x}$:

$$\delta_j = \| x - \hat{x} \| = \| (I - W_j W_j^{\mathrm{T}})(x - \mu_j) \| \quad for\ j = 1,2,\ldots,(Q \times c)$$

Step2. Find the argument for which the reconstruction distance is minimum:

$$k = \mathbf{arg} \min_{j=1}^{Q \times c} \delta_j$$

Step3. Assign class label $\omega_r = \theta'_k$ to the test pattern x, where $\theta'_k \in \Omega$.

Thus, it can be seen that step 1 computes the error of reconstruction distance by using direction and centroid information in one single step for the classification.

8.5 LCD Classifier

This section describes our proposed LCD approach for classification purposes. The LCD is a combination of VQ and VQPCA techniques. Empirical results show significant improvement of LCD classifier over previously discussed classifiers in terms of getting higher percentage accuracy with total parameter requirement no more than VQPCA approach. In our approach the training phase of the classifier is identical to VQPCA classifier thus the total parameter requirement for LCD approach is same as VQPCA approach. However the classification procedure differs. In the classification phase the distance used in VQ classification and the distance used in VQPCA classification are added together with some weighting to form a new distance measure. This combination or addition may reduce expected distortion $E[\| x - \mu_j \|]$ and MSE or root-MSE $E[\| x - \hat{x} \|]$, overall producing improved results for the combination. The improved results achieved could be due to each of the constituent distance performing the best in their local regions in the feature space.

The generalization capability or classification accuracy of a classifier depends on the type of distribution or values used for training and/or testing the classifier. For e.g. if training patterns of each class is spherically distributed, dense, well separated with each other and test patterns are closely matched with their train patterns then techniques such as MDC, VQ, NN and kNN may perform the best; if outliers are present in the training patterns then techniques such as PCA or VQPCA may give poor performance. However for Gaussian data with matching train and test conditions PCA may provide reasonably high classification accuracy (Jain et al., 2000) and VQPCA and LCD may provide even better performance than PCA. In the presence of outliers and complex distributions (unmatched train and test conditions) LCD may provide better performance than other techniques.

The concept of combination of multiple classifiers has been previously applied by Xu et al. (1992) for handwriting recognition. They have illustrated the combination using some basic classifiers such as Bayesian and kNN, and shown three categories of combination which depend upon the levels of information available from the classifiers. Jacobs et al. (1991) suggested supervised learning procedure for systems composed of many separate expert networks. Ho et al. (1994) used multiple classifier system to recognize degraded machine-printed characters and words from large lexicons. Tresp and Taniguchi (1995) presented modular ways for combining estimators. Woods et al. (1996) and Woods (1997) presented a method for combining classifiers that uses estimates of each individual classifier's local accuracy in small regions of feature space surrounding a test pattern. Zhou and Imai (1996) showed a combination of VQ and multi layer perceptron (MLP) for Chinese syllables recognition. Alimoglu and Alpaydin (1997) used the combination of two MLP neural networks for handwritten digit recognition. Kittler et al. (1996, 1990) developed a common theoretical framework for combining classifiers which uses distinct pattern representations. Breukelen van and Duin (1998) showed the use of combined classifiers for the initialization of neural network. Alexandre et al. (2000) combined classifiers using weighted average after Turner and Gosh (1999). Ueda (2000) presented linearly combining multiple neural network classifiers based on statistical pattern recognition theory. Senior (2001) used combination of classifiers for fingerprint recognition. Lei et al. (2002) demonstrated a combination of multiple classifiers for handwritten Chinese character recognition and Yao et al. (2002) used a combination based on fuzzy integral and Bayes method. Similarly several other research work on combinational classifiers have been reported in the literature.

In our approach the training phase parameters μ_j (centroid) and W_j (eigenvector set) are stored with the class label $\theta'_j \in \Omega$ information for the use in the classification phase which is same as the training phase of VQPCA approach. Let in a c-class problem each class is separately partitioned into Q disjoint regions then the classification phase of LCD approach can be illustrated as follows:

8.5.1 Classification

Step1. Compute the distance δ_j^1 between a test pattern x and the centroid μ_j of the disjoint region:

$$\delta_j^1 = \| x - \mu_j \| \text{ for } j = 1,2,\ldots,(Q \times c)$$

Step2. Compute the reconstruction distance δ_j^2 between a test pattern x and its reconstructed pattern $\hat{x}$:

$$\delta_j^2 = \| x - \hat{x} \| = \| (I - W_j W_j^t)(x - \mu_j) \| \text{ for } j = 1,2,\ldots,(Q \times c)$$

Step3. Normalize distance δ_j^1 and δ_j^2 to eliminate the difference in their amplitudes that would allow them to contribute equally in decision making.

$$\hat{\delta}_j^1 = \delta_j^1 / \max_{j=1}^{Q\times c}(\delta_j^1) \text{ and } \hat{\delta}_j^2 = \delta_j^2 / \max_{j=1}^{Q\times c}(\delta_j^2)$$

Step4. Add distance $\hat{\delta}_j^1$ and $\hat{\delta}_j^2$:

$\hat{\delta}_j = \alpha\hat{\delta}_j^1 + (1-\alpha)\hat{\delta}_j^2$ for $j = 1,2,\ldots,(Q \times c)$, where α is a weighting constant in the range $[0,1]$.

Step5. Find the argument for which the combined distance is minimum:

$$k = \arg\min_{j=1}^{Q\times c} \hat{\delta}_j$$

Step6. Assign class label $\omega_r = \theta_k'$ to the test pattern x, where $\theta_k' \in \Omega$.

The classification phase of LCD technique is simple, computationally inexpensive and attains high classification accuracy or low classification error. The distance $\hat{\delta}_j$ in the classification phase depends on the weighting constant α and the two normalized distance $\hat{\delta}_j^1$ and $\hat{\delta}_j^2$. The weighting constant α (in step 4) is a positive constant in the range $[0,1]$. Appropriate value for α should be taken since bad selection may lead to poor classification accuracy. The two normalized distance $\hat{\delta}_j^1$ and $\hat{\delta}_j^2$ are classification distance of VQ and VQPCA techniques respectively. The next sub-section elaborates the choosing of the value of weighting constant.

8.5.2 Choice of α

The optimum or close to optimum performance by LCD classifier can be obtained by selecting the appropriate value of α empirically. We have used speech data (Garofalo, 1986) and image data (Blake and Merz, 1998; Michie, 1994) to select the value of α. In this chapter we have taken α as a numerical constant, however, one can also take α as a probabilistic model which would depend on a test pattern and the distribution of train patterns. This may increase the computation and storage requirements. The discussion on α as a probabilistic model is beyond the scope of this paper. In figure 8.1 and figure 8.2 classification accuracy for LCD technique is computed for dimension h and level Q, where $h = 1,2...,4$ and $Q = 1,2,4,8,16$. The values of α are $0.1, 0.2, ..., 0.9$, where choosing α values close to 0.1 and 0.9 will give performance similar to VQPCA approach and VQ approach respectively. Diverting either upwards ($\alpha = 0.6,...,0.9$) or downwards ($\alpha = 0.4,...,0.1$) from the center value of α (0.5) will make the distance $\hat{\delta}_j$ biased for $\hat{\delta}_j^1$ or $\hat{\delta}_j^2$ respectively. It can be observed from figures 8.1 and 8.2 that at $\alpha = 0.5$ classification accuracy obtained by LCD technique (in figures 8.1 and 8.2 denoted by bold lines) is close to optimum. This implies that when the distance $\hat{\delta}_j^1$ and $\hat{\delta}_j^2$ contribute equally in the decision making for a test pattern in the feature space then the classification accuracy is close to optimum. Thus we have taken $\alpha = 0.5$. Section 8.6 deals with the experimentation of all the discussed classifiers on speech and image data.

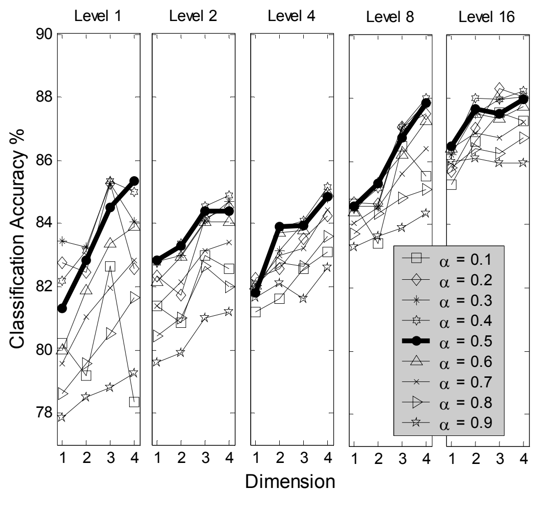


Figure 8.1: Classification accuracy for different values of α on image data.

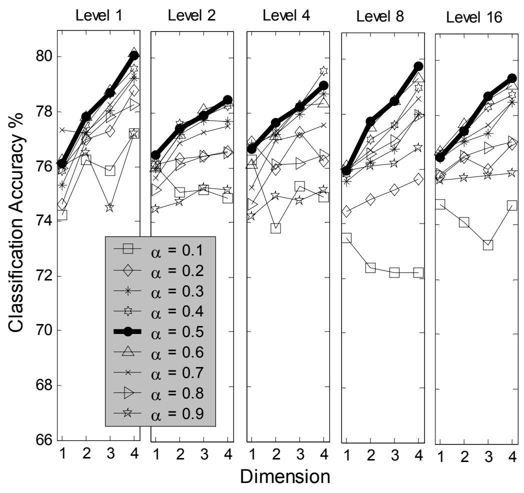


Figure 8.2: Classification accuracy for different values of α on speech data.

8.6 Experimentation

The experimentation section is subdivided into two main parts. The first part demonstrates the classification accuracy for all the techniques given some fixed levels Q and dimensions h. The second part exhibits the effectiveness of all the techniques by showing the maximum achievable classification accuracy by all the discussed techniques, given total parameter requirement and processing time. For all the experiments two sets of machine learning corpuses have been utilized namely TIMIT database (Garofalo, 1986) for speech classification and Sat-Image dataset (Blake and Merz, 1998; Michie et al., 1994) for image classification. From the TIMIT corpus a set of 10 distinct monothongal vowels are extracted, then each vowel is divided into three segments and each segment is used in getting mel-frequency cepstral coefficients with energy-delta-acceleration (MFCC_E_D_A) feature vectors (Young et al., 2002). A total of 9357 MFCC_E_D_A vectors of dimension 39 for training session and a separate set of 3222 vectors for classification are utilized. The second dataset is Sat-Image which consists of 6 distinct classes with 36 dimensions. A sum of 4435 feature vectors is used to train the classifier and a different set of 2000 vectors is used for verifying the performance of the classifier.

In the first part of the experimentation, classification accuracy is measured for all the classifiers given some fixed parameters. Here the accuracy is a function of dimension *h* and level *Q*, where $Q = 1,2,4,8,16$ and $h = 1,2,..4$ for all the levels, except for $Q = 8$, where $h = 1,2,...,10$. Level 8 ($Q = 8$) is taken at random for dimension $h = 1,2,...,10$ to get a general understanding of how the dimension affects the classification accuracy if it is increased continuously.

Not all the techniques depend upon both the dimension *h* and level *Q*; VQ depends upon levels, PCA depends upon dimensions, MDC, NN and kNN depend neither upon dimensions nor on levels, only VQPCA and LCD depend upon dimensions as well as levels. Figure 8.3 (image dataset) and figure 8.4 (speech dataset) illustrate classification accuracy for MDC, VQ, PCA, VQPCA and LCD techniques and table 8.1 depicts classification accuracy for NN and kNN techniques. Usually the MDC technique is a special case of VQ when $Q = 1$, that's why it is represented in the column of Level 1 in figures 8.3 and 8.4.

It can be observed from figure 8.3 (image dataset) that MDC is giving better classification accuracy than PCA; VQ is producing higher classification accuracy at Level 2 and Level 4 than VQPCA, but VQPCA is showing improvement over VQ technique at level 8 and level 16. It is also clear that LCD is performing better than MDC, VQ, PCA and VQPCA at all the levels and dimensions. Increasing the dimension at any given level is improving the classification accuracy of LCD technique. At level 8 and dimension 10 the classification accuracy by LCD is 89.2% which is very close to NN and kNN techniques. It should be noted that NN and kNN techniques produce similar classification accuracy as LCD technique but their processing time and total parameter requirement are severely expensive.

Furthermore, it can be observed from the experiment on speech data (figure 8.4) and table 8.1 that MDC is giving better classification accuracy than NN technique; PCA is improving at dimension 2 over MDC technique; VQPCA is producing better classification accuracy over VQ technique at levels 2 and 4 for dimension 1 but deteriorating at level 8 and level 16. LCD is exhibiting better performance than all the techniques including NN and kNN. The classification accuracy is improving with the increase in dimension at any given level. The classification accuracy by NN and kNN is quite poor for speech data. This may be due to the testing data not matching with their training data.

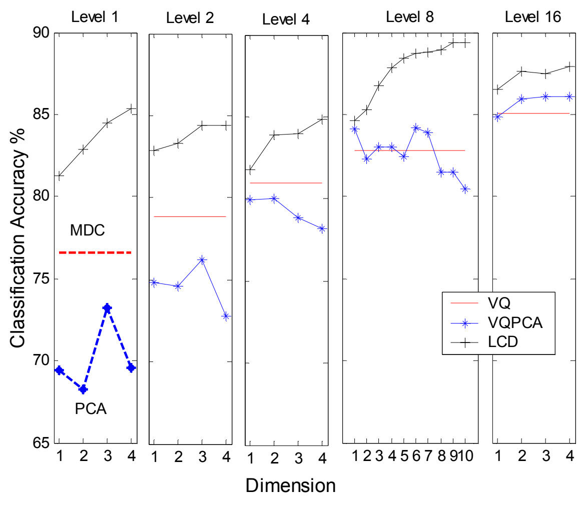


Figure 8.3: Classification accuracy vs. dimensions and levels using MDC, VQ, PCA, VQPCA and LCD on image dataset.

