Feature Extraction and Dimensionality Reduction in Pattern Recognition and Their Application in Speech Recognition

A Dissertation
Presented to
School of Microelectronical Engineering
Faculty of Engineering and Information Technology
Griffith University

Submitted in Fulfillment of
the Requirements of the Degree of
Doctor of Philosophy

by
Xuechuan Wang
November 2002

STATEMENT OF ORIGINALITY

This work has not 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 thesis itself.

Xuechuan Wang, November 2002

ABSTRACT

Conventional pattern recognition systems have two components: feature analysis and pattern classification. Feature analysis is achieved in two steps: parameter extraction step and feature extraction step. In the parameter extraction step, information relevant for pattern classification is extracted from the input data in the form of parameter vector. In the feature extraction step, the parameter vector is transformed to a feature vector. Feature extraction can be conducted independently or jointly with either parameter extraction or classification. Linear Discriminant Analysis (LDA) and Principal Component Analysis (PCA) are the two popular independent feature extraction algorithms. Both of them extract features by projecting the parameter vectors into a new feature space through a linear transformation matrix. But they optimize the transformation matrix with different intentions. PCA optimizes the transformation matrix by finding the largest variations in the original feature space. LDA pursues the largest ratio of *between*-class variation and *within*-class variation when projecting the original feature space to a subspace. The drawback of independent feature extraction algorithms is that their optimization criteria are different from the classifier's minimum classification error criterion, which may cause inconsistency between feature extraction and the classification stages of a pattern recognizer and consequently, degrade the performance of classifiers. A direct way to overcome this problem is to conduct feature extraction and classification jointly with a consistent criterion. Minimum Classification Error (MCE) training algorithm provides such an integrated framework. MCE algorithm was first proposed for optimizing classifiers. It is a type of discriminative learning algorithm but achieves minimum classification error directly. The flexibility of the framework of MCE algorithm makes it convenient to conduct feature extraction and classification jointly. Conventional feature extraction and pattern classification algorithms, LDA, PCA, MCE training algorithm, minimum distance classifier, likelihood classifier and Bayesian classifier, are linear algorithms. The advantage of linear algorithms is their simplicity and ability to reduce feature dimensionalities. However, they have the limitation that the decision boundaries generated are linear and have little computational flexibility. SVM is a recently developed integrated pattern classification algorithm with non-linear formulation. It is based on the idea that the classification that affords dot-products can be computed efficiently in higher dimensional feature spaces. The classes which are not linearly separable in the original parametric space can be linearly separated in the higher dimensional feature space. Because of this, SVM has the advantage that it can handle the classes with complex non-linear decision boundaries. However, SVM is a highly integrated and closed pattern classification system. It is very difficult to adopt feature extraction into SVM's framework. Thus SVM is unable to conduct feature extraction tasks. This thesis investigates LDA and PCA for feature extraction and dimensionality reduction and proposes the application of MCE training algorithms for joint feature extraction and classification tasks. A generalized MCE (GMCE) training algorithm is proposed to mend the shortcomings of the MCE training algorithms in joint feature and classification tasks. SVM, as a non-linear pattern classification system is also investigated in this thesis. A reduced-dimensional SVM (RDSVM) is proposed to enable SVM to conduct feature extraction and classification jointly. All of the investigated and proposed algorithms are tested and compared firstly on a number of small databases, such as Deterding Vowels Database, Fisher's IRIS database and German's GLASS database. Then they are tested in a large-scale speech recognition experiment based on TIMIT database.

ACKNOWLEDGMENTS

I am particularly grateful to my supervisor, Professor Kuldip K. Paliwal who offered prompt, wise and always constructive feedback and advise (despite busy schedules) and who displayed unerring professionalism, graciousness and indefatigable dedication to research. Professor Paliwal's depth of knowledge, insight and untiring work ethic has been and will continue to be a source of inspiration to me.

My special thanks go to Dr. Jun Wei Lu for providing me with numerous and priceless help and advices both in my research and everyday life.

A special thank to my wife, Delia Qinghong Lin, for her love and unfailing support. Her support and help is priceless to the completion of my study.

The long, hard process of completing a thesis would have been completely impossible without the support of many friends and colleagues in Signal Processing Lab over the years.

This thesis would not have been possible without the financial assistance from Overseas Postgraduate Research Scholarship (OPRS) and Microelectronical Engineering Postgraduate Research Scholarship (MEEPRS).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Human beings have dreamed to build a highly intelligent machine that can do things like themselves. The motivation for this effort comes from the practical need to find more efficient ways to accomplish intellectual tasks in many areas, such as manufacturing, biology, clinics, mining, communication and military application. Intellectual tasks include realization, evaluation and interpretation of information that comes from sensors. All of these can be summarized by *perception*. Perception allows human beings to acquire knowledge about the environment, react to it and finally influence it [70]. Although every human being has the ability to perceive information, it is by far impossible to explain the intrinsic mechanics of perception, that is, the algorithms which might be implemented on a computer. It has been of great scientific interest to exploit the mathematical aspect of perception. This is found in the area of artificial intelligence, in which pattern recognition is a core technique that assigns to machines the ability to recognize and classify external objects so as to react to the changing environments. Because of the nature of lacking a complete theory of perception, the study of pattern recognition has led to an abstract mathematical model that provides the theoretical basis for recognizer design.

1.1 Pattern Recognition

Pattern recognition deals with mathematical and technical aspects of classifying different objects through their observable information, such as grey levels of pixels for an image, energy levels in frequency domain for a waveform and the percentage of certain contents in a product. The objective of pattern recognition is achieved in a three-step procedure, as shown in Figure 1.1. The observable information of an unknown object is first transduced into signals that can be analysed by computer systems. Parameters and/or features suitable for classification are then extracted from the collected signals. The extracted parameters and/or features are classified in the final step based on certain types of measures, such as distance, likelihood and Bayesian, over class models.

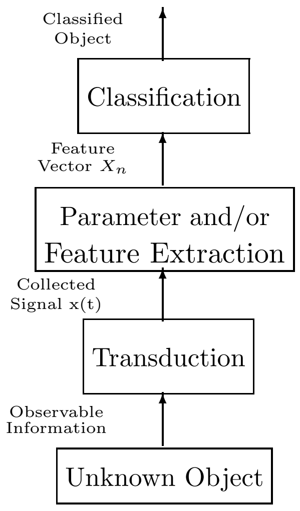


Figure 1.1: A typical pattern recognition procedure.

Transduction step is achieved by physical or chemical methods or apparatus which are closely related to the physical or chemical characteristics of the objects. It is normally beyond the scope of the study of pattern recognition. Thus a typical pattern recognition system consists of two components: feature analysis, which includes parameter extraction and/or feature extraction, and pattern classification. The structure of a conventional pattern recognition system is shown in Figure 1.2.

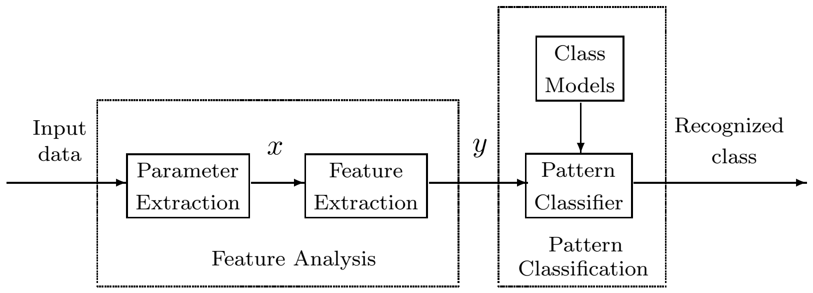


Figure 1.2: Conventional pattern recognition system.

1.1.1 Feature Analysis

Feature analysis is achieved in two steps: parameter extraction and/or feature extraction. In the parameter extraction step, information relevant to pattern classification is extracted from the input data in the form of a p-dimensional parameter vector x. In the feature extraction step, the parameter vector x is transformed to a feature vector y, which has a dimensionality m $(m \leq p)$. If the parameter extractor is properly designed so that the parameter vector x is matched to the pattern classifier and its dimensionality is low, then there is no necessity for the feature extraction step. However in practice, parameter vectors are not suitable for pattern classifiers. For example, parameter vectors have to be decorrelated before applying them to a classifier based on Gaussian mixture models (with diagonal variance matrices). Furthermore, the dimensionality of parameter vectors is normally very high and needs to be reduced for the sake of less computational cost and system complexity. Due to these reasons, feature extraction has been an important part in pattern recognition tasks.

Feature extraction can be conducted independently or jointly with either parameter extraction or classification. Independent feature extraction method is a well-developed area of research. A number of independent feature extraction algorithms have been proposed [19, 27, 42, 46, 48, 80]. Among them, LDA and PCA are the two popular independent feature extraction methods. Both of them extract features by projecting the original parameter vectors onto a new feature space through a linear transformation matrix. But they optimize the transformation matrix with different intentions. PCA optimizes the transformation matrix by finding the largest variations in the original feature space [48, 53, 80]. LDA pursues the largest ratio of *between*-class variation and *within*-class variation when projecting the original feature to a subspace [13, 78, 93]. The drawback of independent feature extraction algorithms is that their optimization criteria are different from the classifier's minimum classification error criterion, which may cause inconsistency between feature extraction and the classification stages of a pattern recognizer and consequently, degrade the performance of classifiers [54].

A direct way to overcome the problem with independent feature extraction algorithms is to conduct feature extraction and classification jointly with a consistent criterion. Integrated feature extraction and classification has become a subject of major importance, recently[76]. The structure of a pattern recognition system using integrated feature extraction and classification algorithm is shown in Figure 1.3. MCE training algorithm provides an ideal integrated framework for joint feature extraction and classification. MCE training algorithm was first proposed for optimizing classifiers [54, 56]. It was derived from discriminant analysis but achieves minimum classification error directly. This direct relationship has made MCE training algorithm widely popular in a number of pattern recognition applications, such as dynamic time-wrapping based speech recognition[20, 57] and Hidden Markov Model (HMM) based speech and speaker recognition[21, 63, 79]. The characteristics of MCE training algorithm also enable it to conduct joint feature extraction and classification tasks easily. In this thesis, we propose the use of MCE training algorithm for integrated feature extraction and classification. A generalized MCE (GMCE) training algorithm is proposed to mend the shortcomings of MCE training algorithm that appear in the joint feature extraction and classification tasks.

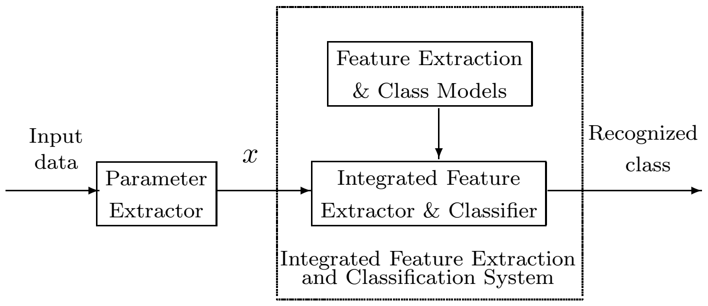

Feature Extraction & Class Models
Input data
Parameter Extractor
x
Integrated Feature Extractor & Classifier
Recognized class
Integrated Feature Extraction and Classification System

Figure 1.3: Integrated pattern recognition system.

Both independent and integrated feature extraction algorithms extract features through a linear transformation matrix. The advantage of linear transformation matrices is their ability to reduce feature dimensionalities. Pattern recognition systems can be benefitted from feature dimensionality reduction, such as less system complexity and computational cost. Therefore, the performances of feature extraction algorithms in feature dimensionality reduction are also investigated in this thesis.

1.1.2 Pattern Classification

The objective of pattern classification is to assign an input feature vector to one of K existing classes based on a classification measure. Conventional classification measures include distance (Mahalanobis or Euclidean distance), likelihood and Bayesian *a posteriori* probability. These measures lead to linear classification methods, i.e., the decision boundaries they generate are linear. Linear methods, however, have the limitation that they have little computational flexibility and are unable to handle complex non-linear decision boundaries. SVM is a recently developed pattern classification algorithm with non-linear formulation. It is based on the idea that the classification that affords dot-products can be computed efficiently in higher dimensional feature spaces [14, 82, 99]. The classes which are not linearly separable in the original parametric space can be linearly separated in the higher dimensional feature space. Because of this, SVM has the advantage that it can handle the classes with complex non-linear decision boundaries. SVM has now evolved into an active area of research [52, 86, 87, 89].

Different from conventional pattern recognition systems, SVM bypasses feature extraction step and uses parameter vectors directly as its input.

However, the dimensionality of parameter vectors in modern pattern recognition systems are normally very high. In speech recognition, for example, the dimensionality is around 40 and in image recognition, it is often more than 100. This leads to high complexity of SVM systems. Furthermore, large amount of irrelevant information that resides in parameter vectors will make the computational expense of SVM unnecessarily high. In this thesis, we investigate the performance of SVM in low-dimensional discriminated feature spaces. A reduced-dimensional SVM (RDSVM) is proposed to adopt feature extraction into SVM training.

1.2 Contributions

The following contributions are made in this thesis:

- *Alternative MCE Training Algorithm*, Chapter 4: The conventional MCE training algorithm uses additive model to formulate the misclassification measure. However, additive model is not suitable for accommodating gradient descent method for optimization. This chapter proposes an alternative form of MCE training algorithm to improve the performance of conventional MCE training algorithm. The proposed algorithm uses a ratio model of misclassification measure, which is more suitable for the gradient descent method than the additive model used conventionally.

- *MCE Training Algorithm for Joint Feature Extraction and Classification*, Chapter 5: Independent feature extraction method is a well-developed area of research. LDA and PCA are the two popular independent feature extraction algorithms. However, they have inconsistent optimization criteria to the minimum classification error objective. This may cause dismatch between the features extraction and the classification and thus degrade the performance of pattern recognition systems. A direct way to mend this drawback is to conduct feature extraction and classification jointly. This chapter proposes the use of MCE training algorithm for joint feature extraction and classification. MCE training algorithm provides an integrated framework and is suitable for this joint task. The corresponding formulation is derived in this chapter.

- *Generalized MCE (GMCE) training Algorithm*, Chapter 6: One significant limitation appearing in the performance of MCE training algorithm in joint feature extraction and classification tasks is that the success of MCE training is highly dependent on the the initialization of the parameter set, especially the transformation matrix. This leads to poor generalization properties of MCE models. A major reason is that

MCE training algorithm employs gradient descent method for model optimization, while the gradient descent method is dependent to the starting point and does not guarantee the global minimum. This chapter proposes a generalized MCE (GMCE) training algorithm to mend this shortcoming of MCE training algorithm in joint feature extraction and classification tasks. GMCE training algorithm achieves the classification objective in two steps. The first step is a initialization step, which searches for a suitable initialization for MCE training. The second step conducts MCE training.

- *Reduced-Dimensional SVM*, Chapter 8: SVM is a recently developed pattern classification algorithm with non-linear formulation. However, it overpasses the feature extraction step and uses parameter vectors directly as input. This causes a number of problems to pattern recognition systems, such as high system complexity and low efficiency. This chapter proposes a reduced-dimensional SVM (RDSVM) to adopt feature extraction into SVM. The proposed RDSVM algorithm has a two-layer structure. The first layer conducts feature extraction and provides a discriminated and/or reduced-dimensional feature space for the second layer. The second layer conducts SVM training in this feature space.

1.3 Thesis Organization

This thesis is mainly concerned with feature extraction and dimensionality reduction algorithms for pattern recognition. It is organized as follows:

Chapter 1: This chapter gives a brief introduction to the main purpose, structure and contributions of this thesis.