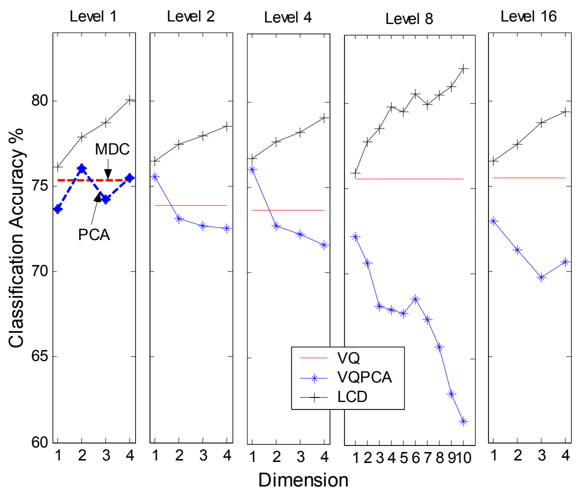


Figure 8.4: Classification accuracy vs. dimensions and levels using MDC, VQ, PCA, VQPCA and LCD on speech dataset.

TABLE 8.1: Classification accuracy for NN and kNN techniques on image and speech datasets.

Technique		Classification accuracy using image dataset	Classification accuracy using speech dataset
NN		90.30	74.05
kNN	3	90.45	75.67
	5	89.70	76.82
	7	90.05	77.56
	9	90.05	78.15
	11	89.35	78.34

In the second part of experimentation, classification accuracy is computed as a function of total parameters and processing time. This would give 3D plot where x and y axes represent total parameters and processing time and z-axis represents classification accuracy. For simplicity, a 3D plot is split into two 2D plots, where one plot shows classification accuracy versus total parameters and the other plot shows classification accuracy versus processing time for the corresponding values of total parameters. The level is taken as $Q = 1,2,4,8,16$ and dimension $h = 1,2,...,10$ for image dataset and $h = 1,2,...,12$ for speech dataset. Figure 8.5.1 and figure 8.5.2 show classification accuracy versus total parameters in logarithmic scale and classification accuracy versus processing time respectively, using all the techniques on image dataset.

For LCD technique, as presented in the figures 8.5.1 and 8.5.2, the first value of classification accuracy is 81.3% at total parameter $10^{2.636}$ (figure 8.5.1) which takes processing time of 2.94 units (figure 8.5.2). The next reported value of classification accuracy in figures 8.5.1 and 8.5.2 is only those which provide better classification accuracy than the present value, i.e. those values are plotted next in the figures which are giving improvement in classification accuracy compared to the previous value. This would help to describe that to achieve a certain range of classification accuracy what is the total parameter requirement and its corresponding processing time. Similar strategy is opted for VQPCA and PCA techniques. For VQ technique there are only four levels and all of them are given which are denoted by 2,4,8 and 16 in the figures 8.5.1 and 8.5.2. MDC and NN have only one value and kNN has got 5 values for $k = 3,5,7,9,11$ which is depicted in the same figures.

It can be observed from the figures (8.5.1 and 8.5.2) that MDC has minimal total parameter requirement and processing time but the classification accuracy is quite poor around 76.6%. The other techniques with same total parameter requirement but with different processing timings are PCA, VQ and LCD (at level 1). Though the processing time is very low for PCA (around 2.53 to 2.99 time units), the performance is quite poor giving classification accuracy in the range of 69.4% to 73.3% which is even lower than MDC. With the same total parameter requirement VQ gives much better performance than PCA in terms of accuracy but the processing time increases as the levels increase towards 16. The classification accuracy of VQPCA is quite poor at the beginning. As the total parameter requirement increases it gives reasonably well results but at the expense of high processing time. It is evident that LCD technique gives high classification accuracy at low total parameter requirement and processing time, for e.g. it gives 85.4% accuracy at $10^{3.033}$ total parameters using only 3.00 units processing time whereas the maximum accuracy obtained by VQ is 85.1% at $10^{3.539}$ total parameters using 23.41 units processing time and VQPCA gives 84.9% at $10^{3.840}$ using 32.81 units processing time. The maximum accuracy achieved by LCD technique (when $Q \leq 16$ and $h \leq 10$) is 90.0% at $10^{4.580}$ using 48.12 units processing time which is very close to NN technique (90.3%) and close to the maximum of kNN (for $k = 3$) technique (90.5%). However the processing time for NN and kNN techniques are 193.37 units and from 196.89 to 220.01 units (for $k = 3,5,7,9,11$) respectively, and the total parameter requirement for both the techniques is $10^{5.203}$, which is quite expensive as compared to LCD and other techniques. Figure 8.6.1 and figure 8.6.2 show classification accuracy vs. total parameters on logarithmic scale and classification accuracy vs. processing time respectively for all the techniques on speech dataset. The plotting scheme is similar to that applied for figures 8.5.1 and 8.5.2.

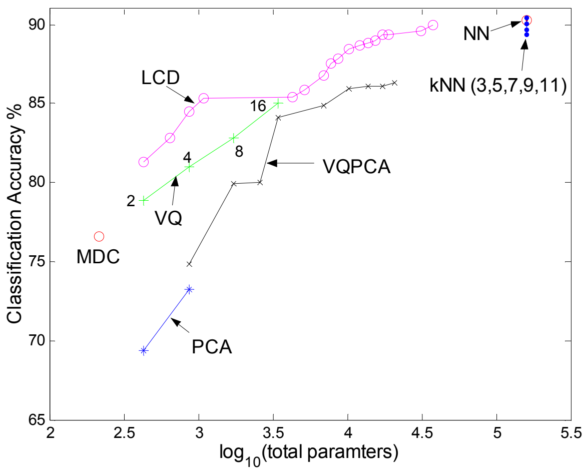


Figure 8.5.1: Classification accuracy vs. $\log_{10}$(total parameters) on image dataset.

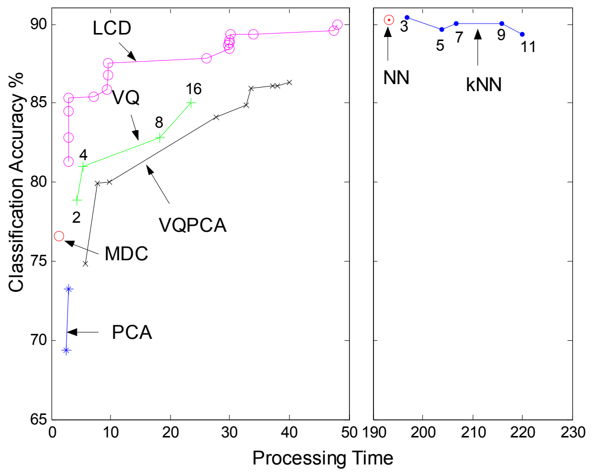


Figure 8.5.2: Classification accuracy vs. processing time on image dataset.

It is evident from figures 8.6.1 and 8.6.2 that LCD technique is performing better than all the other techniques including NN and kNN in terms of achieving higher classification accuracy at low total parameter requirement and low processing time. The classification accuracy of NN technique is even poorer than MDC, PCA and VQ techniques; this means that increasing total parameters does not always help in improving the classification accuracy. The maximum classification accuracy for LCD technique is 84.1% at $10^{3.670}$ using 8.74 units processing time, whereas the nearest technique in terms of accuracy is kNN which is giving 78.3% (for $k = 11$) at $10^{5.562}$ using 794.08 units processing time.

It can be concluded from the experiments on image dataset and speech dataset that LCD technique outperforms MDC, PCA, VQ, VQPCA, NN and kNN techniques in terms of getting reasonably accepted classification accuracy and at the same time maintaining minimal total parameter requirement and processing time. This would enable the user to classify a given object accurately and quickly with minimal implementation cost. The next section presents our concluding remarks.

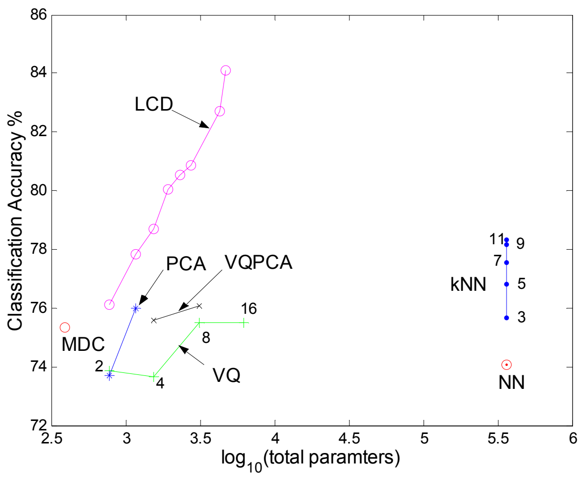


Figure 8.6.1: Classification accuracy vs. $\log_{10}$(total parameters) on speech dataset.

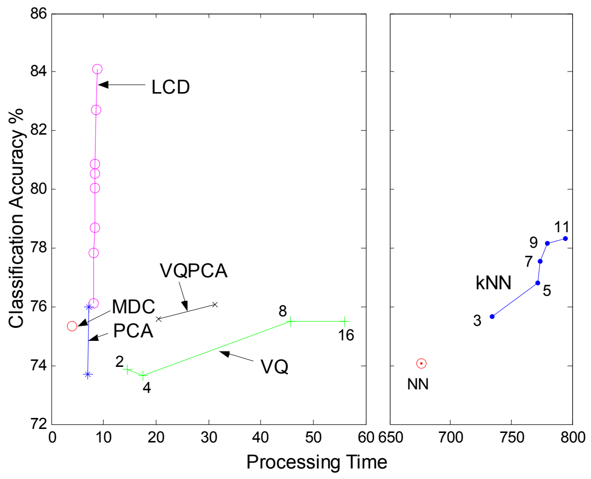


Figure 8.6.2: Classification accuracy vs. processing time on speech dataset.

8.7 Summary

A survey on basic classifiers namely MDC, VQ, PCA, NN and kNN was given. Their classification procedures were illustrated. Then we looked at VQPCA technique which is normally used for representation purposes. We showed how to use VQPCA for classification purposes. However, we found that VQPCA did not give very encouraging performance as a classifier but this gave us initiative to develop combined classifiers.

Next we presented LCD technique which is the combination of VQ and VQPCA techniques. By combining the classifiers we found that the performance improved significantly which was not possible by using either VQ or VQPCA individually. The performance of LCD technique is found to be better than all the other presented techniques. Thus it can classify a given object more accurately at very low implementation cost and processing time, which was demonstrated using speech and image datasets.

It was found that when the weighting coefficient α was close to 0.5 the LCD technique gave close to optimum performance, i.e. when VQ and VQPCA techniques contribute equally in the decision making of a test pattern then the performance is close to optimum.

Chapter 9

Independent Component Analysis

9.1 Abstract

This chapter describes the basic theories and concepts of independent component analysis in solving blind source separation problems.

9.2 Introduction

ICA has emerged as a tool to solve blind source separation (BSS) problem. In the BSS problem it is assumed that an observation or mixture x is modelled from statistically independent and nongaussian components **s**. The mixing matrix A is square and invertible. The elements of **s** are linearly mixed with the mixing matrix A to give the observation x. Both the source signals and the mixing matrix are unknown to the observer. The BSS problem is to identify the source signals only from the observation x. The source signals could be obtained up to their permutation, sign and amplitude only, that is the order and variances of independent components cannot be determined. These indeterminacies are, however, insignificant in most of the applications. Two methods for estimating independent components namely kurtosis and negentropy are discussed. Their fixed point implementations are also illustrated.

9.3 ICA for Blind Source Separation (BSS) Problem

In blind source separation (BSS) problem a set of observed features or patterns are given but their underlying source information is hidden. The independent component analysis (ICA) tool is used to solve BSS problem by finding these hidden source information. It is assumed that hidden source signals are the weighted sum of the observed signals. It is also assumed that the source signals are statistically independent and non-gaussian. For dependent and gaussian source signals it is not possible to solve BSS problem using the ICA technique.

To elaborate the ICA technique, let us assume two source signals (e.g. speech from two amplifiers placed at some distance) are s_1 and s_2; observed signals (e.g. by two microphones situated at some distance) are x_1 and x_2; parameters that depend upon the distance of observers (here microphones) placed are a_{11}, a_{12}, a_{21} and a_{22}. This will give the following problem:

$$x_1 = a_{11}s_1 + a_{12}s_2 \qquad 9.1$$

$$x_2 = a_{21}s_1 + a_{22}s_2 \qquad 9.2$$

The same problem can be written in matrix system as

$$\mathbf{x} = \mathbf{A}\mathbf{s} \qquad 9.3$$

If $\mathbf{A}$ is a square matrix (i.e. number of observation and number of source signals are equal) and non-singular then the source signals can be obtained using the following equation:

$$\mathbf{s} = \mathbf{A}^{-1}\mathbf{x} \qquad 9.4$$

One can easily find source $\mathbf{s}$ if the mixing matrix $\mathbf{A}$ is known, the problem here is, it is unknown. The only known thing is the observation $\mathbf{x}$ and the problem is to find source signal $\mathbf{s}$ from equation 9.3. This is BSS problem and can be solved using the ICA technique.

In order to find the source signal $\mathbf{s}$, some assumptions are made (as previously discussed)

Assumption 1: Source signals are statistically independent.
Assumption 2: At least all but one of the components of $\mathbf{s}$ are non-gaussian.