Chapter 2: This chapter gives a brief introduction to the fundamentals of pattern recognition. It includes the formulation of pattern recognition problems, definitions of some basic concepts, approaches to design feature extractors and classifiers, integrated pattern recognition systems and feature dimensionality problems in pattern recognition.

Chapter 3: This chapter discusses two popular independent feature extraction algorithms — LDA and PCA. In the following chapters, they are used as the references to evaluate integrated feature extraction and classification algorithms.

Chapter 4: This chapter discusses the framework of MCE training algorithm and proposes an alternative form of MCE training algorithm, which uses a ratio model of misclassification measure. The performance of alternative MCE training algorithm is compared to those of conventional MCE training algorithm, LDA and PCA.

Chapter 5: This chapter proposes the use of MCE training algorithm for joint feature extraction and classification tasks. Corresponding formulation is derived. An experiment is carried out on two small databases (Deterding Vowel database and D. German's GLASS database). In the experiment, the performance of MCE training algorithm is compared to those of LDA and PCA.

Chapter 6: This chapter proposes a generalized MCE (GMCE) training algorithm. GMCE has a general searching step to search for a suitable initialization of transformation matrix before MCE training process. The criterion for general searching process is investigated.

Chapter 7: This chapter introduces the formulation of SVM. SVM is employed on vowel classification tasks based on Deterding Vowel database. Its performance is compared to those of MCE and GMCE training algorithms.

Chapter 8: This chapter discusses the shortcomings of SVM and proposes a RDSVM algorithm to adopt feature extraction into SVM. The proposed RDSVM is tested on Deterding Vowel database and its performance is analysed.

Chapter 9: This chapter first introduces the database used in our vowel classification experiments — TIMIT database and the selection of vowels. The setup of the experiments is also introduced. The recognition results of LDA, PCA, MCE, GMCE, SVM and RDSVM are then shown and analysed.

Chapter 10: This chapter concludes the whole thesis and summarizes the conclusion obtained in each chapter.

1.4 Publications Resulting from Research for This Thesis

This thesis has in many parts been shaped by colleagues' and reviewers' comments regarding many of the publications listed below. It has also been shaped by the comments and suggestions resulting from conference presentations.

1. X.Wang and K.Paliwal, "Feature extraction and dimensionality reduction algorithms and their application in vowel recognition", *Pattern Recognition*, Accepted in December 2002.

2. X.Wang and K.Paliwal, "A modified minimum classification error training algorithm for dimensionality reduction", *Journal of VLSI Signal Processing Systems*, vol 32, pp. 19-28, April 2002.

3. X.Wang and K.Paliwal, "Discriminative learning and informative learning in pattern recognition", *9th International Conference on Neural Information Processing*, Singapore, November 2002.

4. X.Wang and K.Paliwal, "Feature extraction for integrated pattern recognition systems", *Fourth Workshop on Signal Processing and Applications*, Brisbane, Australia, December 2002.

5. X.Wang and K.Paliwal, "Generalized minimum classification error training algorithm for dimensionality reduction", *Microelectronic Engineering Research Conference 2001*, Brisbane, Australia, 2001.

6. X.Wang and K.Paliwal, "Using minimum classification error training in dimensionality reduction", *Proceedings of the 2000 IEEE Workshop on Neural Networks for Signal Processing X*, pp. 338-345, Sydney, 2000.

7. X.Wang, K.Paliwal and J. Chen, "Extension of minimum classification error training algorithm", *Microelectronic Engineering Research Conference 1999*, Brisbane, Australia, 1999.

Chapter 2

Fundamentals of Pattern Recognition

2.1 Data Flow in Pattern Recognition Systems

Data flow in a typical pattern recognition system is shown in Figure 2.1. The collected information of an object, $x(t)$, is firstly processed by a parameter extractor. Information relevant to pattern classification is extracted from $x(t)$ in the form of a p-dimensional parameter vector x. x is then transformed to a feature vector y, which has a dimensionality m ($m \leq p$), by a feature extractor. The purpose of feature extraction is to make the input data more suitable for pattern classifier and/or reduce the dimensionality of the input data vectors. Feature vector y is assigned to one of the K classes, $\Omega_1, \Omega_2, \cdots, \Omega_K$, by the classifier based on a certain type of classification criteria.

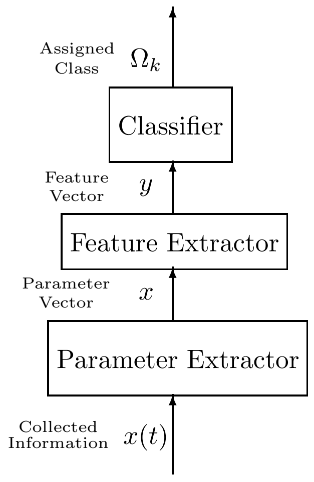


Figure 2.1: Data flow in a typical pattern recognition system.

Designing a pattern recognition system, therefore, includes three parts: the design of parameter extractor, feature extractor and classifier. This thesis concentrates on the last two parts: the design of feature extractor and classifier.

2.2 Definition of Some Basic Concepts

2.2.1 Pattern

Pattern is a quantitative or structural description of an object or some other entity of interest[40]. It is usually arranged in the form of a feature vector as:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

where $x_1, x_2, \ldots, x_n$ are the features. Depending on the measurements of an object, features in a pattern can be either discrete numbers or real continuous values. The requirement on features is that the features can reflect the characteristics of desired objects and differ from those of other objects to the largest extent.

2.2.2 Class

Class or *pattern class* is a set of patterns that share some common properties. The feature vectors of the same type of objects will naturally form one set. Due to the diversity of the objects, the patterns extracted from the same type of objects are seldom identical. This can be interpreted as clusters of points in a n-dimensional space, which are called distributions of classes. Figure 2.2 shows an example of distributions of Fisher's iris data in a two-dimensional space, in which only two out of four dimensions, i.e., petal length and petal width, are used. Since the purpose of pattern recognition is to classify these patterns, the distributions of classes are desired to be separable and not empty. Suppose we have K classes, in a mathematical form, the requirement is:

$$\Omega_k \neq \phi \quad k = 1, \ldots, K; \qquad \Omega_k \bigcap \Omega_l = \phi \quad k \neq l \in \{1, \ldots, K\} \tag{2.1}$$

2.2.3 Classification Criterion

Classification Criterion is also called *decision rule.* The most widely used classification criteria are *distance, Bayes decision rule* and *likelihood.* A brief summary of these criteria is given in the following:

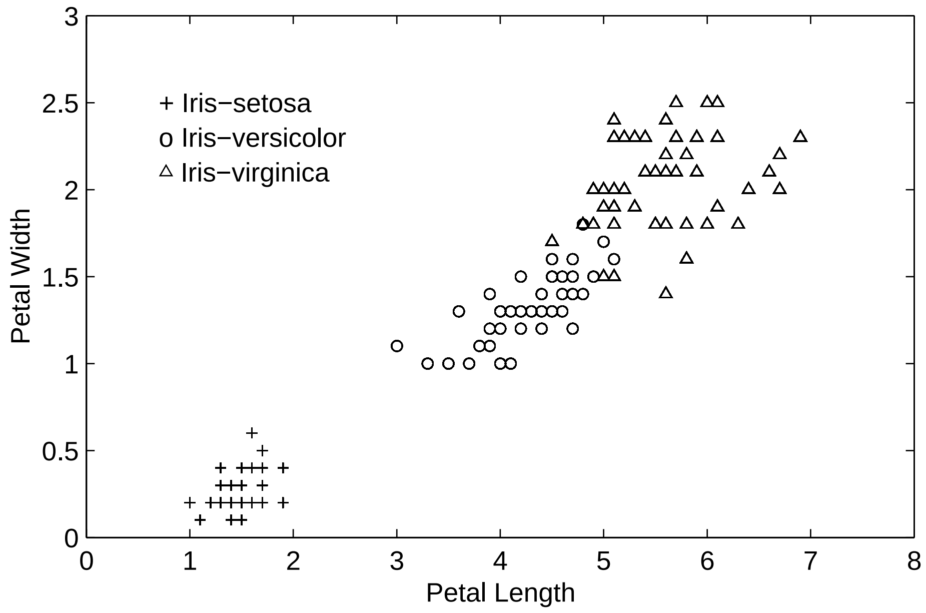


Figure 2.2: Distribution of Fisher's iris data in a two-dimensional space.

- Distance criterion is the simplest and most direct criterion. The basic idea of distance classification criterion is that a data is classified to the class that is closest to it. Euclidean distance and Mahalanobis distance are the two most common forms. Suppose we have K classes, let (μ_i, Σ_i) be the known parameter set of class i, where μ_i is the reference vector of class i, Σ_i is the covariance. The square form of Euclidean distance of an observation vector x from class i is:

$$d_i(x) = \|x - \mu_i\|^2 \tag{2.2}$$

  The square form of Mahalanobis distance of x from class i is:

$$d_i(x) = (x - \mu_i)^T \Sigma_i^{-1} (x - \mu_i) \tag{2.3}$$

  Euclidean distance is in fact a special case of Mahalanobis distance.

- Bayes decision rule is based on the assumption that classification problems are posed in probabilistic terms and all of the relevant probabilities are known. It will assign an observation vector to the class that has the largest *a posteriori* probability $p(\Omega_j|x)$. Suppose we have K classes, $\Omega_1, \Omega_2, \ldots, \Omega_K$, and also we have known the *a priori* probability of each class $P(\Omega_i), i = 1, 2 \ldots, K$ and the conditional probability density $p(x|\Omega_i), i = 1, 2, \ldots, K$, the *a posteriori* probability can be calculated by *Bayes rule*:

$$p(\Omega_j|x) = \frac{p(x|\Omega_j)P(\Omega_j)}{p(x)} = \frac{p(x|\Omega_j)P(\Omega_j)}{\sum_{i=1}^{J} p(x|\Omega_i)P(\Omega_i)} \tag{2.4}$$

- Likelihood criterion is a special case of Bayes classification criterion. It assumes that all of the *a priori* probabilities $P(\Omega_i)$ are equal and the distributions of classes are normal, i.e., $x \sim N(\mu_i, \Sigma_i),\ i = 1, 2 \ldots, K$. Then we have:

$$p(\Omega_j|x) \ = \ p(x|\Omega_i) \tag{2.5}$$

and

$$p(x|\Omega_i) = \frac{1}{|2\pi\Sigma_i|^{1/2}} e^{-\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i)} \tag{2.6}$$

If the parameters of a class are known, likelihood is in fact the *PDF* (probability density function) of the class. Using likelihood can greatly simplify the calculation arisen by using Bayes decision rule. A further logarithm of likelihood is usually taken to make the calculation simpler. The log-likelihood has the form as follows:

$$P_i(x) = -\frac{1}{2} ln|\Sigma_i| - \frac{n}{2} ln 2\pi - \frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i) \tag{2.7}$$

2.2.4 Classifier

A *classifier* first creates a series of functions $g_i(x, \Lambda_i), i = 1, \ldots, K$ as the input-output functions, which are called *discriminant functions.* In a discriminant function $g(x, \Lambda)$, x is the input vector and Λ is the parameter set of the class. Each discriminant function will output a value. Based on these values, the classifier then assigns x to one of the classes following the decision rule:

$$x \in Class\ i \quad if\ g_i(x, \Lambda_i) = \max_{for\, all\, j \in K} g_j(x, \Lambda_j) \tag{2.8}$$

Based on the classification criterion used in the discriminant functions, classifiers can be grouped into *Bayesian classifier, Likelihood classifier* and *distance classifier.* Figure 2.3 shows discriminant functions of these classifiers in a two-class problem.

2.3 Approaches to Designing Feature Extractors

The main task of feature extractor is to select or combine the features that preserve most of the information and remove the redundant components in order to improve the efficiency of the subsequent classifiers without degrading their performances. Feature extraction methods can be grouped into two categories: feature selection method and feature extraction method [62, 75].

2.3.1 Feature Selection Method

Feature selection method generates feature vectors by removing one measurement at a time and maintaining the highest value in some performance

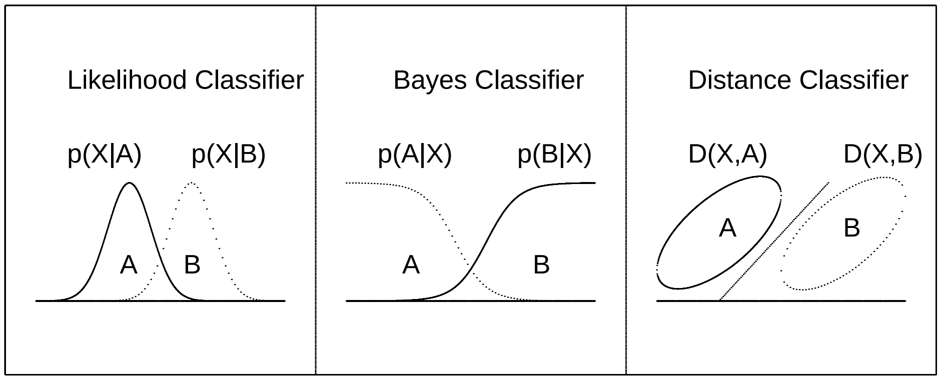


Figure 2.3: Discriminant functions of Bayesian, likelihood and distance classifiers.

indices. Measurements are removed until there is an unacceptable degradation in system performance. Karhunen-Loéve (K-L) expansion [36] and F-ratio [75] are the two major feature selection algorithms.

K-L Expansion

In *K-L* expansion, the input vector x is assumed to be a zero-mean vector. Re-write x in an orthogonal expended form:

$$x = Qc \tag{2.9}$$

where $Q = (q1, \ldots, q_n)$ is an orthogonal matrix formed by n normalized orthogonal basis in the observation space, c is a set of random uncorrelated coefficients. These coefficients c can be calculated by re-arranging equation 2.9 as follows:

$$c = Q^T x \tag{2.10}$$

If we define the covariance matrix of x as:

$$R = E[xx^T] \tag{2.11}$$

Then we have:

$$R = E[Qc(Qc)^T] = QE[cc^T]Q^T \tag{2.12}$$

Since c is uncorrelated, the expectation of cc^T will be diagonal and let it be Λ, so that

$$R = Q\Lambda Q^T \tag{2.13}$$

This means that the diagonal elements of Λ are the eigenvalues of R and Q can be formed by the normalized eigenvectors of R. We can rewrite Q as:

$$Q = \{q_1, \cdots, q_l, q_{l+1}, \cdots, q_n\} = \{Q^{'}Q^{''}\} \tag{2.14}$$

where $q_1, \cdots, q_l$ are the eigenvectors corresponding to first l largest eigenvalues. By discarding $q_{l+1}, \cdots, q_n$, c is formed again to represent x in a lower dimensional space with $Q^{'}$, $c^{'} = Q^{'T}x$. In this newly formed space, most of the variances of x are retained. The error, $e = x - Qc$, due to the selection of first l features can be minimized by the least mean squared method.

F-ratio Method

F-ratio approach selects the features in a different way to K-L method. It selects the features by finding the largest ratio of *between-class* covariance and *within-class* covariance. Suppose we have K classes, $\mu_1, \ldots, \mu_K$ represent means of each classes, which are calculated by:

$$\mu_i = \frac{1}{n_i}\sum_{j=1}^{n_i} x_{ij} \tag{2.15}$$

where n_j is the number of data in class j. Let μ be the overall mean:

$$\mu = \frac{1}{n}\sum_{i=1}^{K}\sum_{j=1}^{n_i} x_{ij} \tag{2.16}$$

where $n = \sum_{i=1}^{K} n_i$ is the total number of data. Then within-class covariance is defined as:

$$S_W = \frac{1}{n}\sum_{i=1}^{K}\sum_{j=1}^{n_i}(x_{ij} - \mu_i)(x_{ij} - \mu_i)^T \tag{2.17}$$

Between-class covariance is defined as:

$$S_B = \frac{1}{K}\sum_{i=1}^{K}(\mu_i - \mu)(\mu_i - \mu)^T \tag{2.18}$$

and *F-ratio* is defined as:

$$F\text{--}ratio = \frac{S_B}{S_W} \tag{2.19}$$

Then the features that can keep the ratio largest will be kept and the others will be discarded.