It can be seen (Hyvärinen et al. 2001) that non-gaussianity is undesirable for independent component analysis. In ICA the components and the mixing matrix are estimated from the mixtures **x**. This is done by estimating a matrix **W** that gives the estimates **y** as

$$\mathbf{y} = \mathbf{W}^{\mathrm{T}}\mathbf{x} \qquad 9.5$$

The components of **y** are maximally independent i.e. maximally non-gaussian. The maximization of non-gaussianity of the components enables estimation of independent components. In ICA mixtures **x** are firstly centered and whitened[1] before any further processing, i.e.

$$\mathbf{y} = \mathbf{W}^{\mathrm{T}}\mathbf{z} \qquad 9.6$$

is taken as a preprocessing step, where **z** is centered and whitened form of **x**.

There are several methods for maximizing non-gaussianity. Two methods readily used are kurtosis and negentropy. These methods are discussed in the next section.

9.4 Measure of Nongaussianity

Two quantitative measure of nongaussianity are kurtosis and negentropy. These are described as follows:

9.4.1 Kurtosis

Kurtosis or univariate kurtosis is a fourth order cumulant of a random variable

[1] The centering process is to subtract the data with its mean i.e. $\mathbf{x} \leftarrow \mathbf{x} - E[\mathbf{x}]$ and whitening process can be done by using the eigenvalue decomposition of the covariance matrix which will give orthogonal matrix **E** and diagonal matrix **D**. Then these matrices are used to find the whitening matrix $\mathbf{V} = \mathbf{E}\mathbf{D}^{-1/2}\mathbf{E}^{\mathrm{T}}$ which will give whitened data $\mathbf{z} = \mathbf{V}\mathbf{x}$.

(Hyvärinen et al. 2001). For zero-mean random variable, kurtosis is defined as:

$$\mathrm{kurt}(y) = \mathrm{E}[y^4] - 3(\mathrm{E}[y^2])^2 \qquad 9.6$$

where y is any component of $\mathbf{y}$ which is equal to the linear combination $\mathbf{w}^{\mathrm{T}}\mathbf{z}$. The vector **w** is the corresponding column vector of **W**. Kurtosis value can be any real number. Random variables with $\mathrm{kurt}(y) > 0$ are considered supergaussian while with $\mathrm{kurt}(y) < 0$ are considered subgaussian. For gaussian random variables and a very few nongaussian variables $\mathrm{kurt}(y) = 0$. Thus nongaussianity can be measured by the absolute value of kurtosis. If the variance of random variables are kept constant (i.e. $\mathrm{E}[\mathbf{y}^2] = 1$) then kurtosis can be computed by the fourth moment of random variables. The main advantage of using kurtosis is its computational simplicity. One of the drawbacks of kurtosis inherited by the fourth order moments is its susceptibility (sensitive) to outliers.

To find the first orthonormal vector **w** of **W** (i.e. $\|\mathbf{w}\|^2 = \mathbf{w}^{\mathrm{T}}\mathbf{w} = 1$), we compute the gradient of kurtosis. The derivative of kurtosis with respect to **w** is

$$\frac{\partial\,|\,\mathrm{kurt}(\mathbf{w}^{\mathrm{T}}\mathbf{z})\,|}{\partial\mathbf{w}} = 4\mathrm{sign}(\mathrm{kurt}(\mathbf{w}^{\mathrm{T}}\mathbf{z}))[E[\mathbf{z}(\mathbf{w}^{\mathrm{T}}\mathbf{z})^3] - 3\mathbf{w}\,\|\mathbf{w}\|^2] \qquad 9.7$$

The gradient of kurtosis can be used in gradient descent algorithm. This yields the following gradient algorithm

$$\Delta\mathbf{w} \propto \mathrm{sign}(\mathrm{kurt}(\mathbf{w}^{\mathrm{T}}\mathbf{z}))[E[\mathbf{z}(\mathbf{w}^{\mathrm{T}}\mathbf{z})^3] - 3\mathbf{w}] \qquad 9.8$$

$$\mathbf{w} \leftarrow \mathbf{w}/\,\|\mathbf{w}\| \qquad 9.9$$

The latter term in equation 9.8 does not change the direction of **w**, it will be changing only the norm of **w** and hence can be excluded from the equation which would further simplify the algorithm. The advantage of using gradient algorithm is that inputs **z** can be used at once in the algorithm making the adaptation fast. On the other hand, the convergence is very slow and strongly depends on the initial settings. A bad choice of learning rate parameter (initial setting) could affect the convergence speed and can also destroy the convergence. Alternatively, one can apply fixed-point algorithm (Hyvärinen and Oja, 1997) for convergence. Fixed-point algorithm is reliable, robust and very fast. It does not depend on any learning rate parameter, thus is free from initial settings. In this case, the algorithm will converge when the 'new' and 'old' values of $\mathbf{w}$ point in the same direction (since sign of $\mathbf{w}$ and $-\mathbf{w}$ define the same direction) i.e. $|(\mathbf{w}^{+})^{\mathrm{T}}\mathbf{w}| \approx 1$ where $\mathbf{w}^{+}$ is the new value of $\mathbf{w}$. The fixed-point algorithm using kurtosis (so called FastICA using kurtosis) can be given as follows:

$$\mathbf{w} \leftarrow E[\mathbf{z}(\mathbf{w}^{\mathrm{T}}\mathbf{z})^{3}] - 3\mathbf{w} \qquad 9.10$$

$$\mathbf{w} \leftarrow \mathbf{w} / \|\mathbf{w}\| \qquad 9.11$$

9.4.2 Negentropy

This measure of nongaussianity is robust but computationally complicated. It is defined as:

$$J(\mathbf{y}) = H(\mathbf{y}_{gauss}) - H(\mathbf{y}) \qquad 9.12$$

where $\mathbf{y}_{gauss}$ is a gaussian random variable and $H(\mathbf{y})$ is the differential entropy with density $p_{\mathrm{y}}(\boldsymbol{\eta})$ defined as:

$$H(\mathbf{y}) = -\int p_{\mathrm{y}}(\boldsymbol{\eta}) \log p_{\mathrm{y}}(\boldsymbol{\eta}) d\,\boldsymbol{\eta} \qquad 9.13$$

For gaussian random variables $J(\mathbf{y}) = 0$ and for different distributions it is nonnegative. In practice, approximation of equation 9.12 is used for negentropy computation since it is difficult to compute differential entropy $H(\mathbf{y})$ due to the unknown density $p_{\mathrm{y}}(\boldsymbol{\eta})$. The approximation for $J(y)$ (1-dimensional negentropy case) can be given as

$$J(y) \propto [E[G(y)] - E[G(v)]]^{2} \qquad 9.14$$

where v is a standardized gaussian variable and G is a nonquadratic function usually defined as:

$$G_1(y) = \frac{1}{a_1}\mathbf{log\,cosh}\,a_1 y \qquad 9.15$$

$$G_2(y) = -\mathbf{exp}(-y^2/2) \qquad 9.16$$

The constant a_1 in equation 9.15 is often taken to be 1. The gradient algorithm can now be defined as

$$\Delta\mathbf{w} = \gamma\, E[\mathbf{z}\,g(\mathbf{w}^{\mathrm{T}}\mathbf{z})] \qquad 9.17$$

$$\mathbf{w} \leftarrow \mathbf{w}/\,\|\,\mathbf{w}\,\| \qquad 9.18$$

where $\gamma = E[G(\mathbf{w}^{\mathrm{T}}\mathbf{z})] - E[G(v)]$ and g is the derivative of the selected function G. Here also the fixed-point algorithm can be applied. The fixed-point algorithm using negentropy is given as follows:

$$\mathbf{w} \leftarrow E[\mathbf{z}\,g(\mathbf{w}^{\mathrm{T}}\mathbf{z})] - E[g'(\mathbf{w}^{\mathrm{T}}\mathbf{z})]\,\mathbf{w} \qquad 9.19$$

$$\mathbf{w} \leftarrow \mathbf{w}/\,\|\,\mathbf{w}\,\| \qquad 9.20$$

where g' is the derivative of g.

9.5 Estimation of Multiple Independent Components

Equations 9.19 and 9.20 are used to find one unit independent component (IC) for negentropy function. Similarly, equations 9.10 and 9.11 are used to find one unit IC for kurtosis function. It is also possible to find more than one ICs. Since ICs are orthogonal to each other, we can take this as a basis to find other ICs. The deflationary orthogonalization process is used to find p ICs (where $p > 1$) one by one. It uses Gram-Schmidt orthonormalization method for finding ICs iteratively. We first estimate p ICs $(\mathbf{w}_1,\ldots\mathbf{w}_p)$ and then orthonormalize the obtained ICs prior to running the algorithm for $(p+1)^{\text{th}}$ independent component $(\text{i.e.}\,\mathbf{w}_{p+1})$. The procedure is illustrated in table 9.1.

Multiple ICs can also be estimated by using symmetric orthogonalization procedure. Here all the column vectors of **W** are estimated in parallel i.e. at once. Therefore the ICs are not estimated one by one, this alleviates the error that may propagate in serial nature of deflationary orthogonalization process. The symmetric orthogonalization procedure for finding independent components is depicted in table 9.2.

9.6 An Illustration for ICA on BSS problem

This section describes the blind source separation using the ICA technique. For an illustration two source signals are taken as depicted in 9.1. The source signals are artificially mixed to get a mixture of signals. The mixed signals are shown in figure 9.2. Here we have applied kurtosis function for finding the estimates of the original source signals. The estimates of the original signals are shown in figure 9.3. The absolute kurtosis versus iteration plot is also shown for the first vector $\mathbf{w}_1$ (figure 9.4a) and for the second vector $\mathbf{w}_2$ (figure 9.4b), where $\mathbf{W}=[\mathbf{w}_1,\mathbf{w}_2]$. It can be observed from figure 9.3 that the estimates are very similar to the original source signals. The convergence of the algorithm using kurtosis is evident from figure 9.4.

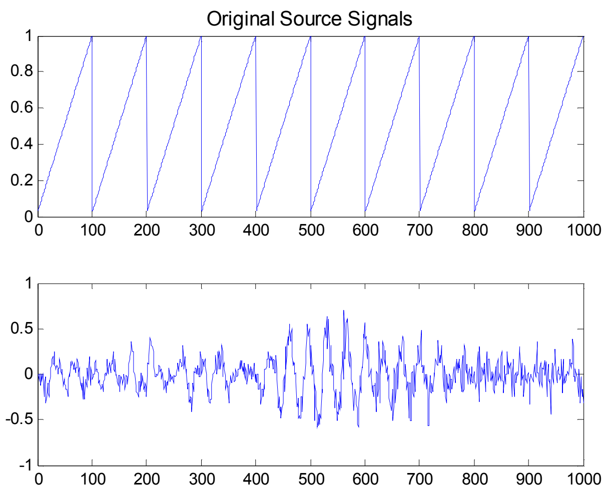


Figure 9.1: Two original source signals for an illustration

TABLE 9.1: Deflationary orthogonalization process for finding independent components

1. Center data **x**.
2. Whiten data **x** to give **z**.
3. Select *m*, the number of independent components to estimate. Set counter $p \leftarrow 1$.
4. Select an initial value of identity norm for $\mathbf{w}_p$, e.g. randomly.
5. Compute $\mathbf{w}_p$ by using either equation 9.10 for kurtosis or equation 9.19 for negentropy process.
6. Do Gram-Schmidt orthogonalization for $\mathbf{w}_p$

$$\mathbf{w}_p \leftarrow \mathbf{w}_p - \sum_{j=1}^{p-1} (\mathbf{w}_p^{\mathrm{T}} \mathbf{w}_j)\mathbf{w}_j \qquad 9.21$$

7. Normalize $\mathbf{w}_p$ using 9.20
8. If $\mathbf{w}_p$ has not converged, go back to step 5.
9. Set $p \leftarrow p+1$ and go to step 4 until $p = m$.

TABLE 9.2: Symmetric orthogonalization process for finding independent components

1. Center data **x**.
2. Whiten data **x** to give **z**.
3. Select *m*, the number of independent components to estimate.
4. Select an initial value of identity norm for $\mathbf{w}_j$ where $j = 1 \ldots m$ (e.g. randomly).
5. Compute $\mathbf{w}_j$ by using either equation 9.10 for kurtosis or equation 9.19 for negentropy process in parallel.
6. Do symmetric orthogonalization of matrix $\mathbf{W} = [\mathbf{w}_1, \ldots \mathbf{w}_m]^{\mathrm{T}}$

$$\mathbf{W} \leftarrow (\mathbf{W}\mathbf{W}^{\mathrm{T}})^{-1/2}\,\mathbf{W} \qquad 9.22$$[2]

7. If not converge, go back to step 5.

[2] Some simpler alternatives of equation 9.22 can be found in (Hyvärinen et al. 2001).