2.3.2 Feature Extraction Method

Feature extraction method generates feature vectors by projecting parameter vectors onto a feature space through a linear transformation $T_{p\times m}$, $p \geq m$:

$$y = T^T x \tag{2.20}$$

where y is feature vector and x is parameter vector. In independent feature extraction algorithms, transformation T is optimized separately from the class models with different criterion, while in integrated feature extraction and classification algorithms, T is optimized synchronously with class models.

LDA and PCA are the two popular independent feature extraction algorithms. They optimize the transformation T with different intentions. LDA optimizes T by maximizing the ratio of *between*-class variation and *within*-class variation. PCA obtains T by searching for the directions that have the largest variations. Therefore LDA and PCA project parameter vectors along different directions. Figure 2.4 shows the difference between the projecting directions of LDA and PCA when projecting the parameter vectors from a two-dimensional parametric space onto a one-dimensional feature space. A detailed discussion of LDA and PCA will be given in Chapter 3.

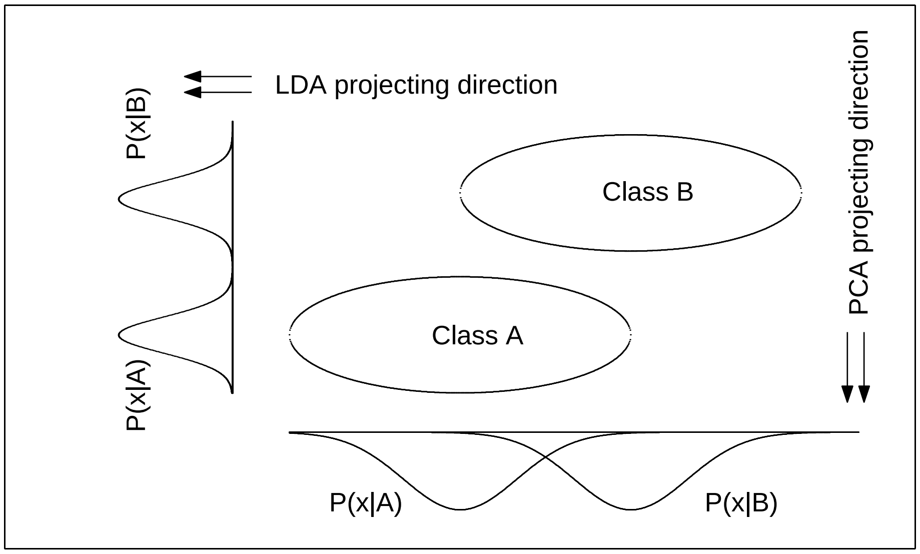


Figure 2.4: A comparison of the directions in which LDA and PCA project data from a two-dimensional space onto a one-dimensional space.

In this thesis, MCE training algorithm is applied to integrated feature extraction and classification systems due to its flexible framework. Corresponding formulation and investigation on MCE's performance in integrated feature extraction and classification tasks will be given in Chapter 4 and 5.

2.4 Approaches to Designing Classifiers

2.4.1 Procedure of Training Classifiers

When designing a classifier, we usually have no knowledge about the classes, especially the information of class distributions. What we have is only a bunch of data obtained from observations. Classifiers have to be built up based on these observed data. Normal process of building up a classifier includes the initialization of the classifier, estimation of error and adjustment of the parameters in the classifier, as shown in Figure 2.5. This is often called a training process. Since the classifiers assign observations to classes based

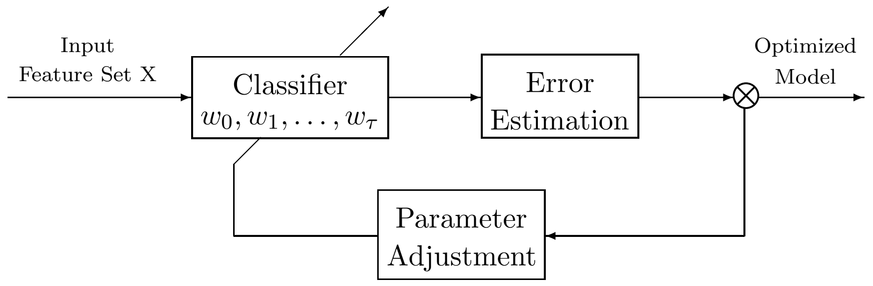


Figure 2.5: Training process of a classifier.

on the output of discriminant functions, the success of classifiers is highly dependent on the selection of discriminant functions. Unfortunately, it is often very difficult to find a suitable parametric form of discriminant functions for classification. So in some approaches, an unstructured estimation of discriminant functions is used. These methods are called *non-parametric training*. *Non-parametric training* approaches, however, in some cases, can be very complex and require a large number of samples to give accurate results. Thus this leads to the consideration of simpler procedures for designing classifiers. In particular, the mathematical forms of discriminant functions are pre-specified and a small set of parameters is left to be determined. This type of approaches is called *parametric training*. The following subsections will give a brief introduction to both *non-parametric training* and *parametric training*.

2.4.2 Non-parametric Training

The most fundamental technique of non-parametric approaches situates on the fact that the probability P of an observation vector x that falls into a region $\mathcal{R}$ is given by

$$P = \int_{\mathcal{R}} p(x)dx \tag{2.21}$$

Thus the probability density function of x can be estimated by estimating the probability P. Suppose that n samples $x_1, \ldots, x_n$ are independently drawn by the probability density $p(x)$. Obviously, a good estimate of the probability P that k of n samples fall into $\mathcal{R}$ is [12]:

$$P = \frac{k}{n} \tag{2.22}$$

If we assume that $p(x)$ is continuous and the region $\mathcal{R}$ is so small that $p(x)$ does not vary significantly within it, then the right side of equation (2.21) can be re-written as:

$$\int_{\mathcal{R}} p(x)dx \approx p(x)V \tag{2.23}$$

where V is the volume of $\mathcal{R}$. Combining (2.21), (2.22) and (2.23), the estimate of $p(x)$ is obtained as follows:

$$\hat{p}(x) \approx \frac{k/n}{V} \tag{2.24}$$

If we fix V and increase n, the ratio k/n will converge as desired. But what we have obtained is an estimate of the average value, P/V, of $p(x)$ over $\mathcal{R}$. If an estimate of density at x, $p(x)$, is desired rather than an averaged estimate of $p(x)$ over a region, we have to let V approach zero. However, if we fix n and let $V \rightarrow 0$, the region will eventually become so small that $\hat{p}(x)$ will approach zero and become useless. Since in practice n is always limited, volume V can not be be arbitrarily small. If $\hat{p}(x)$ is to converge to $p(x)$, three conditions have to be satisfied:

$$\begin{array}{ll} (1) & \lim_{n\rightarrow\infty} V = 0 \\ (2) & \lim_{n\rightarrow\infty} k = \infty \\ (3) & \lim_{n\rightarrow\infty} k/n = 0. \end{array} \tag{2.25}$$

The first condition ensures that the region average will converge to $p(x)$; the second condition ensures that the ratio k/n will converge to P and the last condition ensures that estimate in equation (2.24) converges. There are two common approaches to obtain V and k that satisfy the conditions. One is called *Parzen estimate*, which fixes V and obtains the value of k by counting the number of training data falling in V. The other approach is called *k-nearest neighbour estimate* or *k-NN*, which fixes k and evaluate V by finding the volume of the region which captures the k nearest neighbours of x [50].

Parzen estimate

In *Parzen estimate*, an initial region around the data point x, $\mathcal{R}_x$, is set up and shrunk by specifying the volume V as a function of n. The region is usually assumed to be a d-dimensional hypercube and the length of each side be r, then the volume is given by:

$$V = r^d \tag{2.26}$$

In some cases, $\mathcal{R}_x$ is assumed to be a hypersphere. Given the radium r, the volume is then:

$$V = \int_{L(x)} dy = \frac{\pi^{d/2}}{\Gamma(\frac{d+2}{2})} |\Sigma|^{1/2} r^d \tag{2.27}$$

where Σ is the covariance of n samples [36]. The number of the samples that fall into $\mathcal{R}_x$, k, is often given by a kernel function $k(x - x_i)$, which is set up under the condition $\int k(x)dx = 1$. Then the estimate is obtained by:

$$\hat{p}(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{V}k(x - x_i) \tag{2.28}$$

The selection of kernel function is usually limited to either a uniform or a normal kernel in high dimensional space. For a uniform kernel,

$$K(y) = \begin{cases} 1 & \text{if } y \text{ inside } \mathcal{R}_x \\ 0 & \text{if } y \text{ outside } \mathcal{R}_x \end{cases} \tag{2.29}$$

For a normal kernel,

$$K(y) = e^{-\frac{1}{2}(y-x)^T r\Sigma(y-x)} \tag{2.30}$$

Clearly, the choice of r, which decides volume V, has a major effect on $p(x)$. r can be optimized by minimizing the *mean-square error* between $\hat{p}(x)$ and $p(x)$ with respect to r, which is represented as follows:

$$\begin{aligned} &MSE\{\hat{p}(x)\} = E\{[\hat{p}(x) - p(x)]^2\} \\ &\nabla(MSE\{\hat{p}(x)\}) = 0 \end{aligned} \tag{2.31}$$

k-NN estimate

One of the problems encountered in *Parzen estimate* approach is that the results are very sensitive to the initial choice of V(or r). Furthermore, it may be the case that a volume that works well for one value of x might be totally unsuitable elsewhere. One remedy for these problems is *k-NN* method. In *k-NN*, V becomes a function of the data and is extended until k samples are captured [12, 36]. These k samples are the k nearest neighbours of x.

Suppose we have N classes, for *k-NN* rule, each class $\Omega_i, i = 1, \ldots, N$ is represented by a set of known points, $z_j^{(i)}, j = 1, \ldots, n_i$, in the feature space. For each observation vector x, a *k-NN* list $d(x, z_j^{(i)})$ is made for all classes, where $d(x, z_j^{(i)})$ usually uses the distance of observation x to the jth point in Ω_i, instead of the probability for the sake of simplicity of calculation. The distance measure is defined in terms of a metric $d(x, y)$. One of the widely used metrics is the Minowski metric,

$$d(x, y) = \|x - y\|_p = \left(\sum_{i=1}^{n}|x_i - y_i|^p\right)^{1/p} \tag{2.32}$$

When $p = 2$ this metric becomes the well-known Euclidean distance.

Among the non-linear metrics, quadratic metrics are of most practical interest and defined in the following equation:

$$d(x, y) = (x - y)^t A(x - y) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}(x_i - y_i)(x_j - y_j) \tag{2.33}$$

where A is an $n \times n$ positive-definite real symmetric matrix. A special case of quadratic metrics is the Mahalanobis distance.

These lists of distance measures are used to define the distance of x to Ω_i. There are a number of *k-NN* rules to define the distance. The simplest one defines the distance as the smallest distance in Ω_i:

$$D(x, \Omega_i) = \arg\min \ d(x, z_j^{(i)}), \quad j = 1, \ldots, n_i \tag{2.34}$$

Another popular *k-NN* rule defines the distance as the average distance of x to the K nearest neighbours in Ω_i:

$$D(x, \Omega_i) = \frac{1}{k} \sum_{j=1}^{K} d(x, z_j^{(i)}) \tag{2.35}$$

Still another *k-NN* rule defines the distance based on a majority decision, in which the decision rule first finds k nearest neighbours within all the nearest neighbour lists, then the distance to class Ω_i is represented by the number of points appeared in the k nearest neighbour list:

$$D(x, \Omega_i) = \frac{1}{l_i} \tag{2.36}$$

where l_i is the number of points in the k nearest neighbour list which are associated with class Ω_i.

The observation x is then classified following the decision rule:

$$x \in \Omega_i \quad if \ D(x, \Omega_i) = \arg\min_{for \ all \ n \neq i} \ D(x, \Omega_n) \tag{2.37}$$

2.4.3 Parametric Training

In typical pattern classification problems, most of the difficulties arise from the estimation of class-conditional densities. If the conditional densities are parameterized by our general knowledge about the problems, the severity of these difficulties can be reduced significantly. For example, if we assume that $p(x|\Omega_i)$ is a Gaussian density with mean μ_i and covariance Σ_i, the problem of estimating a density function $p(x|\Omega_i)$ is simplified to that of estimating the parameters μ_i and Σ_i. The approaches that use this strategy are called *parametric training* approaches or *supervised learning* approaches. *Distance estimate*, *Maximum Likelihood estimate*, and *Bayesian estimate* are three common parametric training methods. The following sub-sections will give detailed discussion on them.

Distance Estimate

In distance estimate, the data is assumed to distribute in many hyperspheres and each hypersphere can be represented by a reference point μ, which is a p-dimensional vector, and a dispersion matrix Σ of the hypersphere in different directions. Σ is required to be an positive definite, symmetric and non-singular matrix. Thus the problem is simplified to find μ and Σ.

Suppose we have K classes and K sets of samples for them. The distance of a sample x to class Ω_i is defined as the distance to the reference vector of the class:

$$d(x, \Omega_i) = (x - \mu^{(i)})^T \Sigma^{(i)} (x - \mu^{(i)}) \tag{2.38}$$

Mahalanobis distance is a special case of this definition when $\Sigma^{(i)}$ is the covariance matrix of Ω_i. The total distance in a class Ω_i is defined as the sum of the distances of the samples to Ω_i:

$$D(\Omega_i) = \sum_{j=1}^{n_i} d(x_j^{(i)}, \Omega_i) \tag{2.39}$$

where n_i is the number of samples in class Ω_i. Obviously, the distance of a data to its desired class should be the smallest among the distances of it to all the classes. Therefore, the problem of estimating the parameters becomes how to choose $\{\hat{\mu}^{(i)}, \hat{\Sigma}^{(i)}\}$ so that the total distance in Ω_i is the minimum.

Gradient descent method is often preferred to minimize $D(\Omega_i)$. However, the total distance defined in (2.39) is still not suitable for differentiation because $D(\Omega_i)$ is not continuous over Ω_i. So it is usually smoothed by a monotonical differentiable function so that the normal gradient descent method can be employed to obtain the estimated $\{\hat{\mu}^{(i)}, \hat{\Sigma}^{(i)}\}$. There is a number of choices of smooth functions. Sigmoid function and exponential function are two popular forms among them:

- *a) Sigmoid function*

  $$L[D(\Omega_i)] = \frac{1}{1 + e^{-\xi(D(\Omega_i) + \alpha)}} \tag{2.40}$$

  where $0 < \xi < 1$ and α is a constant and usually set to 0.

- *b) Exponential function*

  $$L[D(\Omega_i)] = \begin{cases} (D(\Omega_i))^{\xi}, & D(\Omega_i) > 0 \\ 0, & D(\Omega_i) \le 0 \end{cases} \tag{2.41}$$

  where $\xi > 0$ and $\xi \to 0$.

The samples from different classes are assumed to be uncorrelated so that we can optimize each class independently. Let $\mu^{(i)}$ be a p-dimensional vector, $\Sigma^{(i)}$ be a $p \times p$ matrix and the smooth function be sigmoid function. Then the gradient of $L[D(\Omega_i)]$ with respect to $\mu^{(i)}$ and $\Sigma^{(i)}$ is calculated as follows:

$$\nabla_{\mu^{(i)},\Sigma^{(i)}} L = \begin{bmatrix} \xi L(1-L)\frac{\partial D(\Omega_i)}{\partial \mu_1^{(i)}} \\ \vdots \\ \xi L(1-L)\frac{\partial D(\Omega_i)}{\partial \mu_p^{(i)}} \\ , \\ \xi L(1-L)\frac{\partial D(\Omega_i)}{\partial \sigma_{11}^{(i)}} \\ \vdots \\ \xi L(1-L)\frac{\partial D(\Omega_i)}{\partial \sigma_{pp}^{(i)}} \end{bmatrix} \tag{2.42}$$

The optimized parameters are obtained by setting $\nabla_{\mu^{(i)},\Sigma^{(i)}} L = 0$. The procedure of distance estimate is shown in Figure 2.6.

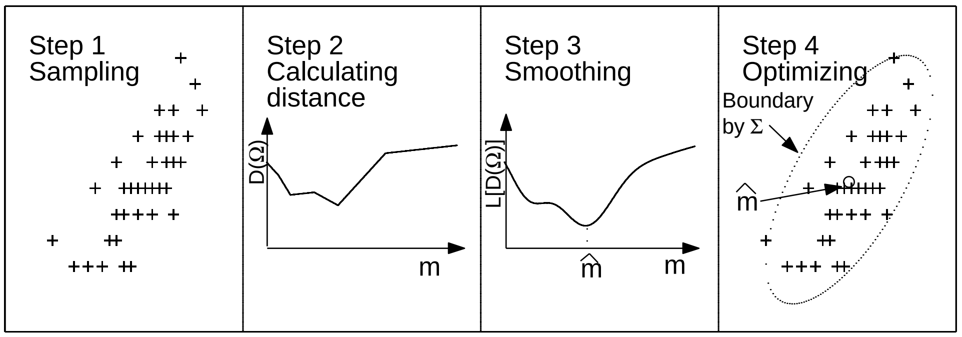


Figure 2.6: Procedure of distance estimate.

Maximum Likelihood Estimate

Suppose we have K classes and a set of samples, $\mathcal{X}_i$, for each class. The samples are drawn identically and independently. It is assumed that $p(x|\Omega_i)$ has a known parametric form and is determined uniquely by the parameter set θ_i. We use $p(x|\theta_i)$ to represent $p(x|\Omega_i, \theta_i)$ to show the dependence of $p(x|\Omega_i)$ on θ_i. Then the problem is changed to use the samples to obtain an appropriate estimate for the unknown parametric sets $\theta_1, \cdots, \theta_K$.

It is plausible to assume that samples from different sets are uncorrelated so that we can work with each class separately. For a single class, suppose that the sample set, $\mathcal{X}$, contains n samples, $\mathcal{X} = x_1, \ldots, x_n$. Then, since the samples are drawn independently, the conditional probability,

$$p(\mathcal{X}|\theta) = \prod_{j=1}^{n} p(x_j|\theta) \tag{2.43}$$

is a function of θ and called the likelihood of θ with respect to the set of samples. The maximum likelihood estimate of θ is to value $\hat{\theta}$ that maximizes $p(\mathcal{X}|\theta)$(as shown in Figure 2.7) Likelihood is usually embedded into a monotonically increasing or decreasing function so that $\hat{\theta}$ can be found by standard differential methods. Logarithm of the likelihood is the most common form used.

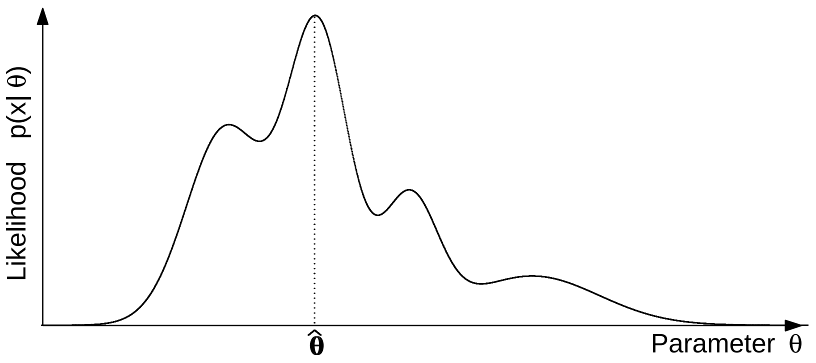


Figure 2.7: Maximum likelihood estimate for a parameter θ.

Let $\theta = (\theta_1, \ldots, \theta_p)^t$ be the p-dimension vector, let ∇_θ be the gradient operator:

$$\nabla_\theta = \begin{bmatrix} \frac{\partial}{\partial\theta_1} \\ \vdots \\ \frac{\partial}{\partial\theta_p} \end{bmatrix} \tag{2.44}$$

The log-likelihood is defined as follows:

$$\begin{aligned} l(\theta) &= log\ p(\mathcal{X}|\theta) \\ &= \textstyle\sum_{j=1}^{n} log\ p(x_j|\theta) \end{aligned} \tag{2.45}$$

The gradient of log-likelihood with respect to θ is:

$$\begin{aligned} \nabla_\theta l &= \textstyle\sum_{j=1}^{n} \nabla_\theta log\ p(x_j|\theta) \\ &= \begin{bmatrix} \sum_{j=1}^{n} \frac{\partial log\ p(x_j|\theta)}{\partial\theta_1} \\ \vdots \\ \sum_{j=1}^{n} \frac{\partial log\ p(x_j|\theta)}{\partial\theta_p} \end{bmatrix} \end{aligned} \tag{2.46}$$

The maximum likelihood estimate for θ can be easily obtained from the set of p equations $\nabla_\theta l = 0$.

Bayesian Estimate

The key problem of Bayes estimate is the calculation of *a posteriori* probabilities $P(\Omega_i|x)$. Bayes rule allows us to compute these probabilities from the *a priori* probabilities $P(\Omega_i)$ and the class-conditional probability densities $P(x|\Omega_i)$. However, both $P(\Omega_i)$ and $P(\Omega_i|x)$ are unknown. So we have to compute them by using the information that resides in the samples. Let $\mathcal{X} = \{x_1, \ldots, x_K\}$ be the set of samples from K classes. If our goal is to compute *a posteriori* probabilities $P(\Omega_i|x, \mathcal{X})$ from the samples, then:

$$p(\Omega_j|x, \mathcal{X}) = \frac{p(x|\Omega_j, \mathcal{X})P(\Omega_j|\mathcal{X})}{\sum_{i=1}^{K} p(x|\Omega_i, \mathcal{X})P(\Omega_i|\mathcal{X})} \tag{2.47}$$

Without the loss of generality, we assume that the true values of the *a priori* probabilities are known, i.e., $P(\Omega_i|\mathcal{X}) = P(\Omega_i), i = 1, \ldots, K$. Then $P(\Omega_i|x, \mathcal{X})$ can be re-written as:

$$p(\Omega_j|x, \mathcal{X}) = \frac{p(x|\Omega_j, \mathcal{X})P(\Omega_j)}{\sum_{i=1}^{K} p(x|\Omega_i, \mathcal{X})P(\Omega_i)} \tag{2.48}$$

Once again we assume that the samples in Ω_i have no influence on $P(\Omega_j|x)$ if $j \neq i$. This allows us to work with each class separately. We can simplify the notation from $p(x|\Omega_j, \mathcal{X})$ to $p(x|\mathcal{X})$ to remove the class distinction. Computing $p(x|\mathcal{X})$ forms the central problem of Bayesian learning.

Let θ be the parameter vector, which is unknown. Then $p(x|\mathcal{X})$ can be computed by integrating the joint density $p(x, \theta|\mathcal{X})$ over θ:

$$\begin{aligned} p(x|\mathcal{X}) &= \int p(x, \theta|\mathcal{X})d\theta \\ &= \int p(x|\theta)p(\theta|\mathcal{X})d\theta \end{aligned} \tag{2.49}$$

Equation (2.49) links $p(x|\mathcal{X})$ to the *a posteriori* probability $p(\theta|\mathcal{X})$. If $p(\theta|\mathcal{X})$ peaks very sharply above some value $\hat{\theta}$, we obtain $p(x|\mathcal{X}) \approx p(x|\hat{\theta})$. $p(\theta|\mathcal{X})$ can be calculated by Bayes rule:

$$p(\theta|\mathcal{X}) = \frac{p(\mathcal{X}|\theta)p(\theta)}{\int p(\mathcal{X}|\theta)p(\theta)d\theta} \tag{2.50}$$

By the independent assumption:

$$p(\mathcal{X}|\theta) = \prod_{i=1}^{n} p(x_i|\theta) \tag{2.51}$$

where n is the number of samples in $\mathcal{X} = \{x_1, \ldots, x_n\}$. Equation (2.51) can be re-written in a recursive form:

$$p(\mathcal{X}^s|\theta) = p(x_n|\theta)p(\mathcal{X}^{s-1}|\theta) \tag{2.52}$$

where s represents sth iteration. Substitute Eq. (2.52) into Eq. (2.50) with the understanding that $p(\theta|\mathcal{X}^0) = p(\theta)$, we obtain the recursive form of $p(\theta|\mathcal{X})$:

$$p(\theta|\mathcal{X}^s) = \frac{p(x_n|\theta)p(\theta|\mathcal{X}^{s-1})}{\int p(x_n|\theta)p(\theta|\mathcal{X}^{s-1})d\theta} \tag{2.53}$$

Repeated use of Eq. (2.53), $p(\theta|\mathcal{X}^s)$ will eventually converge to a Dirac delta function centered about the true values of θ. Thus from Eq. (2.49) we can see that $p(x|\mathcal{X}^s)$ will converge to $p(x)$. This procedure is called *Bayesian learning*.

2.5 Non-linear Classifier

The decision boundaries generated by conventional classifiers, such as distance, likelihood and Bayesian classifiers, are linear boundaries, as shown in Figure 2.8. The limitation of linear decision boundaries is that it lacks computational flexibility and is not suitable for handling the classes with complex distributions.

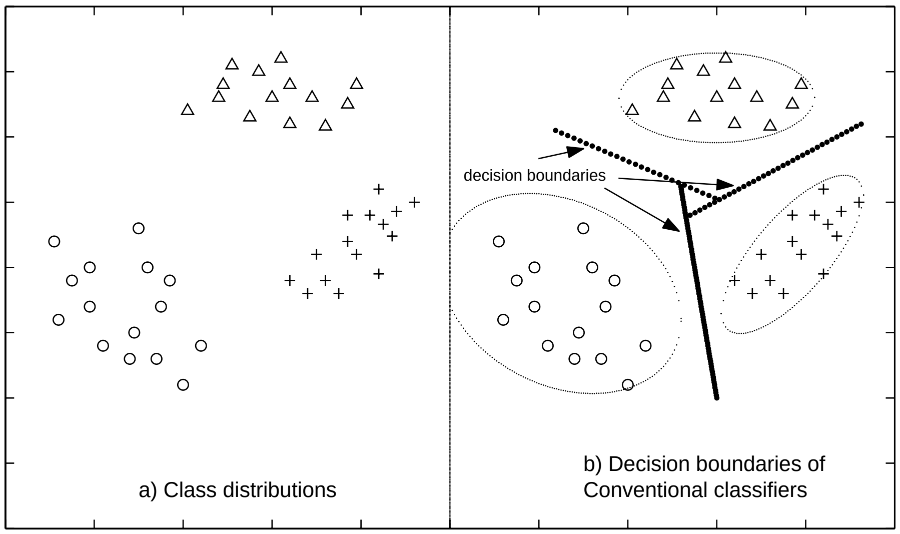


Figure 2.8: Linear decision boundaries of conventional classifiers.

Non-linear classifier is a recently developed classification technique. It first projects parameter vectors onto a higher-dimensional feature space through a non-linear kernel function. Kernel function is often defined as the dot product between a parameter vector and a reference vector:

$$k(x, y) = (x \cdot y) \tag{2.54}$$

In this higher-dimensional feature space, non-linear class boundaries may become linear, as shown in Figure 2.9. Then the decision plane is pursuit in the feature space. The projection of decision plane in the parameter space is a non-linear decision boundary. Therefore, non-linear classifiers have the advantage of handling the classes with complex distributions.

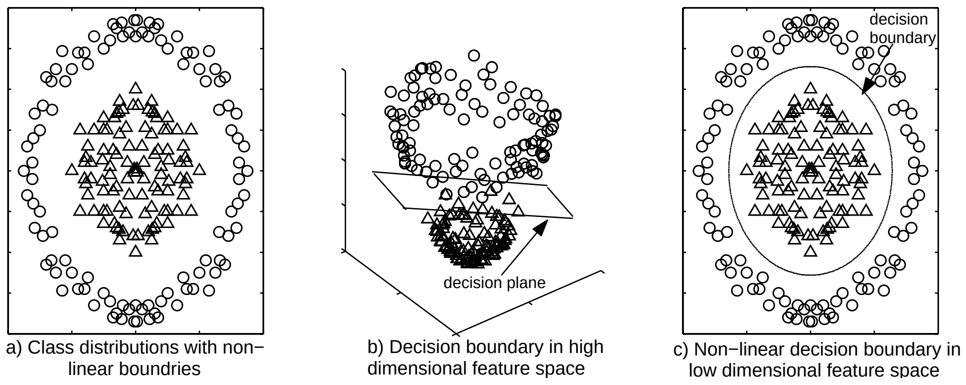


Figure 2.9: Non-linear decision boundaries of non-linear classifiers.

Kernel Discriminant Analysis (KDA) [82, 68], Kernel Principal Component Analysis (KPCA) [69] and SVM [17, 18, 72, 73, 85, 87, 89, 99] are the three major non-linear classification algorithms. In this thesis, we will concentrate on SVM. The formulation of SVM and corresponding experiments are introduced in the following chapters.

2.6 The Curse of Dimensionality

2.6.1 Problems caused by High Dimensionality

Computational efficiency is an important problem to pattern recognition systems, especially the real-time systems. The amount of computations required for pattern recognition and the amount of data required for training systems grows exponentially with the increase of the dimensionality of the feature vectors. This is what Bellman called “the curse of dimensionality” [7, 60] .

In practical pattern recognition applications, it is often the way to consider adding new features if the performance of a pattern recognition system is inadequate, because it is reasonable to believe that the Bayes risk can not possibly be increased by adding new features. Unfortunately, it has been frequently observed in practice that the inclusion of new additional features sometimes leads to poorer rather than better performance[16, 107]. There are many reasons for this apparent paradox. Firstly, with the increase of the dimensionality of feature vectors, more training data are needed to train the system models. However, the number of training data is finite in practice. When the number of features increases faster than the increase of the number of training data, the system models obtained may lose the generalization properties because of insufficient training data. This will lead to the degradation of the performances of pattern recognition systems. Another reason is that when new features are added, both relevant and irrelevant information to pattern classification are brought into pattern recognition systems because it is sometimes difficult to determine necessary or useless features before pattern recognition applications [16]. Some irrelevant information that resides in features may bring errors into pattern recognition systems and degrade the systems' performances.

2.6.2 Feature Dimensionality Reduction

Reducing the dimensionality of feature vectors is the most direct way to solve the problems caused by high feature dimensionalities. Feature dimensionality reduction is normally achieved in feature extraction step in pattern recognition systems. Both feature selection and extraction methods introduced in Section 2.3 are able to conduct feature dimensionality reduction tasks. Feature extraction method, however, shows significant advantages over feature selection method in practice [75]. This thesis will concentrate on feature extraction method for feature dimensionality reduction.