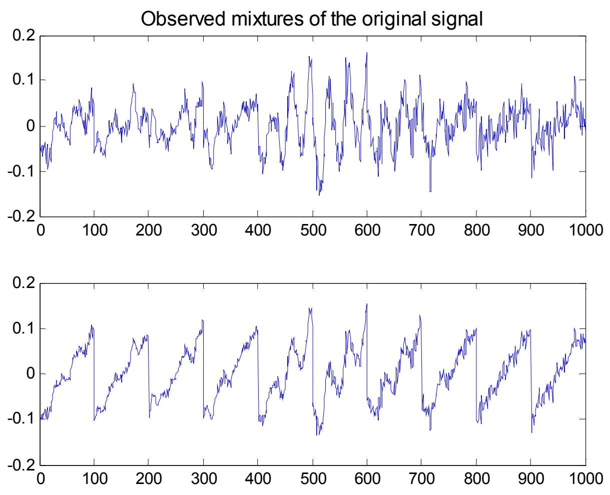


Figure 9.2: Observed mixed signals

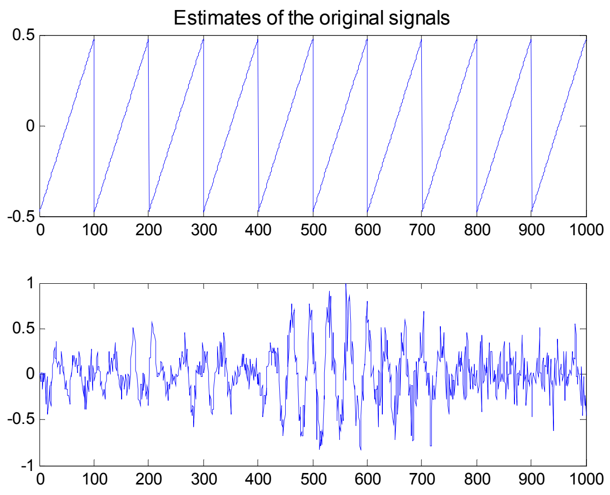


Figure 9.3: The estimates of the original signals

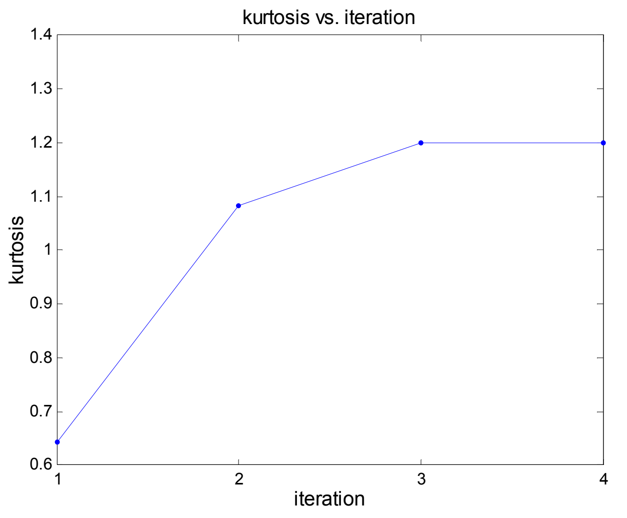


Figure 9.4a: Absolute kurtosis values as a function of iteration for $\mathbf{w}_1$.

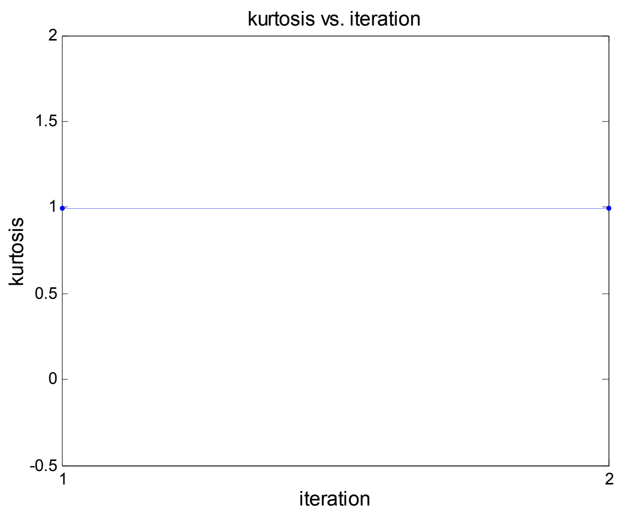


Figure 9.4b: Absolute kurtosis values as a function of iteration for $\mathbf{w}_2$.

9.7 Summary

The basic theories and concepts of independent component analysis (ICA) have been described. The fixed-point implementation of ICA (FastICA) is also described. The experimentation using two mixed signals are shown for kurtosis function. It was seen that the ICA technique is able to separate efficiently the mixed observed signals provided the source signals are statistically independent and nongaussian. It was also observed that the absolute kurtosis values as a function of iteration monotonically increases (converges) to some positive value.

Chapter 10

Subspace Independent Component Analysis Using Vector Kurtosis

10.1 Abstract

This discussion presents a new perspective of subspace independent component analysis (ICA). The notion of a function of cumulants (kurtosis) is generalized to vector kurtosis. This vector kurtosis is utilized in the subspace ICA algorithm to estimate subspace independent components. One of the main advantages of the presented approach is its computational simplicity. The experiments have shown promising results in estimating subspace independent components.

10.2 Introduction

Independent component analysis is a widely accepted tool in solving blind source separation (BSS) problems. In BSS problem a set of observations is given but the underlying source information is hidden. The mixing weights of this underlying source information are also not known to the observer. The BSS problem is thus to identify the source signals and/or the mixing weights. The assumptions in the basic ICA model include the source signals being mutually independent and having nongaussian distributions. In the BSS problem an $\mathrm{M}\times 1$ vector of observation **x** is modelled from statistically independent and nongaussian components **s** of size $\mathrm{M}\times 1$:

$$\mathbf{x} = \mathbf{A}\mathbf{s} \qquad 10.1$$

where **A** is a square and invertible mixing matrix of size $\mathrm{M}\times\mathrm{M}$. The elements of $s = [\mathrm{s}_1,\ldots,\mathrm{s}_\mathrm{M}]^\mathrm{T}$ are linearly mixed with the mixing matrix **A** to give the observation **x**. The source signals could be obtained up to their permutation, sign and amplitude only, that is the order and variances of independent components cannot be determined (as discussed in chapter 11). These indeterminacies are, however, insignificant in most of the applications.

Some techniques (Hyvärinen and Hoyer, 2000; Cardoso, 1998) have evolved in recent years that relax the assumptions of basic ICA model and generalize the problem. These techniques are a generalization of basic ICA model and are known as multidimensional ICA (MICA) (Cardoso, 1998) and subspace ICA (Hyvärinen and Hoyer, 2000) model. In MICA or subspace ICA it is not assumed that all the source signals are independent, instead it is assumed that some components that usually come in *n*-tuples or the elements of subspaces are mutually non-independent. However, the non-independencies among different *n*-tuples or subspaces are not allowed.

In this chapter we present a new perspective of subspace ICA model. Unlike MICA (CArdoso, 1998) or subspace ICA (Hyvärinen and Hoyer, 2000) we have not applied an additive model. However, the multiplicative model as of basic ICA has been utilized except that it is partitioned into sub-matrices and sub-vectors. Then we generalize the notion of kurtosis (Hyvärinen et al., 2001) to vector kurtosis for our model and show the relationship of the optimized vector kurtosis to the subspace independent components. This approach would solve the BSS problem even when not all the components are independent i.e. it accounts for a generalized problem. One of the advantages of our subspace ICA algorithm is its computational simplicity due to the use of vector (generalized) kurtosis function.

10.3 Evaluation of Independent Components by Maximizing a Quantitative Measure of Nongaussianity

Independent components can be estimated by the maximization of nongaussianity. Two quantitative measures of nongaussianity readily used in ICA estimation are kurtosis and negentropy (Hyvärinen et al., 2001).

10.3.1 Kurtosis

Kurtosis or univariate kurtosis is a fourth order cumulant of a random variable. For zero-mean random variable, kurtosis is defined as:

$$\mathrm{kurt}(y) = \mathrm{E}[y^4] - 3(\mathrm{E}[y^2])^2 \qquad 10.2$$

Kurtosis value can be any real number. Random variables with $\mathrm{kurt}(y) > 0$ are considered supergaussian while with $\mathrm{kurt}(y) < 0$ are considered subgaussian. For gaussian random variables and a very few nongaussian variables $\mathrm{kurt(y)} = 0$. Thus nongaussianity can be measured by the absolute value of kurtosis. If the variance of random variables are kept constant (i.e. $\mathrm{E[y^2]} = 1$) then kurtosis can be computed by the fourth moment of random variables. The main advantage of using kurtosis is its computational simplicity. One of the drawbacks of kurtosis inherited by the fourth order moments is its susceptibility (sensitivity) to outliers (Hyvärinen et al., 2001).

10.4 Subspace ICA and MICA

Cardoso (1998) introduced the notion of MICA by generalizing basic ICA model. MICA is an additive model which is derived from the multiplicative model. Its components $\mathbf{s}_i$ are vector-valued, instead of scalar-valued as of equation 10.1 and not all the elements of $\mathbf{s}_i$ are assumed to be independent. MICA was estimated by maximum likelihood (ML) estimation and illustrated on foetal ECG dataset (De Moor et al., 1997). The author argued that the dataset was well modelled by MICA decomposition into one bi-dimensional component (mother) and one mono-dimensional component (foetal).

Hyvärinen and Hoyer (2000) combined the technique of MICA and the principle of invariant-feature subspaces[1] (Kohonen, 1996) to explain the emergence of phase- and shift-invariant features. The authors call the *n* elements of $\mathbf{s}_i$ as the subspace spanned by a set of *n* basis vectors an independent subspace and referred the algorithm as independent subspace analysis (ISA) or subspace ICA and estimated subspace independent components by ML estimation. Thus different subspaces are mutually independent but the entries of each subspace are not independent. The probability density of each subspace is considered to be spherically symmetric, i.e. it depends only on the norm of the projection.

10.5 Subspace ICA Model: A New Perspective

We take the multiplicative model and partition the entries of matrix and vectors:

$$\mathbf{x} = \mathbf{A}\mathbf{s} \qquad \text{or} \qquad \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_M \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{11} & \cdots & \mathbf{A}_{1M} \\ \vdots & \ddots & \vdots \\ \mathbf{A}_{M1} & \cdots & \mathbf{A}_{MM} \end{bmatrix} \begin{bmatrix} \mathbf{s}_1 \\ \vdots \\ \mathbf{s}_M \end{bmatrix} \qquad 10.3$$

where $\mathbf{x}_j$ and $\mathbf{s}_j$ of $\mathbf{x}$ and $\mathbf{s}$ respectively are vectors of dimension d and can be defined as $\mathbf{x}_j = [x_1^j, x_2^j, ..., x_d^j]^T$ and $\mathbf{s}_j = [s_1^j, s_2^j, ..., s_d^j]^T$ for $j = 1, ..., M$. Partitioned matrix $\mathbf{A}$ (equation 10.3) is of size $Md \times Md$ since its entries $\mathbf{A}_{ij}$ are matrices of size $d \times d$. We made the following assumptions for our model:

<u>*Assumption1*</u>: Components $\mathbf{s}_j$ are vector-valued, nongaussian, mutually independent and of identity covariance.

<u>*Assumption2*</u>: Entries of $\mathbf{s}_j$ are not independent and all are of equal dimension d.

<u>*Assumption3*</u>: Sample data is centered and whitened.

[1] The principle of invariant-feature subspace is that invariant-feature can be considered as a linear subspace in a feature space and its value can be computed by taking the norm of the projection on that subspace.

To estimate subspace independent components we take a d-dimensional vector $\mathbf{y}$ which is defined as:

$$\mathbf{y} = \mathbf{B}^T\mathbf{x} = \sum_{j=1}^{M} \mathbf{B}_j^T \mathbf{x}_j \qquad 10.4$$

where size of $\mathbf{B}$ and $\mathbf{B}_j$ are $Md \times d$ and $d \times d$ respectively. Given equation 10.4, now the problem is to identify and/or estimate subspace independent components from the observation $\mathbf{x}$ only. The problem is solved in section 10.5.2.

10.5.1 Extension of Univariate Kurtosis to Vector Kurtosis

Univariate kurtosis or simply kurtosis (section 10.3.1) is utilized when the variable y is a scalar quantity or one-dimensional vector. It does not accommodate for multidimensional features. To solve the multidimensional problem we first need to extend the basic kurtosis function. The natural generalization of basic kurtosis function for any vector $\mathbf{y}$ can be given as:

$$\text{kurt}(\mathbf{y}) = E[(\mathbf{y}^T \mathbf{y})^2] - 3(E[\mathbf{y}^T \mathbf{y}])^2 \qquad 10.5$$

which is a multidimensional equivalent of equation 10.2. There is no covariance term in equation 10.5. This is due to one of our assumptions that the sample data is whitened ($E[\mathbf{y}\mathbf{y}^T] = I_{d\times d}$). We refer to this generalized kurtosis as vector kurtosis. As of kurtosis function, vector kurtosis is computationally simple but sensitive to the outliers.