Feature dimensionality reduction can be easily achieved by reducing the rank of linear transformation shown in Eq. (2.20), where the transformation T is a $p \times m$, $p \geq m$ matrix, p is the dimensionality of parameter vectors and m feature vectors. Both LDA and PCA can be used for dimensionality reduction. This thesis proposes the use of MCE training algorithm for feature dimensionality reduction. These algorithms will be introduced in the following chapters.

Feature dimensionality reduction is an active area of research because of its importance [55, 59, 75, 78, 93]. In this thesis, all the feature extraction experiments will include feature dimensionality reduction tasks. The performances of feature extraction algorithms in feature dimensionality reduction will be investigated.

2.7 Summary of Chapter

This chapter gives a brief introduction to the fundamentals of pattern recognition. It includes the formulation of pattern recognition problems, definition of some basic concepts, approaches to design feature extractors and classifiers and introduction to feature dimensionality reduction problems in pattern recognition.

Chapter 3

Independent Feature Extraction

3.1 Linear Feature Extraction Formulation

Linear feature extraction method is the most basic way of extracting feature vectors. It projects parameter vectors from parametric space onto feature space through a linear transformation matrix T. Suppose the input observation vector x be an p-dimensional vector and T be a $p \times m$ $(p \geq m)$ matrix. The extracted feature vector y is:

$$y = T^T x \tag{3.1}$$

The difference between linear feature extraction algorithms is that they optimize T by different criteria. A number of algorithms have been proposed to seek the optimized T. LDA and PCA are the most popular ones among them. Briefly speaking, LDA optimizes T by maximizing the ratio of between-class variation and within-class variation; PCA obtains T by searching for the directions that have the largest variations. In the following subsections, a detailed discussion of each of them will be given.

3.2 Linear Discriminant Analysis

3.2.1 Fisher's Linear Discriminants

The goal of Fisher's linear discriminant is to well separate the classes by projecting classes' samples from p-dimension space onto a finely orientated line. For a K-class problem, $c = \min(K-1, p)$ different lines will be involved. Thus the projection is from a p-dimensional space to a c-dimensional space[28].

Suppose we have K classes, $\mathcal{X}_1, \mathcal{X}_2, \cdots, \mathcal{X}_K$. Let the ith observation vector from the $\mathcal{X}_j$ be x_{ji}, where $j = 1, \ldots, K$, $i = 1, \ldots, N_j$ and N_j is the number of observations from class j. The sample mean vector μ_j and the covariance matrix S_j of class j are given by:

$$\mu_j = \frac{1}{N_j} \sum_{i=1}^{N_j} x_{ji} \tag{3.2}$$

and

$$S_j = \frac{1}{N_j} \sum_{i=1}^{N_j} (x_{ji} - \mu_j)(x_{ji} - \mu_j)^T \tag{3.3}$$

The within-class covariance matrix S_W is given by:

$$S_W = \sum_{j=1}^{K} S_j \tag{3.4}$$

Define the overall mean μ and the total covariance matrix S_T as:

$$\mu = \frac{1}{N} \sum_{j=1}^{K} \sum_{i=1}^{N_j} x_{ji} = \frac{1}{N} \sum_{j=1}^{K} N_j \mu_j \tag{3.5}$$

and

$$S_T = \sum_{j=1}^{K} \sum_{i=1}^{N_j} (x_{ji} - \mu)(x_{ji} - \mu)^T \tag{3.6}$$

where $N = \sum_{j=1}^{K} N_j$. Then it follows that:

$$\begin{aligned} S_T &= \textstyle\sum_{j=1}^{K} \sum_{i=1}^{N_j} (x_{ji} - \mu_j + \mu_j - \mu)(x_{ji} - \mu_j + \mu_j - \mu)^T \\ &= \textstyle\sum_{j=1}^{K} \sum_{i=1}^{N_j} (x_{ji} - \mu_j)(x_{ji} - \mu_j)^T + \sum_{j=1}^{K} \sum_{i=1}^{N_j} (\mu_j - \mu)(\mu_j - \mu)^T \\ &= S_W + \textstyle\sum_{j=1}^{K} N_j (\mu_j - \mu)(\mu_j - \mu)^T \end{aligned} \tag{3.7}$$

It is natural to define the second term in Eq.(3.7)the between-class covariance matrix, so that we have:

$$S_B = \sum_{j=1}^{K} N_j (\mu_j - \mu)(\mu_j - \mu)^T \tag{3.8}$$

and

$$S_T = S_W + S_B \tag{3.9}$$

The projection from a p-dimensional space to an m-dimensional space is accomplished by m discriminant functions:

$$y_i = w_i^t x \qquad\qquad i = 1, 2, \cdots, m. \tag{3.10}$$

Eq. (3.10) can be re-written in matrix form:

$$y = W^t x \tag{3.11}$$

Then corresponding mean and covariance matrix of y are defined as:

$$\tilde{\mu}_j = \frac{1}{N_j} \sum_{i=1}^{N_j} y_{ji} \tag{3.12}$$

$$\tilde{\mu} = \frac{1}{N} \sum_{j=1}^{K} N_j \tilde{\mu}_j \tag{3.13}$$

$$\tilde{S}_W = \sum_{j=1}^{K} \sum_{i=1}^{N_j} (y_{ji} - \tilde{\mu}_j)(y_{ji} - \tilde{\mu}_j)^T \tag{3.14}$$

and

$$\tilde{S}_B = \sum_{j=1}^{K} N_j (\tilde{\mu}_j - \tilde{\mu})(\tilde{\mu}_j - \tilde{\mu})^T \tag{3.15}$$

It is straightforward to show that:

$$\tilde{S}_W = W^T S_W W \tag{3.16}$$

and

$$\tilde{S}_B = W^T S_B W \tag{3.17}$$

Fisher's linear discriminant is then defined as the linear functions $W^T x$ for which the criterion function

$$J(W) = \frac{|\tilde{S}_B|}{|\tilde{S}_W|} = \frac{W^T S_B W}{W^T S_W W} \tag{3.18}$$

is maximum.

It can be shown that the solution of (3.18) is that the ith column of an optimal W is the generalized eigenvector corresponding to the ith largest eigenvalue of matrix $S_W^{-1} S_B$.

3.2.2 Generalized LDA

Linear discriminant function is usually written as:

$$g(x) = w_0 + \sum_{i=1}^{p} w_i x_i \tag{3.19}$$

and can be extended to non-linear form by adding non-linear terms into the RHS of Eq. (3.19). If we express these non-linear terms as:

$$y_i = \phi_i(x) \qquad\qquad i = 1, 2, \cdots, d \tag{3.20}$$

where d is the number of non-linear terms. Then we obtain the generalized linear discriminant function:

$$g(x) = a_0 + a_1 \phi_1(x) + \cdots + a_d \phi_d(x) = a^T y \tag{3.21}$$

This function is not linear in x any more, but it is linear in y. The common approach to finding the solution of Eq. (3.21) is to define a criterion function $J(a)$ first, and then minimize $J(a)$ subject to $a^T y > 0$. Gradient descent method is often employed in the procedure of minimization. The core of this method is the regression equation:

$$a_{k+1} = a_k - \eta_k \nabla J(a_k) \tag{3.22}$$

where k and $k+1$ are index of the iteration steps, η_k is a positive scale factor that sets the step size and is usually called *learning rate* and $\nabla J(a_k)$ is the gradient of $J(a)$ at the point $a = a_k$.

Construction of $J(a)$ is in fact to find some analytically tractable scalar functions so that the inequalities $a^T y_i > 0$ can be readily solved by the gradient descent method. Three essential criterion functions are summarized in the following. Details can be found in [28].

A. Perceptron Criterion Function

The perceptron criterion function is defined as:

$$J_p(a) = \sum_{y \in \mathcal{Y}} (-a^T y) \tag{3.23}$$

where $\mathcal{Y}$ is the set of samples that are misclassified. This function is proportional to the sum of the distances from the misclassified samples to the decision boundaries. The problem with J_p is that the gradient of it is not continuous.

B. Squared Distance Criterion

The squared distance criterion is a close relative to perceptron criterion, but distinct by its continuous gradient. It is defined as:

$$J_q(a) = \sum_{y \in \mathcal{Y}} (a^T y)^2 \tag{3.24}$$

or

$$J_r(a) = \frac{1}{2} \sum_{y \in \mathcal{Y}} \frac{(a^T y - b)^2}{\|y\|^2} \tag{3.25}$$

where again $\mathcal{Y}$ is the set of misclassified samples and b is a margin vector. The second definition of J_r avoids the problem arising from (3.24) that the value of J_q can be dominated by the longest vectors.

C. Minimum Squared Error Criterion

Unlike the above two criteria which consider only the misclassified samples, minimum squared error criterion takes into account the entire samples. Let b be arbitrarily specified margin vector, minimum squared error is defined as:

$$J_s(a) = \sum_{i=1}^{N} (a^T y_i - b_i)^2 \tag{3.26}$$

It can be shown that the solution of Eq. (3.26) depends on the choice of the margin vector b.

3.2.3 Development of LDA

Traditional LDA can be generalized into a two-step procedure: defining the discriminant functions and looking for a solution by incorporating the discriminant functions into a criterion function that is suitable for gradient descent search procedure. The inadequacies in these two steps are:

a. In the second step, the decision rule of classification

$$C(x) = C_i \qquad if \;\; g_i(x) = \max_{for\ all\ j=1,\cdots,K} g_j(x) \tag{3.27}$$

does not appear in a functional form in the overall criterion function for optimization. Therefore there exists an inconsistency between the criterion function and minimum classification error probability objective.

b. The initial purpose of defining discriminant functions is to reduce the dimensionality, while the class information is not considered. Therefore the discriminant functions are unable to give an adequate description of classes, such as class distributions and class boundaries.

The first problem with LDA is often mended by embedded the decision rule into a function of *misclassification measure*, in which the discriminant functions are combined. This leads to the development of *Minimum Classification Error* (MCE) training algorithm, which will be thoroughly discussed in Chapter 4.

The improvement on the second inadequacy of LDA is made by redefining the discriminant functions as the linear functions f:

$$f(x) = w \cdot x + b \qquad with \;\; x \in \mathcal{X}, \;\; b \in \mathcal{R}^p \tag{3.28}$$

then maximizing the distance between the hyperplane that separates the classes and the closest samples to the hyperplane. The solution of this problem leads to the development of *Support Vector Machine* (SVM), which will be discussed in Chapter 7.

3.3 Principal Component Analysis

3.3.1 A Brief History of PCA

The earliest descriptions of PCA appear to be proposed by Pearson in 1901 [77] and Hotelling in 1933 [48]. In Pearson's paper, the main concern was to find lines and planes which best fit a set of points in a p-dimensional space and the geometric optimization problems considered lead to *principal components* (PCs). It seems that little relevant work has been published in the 32 years between Pearson's and Hotelling's papers. Hotelling's motivation is that there may be a smaller 'fundamental set of independent variables' which determines the values of the original p variables. The term 'components' was introduced and they were chosen to maximize their successive contributions to the total of the variances of the original variables. Hotelling called the components derived in this way the 'principal components' and the analysis to find these components was then christened the 'method of principal components'. Hotelling derived the PCs by using Lagrange multipliers and showed in his paper how to find PCs by the power method.

In 1939, Girshick [39] investigated the asymptotic sampling distributions of the coefficients and variances of PCs. But apart from Girshick's work, there appears to be little work on the development of different applications of PCA during nearly three decades following the publication of Hotelling's paper. Not until 1963, based on the earlier work by Girshick(1939), Anderson(1963) discussed the asymptotic sampling distributions of the coefficients and variances of the sample PCs, which has built up the fundamental framework of PCA [5]. Rao(1964) provided a large number of new ideas concerning uses, interpretations and extensions of PCA [80]. Gower(1966) discussed some links between PCA and various other statistical techniques and provided a number of geometric insights [41].

Despite the simplicity of the technique, much research is still being carried out in the general area of PCA. Apart from being used basically as a dimensionality reduction tool, PCA is also widely used for feature extraction, data compression and preprocession for pattern recognition, etc.

3.3.2 Definition and Derivation of PCA

The central idea of PCA is to reduce the dimensionality of a data set which consists of a large number of interrelated variables, while retaining as much as possible the variation present in the data set.

Suppose x is a p-dimensional random vector. PCA first looks for a linear function $\alpha_1^T x$ of x which has maximum variance, where $\alpha_1 = \{\alpha_{11}, \alpha_{12}, \cdots, \alpha_{1p}\}$ is a p-dimensional vector and

$$\alpha_1^T x = \alpha_{11}x_1 + \alpha_{12}x_2 + \cdots + \alpha_{1p}x_p = \sum_{i=1}^{p} \alpha_{1i}x_i \tag{3.29}$$

Then it looks for a second linear function $\alpha_2^T x$ which is uncorrelated with $\alpha_1^T x$ and has the second maximum variance. Repeat this procedure until the desired kth linear function $\alpha_k^T x$ is found. These k variables, $\alpha_1^T x, \alpha_2^T x, \cdots, \alpha_k^T x$, are called k principle components(PCs). In general, up to p PCs can be found. The mathematical expression of constraints on $\alpha_i, (i = 1, 2, \cdots, p)$ are:

$$\alpha_i^T \alpha_j = \begin{cases} 1, & if \quad i = j \\ 0, & if \quad i \leq j \end{cases} \tag{3.30}$$

Consider the first PC, $\alpha_1^T x$. α_1 maximizes $var[\alpha_1^T x] = \alpha_1^T \Sigma \alpha_1$ subject to $\alpha_1^T \alpha_1 = 1$. Use Lagrange multiplier, we have:

$$\alpha_1^T \Sigma \alpha_1 - \lambda_1(\alpha_1^T \alpha_1 - 1) \tag{3.31}$$

where λ_1 is a Lagrange multiplier. Differentiation (3.31) with respect to α_1 gives:

$$(\Sigma - \lambda_1 I_p)\alpha_1 = 0 \tag{3.32}$$

where I_p is the $(p \times p)$ identity matrix. Thus, λ_1 is the eigenvalue of Σ and α_1 is the corresponding eigenvector. Note the quantity to be maximized is:

$$\alpha_1^T \Sigma \alpha_1 = \alpha_1^T \lambda_1 \alpha_1 = \lambda_1 \alpha_1^T \alpha_1 = \lambda_1 \tag{3.33}$$

Thus, λ_1 must be the largest eigenvalue and α_1 is the corresponding eigenvector.

Consider the second PC, $\alpha_2^T x$, maximizes $\alpha_2^T x \alpha_2$ subject to being uncorrelated with the first PC, $\alpha_1^T x$, that is:

$$cov[\alpha_1^T x, \alpha_2^T x] = 0 \tag{3.34}$$

If choosing $\alpha_2^T \alpha_1 = 0$ to specify the relationship in (3.34), the quantity to be maximized is:

$$\alpha_2^T \Sigma \alpha_2 - \lambda_2(\alpha_2^T \alpha_2 - 1) - \phi \alpha_2^T \alpha_1 \tag{3.35}$$

where λ_2 and ϕ are Lagrange multipliers. Differentiation of (3.35) with respect to α_2 gives:

$$\Sigma \alpha_2 - \lambda_2 \alpha_2 - \phi \alpha_1 = 0 \tag{3.36}$$

and multiplication of (3.36) on the left by α_1^T gives:

$$\alpha_1^T \Sigma \alpha_2 - \lambda_2 \alpha_1^T \alpha_2 - \phi \alpha_1^T \alpha_1 = 0 \tag{3.37}$$

Eq. (3.37) can be reduced to:

$$\begin{aligned} &\Sigma \alpha_2 - \lambda\_2alpha_2 = 0 \\ &\phi = 0 \end{aligned} \tag{3.38}$$

Again $\lambda_2 = \alpha_2^T \Sigma \alpha_2$, therefore, λ_2 is the second largest eigenvalue and α_2 is the corresponding eigenvector.