10.5.2 Relation of Optimized Vector Kurtosis to the Subspace Independent Components

In this section we discuss how the subspace independent components are related to the optimization of vector kurtosis. Let us consider the subspace independent component (equation 10.4) again. From equations 10.3 and 10.4, the component can be written as:

$$\mathbf{y} = \mathbf{B}^{\mathrm{T}}\mathbf{A}\mathbf{s} = \mathbf{Q}^{\mathrm{T}}\mathbf{s} = \sum_{i=1}^{\mathrm{M}} \mathbf{Q}_i^{\mathrm{T}}\, \mathbf{s}_i \qquad 10.6$$

where size of $\mathbf{Q}$ and $\mathbf{Q}_i$ are $\mathrm{Md}\times\mathrm{d}$ and $\mathrm{d}\times\mathrm{d}$ respectively. Equation 10.6 is a linear combination of vectors $\mathbf{s}_i$. To show the relationship, consider two observations ($\mathrm{M}=2$) s_1 and s_2 each of dimension d. This would simplify equation 10.6 as:

$$\mathbf{y} = \mathbf{Q}_1^{\mathrm{T}}\, \mathbf{s}_1 + \mathbf{Q}_2^{\mathrm{T}}\, \mathbf{s}_2 \qquad 10.7$$

Using the additive property of kurtosis (which can be shown for vector kurtosis as well) we can say:

$$\begin{aligned} f(\mathbf{Q}_1,\mathbf{Q}_2) &= \mathrm{kurt}(\mathbf{y}) = \mathrm{kurt}(\mathbf{Q}^{\mathrm{T}}\mathbf{s}) \\ &= \mathrm{kurt}(\mathbf{Q}_1^{\mathrm{T}}\mathbf{s}_1) + \mathrm{kurt}(\mathbf{Q}_2^{\mathrm{T}}\mathbf{s}_2) \end{aligned} \qquad 10.8$$

where $\mathrm{kurt}(\mathbf{Q}_j^{\mathrm{T}}\mathbf{s}_j) = \mathrm{E}[(\mathbf{s}_j^{\mathrm{T}}\mathbf{Q}_j\mathbf{Q}_j^{\mathrm{T}}\mathbf{s}_j)^2] - 3(\mathrm{E}[(\mathbf{s}_j^{\mathrm{T}}\mathbf{Q}_j\mathbf{Q}_j^{\mathrm{T}}\mathbf{s}_j)])^2$. Now we put a constraint g on $\mathbf{Q}$ (since $\mathrm{E}[\mathbf{y}\mathbf{y}^{\mathrm{T}}] = \mathrm{I}_{\mathrm{d}\times\mathrm{d}}$):

$$\begin{aligned} \mathrm{E}[\mathbf{y}^{\mathrm{T}}\mathbf{y}] &= \mathrm{E}[(\mathbf{Q}_1^{\mathrm{T}}\mathbf{s}_1 + \mathbf{Q}_2^{\mathrm{T}}\mathbf{s}_2)^{\mathrm{T}}(\mathbf{Q}_1^{\mathrm{T}}\mathbf{s}_1 + \mathbf{Q}_2^{\mathrm{T}}\mathbf{s}_2)] \\ &= \mathrm{E}[\mathbf{s}_1^{\mathrm{T}}\mathbf{Q}_1\mathbf{Q}_1^{\mathrm{T}}\mathbf{s}_1] + \mathrm{E}[\mathbf{s}_2^{\mathrm{T}}\mathbf{Q}_2\mathbf{Q}_2^{\mathrm{T}}\mathbf{s}_2]\{\because \mathbf{s}_1 \text{ and } \mathbf{s}_2 \text{ are independent and } \mathrm{E}[\mathbf{s}_1] = \mathrm{E}[\mathbf{s}_2] = 0_{\mathrm{d}\times 1}\} \end{aligned}$$

again $\mathrm{E}[\mathbf{y}^{\mathrm{T}}\mathbf{y}] = \mathrm{E}[\mathit{trace}(\mathbf{y}^{\mathrm{T}}\mathbf{y})] = \mathrm{E}[\mathit{trace}(\mathbf{y}\mathbf{y}^{\mathrm{T}})] = \mathit{trace}(\mathrm{E}[\mathbf{y}\mathbf{y}^{\mathrm{T}}]) = \mathit{trace}(\mathrm{I}_{\mathrm{d}\times\mathrm{d}}) = \mathrm{d}$ and using equation 10.7

$$\mathrm{E}[\mathbf{y}\mathbf{y}^{\mathrm{T}}] = \mathbf{Q}_1^{\mathrm{T}}\mathbf{Q}_1 + \mathbf{Q}_2^{\mathrm{T}}\mathbf{Q}_2 = \mathrm{I}_{\mathrm{d}\times\mathrm{d}} \qquad 10.9$$

therefore, constraint g can be written as:

$$g(\mathbf{Q}_1,\mathbf{Q}_2) = \mathrm{E}[\mathbf{y}^{\mathrm{T}}\mathbf{y}] - \mathrm{d} = \mathrm{E}[\mathbf{s}_1^{\mathrm{T}}\mathbf{Q}_1\mathbf{Q}_1^{\mathrm{T}}\mathbf{s}_1] + \mathrm{E}[\mathbf{s}_2^{\mathrm{T}}\mathbf{Q}_2\mathbf{Q}_2^{\mathrm{T}}\mathbf{s}_2] - \mathrm{d} = 0 \qquad 10.10$$

From equation 10.9 it can be stated that column vectors of rectangular matrix $\mathbf{Q}$ are orthonormalized. The optimization problem can now be solved by finding $\mathbf{Q}_1$ and $\mathbf{Q}_2$ that occur at constrained relative-extremum of $f(\mathbf{Q}_1,\mathbf{Q}_2)$ (equation 10.8) under the constrained curve $g(\mathbf{Q}_1,\mathbf{Q}_2)$ (equation 10.10) using the method of Lagrange multipliers:

$$\nabla_{(\mathbf{Q}_1,\mathbf{Q}_2)} f(\mathbf{Q}_1,\mathbf{Q}_2) = \lambda \nabla_{(\mathbf{Q}_1,\mathbf{Q}_2)} g(\mathbf{Q}_1,\mathbf{Q}_2) \qquad \text{where } \lambda \neq 0 \qquad 10.11$$

Solving for the derivatives of functions f and g (partial proof of equations 10.12 and 10.13 can be viewed in appendix 10.1), we get

$$\begin{aligned}\nabla_{(\mathbf{Q}_1,\mathbf{Q}_2)} f(\mathbf{Q}_1,\mathbf{Q}_2) = {} & \{4\mathrm{E}[(\mathbf{s}_1^{\mathrm{T}}\mathbf{Q}_1\mathbf{Q}_1^{\mathrm{T}}\mathbf{s}_1)(\mathbf{s}_1\mathbf{s}_1^{\mathrm{T}}\mathbf{Q}_1)] - 12\mathrm{E}[\mathbf{s}_1^{\mathrm{T}}\mathbf{Q}_1\mathbf{Q}_1^{\mathrm{T}}\mathbf{s}_1]\mathrm{E}[\mathbf{s}_1\mathbf{s}_1^{\mathrm{T}}\mathbf{Q}_1]\}\,\hat{i}_{\mathrm{Q}_1} + \\ & \{4\mathrm{E}[(\mathbf{s}_2^{\mathrm{T}}\mathbf{Q}_2\mathbf{Q}_2^{\mathrm{T}}\mathbf{s}_2)(\mathbf{s}_2\mathbf{s}_2^{\mathrm{T}}\mathbf{Q}_2)] - 12\mathrm{E}[\mathbf{s}_2^{\mathrm{T}}\mathbf{Q}_2\mathbf{Q}_2^{\mathrm{T}}\mathbf{s}_2]\mathrm{E}[\mathbf{s}_2\mathbf{s}_2^{\mathrm{T}}\mathbf{Q}_2]\}\,\hat{i}_{\mathrm{Q}_2}\end{aligned} \qquad 10.12$$

and $\nabla_{(\mathbf{Q}_1,\mathbf{Q}_2)} g(\mathbf{Q}_1,\mathbf{Q}_2) = 2\mathrm{E}[\mathbf{s}_1\mathbf{s}_1^{\mathrm{T}}\mathbf{Q}_1]\,\hat{i}_{\mathrm{Q}_1} + 2\mathrm{E}[\mathbf{s}_2\mathbf{s}_2^{\mathrm{T}}\mathbf{Q}_2]\,\hat{i}_{\mathrm{Q}_2}$ 10.13

substituting equations 10.12 and 10.13 in equation 10.11 and comparing $\hat{i}_{\mathrm{Q}_1}$ terms, we get:

$$4\mathrm{E}[(\mathbf{s}_1^{\mathrm{T}}\mathbf{Q}_1\mathbf{Q}_1^{\mathrm{T}}\mathbf{s}_1)(\mathbf{s}_1\mathbf{s}_1^{\mathrm{T}}\mathbf{Q}_1)] - 12\mathrm{E}[\mathbf{s}_1^{\mathrm{T}}\mathbf{Q}_1\mathbf{Q}_1^{\mathrm{T}}\mathbf{s}_1]\mathrm{E}[\mathbf{s}_1\mathbf{s}_1^{\mathrm{T}}\mathbf{Q}_1] = \lambda 2\mathrm{E}[\mathbf{s}_1\mathbf{s}_1^{\mathrm{T}}\mathbf{Q}_1] \qquad 10.14$$

It is evident from equation 10.14 that $\mathbf{Q}_1 = 0_{\mathrm{d\times d}}$ is one of the solutions. The corresponding value of $\mathbf{Q}_2$ for this value of $\mathbf{Q}_1$ can be obtained by substituting $\mathbf{Q}_1 = 0_{\mathrm{d\times d}}$ in constraint curve (equation 10.10), which yields:

$$g(0_{\mathrm{d\times d}},\mathbf{Q}_2) = \mathrm{E}[\mathbf{s}_2^{\mathrm{T}}\mathbf{Q}_2\mathbf{Q}_2^{\mathrm{T}}\mathbf{s}_2] - \mathrm{d} = 0$$

or $\mathrm{E}[\mathbf{s}_2^{\mathrm{T}}\mathbf{Q}_2\mathbf{Q}_2^{\mathrm{T}}\mathbf{s}_2] = \mathrm{d}$; or $\mathit{trace}(\mathbf{Q}_2^{\mathrm{T}}\mathbf{Q}_2) = \mathrm{d}$ 10.15

Equations 10.9 and 10.15 imply that $\mathbf{Q}_2^{\mathrm{T}}\mathbf{Q}_2 = \mathrm{I}_{\mathrm{d\times d}}$. These values suggest that the norm of y is equal to the norm of one of the subspace independent components $\|\mathbf{y}\|^2 = \mathbf{y}^T\mathbf{y} = (\mathbf{Q}_i^T\mathbf{s}_i)^T(\mathbf{Q}_i^T\mathbf{s}_i) = \mathbf{s}_i^T\mathbf{s}_i = \|\mathbf{s}_i\|^2$ or $\|\mathbf{y}\| = \|\mathbf{s}_i\|$. Therefore, for any whitened data **z** (which can be achieved for example by eigenvalue decomposition procedure of covariance of sample data **x**), we search for $\mathbf{W}^T\mathbf{z}$ (where **W** is a rectangular matrix of the same size as $\mathbf{Q}$) that maximizes vector kurtosis. We see that $Q = (\mathbf{VA})^T\,\mathbf{W}$ and $\mathbf{Q}^T\mathbf{Q} = (\mathbf{W}^T\mathbf{VA})(\mathbf{A}^T\mathbf{V}^T\mathbf{W}) = \mathbf{W}^T\mathbf{W}$. It can also be observed from equation 10.9 that $\mathbf{Q}^T\mathbf{Q} = \mathbf{Q}_1^T\mathbf{Q}_1 + \mathbf{Q}_2^T\mathbf{Q}_2 = \mathrm{I}_{d\times d}$. Thus we maximize $\mathbf{W}^T\mathbf{z}$ under the constraint $\mathbf{W}^T\mathbf{W} = \mathrm{I}_{d\times d}$. This **W** will give first subspace independent component and second subspace IC will be mutually orthogonal to the first one. Altogether there are M subspace ICs. The p^{th} subspace IC is orthogonal to all the previous $1\ldots p-1$ subspace ICs. The same algorithm needs to be run M times to get all the subspace ICs. It is therefore rather appropriate to define a square matrix $\mathbf{\Lambda}$ of size $\mathrm{Md}\times\mathrm{Md}$ that consists of M rectangular matrices **W** such that $\mathbf{\Lambda} = [\mathbf{W}_1 \ldots \mathbf{W}_M]$. Therefore the objective is to find all **W** to get projection $\mathbf{\Lambda}^T\mathbf{z}$.

10.5.3 Fixed Point Algorithm Using Vector Kurtosis

In this section we discuss the fixed-point algorithm (Hyvärinen and Oja, 1997) for finding the projection matrix $\mathbf{W} \in \mathbf{\Lambda}$ which would enable us to find subspace independent components. Let the whitened data **z** be a set of vectors defined as $\mathbf{z} = [\mathbf{z}_1^T, \ldots, \mathbf{z}_M^T]^T$, where $\mathbf{z}_j$ is a vector of d dimension and given by $[z_1^j, \ldots, z_d^j]^T$ (z_i^j are scalar quantities). For a projection matrix **W** of size $\mathrm{Md}\times\mathrm{d}$ the gradient of absolute value of vector kurtosis can be computed as (see appendix 10.1 for the proof)

$$\frac{\partial\,|\,\mathrm{kurt}(\mathbf{W}^T\mathbf{z})\,|}{\partial\mathbf{W}} = 4\mathrm{sign}(\mathrm{kurt}(\mathbf{W}^T\mathbf{z}))\big[\mathrm{E}[(\mathbf{z}^T\mathbf{W}\mathbf{W}^T\mathbf{z})(\mathbf{z}\mathbf{z}^T\mathbf{W})] - 3\mathrm{E}[\mathbf{z}^T\mathbf{W}\mathbf{W}^T\mathbf{z}]E[\mathbf{z}\mathbf{z}^T\mathbf{W}]\big]$$

For whitened data **z** and normalized[2] **W**, the fixed-point algorithm for subspace ICA model (see appendix 10.1) would be $\mathbf{W} \leftarrow \mathrm{E}[(\mathbf{z}^T\mathbf{W}\mathbf{W}^T\mathbf{z})(\mathbf{z}\mathbf{z}^T\mathbf{W})] - 3\mathrm{d}\mathbf{W}$. The

[2] The term normalization for **W** is meant orthonormalization of the column vectors of **W**. Here we used this term to make distinction between the orthonormalization process of one **W** (say $\mathbf{W}_j$) with another (say $\mathbf{W}_k$) and to that of orthonormalization of column vectors within **W**.

algorithm will converge when the norm of new and old values of **W** point in the same direction, i.e. $\|(\mathbf{W}^{+})^{\mathrm{T}}\mathbf{W}\| \approx \|\mathrm{I}_{\mathrm{d}\times\mathrm{d}}\|$(where $\mathbf{W}^{+}$ is the new value of **W** and $\|\bullet\|$ is Frobenius norm). The iterative process can also be terminated when the vector kurtosis stops increasing.

10.5.4 Orthonormalization of a Rectangular Matrix

This procedure used in subspace ICA is briefly explained here since it is slightly different from the regular vector orthonormalization procedure.

Orthonormalization: The orthonormalization of p rectangular matrix $\mathbf{W}_p \in \mathbf{\Lambda}$ can be computed by Gram-Schmidt process:

1. $\mathbf{W}_p \leftarrow \mathbf{W}_p - \sum_{j=1}^{p-1} \mathbf{W}_j \mathbf{W}_j^{\mathrm{T}} \mathbf{W}_p$ (orthogonalize **W**)
2. $\mathbf{W}_p \leftarrow \mathbf{W}_p (\mathbf{W}_p^{\mathrm{T}} \mathbf{W}_p)^{-1/2}$ (normalize **W**)

For orthonormalization of $\mathbf{W}_p$ check if the following two conditions are satisfied:

1. $\mathbf{W}_p^{\mathrm{T}} \mathbf{W}_p = \mathrm{I}_{\mathrm{d}\times\mathrm{d}}$
2. $(\mathbf{W}_i + \mathbf{W}_j)^{\mathrm{T}} (\mathbf{W}_i + \mathbf{W}_j) = \mathbf{W}_i^{\mathrm{T}} \mathbf{W}_i + \mathbf{W}_j^{\mathrm{T}} \mathbf{W}_j$ (from Pythagorean Theorem)

or $\mathbf{W}_i^{\mathrm{T}} \mathbf{W}_j + \mathbf{W}_j^{\mathrm{T}} \mathbf{W}_i = 0_{\mathrm{d}\times\mathrm{d}}$ where $i = p$ and $j = p-1$ for $p \geq 2$. If the above two conditions are not satisfied then the Gram-Schmidt orthonormalization procedure should be repeated until both the conditions are satisfied or the values of $\mathbf{W}_p^{\mathrm{T}} \mathbf{W}_p$ and $\mathbf{W}_i^{\mathrm{T}} \mathbf{W}_j + \mathbf{W}_j^{\mathrm{T}} \mathbf{W}_i$ meet some predefined thresholds.

10.5.5 Deflationary Orthogonalization Procedure for Subspace ICA

Deflationary orthogonalization procedure can be used to estimate subspace independent components one by one. We first estimate p matrices and then orthonormalize the obtained matrices prior to running the algorithm for

$(p+1)^{\text{th}}$ matrix. The size of matrix $\mathbf{W}_p$ is $\mathrm{Md} \times \mathrm{d}$. The procedure is illustrated as in table 10.1.

For special case, when $\mathrm{d}=1$ (one-dimensional vector or scalar quantity) then the subspace ICA procedure will be reduced to the basic ICA procedure.

TABLE 10.1: Deflationary orthogonalization procedure for subspace ICA technique

1. Center data **x**.
2. Whiten data **x** to give **z**.
3. Select M, the number of subspace independent components and dimension d for each of the subspaces. Set counter $p \leftarrow 1$.
4. Select an initial value of identity norm for $\mathbf{W}_p$, e.g. randomly.
5. Let $\mathbf{W}_p \leftarrow \mathrm{E}[(\mathbf{z}^{\mathrm{T}}\mathbf{W}_p\mathbf{W}_p^{\mathrm{T}}\mathbf{z})(\mathbf{z}\mathbf{z}^{\mathrm{T}}\mathbf{W}_p)] - 3\mathrm{d}\,\mathbf{W}_p$.
6. Do orthonormalization for $\mathbf{W}_p$ (see section 10.5.4).
7. If $\mathbf{W}_p$ has not converged, go back to step 5.
8. Set $p \leftarrow p+1$ and go to step 4 until $p = \mathrm{M}$.

10.6 Illustration Using Foetal ECG

The subspace ICA model is illustrated on foetal ECG dataset (De Moor et al., 1997). The dataset consists of 2500 ECG points sampled at 500 Hz. We considered samples of four electrodes located on the abdomen of a pregnant woman. These observed samples are the mixtures of the cardiac rhythms of the mother and her foetus. The starting second of signals taken by each electrode are depicted in figure 10.1. In our model we assume two independent observations ($\mathrm{M}=2$) and the dimension of each observation vector to be two as well (i.e. each observation vector has 2 non-independent components). From figure 10.1, row 1 and row 2 are assumed to be the ‘first-subspace’ and row 3 and row 4 are assumed to be the ‘second-subspace’. Therefore row 1 and row 2 are dependent components; similarly row 3 and row 4 are dependent components. But dependencies between the two different subspaces are not allowed, i.e. they are considered as mutually independent.

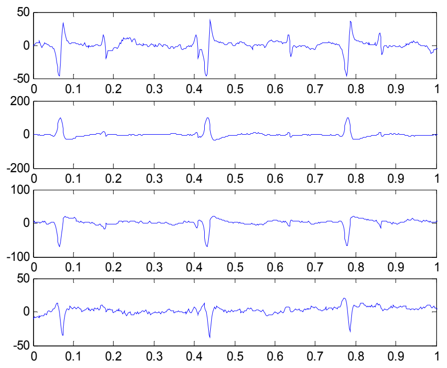


Figure 10.1: Observed ECG from 4 electrodes located on the abdomen of a pregnant woman

The convergence of subspace ICA using vector kurtosis for two subspace components with foetal ECG dataset is illustrated in figure 10.2. The absolute of vector kurtosis ($|\text{kurt}|$) is displayed as a function of iteration for the ‘first-subspace’ (figure 10.2a) and ‘second-subspace’ (figure 10.2b) components. It can be seen from both the figures that $|\text{kurt}|$ attained some finite value and converged after a few iterations.

The subspace independent components estimated by subspace ICA method using vector kurtosis are depicted in figure 10.3. The first two rows of the figure show the cardiac rhythms of the mother and the last row shows the cardiac rhythms of the foetus. The third row of the figure does not precisely follow any cardiac rhythm and is thus considered as noise being emitted from the electrodes. It can be seen that subspace ICA is well modelled on ECG dataset and is able to extract hidden cardiac rhythms.

The subspace ICA model using vector kurtosis has estimated the rhythms in a similar fashion as MICA model (Cardoso, 1998) has on the same foetal ECG database. This proves the validity of our approach. Although some finer points remain unanswered at this stage (which we have included in the ‘summary and future work’ section), the prime objective of introducing the concept of vector kurtosis for subspace ICA model is achieved.

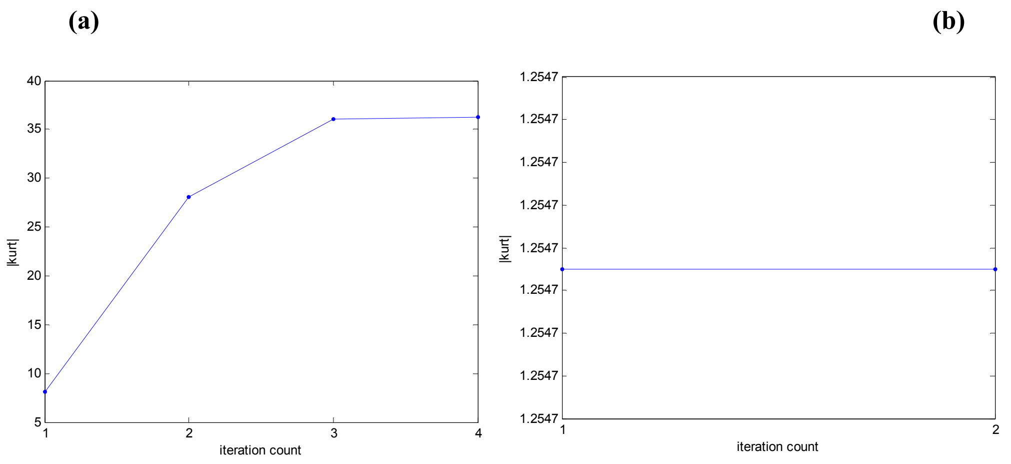


Figure 10.2: The convergence of subspace ICA algorithm using absolute of vector kurtosis on foetal ECG dataset.

a) first-subspace component

b) second-subspace component

10.7. Summary and Future Work

We have presented a new perspective of subspace ICA algorithm. The subspace ICA model is derived by partitioning the multiplicative model of basic ICA. The idea of kurtosis is extended to vector kurtosis to solve generalized version of BSS problem, i.e. when dependent components are involved. The relationship between the optimization of vector kurtosis and subspace independent components, which enabled us to estimate subspace independent components by maximizing vector kurtosis is established. It is seen that the approach works well on ECG dataset. Some essential questions are included here under to be answered in future:

- How to appropriately select the value of *d*?
- If two or more signals are linearly dependent then it is possible to have

reduced rank covariance matrix $E[\mathbf{zz}^T]$. How to apply the algorithm on reduced rank cases?

- How to select the value of M if the number of sources is completely unknown to the observer?
- The presented model can also be applied on negentropy measure.

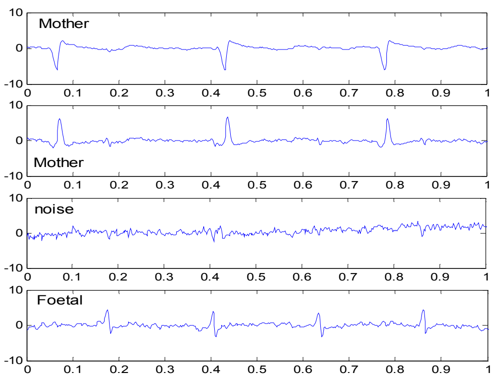


Figure 10.3: The estimated cardiac rhythms of the mother and her foetus using subspace ICA algorithm.

Appendix 10.1:

Lemma: Let vector kurtosis $\mathrm{kurt}(\mathbf{W}^{\mathrm{T}}\mathbf{z}) = \mathrm{E}[(\mathbf{z}^{\mathrm{T}}\mathbf{W}\mathbf{W}^{\mathrm{T}}\mathbf{z})^2] - 3(\mathrm{E}[\mathbf{z}^{\mathrm{T}}\mathbf{W}\mathbf{W}^{\mathrm{T}}\mathbf{z}])^2$ be a differentiable function of an $m \times n$ rectangular matrix **W** for $m \geq n$; z be any vector of size $m \times 1$. The gradient of $\mathrm{kurt}(\mathbf{W}^{\mathrm{T}}\mathbf{z})$ is defined as $\nabla_{\mathbf{W}} \mathrm{kurt}(\mathbf{W}^{\mathrm{T}}\mathbf{z}) = 4\mathrm{E}[(\mathbf{z}^{\mathrm{T}}\mathbf{W}\mathbf{W}^{\mathrm{T}}\mathbf{z})(\mathbf{z}\mathbf{z}^{\mathrm{T}}\mathbf{W})] - 12\mathrm{E}[\mathbf{z}^{\mathrm{T}}\mathbf{W}\mathbf{W}^{\mathrm{T}}\mathbf{z}]\mathrm{E}[\mathbf{z}\mathbf{z}^{\mathrm{T}}\mathbf{W}]$. In the case of whitened **z** and normalized **W**, the second term of the equation will be $12n\mathbf{W}$.

Proof: Let the scalar function be defined as $h(\mathbf{W}) = (\mathbf{z}^{\mathrm{T}}\mathbf{W}\mathbf{W}^{\mathrm{T}}\mathbf{z})$. The derivative of h with respect to **W** will then be given as:

$$\frac{\partial(h(\mathbf{W}))}{\partial \mathbf{W}} = \frac{\partial}{\partial \mathbf{W}}(\mathbf{z}^{\mathrm{T}}\mathbf{W}\mathbf{W}^{\mathrm{T}}\mathbf{z})$$

or $$\partial(\mathbf{z}^{\mathrm{T}}\mathbf{W}\mathbf{W}^{\mathrm{T}}\mathbf{z}) = \partial(trace(\mathbf{z}^{\mathrm{T}}\mathbf{W}\mathbf{W}^{\mathrm{T}}\mathbf{z})) = trace(\mathbf{z}^{\mathrm{T}}\,\partial(\mathbf{W}\mathbf{W}^{\mathrm{T}})\,\mathbf{z})$$

$$= 2trace(\mathbf{z}\mathbf{z}^{\mathrm{T}}\mathbf{W}(\partial\,\mathbf{W})^{\mathrm{T}})\ \{\text{since}\ tr(\mathbf{A}^{\mathrm{T}}) = tr(\mathbf{A})\ \text{and}\ tr(\mathbf{AB}) = tr(\mathbf{BA})\}$$

or $$h(\mathbf{W})' = 2(\mathbf{z}\mathbf{z}^{\mathrm{T}}\mathbf{W}) \qquad \text{A1}$$

Therefore the derivative of vector kurtosis (from equation A1) can be written as:

$$\nabla_{\mathrm{W}} \mathrm{kurt}(\mathbf{W}^{\mathrm{T}}\mathbf{z}) = 2\mathrm{E}[h(\mathbf{W})h(\mathbf{W})'] - 6\mathrm{E}[h(\mathbf{W})]\mathrm{E}[h(\mathbf{W})']$$

$$= 4\mathrm{E}[(\mathbf{z}^{\mathrm{T}}\mathbf{W}\mathbf{W}^{\mathrm{T}}\mathbf{z})(\mathbf{z}\mathbf{z}^{\mathrm{T}}\mathbf{W})] - 12\mathrm{E}[\mathbf{z}^{\mathrm{T}}\mathbf{W}\mathbf{W}^{\mathrm{T}}\mathbf{z}]\mathrm{E}[\mathbf{z}\mathbf{z}^{\mathrm{T}}\mathbf{W}] \qquad \text{A2}$$