By using the same strategy, it can be shown that the coefficient vector α_k of kth PC $(k = 1, 2, \cdots, p)$ is the eigenvector corresponding to the kth largest eigenvalue of Σ.

3.3.3 PCA for Feature Dimensionality Reduction in Classification

For a given p-dimensional data set $\mathcal{X}$, the m principal axes $T_1, T_2, \cdots, T_m$, where $1 \leq m \leq p$, are orthonomal axes onto which the retained variance is maximum in the projected space. Generally, $T_1, T_2, \cdots, T_m$ can be given by the m leading eigenvectors of the sample covariance matrix $S = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^T(x_i - \mu)$, where $x_i \in \mathcal{X}$, μ is the sample mean and N is the number of samples, so that:

$$ST_i = \lambda_i T_i \qquad i \in 1, \cdots, m \tag{3.39}$$

where λ_i is the ith largest eigenvalue of S. The m principal components of a given observation vector $x \in \mathcal{X}$ are given by:

$$y = [y_1, \cdots, y_m] = [T_1^T x, \cdots, T_m^T x] = T^T x \tag{3.40}$$

The m principal components of x are then uncorrelated in the projected space. In multi-class problems, the variations of data are determined on a global basis [58], that is, the principal axes are derived from a global covariance matrix:

$$\hat{S} = \frac{1}{N}\sum_{j=1}^{K}\sum_{i=1}^{N_j}(x_{ji} - \hat{\mu})(x_{ji} - \hat{\mu})^T \tag{3.41}$$

where $\hat{\mu}$ is the global mean of all the samples, K is the number of classes, N_j is the number of samples in class j, $N = \sum_{j=1}^{K} N_j$ and x_{ji} represents the ith observation from class j. The principal axes $T_1, T_2, \cdots, T_m$ are therefore the m leading eigenvectors of $\hat{S}$:

$$\hat{S}T_i = \hat{\lambda}_i T_i \qquad i \in 1, \cdots, m \tag{3.42}$$

where $\hat{\lambda}_i$ is the ith largest eigenvalue of $\hat{S}$.

An assumption made for dimensionality reduction by PCA is that most information of the observation vectors is contained in the subspace spanned by the first m principal axes, where $m < p$. Therefore, each original data vector can be represented by its principal component vector:

$$y = T^T x \tag{3.43}$$

where $T = [T_1, \cdots, T_m]$ is a $p \times m$ matrix.

The merit of PCA is that the extracted features have the minimum correlation along the principal axes. On the other hand, there are some defects that reside in PCA. First, as mentioned in [62], PCA is a scale-sensitive method, i.e., the principal components may be dominated by the elements with large variances. Another problem with PCA is that the directions of maximum variance are not necessarily the directions of maximum discrimination since there is no attempt to use the class information, such as the between-class scatter and within-class scatter.

3.4 Summary of Chapter

In this chapter two popular feature extraction and dimensionality reduction methods — LDA and PCA are discussed. In the following chapters, they will be used as references to evaluate other feature extraction and dimensionality reduction methods.

Chapter 4

MCE Training Algorithm

4.1 A Brief Review on MCE Training Algorithm

The input observation data of a pattern classification system usually contains large amount of irrelevant information that may increase computation expenses and degrade the performance of the system. Feature extraction is needed to remove the irrelevant information from the raw data. Conventionally, feature extractors, such as PCA and LDA, deal with data separately from classifiers. Meanwhile, the suitable criterion for classification is the minimum probability of error classification, which has no direct link to feature extraction criteria. This inconsistency forces feature extractors and classifiers to be trained separately. However, the optimized feature extractors and classifiers may cause the problem of dismatch between feature extractors and classifiers and thus do not necessarily make the whole pattern classification system effective. One possible way to solve this problem is to train the feature extractors and classifiers together with consistent criteria. In this case, MCE training algorithm is a suitable framework to achieve this goal.

MCE training algorithm is a type of discriminant training algorithm. It is proposed to mend the shortcomings of traditional discriminant training [54]. As pointed out by Juang and Katagiri [56], traditional discriminant training algorithms are inadequate in that the decision rule in classification does not appear in the overall criterion functions and there is an inconsistency between the criterion function and the minimum classification error objective. MCE training algorithm bridges this gap by introducing a classification measure, in which the decision rule is embedded, into the overall criterion functions.

Defining misclassification measure therefore stays in the core of the framework of MCE training algorithm. The basic way of defining misclassification measure is to embed the decision rule of classification into it so that the extracted features are ready for minimizing the classification error.

A popular way to define misclassification measure is through discriminant functions.

Let $g_j(x,\Lambda),(j=1,2,\cdots,K)$ be the discriminant functions, which indicates the degree that an observation x belongs to class k over the the parameter set Λ. The classifier makes the classification decision by the rule:

$$x \in Class\ k \quad if\ g_k(x,\Lambda) = \max_{for\, all\, i \in K} g_i(x,\Lambda) \tag{4.1}$$

Suppose x belongs to class k, Misclassification measure of x over Λ, $d(x,\Lambda)$, is then defined as the difference between discriminant function $g_k(x,\Lambda)$ and a combination of other discriminant functions $g_j(x,\Lambda), j=1,2,\cdots,K$ and $j \neq k$:

$$d_k(x,\Lambda) = g_k(x,\Lambda) - \mathcal{G}[g_1(x,\Lambda)),\cdots,g_{k-1}(x,\Lambda),g_{k+1}(x,\Lambda),\cdots,(g_K(x,\Lambda)] \tag{4.2}$$

so that when the misclassification measure $d_k(x,\Lambda)$ is minimized, the optimized features will automatically satisfy the minimum classification error criterion (4.1). Therefore MCE training achieves minimum classification error in a more direct manner than traditional discriminant learning. Furthermore, the simplicity of MCE algorithm makes it easy to apply MCE training algorithm to other frameworks. As a result, besides feature extraction, MCE training algorithm has been used in a number of pattern classification applications, such as dynamic time-wrapping based speech recognition [63] and HMM based speech and speaker recognition [8, 65].

4.2 Derivation of MCE Formulation

4.2.1 Conventional MCE Training Algorithm

Consider an input vector x, the classifier makes its decision by the decision rule expressed in Eq.(4.1). This criterion can be rewritten as:

$$x \in Class\ k \quad if \quad g_k(x,\Lambda) - \max_{for\, all\, i \neq k} g_i(x,\Lambda) > 0 \tag{4.3}$$

Thus, the higher the value of function $g_k(x,\Lambda) - \max_{for\ all\ i\neq k} g_i(x,\Lambda)$, the more reliable the classification result. This means that we can use the negative of this function as a measure of misclassification. The form of Eq. (4.3), however, is not suitable for optimization since it is not differentiable. In [54], a modified differentiable version of Eq. (4.3) is introduced as a misclassification measure. For the kth class, the definition is given by

$$d_k(x,\Lambda) = -g_k(x,\Lambda) + [\frac{1}{N-1}\sum_{for\, all\, i\neq k}(g_i(x,\Lambda))^{\eta}]^{1/\eta}, \tag{4.4}$$

where η is a positive number and $g_k(x, \Lambda)$ is the discriminant of observation x to its known class k. When η approaches ∞, it reduces to

$$d_k(x, \Lambda) = -g_k(x, \Lambda) + g_j(x, \Lambda), \tag{4.5}$$

where class j has the largest discriminant value among all the classes other than class k. Obviously, $d_k(x, \Lambda) > 0$ implies misclassification, $d_k(x, \Lambda) < 0$ means correct classification and $d_k(x, \Lambda) = 0$ suggests that x sits on the boundary. A loss function is then defined to smooth misclassification measure. Sigmoid function is often chosen since it is a smooth zero-one monotonic function suitable for gradient descent algorithm. Loss function is given as:

$$l_k(x, \Lambda) = f(d_k(x, \Lambda)) = \frac{1}{1 + e^{-\xi d_k(x, \Lambda)}} \tag{4.6}$$

where $\xi > 0$. For a training set $\mathcal{X}$, the empirical loss is defined as:

$$L(\Lambda) = E\{l_k(x, \Lambda)\} = \sum_{k=1}^{K} \sum_{i=1}^{N_k} l_k(x^{(i)}, \Lambda) \tag{4.7}$$

where N_k is the number of samples in class k. Clearly, minimizing the above empirical loss function will lead to the minimization of the classification error. As a result, Eq.(4.7) is called the MCE criterion[56, 54, 76]. The class parameter set Λ can be obtained by minimizing the loss function through the steepest gradient descent algorithm. This is an iterative algorithm and the iteration rules are:

$$\Lambda_{t+1} = \Lambda_t - \epsilon \nabla L(\Lambda)|_{\Lambda=\Lambda_t} \tag{4.8}$$

$$\nabla L(\Lambda) = \begin{bmatrix} \partial L / \partial \lambda_1 \\ \vdots \\ \partial L / \partial \lambda_d \end{bmatrix} \tag{4.9}$$

where t denotes t-th iteration, $\lambda_1, \cdots, \lambda_d \in \Lambda$ are parameters, $\epsilon > 0$ is the adaptation constant. For $s = 1, 2, \cdots, d$, the gradient $\nabla L(\Lambda)$ can be computed as follows:

$$\frac{\partial L}{\partial \lambda_s} = \xi \sum_{i=1}^{N_k} L^{(i)}(1 - L^{(i)}) \frac{\partial g_k(x^{(i)}, \Lambda)}{\partial \lambda_s}, \quad if\ \lambda_s \in\ class\ k \tag{4.10}$$

$$\frac{\partial L}{\partial \lambda_s} = -\xi \sum_{i=1}^{N_j} L^{(i)}(1 - L^{(i)}) \frac{\partial g_j(x^{(i)}, \Lambda)}{\partial \lambda_s}, \quad if\ \lambda_s \in\ class\ j \tag{4.11}$$

4.2.2 An Alternative MCE Training Algorithm

The purpose of defining misclassification measure is to obtain the largest discrimination between $g_k(x,\Lambda)$ and $g_j(x,\Lambda)$. Basically, we want $g_k(x,\Lambda)$ to be as large as possible while $g_j(x,\Lambda)$ to be as small as possible. The control of the joint behavior of $g_k(x,\Lambda)$ and $g_j(x,\Lambda)$ is essential to the success of MCE training. The conventional definitions in Eq.(4.4) and (4.5) uses an additive combination between $g_k(x,\Lambda)$ and $g_j(x,\Lambda)$. Additive combination, however, are linear combination. Its absolute value has no limitation, which is easy to make the gradient descent search process divergent. Furthermore, additive combination is a loose combination and has weak control of the joint behavior of $g_k(x,\Lambda)$ and $g_j(x,\Lambda)$. To enhance MCE's ability of controlling the joint behavior of discriminant functions, we propose an alternative definition of misclassification measure which uses a ratio combination between $g_k(x,\Lambda)$ and $g_j(x,\Lambda)$. The ratio combination is a non-linear combination with absolute value limitations and has strong control of the joint behavior of $g_k(x,\Lambda)$ and $g_j(x,\Lambda)$. The alternative definition also comes from Bayes decision rule. Since the values of discriminant functions are all positive, we therefore can rewrite Eq. (4.1) as follows:

$$x \in Class\ k \quad if \quad \frac{\max_{for\,all\,i\neq k} g_i(x,\Lambda)}{g_k(x,\Lambda)} < 1 \tag{4.12}$$

The misclassification measure, $d_k(x,\Lambda)$ is then defined as an approximate of the L.H.S of Eq. (4.12):

$$d_k(x,\Lambda) = \frac{[\frac{1}{N-1}\sum_{for\,all\,i\neq k} g_i(x,\Lambda)^{\eta}]^{1/\eta}}{g_k(x,\Lambda)} \tag{4.13}$$

To the extreme case, i.e. $\eta \to \infty$, Eq. (4.13) becomes:

$$d_k(x,\Lambda) = \frac{g_j(x,\Lambda)}{g_k(x,\Lambda)} \tag{4.14}$$

The loss function still uses the Sigmoid function. The class parameters are optimized using the same adaptation rules as shown in Eq. (4.8) and (4.9). The gradients of Λ, $\nabla L(\Lambda)$, are calculated as follows:

$$\frac{\partial L}{\partial \Lambda_s} = -\xi \sum_{i=1}^{N_j} L^{(i)}(1-L^{(i)}) \frac{g_j(x^{(i)},\Lambda)}{[g_k(x^{(i)},\Lambda)]^2} \frac{\partial g_k(x^{(i)},\Lambda)}{\partial \Lambda_s}, \ if\ \Lambda_s \in\ class\ k \tag{4.15}$$

and

$$\frac{\partial L}{\partial \Lambda_s} = \xi \sum_{i=1}^{N_k} L^{(i)}(1-L^{(i)}) \frac{1}{g_k(x^{(i)},\Lambda)} \frac{\partial g_j(x^{(i)},\Lambda)}{\partial \Lambda_s}, \ \ if\ \Lambda_i \in\ class\ j \tag{4.16}$$

where $\Lambda_s \in \Lambda, s = 1, \cdots, d$.

4.2.3 A Comparison of Two Forms of MCE training Algorithms

The proposed alternative form of MCE training algorithm differs from the conventional one in that the misclassification measure is a non-linear combination of discriminant functions. To compare these two forms of MCE training algorithms, we use a $g_k(x,\Lambda)$-$g_j(x,\Lambda)$ decision plane to show their behaviors in the training process. The vertical axis of the decision plane is $g_k(x,\Lambda)$, which represents the discriminant of a vector x to its desired class k. The horizontal axis is $g_j(x,\Lambda)$, representing the largest discriminant of x among all the classes other than k. The decision line is $g_k(x,\Lambda) = g_j(x,\Lambda)$. Driven by the training algorithms, all the training data move in this plane

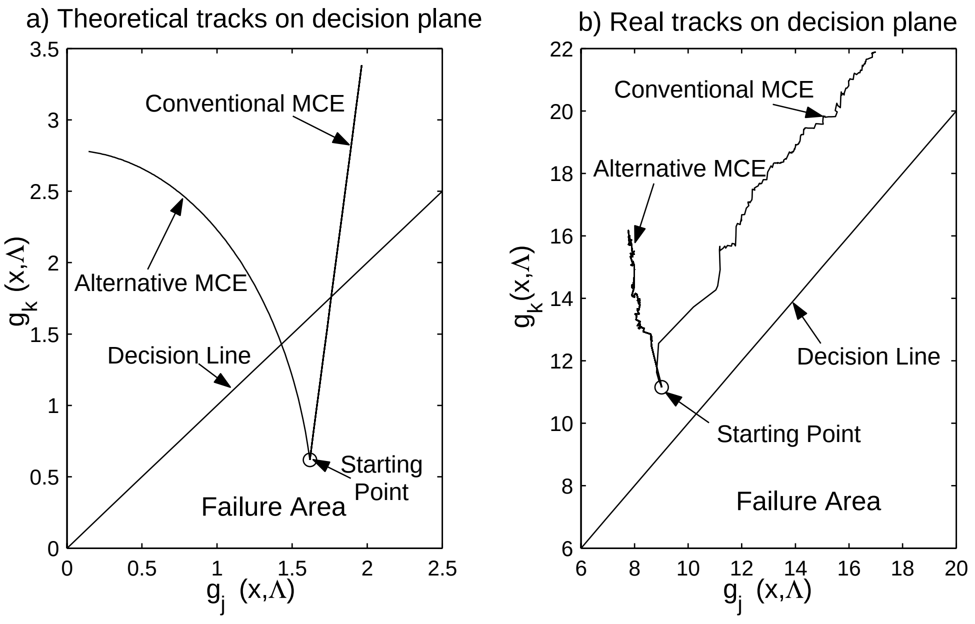


Figure 4.1: Theoretical and practical tracks of a data moving in the decision plane.

throughout the training process. The behaviors of the training algorithms are therefore demonstrated by the tracks of data. If the data move towards the top left of the decision plane, both the absolute and relative differences between $g_k(x,\Lambda)$ and $g_j(x,\Lambda)$ increase. Therefore the training process is effective and robust. If, however, the data move towards the top right of the plane, the relative difference between $g_k(x,\Lambda)$ and $g_j(x,\Lambda)$ does not increase significantly despite an increase in the absolute difference. Furthermore, the training can become divergent if not precisely controlled. In this case the training process is not desirable. Fig. 4.1a shows the theoretical behaviors of these two forms of MCE in the training process and Fig. 4.1b shows their practical behavior. The data used in Fig. 4.1b is randomly selected from Deterding Vowels database. These two figures show that the proposed alternative MCE training algorithm drives training data to the top left of the decision plane both theoretically and practically and thus is robust. The conventional MCE training algorithm drives training data to the top right of the decision plane and is therefore not as robust as the alternative one. The following section will give the experiment results of both the alternative and the conventional MCE training algorithms on some small databases.