However, if data **z** is whitened ($\mathrm{E}[\mathbf{z}\mathbf{z}^{\mathrm{T}}] = \mathrm{I}_{m\times m}$) and rectangular matrix **W** is normalized ($\mathbf{W}^{\mathrm{T}}\mathbf{W} = \mathrm{I}_{n\times n}$) then equation A2 can be rewritten as:

$$\nabla_{\mathbf{W}} \mathrm{kurt}(\mathbf{W}^{\mathrm{T}}\mathbf{z}) = 4\mathrm{E}[(\mathbf{z}^{\mathrm{T}}\mathbf{W}\mathbf{W}^{\mathrm{T}}\mathbf{z})(\mathbf{z}\mathbf{z}^{\mathrm{T}}\mathbf{W})] - 12n\,\mathbf{W} \qquad \text{A3}$$

$\because$ $$\mathrm{E}[\mathbf{z}^{\mathrm{T}}\mathbf{W}\mathbf{W}^{\mathrm{T}}\mathbf{z}] = \mathrm{E}[trace(\mathbf{z}^{\mathrm{T}}\mathbf{W}\mathbf{W}^{\mathrm{T}}\mathbf{z})] = trace(\mathbf{W}^{\mathrm{T}}\,\mathrm{E}[\mathbf{z}\mathbf{z}^{\mathrm{T}}]\,\mathbf{W})$$

$$= trace(\mathbf{W}^{\mathrm{T}}\mathbf{W}) = trace(\mathrm{I}_{n\times n}) = n$$

and $\mathrm{E}[\mathbf{z}\mathbf{z}^{\mathrm{T}}\mathbf{W}] = \mathrm{E}[\mathbf{z}\mathbf{z}^{\mathrm{T}}]\,\mathbf{W} = \mathbf{W}$

References

Ahalt, S.C., Krishnamurthy, A., Cheen, P., Melton, D., "Competitive learning algorithms for vector quantization", *Neural Networks*, vol. 3, pp. 277-290, 1990.

Akansu, A.N., Kadur, M.S., "Subband coding of video with adaptive vector quantization", *In Proc. ICASSP*, pp. 2109-2112, 1990.

Alexandre, L.A., Campilho, A.C., Kamel, M., "Combining independent and unbiased classifiers using weighted average", *Internl. Conf. Pattern Recognition*, vol. 2, pp. 495-498, 2000.

Alimoglu, F., Alpaydin, E., "Combining multiple representations and classifiers for pen-based handwritten digit recognition", *Internl. Conf. Document Analysis and Recognition*, vol. 2, pp. 637-640, 1997.

Anton, H., "Calculus", *John Wiley and Sons*, New York, 1995.

Antonini, M., Barlaud, M., Mathieu, P., Daubechies, I., "Image coding using vector quantization in the wavelet transform domain", *In Proc. ICASSP*, pp. 2297-2300, 1990.

Aravind, R., Gersho, A., "Low-rate image coding with finite-state vector quantization", *In Proc. ICASSP*, pp. 137-140, 1986.

Baldi, P., Hornik, K., "Learning in linear neural networks: a survey", *IEEE Trans. Neural Network*, vol. 6, no. 4, pp. 837-858, 1995.

Barba, D., Hanen, J., "The use of human visual model in sub-band coding of color video signal with adaptive chrominance signal vector quantization", *In Proc. of Visual Comm. and Image Processing '91*, vol. 1605, pp. 408-419, 1991.

Belhumeur, P.N., Hespanha, J.P., Kriegman, D.J., “Eigenfaces vs. fisherfaces: recognition using class specific linear projection”, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 711-720, 1997.

Bilmes, J., “A gentle tutorial on the EM algorithm and its application to the parameter estimation for Gaussian mixture and Hidden Markov Models”, *Technical Report ICSI-TR-97-021*, University of Berkeley, 1997.

Blake, C.L., Merz, C.J., “UCI repository of machine learning databases”, *http://www.ics.uci.edu/~mlearn*, Irvine, CA, University of Calif., Dept. of Information and Comp. Sci., 1998.

Breukelen van, M., Duin, R.P.W., “Neural network initialization by combined classifiers”, *Internl. Conf. Pattern Recognition*, vol. 1, pp. 215-218, 1998.

Cardoso, J.-F., “Multidimensional independent component analysis”, Proc. *ICASSP*, vol. 37, 1998.

Chaffee, D.L., “Applications of rate distortion theory to the bandwidth compression of speech signals”, *PhD Thesis*, Univ. of Calif. at Los Angeles, 1975.

Chen, Y., Chu, M., Chang, E., Liu, J., Liu, R., “Voice conversion with smoothed GMM and MAP adaptation”, *Eurospeech*, pp. 2413-2416, 2003.

Chen, L.-F., Liao, H.-Y.M., Ko, M.-T., Lin, J.-C., Yu, G.-J., “A new LDA-based face recognition system which can solve the small sample size problem”, *Pattern Recognition*, vol. 33, pp. 1713-1726, 2000.

Chen, X., Yang, J., Zhang, J., Waibel, A., “Automatic detection and recognition of signs from natural scenes”, *IEEE Trans. on Image Processing*, vol. 13, no. 1, pp. 87-99, 2004.

Chin-Teng, L., Shi-An, C., Chao-Hui, H., Jen-Feng, C., “Cellular neural networks and PCA neural networks based rotation/scale invariant texture classification”, *IEEE International Jnt. Conf. Neural Networks*, vol. 1, pp. 158, 2004.

Cosman, P.C., Gray, R.M., Vetterli, M., “Vector quantization of image subbands: a survey”, *IEEE Trans. on Image Processing*, vol. 5, no. 2, pp. 202-225, 1996.

Datta, P., Kibler, D., “Symbolic nearest mean classifiers”, *Proc. of the 14th National Conf. on Artificial Intelligence*, San Mateo, CA, pp. 82-87, 1997.

De la Torre, F., Black, M.J., “Robust principal component analysis for computer vision”, *Int. Conf. on Computer Vision, ICCV-2001*, Vancouver, BC, vol. I, pp. 362-369, 2001.

De Moor, B.L.R., Gersem, P.D., Schutter, B.D., Favoreel, W., (eds.), “Daisy: Database for the identification of systems”, *http://www.esat.kuleuven.ac.be/sista/daisy*, 1997.

Dempster, A., Laird, N., Rubin, D., “Maximum likelihood from incomplete data via the EM algorithm”, *Jnrl. of th Royal Statistical Society Series B* 39, pp. 1-38, 1977.

Diamantaras, K.I., “Asymmetric PCA neural networks for adaptive blind source separation”, *Neural Networks for Signal Processing VIII Proc. IEEE Signal Processing Society Worksop*, pp. 103-112, 1998.

Dony, R.D., Haykin, S., “Optimally adaptive transform coding”, *IEEE Trans. on Image Processing*, vol. 4, pp. 1358-1370, 1995.

Dony, R.D., Haykin, S., “Image segmentation using a mixture of principal components representation”, *IEE Proceedings: Vision, Image and Signal Processing*, vol. 144, no. 2, pp. 73-80, 1997.

Dony R.D., Haykin, S., “Compression of SAR images using KLT, VQ and mixture of principal components”, *IEE Proc.-Radar Sonar Navig.*, vol. 144, no. 3, pp. 113-120, 1997b.

Di Maio, V., Marciano, F., “Automatic classification of neural spike activity: an application of minimum distance classifiers”, *Cybernetics and Systems*, vol. 34, no. 3 pp. 173-192, 2003.

Duda, R.O, Hart, P.E., “Pattern classification and scene analysis”, *John Wiley and Sons*, New York, 1973.

Duin, R.P.W., “Small sample size generalization”, in G. Borgefors (Eds.), *SCIA’95, Proc. 9th Scandinavian Conf. Image Analysis*, vol. 2, pp. 957-964, 1995.

Feng, C., Suntherland, A., King, S., Muggleton, S., Henry, R., “Comparison of Machine Learning Classifiers to Statistics and Neural Networks”, *AI & Stats Conf.*, pp. 41-52, 1993.

Fisher, R.A., “The statistical utilization of multiple measurements”, *Ann. Eugenics*, vol. 8, pp. 376-386, 1936.

Fritzke, B., “The LBG-U Method for vector quantization – an improvement over LBG inspired from neural networks”, *Neural Processing Letters*, vol. 5, no. 1, pp. 35-45, 1997.

Fukunaga K., “Introduction to statistical pattern recognition”, *Academic Press Inc., Hartcourt Brace Jovanovich, Publishers*, 1990.

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

Gersho, A., Gray, R.M., “Vector quantization and signal compression”, *Kluwer Academic Publishers*, Norwell, Massachusetts, 1992.

Ghahramani, Z., "Solving inverse problems using an EM approach to density estimation", *In Proceedings of the 1993 Connectionist Models Summer School*, Lawrence Erlbaum Publishers, pp. 316-323, 1994.

Ghahramani, Z., Jordan, M.I., "Supervised learning from incomplete data via an EM approach", in J.D. Cowan, G. Tesauro, J. Alspector, eds, *Advances in Neural Information Processing Systems 6*, Morgan Kaufmann, San Mateo, Calif., pp.120-127, 1994.

Golub, G.H, van Loan C.F., Matrix Computations, 3 ed., *John Hopkins University Press*, Baltimore, 1996.

Gray R. M., "Vector quantization", *IEEE ASSP Magazine*, pp. 4-29, 1984.

Griguolo, S., "Pixel-by-pixel clustering for vegetation monitoring", *Int. Conf. on "Alerte précoce et suivi de l'Environment"*, Niamey, Niger, 1994.

Haeb-Umbach, R., Ney, H., "Linear discriminant analysis for improved large vocabulary continuous speech recognition", *Proceedings. Internl. Conf. on Acoustics, Speech and Signal Processing*, vol. 1, pp. 13-16, 1992.

Hastie, T., "Principal curves and surfaces", *PhD Thesis*, Stanford University, Calif., November 1984.

Ho, T.K., Hull, J.J., Srihari, S.N., "Decision combination in multiple classifier systems", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 16, no. 1, pp. 66-75, 1994.

Hofmann, T., Buhmann, J., "Pairwise data clustering by deterministic annealing", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 19, no. 1, pp. 1-14, 1997.

Hyvärinen, A. , "Fast and robust fixed-point algorithms for independent component analysis", *IEEE Trans. on Neural* Networks, vol. 10, no. 3, pp. 626-634, 1999.

Hyvärinen, A., Oja, E., “A fast fixed-point algorithm for independent component analysis”, *Neural Computation*, vol. 9, no. 7, 1483-1492, 1997.

Hyvärinen, A., Hoyer, P., “Emergence of phase- and shift-invariant features by decomposition of natural images into independent feature subspaces”, *Neural Computation*, vol. 12, pp. 1705-1720, 2000.

Hyvärinen, A., Karhunen, J., Oja. E., “Independent Component Analysis”, *Wiley-Intersci. Pub.*, 2001.

Itakura, F. Saito, S., “Analysis synthesis telephony based upon maximum likelihood method”, *Repts. of the 6th International Cong. Acoust.,* Y. Kohasi, ed., Tokyo, C-5-5, C17-20, 1968.

Jacobs, R.A., Jordan, M.I., Nowlan, S.J., Hinton, G.E., “Adaptive mixtures of local experts”, *Neural Computation*, vol. 3, pp. 79-87, 1991.

Jain, A.K., Duin, R.P.W., Mao, J., “Statistical pattern recognition: a review”, *IEEE Trans. on Pattern Anal. Machine Intelligence*, vol. 22, no. 1, pp. 4-37, 2000.

Jian, H., Yuen, P.C., Wen-Sheng, C., “A novel subspace LDA algorithms for recognition of face images with illumination and pose variations”, *Proceedings. Internl. Conf. on Machine Learning and Cybernetics*, vol. 6, pp. 3589-3594, 2004.

Kambhatla, N., “Local models and Gaussian mixture models for statistical data processing”, *PhD Thesis*, Oregon Graduate Inst. of Sci. and Technology, 1996.

Kambhatla N., Leen T.K., “Dimension Reduction by local PCA”, *Neural Computing*, vol. 9, pp. 1493-1516, 1997.

Karayiannis, N.B., “A Methodology for clustering fuzzy algorithms for learning vector quantization”, *IEEE Trans. on Neural Networks*, vol. 8, no. 3, pp. 505-518, 1997.

Kittler, J., Hatef, M., Duin, R.P.W., Matas, J., "On combining classifiers", *Internl. Conf. Pattern Recognition*, vol. 2, pp. 897-901, 1996.

Kittler, J., Hatef, M., Duin, R.P.W., Matas, J., "On combining classifiers", *IEEE Trans. Pattern Analysis Machine Intelligence*, vol. 20, no. 3, pp. 226-239, 1998.

Kohonen, T., "Emergence of invariant-feature detectors in the adaptive-subspace self-organizing map", *Bio. Cyber.*, vol. 75, pp. 281-291, 1996.

Koren Y., Carnel L. "Robust Linear Dimensionality Reduction", *IEEE Trans. on Visualization and Computer Graphics*, vol. 10, no. 4, pp. 459-470, 2004.

Lambrou, T., Linney, A.D., Speller, R.D., Todd-Pokropek, A., "Statistical classification of digital mamograms using features from the spatial and wavelet domains", *Medical Image Understanding and Anal.*, Portsmouth, UK, 22-23 July, 2002.

Lee, C., Chen, L., "A fast search algorithm for vector quantization using mean pyramids of codewords", *IEEE Trans. on Comm.*, vol. 43, no. 2/3/4, pp.1697-1702, 1995.