4.3 Classification Experiments on Small Databases

4.3.1 Databases

An evaluation of MCE training algorithms, LDA and PCA is made on several different databases. The first one is Deterding vowel database, which has 11 vowel classes as shown in the Table 4.1. Each of these 11 vowels are uttered 6 times by 15 different speakers. This gives a total of 990 vowel tokens. A central frame of speech signal is excised from each of these 990 vowel tokens. A 10th order linear prediction analysis is carried out for each frame resulting in 10 log-area parameters. These 10 parameters define the original 10 dimensional feature space. 528 frames from the eight speakers are used to train the models and 462 frames from the seven speakers are used to test the models.

vowel	word	vowel	word	vowel	word	vowel	word
i	heed	O	hod	I	hid	C:	hoard
E	head	U	hood	A	had	u:	who'd
a:	hard	3:	heard	Y	hud		

Table 4.1: Vowels and words used in Deterding Vowels database.

The second database used is D. German's GLASS database which contains the measurements of the chemical constitutions in terms of their oxide content (Na, Mg, Al, Si, K, Ca, Ba and Fe) and the refractive index of the glass, manufactured through two different processes. The database has 163 instances, of which 87 measurements are made on glass manufactured through the float process and 76 on glass through non-float process. Each measurement has 10 numeric-valued attributes.

The third database is BREAST CANCER WISCONSIN(BCW) database donated by Olvi Mangasarian. This database contains 699 instances and 2 classes (malignant and benign). Each data has 9 integer-valued attributes.

The forth database is IONOSPHERE database. It is from V. Sigillito. It has 2 classes, 351 instances and 34 numeric attributes. This database is used for classification of radar returns from the ionosphere.

The fifth database is the famous IRIS data. It is Fisher's classical data. It has 3 classes, 4 numeric attributes and 150 instances. One class is linearly separable from the other two, but the other two are not linearly separable from each other.

The last database is WINE data, which is donated by Stefan Aeberhard. It uses chemical analysis to determine the origin of wines. There are 3 classes, 178 instances and 13 attributes. The first attribute is the class attribute.

The reason for using these databases is that they have been studied by a number of researchers, such as H.Brunzell & J.Eriksson [16], B.Tian & M.R.Azimi-Sadjadi [94] and S.Aeberhard, O. de Vel & D.Coomansso [1]. The experiment results will be comparable by using these databases.

4.3.2 Classifier

In order to evaluate the performance of the independent feature extraction algorithms (PCA and LDA) and MCE training algorithm, we have used a minimum distance classifier. Here, a feature vector y is classified to j-th class if the distance $d_j(y)$ is less than the other distances $d_i(y), \quad i = 1, \cdots, K$. We use Mahalanobis distance measure to compute the distance of a feature vector from a given class. Thus, the distance $d_i(y)$ is computed as follows:

$$d_i(y) = (y - \mu_i)^T \Sigma_i^{-1} (y - \mu_i) \tag{4.17}$$

where μ_i is the mean vector of class i and Σ_i is the covariance matrix. In our experiments, we use full covariance matrix.

4.3.3 Classification Results

DATABASE	MCE(con)	MCE(alt)	LDA	PCA
VOWELS(Train)	85.6	99.1	97.7	97.7
VOWELS(Test)	53.7	55.8	51.3	49.1
GLASS	76.7	83.4	63.2	61.4
BCW	98.4	98.7	92.1	90.5
IONOSPHERE	95.2	98.9	62.4	61.8
IRIS	98.0	99.0	98.0	98.0
WINE	100.0	100.0	100.0	100.0

Table 4.2: Results on different databases (in %).

Four algorithms, conventional MCE training algorithm, alternative MCE training algorithm, LDA and PCA, are used in the classification experiments. The four algorithms will do feature extraction first, then the extracted features by the four algorithms will be used in a classification task independently. The classifier used in the classification task is Mahalanobis distance classifier. For the sake of convenience, we denote the conventional MCE training algorithm as MCE(con) and the alternative MCE training algorithm as MCE(alt) in the figures and tables. Table 4.2 shows the results of employing these four algorithms on the above six databases. The following observations can be made from these results:

- Both the conventional MCE and the alternative MCE training algorithms generally perform better than LDA and PCA on these databases. On GLASS data the differences between the recognition rates of two MCEs and LDA and PCA are between 13.5% $\sim$ 22.0%. On IONOSPHERE data, the differences are even higher, which are between 32.8% $\sim$ 37.1%.

- The alternative MCE training algorithm performs better than the conventional MCE training algorithm. On the VOWELS(train), the recognition rate of alternative MCE is 13.5% higher than conventional MCE. On other database, the recognition rates of alternative MCE are usually 1% $\sim$ 6% higher than those of conventional MCE.

- The performance of LDA is slightly better that that of PCA on some of the databases, such as VOWELS(test), GLASS, BCW and IONOSPHERE. Their general performances are similar on other databases.

Compared to the results given in literature, Brunzell & Eriksson [16] achieve the correct classification rate of 97.1% on BCW and 82.9% on IONOSPHERE by using Mahalanobis Linear Transformation (MLT) classifier. GLASS data is not a well separated dataset. The highest recognition rate achieved by Brunzell & Eriksson is 69.3%. WINE and IRIS datasets are very well separated datasets. Brunzell & Eriksson's results are very close to the results shown in Table 4.2.

4.4 Conclusion

The classification results on a number of databases show that MCE training algorithm, as an integrated framework of feature extractors and classifiers, has a significant improvement over common feature extraction algorithms, such as LDA and PCA, when employing the same classifier. The conventional MCE training algorithm uses an additive model in misclassification measure, which is not suitable for gradient descent method. This chapter proposes an alternative MCE training algorithm, which employs a ratio model in misclassification measure. The experiment results show that the alternative MCE training algorithm is more robust and has a better performance than the conventional MCE training algorithm.

4.5 Summary of Chapter

In this chapter, the framework of MCE training algorithm is introduced. An alternative MCE training algorithm, which uses a ratio model in misclassification measure, is proposed. The alternative MCE training algorithm, the conventional MCE training algorithm, LDA and PCA are evaluated in an experiment over six popular databases and the corresponding results are compared.

Chapter 5

Integrated Feature Extraction & Classification

5.1 Introduction

Independent feature extraction methods, such as LDA and PCA, extract features by transforming parameter vectors from parametric space to feature space $\mathcal{F}$ through a linear transformation T. But they optimize T independently. Their optimization criteria are different from the minimum classification error objective. Independent feature extraction and classification may cause inconsistency between feature extraction and the classification stages of a pattern recognizer and consequently, degrade the performance of classifiers [54]. A direct way to overcome this problem is to conduct feature extraction and classification jointly with a consistent criterion.

Since MCE training algorithm is derived from discriminant analysis and used for optimizing classifiers, it provides a suitable integrated framework for joint feature extraction and classification. In this chapter, we propose the use of MCE training algorithm for integrated feature extraction and classification and derive the corresponding formulation.

5.2 MCE Training Algorithms for Integrated Feature Extraction and Classification Tasks

5.2.1 Integrated Training Procedure

As with other feature extraction methods, MCE training algorithm extracts features through a linear transformation matrix $T_{p\times m}$, where $p \geq m$,

$$y = T^T x \tag{5.1}$$

Let the class parameter set in feature space $\mathcal{F}$ be $\tilde{\Lambda}$. The discriminant functions in $\mathcal{F}$ becomes:

$$g_i(y, \tilde{\Lambda}) = g_i(T^T x, \tilde{\Lambda}) \qquad i = 1, \cdots, K \tag{5.2}$$

Include T in the expanded class parameter set $\Phi = (T, \tilde{\Lambda}),\ \Phi \in \mathcal{F}$. Thus T can be optimized synchronously with $\tilde{\Lambda}$ in the framework of MCE training algorithm. The integrated training procedure is shown as followings:

- **Step1**. Initialize T with an identity matrix.
- **Step2**. Transform parameter vectors into feature space $\mathcal{F}$ through the initial T and initialize $\tilde{\Lambda}$ in $\mathcal{F}$.
- **Step3**. Calculate the gradients of each element of T and $\tilde{\Lambda}$ in $\mathcal{F}$.
- **Step4**. Update T and $\tilde{\Lambda}$ by the steepest gradient descent method.
- **Step5**. Calculate the empirical loss. If the stop criterion is not met, goto step 3, otherwise stop.

The stop criterion of MCE training algorithm is defined as follows:

$$\Delta L(T, \tilde{\Lambda}) = L_{t+1}(T, \tilde{\Lambda}) - L_t(T, \tilde{\Lambda}) \leq Threshold \tag{5.3}$$

where $L(T, \tilde{\Lambda})$ is the empirical loss and t and $t+1$ are iteration steps.

5.2.2 Formulation for Integrated Tasks

In the integrated feature extraction and classification tasks, $\tilde{\Lambda}$ is initialized and optimized in the feature space $\mathcal{F}$ rather than the original parametric space. Accordingly, misclassification measure needs to be reformulated over the new class parameter set $\Phi = (T, \tilde{\Lambda})$ in $\mathcal{F}$. For the conventional MCE training algorithm, the misclassification measure is redefined as:

$$d_k(y, \tilde{\Lambda}) = d_k(T^T x, \tilde{\Lambda}) = -g_k(T^T x, \tilde{\Lambda}) + [\frac{1}{N-1} \sum_{for\,all\,i \neq k} (g_i(T^T x, \tilde{\Lambda}))^\eta]^{1/\eta} \tag{5.4}$$

When η approaches ∞, the misclassification measure becomes:

$$d_k(y, \tilde{\Lambda}) = d_k(T^T x, \tilde{\Lambda}) = -g_k(T^T x, \tilde{\Lambda}) + g_j(T^T x, \tilde{\Lambda}) \tag{5.5}$$

For the alternative MCE training algorithm proposed in Chapter 4, the misclassification measure is redefined as:

$$d_k(y, \tilde{\Lambda}) = d_k(T^T x, \tilde{\Lambda}) = \frac{[\frac{1}{N-1} \sum_{for\,all\,i \neq k} g_i(T^T x, \tilde{\Lambda})^\eta]^{1/\eta}}{g_k(T^T x, \tilde{\Lambda})} \tag{5.6}$$

To the extreme case, i.e. $\eta \to \infty$, Eq. (5.6) becomes:

$$d_k(y, \tilde{\Lambda}) = d_k(T^T x, \tilde{\Lambda}) = \frac{g_j(T^T x, \tilde{\Lambda})}{g_k(T^T x, \tilde{\Lambda})} \tag{5.7}$$

The loss of classifying an observation vector x is then calculated via its transformed vector y:

$$l(x, \tilde{\Lambda}, T) = l(d_k(y, \tilde{\Lambda})) = l(d_k(T^T x, \tilde{\Lambda})) = \frac{1}{1 + e^{-\alpha d(T^T x, \tilde{\Lambda})}} \tag{5.8}$$

The empirical loss over the whole observation set is given by:

$$L(\tilde{\Lambda}, T) = E\{l(d_k(y, \tilde{\Lambda}))\} = E\{l(d_k(T^T x, \tilde{\Lambda}))\} \tag{5.9}$$

Since Eq. (5.9) is a function of T and $\tilde{\Lambda}$, the elements in T can be optimized together with the parameter set $\tilde{\Lambda}$ in the same gradient descent procedure. $\tilde{\Lambda}$ can be optimized by still using function Eq. (4.8), Eq. (4.10) and Eq. (4.11). Inherited from these equations, T is optimized by the following adaptation rule:

$$T_{sq}(t+1) = T_{sq}(t) - \epsilon \left. \frac{\partial L}{\partial T_{sq}} \right|_{T_{sq} = T_{sq}(t)} \tag{5.10}$$

where t denotes tth iteration, ϵ is the adaptation constant or learning rate and s and q are the row and column indicators of transformation matrix T. For the conventional MCE training algorithm,

$$\frac{\partial L}{\partial T_{sq}} = \xi \sum_{k=1}^{K} \sum_{i=1}^{N_k} L^{(i)}(1 - L^{(i)}) \left(\frac{\partial g_k(T^T x^{(i)}, \tilde{\Lambda})}{\partial T_{sq}} - \frac{\partial g_j(T^T x^{(i)}, \tilde{\Lambda})}{\partial T_{sq}} \right) \tag{5.11}$$

and for the alternative MCE training algorithm,

$$\frac{\partial L}{\partial T_{sq}} = \xi \sum_{k=1}^{K} \sum_{i=1}^{N_k} L^{(i)}(1 - L^{(i)}) \frac{\frac{\partial g_j(T^T x^{(i)}, \tilde{\Lambda})}{\partial T_{sq}} g_k(T^T x^{(i)}, \tilde{\Lambda}) - \frac{\partial g_k(T^T x^{(i)}, \tilde{\Lambda})}{\partial T_{sq}} g_j(T^T x^{(i)}, \tilde{\Lambda})}{[g_k(T^T x^{(i)}, \tilde{\Lambda})]^2} \tag{5.12}$$

5.3 Results on Some Small Databases

5.3.1 Databases and Classifiers

An evaluation of using MCE training algorithm for integrated feature extraction and classification is made on two of the six databases used in the previous chapter. One is Deterding vowel database, and the other is German's GLASS data. The reasons for choosing these two databases for the evaluation are:

- Compared to other databases, the DETERDING and GLASS databases are difficult to classify. The results of dimensionality reduction algorithms on these two databases will provide more information than those on "easy" databases.

- The other databases are well separated databases. Most classification algorithms can obtain fairly high correct classification rate on them, for example, WINE data can be completely classified. Therefore, they are not suitable for evaluating the performance of dimensionality reduction algorithms.

Minimum distance classifier based on Mahalanobis distance measure and full class covariance matrices are used in the MCE training procedure. The experiment setup is identical to that in the previous chapter so that the results are comparable. In the experiments, identity matrix is used as the initial transformation matrix and the class parameters in $\tilde{\Lambda}$ are initialized as class statistical means and covariances.