Lei, L., Xiao-Long, W., Bing-Quan, L., "Combining multiple classifiers based on statistical method for handwritten Chinese character recognition", *Int. Conf. Machine Learning and Cybernetics*, vol. 1, pp. 252-255, 2002.

Lewenstein, K., Chojnacki, M., "Minimum distance classifiers in coronary artery disease diagnosing", *Modelling in Mechatronics*, Kazimierz Dolny, Poland, 2004.

Lieb, M., Haeb-Umbach, R., "LDA derived cepstral trajectory filters in adverse environment conditions", *Proceedings. Internl. Conf. on Acoustics, Speech and Signal Processing*, vol. 2, pp. II1105-II1108, 2000.

Linde Y., Buzo A., Gray R.M., "An algorithm for vector quantization design", *IEEE Trans. on Comm.*, COM-28, no.1, pp. 84-94, 1980.

Lotlikar, R., Kothari, R., “Fractional-step dimensionality reduction”, *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 6, pp. 623-627, 2000.

Lu, J., Plataniotis, K.N., Venetsanopoulos, A.N., “Face recognition using LDA-based algorithms”, *IEEE Trans. Neural Networks*, vol. 14, no. 1, pp. 195-200, 2003.

Magnus, J.R., Neudecker, H., “Matrix differential calculus with applications in statistics and econometrics”, *John Wiley and Sons Ltd.*, 1994.

Makhoul, J., Roucos, S., Gish, H., “Vector quantization in speech coding”, *Proc. of the IEEE*, vol. 73, no. 11, pp. 1551-1588, 1985.

Mardia, K.V., Kent, J.T., Bibby, J.M., “Multivariate analysis”, *Academic Press*, New York, 1979.

Michie, D., Spiegelhalter, D.J., Taylor, C.C., (Eds.), “Machine learning, neural and statistical classification”, *Ellis Horwood*, 1994.

Mika, S., Rätsch, G., Weston, J., Schölkopf, B., Smola, A., Müller, K.-R., “Constructing descriptive and discriminative nonlinear features: Rayleigh coefficients in kernel feature spaces”, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 25, no. 5, pp. 623-628, 2003.

Oja E., “A Simplified Neuron Model as a Principal Component Analyzer”, *J. of Mathematical Biology*, Vol. 15, pp. 267-273, 1982.

Oja, E., “Subspace methods of pattern recognition”, *Research Studies Press*, New York, 1983.

Oja, E., Parkkinen, J., “On subspace clustering”, *Seventh Internl. Conf. on Pattern Recognition*, vol. 2, pp. 692-695, 1984.

Oliveira, P.R., Romero, R.F., “A comparision between PCA neural networks and the JPEG standard for performing image compression”, *Workshop on Cybernetic Vision*, pp. 112-116, 1996.

Ormoneit, D., Tresp, V., “Improved Gaussian mixture density estimates using Bayesian penalty terms and network averaging”, *Technical Report FKI-205-95*, Technische Universität München, 1995.

Paclik, P., Duin, R.P.W., “Dissimilarity-based classification of spectra: computational issues”, *Real-Time Imaging*, vol. 9, pp. 237-244, 2003.

Paliwal K.K., Atal B.S., “Efficient Vector Quantization of LPC Parameters at 24 Bits/Frame. *IEEE Trans. Speech and Audio Processing*, vol. 1, no. 1, pp. 3-14, 1993.

Park, A., Hazen, T.J., “ASR dependent techniques for speaker recognition”, *MIT Laboratory for Comp. Sc.*, pp. 479-480, 2003.

Patané, G., Russo, M., “The enhanced LBG algorithm”, *Neural Networks*, vol. 14, pp. 1219-1237, 2001.

Peper, F., Noda, H., “A class of simple nonlinear 1-unit PCA neural networks”, *IEEE International Conf. Neural Networks*, vol. 1, pp. 285-289, 1995.

Potamianos, G., Graf, H.P., “Linear discriminant analysis for speechreading”, *IEEE Second Workshop on Multimedia Signal Processing*, pp. 221-226, 1998.

Proakis, J.G., Manolakis, D.G., “Digital Signal Processing principles, algorithms, and applications”, *Prentice Hall International*, 1996.

Rao, C.R., “The utilization of multiple measurements in problems of biological classification”, *J. Royal Statistical Soc.*, B, vol. 10, pp. 159-203, 1948.

Raudys, S., Duin, R.P.W., “Expected classification error of the Fisher linear classifier with pseudo-inverse covariance matrix”, *Pattern Recognition Letters*, vol. 19, pp. 385-392, 1998.

Reddy, K.M., Herron, T.J., “Computing the Eigen Decomposition of a Symmetric Matrix in Fixed-point Algorithms”, *IEEE Bangalore Section-Tenth Annual Symposium*, 2001.

Roweis, S., “EM Algorithms for PCA and SPCA”, *Neural Information Processing Systems*, vol. 10, pp. 626-632, 1997.

Sahin, F., “A radial basis function approach to a color image classification problem in a real time industrial application”, *PhD Thesis*, State University, Virginia, 2000.

Schilling, R.J., Harris, S.L., “Applied Numerical Methods for Engineers Using Matlab and C”, *Brooks/Cole Publishing Company*, 2000.

Schölkopf, B., Smola, A, Müller, K-R., “Nonlinear component analysis as a kernel eigenvalue problem”, *Neural Computation*, vol. 10, pp. 1299-1319, 1998.

Senior, A., “Combination Fingerprint Classifier”, *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 23, no. 10, pp. 1165-1174, 2001.

Sharma, A., “Signal modeling in speaker recognition”, *Botswana Journal of Technology*, vol. 14, no. 1, pp. 59-66, 2005.

Sharma, A., “Radioactive mineral identification based on FFT Radix-2 algorithm”, *IEE Letters*, vol. 40, no. 9, pp. 536-537, 2004.

Sharma, A., Paliwal, K.K., Onwubolu, G.C., “Class-dependent PCA, MDC and LDA: A Combined Classifier for Pattern Classification”, *Pattern Recognition*, vol. 39, issue 7, pp. 1215-1229, 2006.

Sharma, A., Paliwal, K.K., Onwubolu, G.C., "Pattern classification: an improvement using combination of VQ and PCA based techniques", *American Journal of Applied Sciences*, vol. 2, no. 10, pp. 1445-1555, 2005.

Sharma, A., Paliwal, K.K., Onwubolu, G.C., "Splitting technique initialization in local PCA", *Journal of Computer Science*, vol. 2, no. 1, pp. 53-58, 2006b.

Sharma, A., Paliwal, K.K., "Subspace Independent Component Analysis using Vector Kurtosis", *Pattern Recognition*, vol. 39, issue 11, pp. 2223-2226, 2006.

Sharma, A., Paliwal, K.K., "Rotational Linear Discriminant Analysis for Dimensionality Reduction", *Machine Learning*, 2006b (under review).

Sharma, A., Paliwal, K.K., "Fast Principal Component Analysis using Fixed-Point Algorithm", *Pattern Recognition Letters*, 2006c (under review).

Sharma, A., Paliwal, K.K., "A Gradient Linear Discriminant Analysis for Small Sample Sized Problem", *Pattern Analysis and Application*, 2006d (under review).

Sharma, A., Paliwal, K.K., "Rotational Linear Discriminant Analysis Using Bayes Rule for Dimensionality Reduction", *Journal of Computer Science*, vol. 2, no. 9, pp. 754-757, 2006e.

Sharma, A., Paliwal, K.K., "A New Gradient Linear Discriminant Analysis Technique for Small Sample Size Problem", *Applied Intelligence*, 2006f (under review).

Siohan, O., "On the robustness of linear discriminant analysis as a preprocessing step for noisy speech recognition", *Proceedings. Internl. Conf. on Acoustics, Speech and Signal Processing*, vol. 1, pp. 125-128, 1995.

Sirovich, L., "Turbulence and the dynamics of coherent structures", *Quarterly of Applied Mathematics*, XLV, pp. 561-571, 1987.

Skurichina, M., Duin, R.P.W., “Stabilizing classifiers for very small sample size”, *Proc.释Internl. Conf. Pattern Recognition*, vol. 2, pp. 891-896, 1996.

Skurichina, M., Duin, R.P.W., “Regularization of linear classifiers by adding redundant features”, *Pattern Analysis and Applications*, vol. 2, no. 1, pp. 44-52, 1999.

Soong, F.K., Rosenberg, A.E., Juang, B., “A vector quantization approach to speaker recognition”, *AT&T Technical Jnrl.*, vol. 66, issues 2, pp. 14-26, 1986.

Swets, D.L., Weng, J., “Using discriminative eigenfeatures for image retrieval”, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 18, no. 8, pp. 831-836, 1996.

Tae-Kyun, K., Hyunwoo, K., Wonjun, H., Seok-Cheol, K., Kittler, J., “Face description based on decomposition and combining of a facial space with LDA”, *Proceedings. Internl. Conf. on Image Processing*, vol. 3, pp. III-877 – III-880, 2003.

Tao, F., Jiangang, L., Zhi, W., Youxian, S., “An image compressing algorithm based on PCA/SOFM hybrid neural network”, *Industrial Electronics Conf. (IECON ’03)*, vol. 3, pp. 2103-2107, 2003.

Tibshirani, R., “Principal curves revisited”, *Statistics and Computing*, vol. 2, pp. 183-190, 1992.

Tipping, M.E., Bishop, C.M., “Mixtures of probabilistic principal component analysers”, *Tech. Rep. NCRG/97/003*, Neural Computing Research Group, Aston University, 1998.

Tresp, V., Taniguchi, M., “Combining estimators using non-constant weighting functions”, In G. Tesauro, D.S. Touretzky, T.K. Leen (Eds.), *Advances in Neural Info. Processing Systems 7*, MIT press, Cambridge, 1995.

Toth, D., Condurache, A., Aach, T., “A two-stage-classifier for defect classification in optical media inspection”, *16th Int. Conf. on Pattern Recognition (ICPR'02)*, vol. 4, pp. 373-376, 2002.

Turner, K., Gosh, J., “Linear and order statistics combiners for pattern classification”, In A. Sharkey, ed., *Combining Artificial Neural Nets*, Springer-Verlag, pp. 127-161, 1999.

Ueda, N., “Optimal linear combination of neural networks for improving classification performance”, *IEEE Trans. Pattern Analysis Machine Intelligence*, vol. 22, no. 2, pp. 207-215, 2000.

Wang, S., Liang, Y., Ma, F., “An adaptive robust PCA neural network”, *IEEE Jnt. Conf. Neural Networks*, vol. 3, pp. 2288-2293, 1998.

Woods, K., “Combination of multiple classifiers using local accuracy estimates”, *IEEE Trans. Pattern Analysis Machine Intelligence*, vol. 19, no. 4, pp. 405-410, 1997.

Woods, K., Bowyer, K., Kegelmeyer, W.P., “Combination of multiple classifiers using local accuracy estimates”, *IEEE Comp. Soc. Conf. Computer Vision and Pattern Recognition CVPR ‘96*, pp. 391-396, 1996.

Wu, X.-J., Kittler, J., Yang, J.-Y., Messerm, K., Wang, S., “A new direct LDA (D-LDA) algorithm for feature extraction in face recognition”, *Proceedings. Internl. Conf. on Pattern Recognition*, vol. 4, pp. 545-548, 2004.

Xiaogang, W., Xiaoou, T., “A unified framework for subspace face recognition”, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 26, no. 9, pp. 1222-1228, 2004a.

Xiaogang, W., Xiaoou, T., “Using random subspace to combine multiple features for face recognition”, *Proceedings. Sixth IEEE Internl. Conf. Automatic Face and Gesture Recognition*, pp. 284-289, 2004b.

Xu, L., Krzyżak, A., Suen, C.Y., "Methods of combining multiple classifiers and their applications to handwriting recognition", *IEEE Trans. on Systems Man. and Cybernetics*, vol. 22, no. 3, pp. 418-435, 1992.

Xu, L., "Robust principal component analysis by self-organizing rules based on statistical physics approach", *IEEE Trans. Neural Networks*, vol. 6, no. 1, pp. 131-143, 1995.

Yang, J., Frangi, A.F., Yang, J.-y., Zhang, D., Zhong, J., "KPCA plus LDA: a complete kernel Fisher discriminant framework for feature extraction and recognition", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 27, no.2, pp. 230-244, 2005.

Yao, M., Pan, X., He, T., Zhang, R., "An improved combination method of multiple classifiers based on fuzzy integrals", *World Congress on Intelligent Control and Automation*, vol. 3, pp. 2445-2447, 2002.

Young, S., Evermann, G., Hain, T., Kershaw, D., Moore, G., Odell, J., Ollason, D., Povey, D., Valtchev, V., Woodland, P., "The HTK Book Version 3.2", *Cambridge University*, Cambridge, England, 2002.

Yu, H., Yang, J., "A direct LDA algorithm for high-dimensional data-with application to face recognition", *Pattern Recognition*, vol. 34, pp. 2067-2070, 2001.

Zhao, W., Chellappa, R., Nandhakumar, N., "Empirical performance analysis of linear discriminant classifiers", *Proceedings. IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 164-169, 1998.

Zhao, W., Chellappa, R., Phillips, P.J., "Subspace linear discriminant analysis for face recognition", *Tech. Rep. CAR-TR-914*, Center for Automation Research, University of Maryland, College Park, 1999.

Zhou, L., Imai, S., "Chinese all syllables recognition using combination of multiple classifiers", *ICASSP*, vol. 6, pp. 3494-3497, 1996.

Zhou, X.S., Huang, T.S., “Small Sample Learning during Multimedia Retrieval Using BiasMap”, *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition*, vol. 1, pp. 11-17, 2001.