5.3.2 Results and Observations

Both the conventional and the alternative MCE training algorithms are used in the integrated feature extraction and classification experiments. Their results are compared to those of LDA and PCA. For the sake of simplicity and consistency, we still denote the conventional MCE training algorithm as MCE(con) and the alternative MCE training algorithm as MCE(alt) in the figures and tables. Fig. 5.1 shows the results when using the Deterding Vowels database in different dimensional spaces. The following observations can be made from it:

- During the training process, all of the four algorithms have the best performance when the training is carried out in the feature space that has the equal dimensionality to the parametric space. But this is not the case in testing where the best results are usually obtained when the dimensionality of the feature space has been reduced nearly by half.

- During the training process, recognition rates decrease with the reduction of dimensionality. In testing, however, no regular pattern of changes in recognition rates with dimensionality can be observed.

- The performance of the alternative MCE training algorithm is the best in all four algorithms on training data throughout all the dimensions.

- The alternative MCE training algorithm performs better than the other three algorithms on most dimensions on testing data.

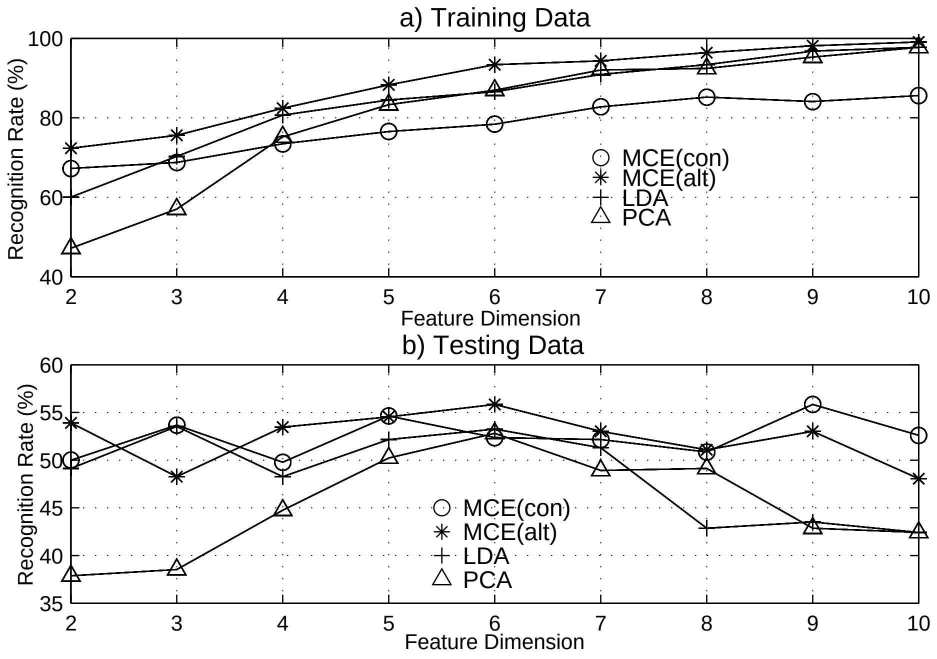


Figure 5.1: Comparison of the recognition rates of MCE(con), MCE(alt), LDA and PCA on Deterding database.

- The conventional MCE training algorithm has the poorest performance in high-dimensional feature spaces on training data, while its performance on testing data are better than those of LDA and PCA.

Table 5.1 shows the results of the four algorithms obtained on GLASS database. The results of the conventional and the alternative MCE training algorithms are given in Columns 2 and 3 respectively and the results of LDA and PCA are in Columns 4 and 5, respectively. Similar observations can be made from the table, except:

- The best recognition rates of the four algorithms do not appear on the highest dimension. For example, the best recognition rate of LDA appears on dimension 2, PCA dimension 6, conventional MCE dimension 6 and alternative MCE dimension 3.

- The performances of both the conventional and the alternative MCE training algorithms are very encouraging. The correct classification rates of the conventional MCE training algorithm are on average around 13% higher than those of LDA and around 20% higher than those of PCA. The alternative MCE training algorithm performs better than the conventional MCE training algorithm. The correct classification rates of alternative MCE training algorithm are on average around 4% higher than those of the conventional MCE training algorithm.

DIM	MCE(con)	MCE(alt)	LDA	PCA
2	77.9	77.9	68.1	48.5
3	75.5	83.4	64.4	49.1
4	79.1	80.4	63.2	60.7
5	77.3	80.4	65.0	63.2
6	79.7	82.8	62.0	63.8
7	76.7	83.4	63.2	61.4

Table 5.1: Results on GLASS data (in %).

5.4 Conclusion and Findings

Following conclusion can be drawn from the above results:

- The results show that the framework of MCE training algorithm is suitable for integrated feature and classification tasks.
- Both the conventional and the alternative MCE training algorithms perform better than LDA and PCA, which are independent feature extraction algorithms, in the experiments.
- The alternative MCE training algorithm has a better performance than the conventional MCE training algorithm.
- Compared to PCA, LDA has a slightly better performance than PCA.
- The results of the experiments show that integrated feature extraction and classification algorithms has significant advantage over independent feature extraction algorithms.

The results, however, present some facts that conflict the common understanding of feature dimensionality reduction. These facts are:

- The classification results on the testing data of Deterding Vowels database do not change as smoothly as those on the training data in different dimensions. For example, the correct classification rate of the alternative MCE training algorithm has a sharp fall on dimension 3. The recognition rates of both the conventional MCE training algorithm and LDA has a deep "valley" on dimension 4. Similar situations appear on the results on GLASS database as well.
- The highest recognition rates on testing data do not always happen on the highest dimensions as those training data. For example, the alternative MCE training algorithm has its highest recognition rate on dimension 6 on Deterding Vowels database. LDA has its highest recognition rate in dimension 2 on GLASS database.

- Similarly, the lowest recognition rates on testing data do not always happen on the lowest dimensions. For example, the lowest recognition rate of the alternative MCE training algorithm appears on dimension 10 on Deterding Vowels database, which is the dimensionality of the observation space.

Similar facts can also be found in Brunzell and Eriksson's work [16]. Normally, the recognition rate on testing data is an index of the generalization of the class models obtained from training process. These facts show that feature dimensionality reduction does not necessarily lead to a degradation of the generalization of class models. However, feature dimensionality reduction will definitely lead to the loss of class information. This implies that some information carried by features is useless for discriminating classes and the loss of it will not affect the generalization of class models. From discrimination point of view, such information will exist in more than one class and will cause confusion between classes. How to parameterize this confusing information and remove it from features will be a research problem for discriminative learning.

5.5 Summary of Chapter

In this chapter, the use of MCE training algorithm for integrated feature extraction and classification tasks is proposed. The corresponding formulation of the algorithm is derived. An experiment is carried out on two databases, Deterding Vowels database and GLASS database. In the experiment, the conventional and the alternative MCE training algorithms and other two independent feature extraction algorithms — LDA and PCA are employed. The performances of both the conventional and the alternative MCE training algorithms are compared to those of LDA and PCA.

Chapter 6

Generalized MCE Training Algorithm

6.1 Introduction

One of the major concerns with MCE training is the generalization of the models trained. This is because the gradient descent method used in MCE training algorithm for model optimization does not guarantee the global minimum value. Fig. 6.1 gives an example of how gradient descent method searches for the optimal value. The optimization procedure of gradient descent method indicates that the optimality of MCE training algorithm is largely dependent on the choice of the starting point, i.e. the initialization of the parameters.

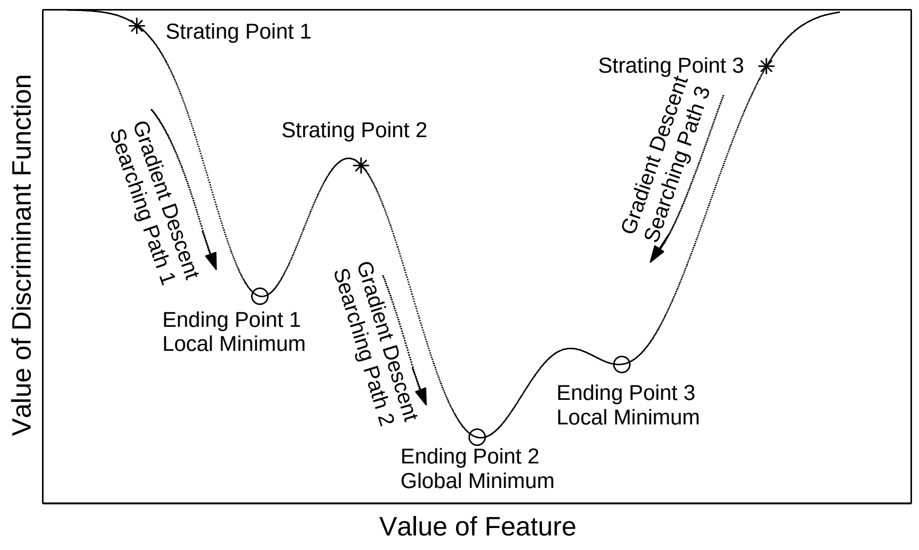

Strating Point 1
Gradient Descent Searching Path 1
Ending Point 1 Local Minimum
Strating Point 2
Gradient Descent Searching Path 2
Ending Point 2 Global Minimum
Strating Point 3
Gradient Descent Searching Path 3
Ending Point 3 Local Minimum
Value of Discriminant Function
Value of Feature

Figure 6.1: Effects of the choices of the starting point on MCE training.

A popular method of initializing the parameters of MCE training algorithm is given by Paliwal, Bacchiani and Sagisaka in [76]. In this method, the transformation matrix is initialized by an identity matrix. The class parameters are initialized by their maximum likelihood estimates (i.e. by their conditioned means and/or variances). In feature dimensionality reduction tasks, transformation matrix is still initialized by an identity matrix but the last $(p - m)$ columns will be discarded, where $m < p$, p is the dimensionality of observation space and m is the dimensionality of feature space. For convenience, this method is denoted as the normal initialization method. This method in fact is equivalent to spanning the new reduced-dimensional feature space by the first m dimensions of the parametric space. The other $m - p$ dimensions will be removed. However, in many cases, this is a convenient way of initialization rather than an effective way because the classification criterion has not been considered in the initialization. Figure 6.2 shows an example of the ineffectiveness of this type of initialization method. In this example, MCE training algorithm is applied to a dimensionality reduction task. The dimensionalities of feature spaces used are from 3 to 8. Two training processes with different initial transformation matrix are used in the example. The first one initializes the transformation matrix by the above method. The second one initializes the transformation matrix still by an identity matrix but the columns kept in the transformation matrix are selected manually. The sequential numbers of columns selected for the initial transformation matrix are listed as follows:

- dimension 3 — Column 0, 1 and 4
- dimension 4 — Column 0, 1, 3 and 4
- dimension 5 — Column 1, 2, 3, 5 and 8
- dimension 6 — Column 0, 1, 2, 3, 4, and 7
- dimension 7 — Column 0, 1, 2, 3, 4, 5 and 7
- dimension 8 — Column 0, 1, 2, 3, 4, 5, 7 and 8

Other parameters, such as the adaptation coefficient, are identical in the two training processes. The database used is Deterding Vowels database. The results are obtained on the testing dataset. The six small figures (from a) to f)) in Figure 6.2 record the whole training process on each dimension (dimension 3 to dimension 8), respectively. The horizontal axis of each figure represents the number of iteration. The maximum number of iteration is 3000. The vertical axis is the recognition rate. In this figure, the results obtained by the first training process are represented by the "-$\circ$-" curves. The results obtained by the second training process are represented by the "-Δ-" curves.

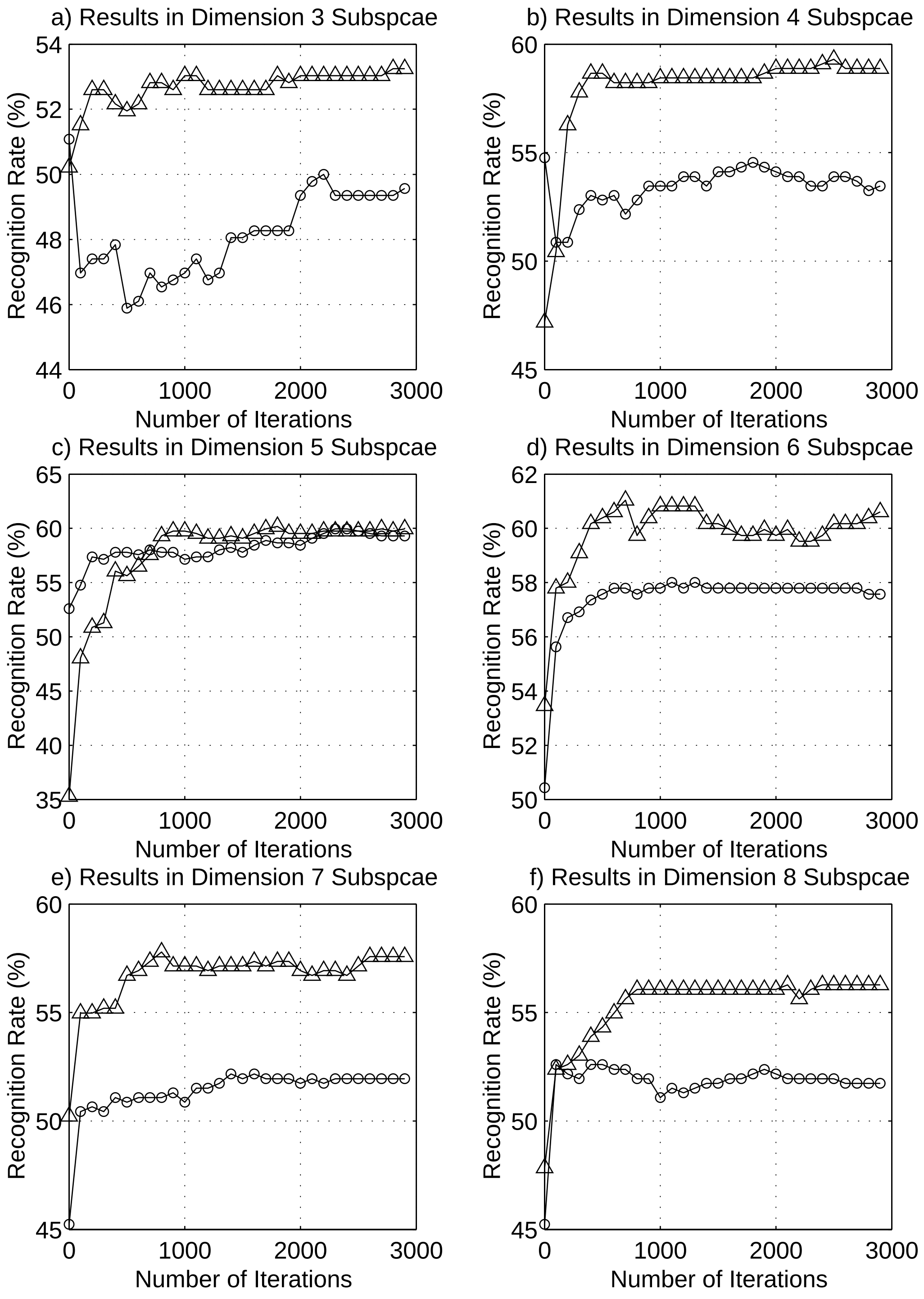


Figure 6.2: Results of different initialization of the transformation matrix on MCE training process. (Deterding database : testing set) ○ – Initialized by the normal initialization method given in [76]; Δ – Manually initialized.

The results show that the first MCE training process, in which the transformation matrix is initialized by the normal initialization method given in [76] is not as effective as the second one, i.e. manually initialized MCE training process. This implies that the initialization of the transformation matrix can affect MCE training process significantly. A properly initialized transformation matrix will help to increase the generalization ability of the models optimized by MCE training process. However, no work on the initialization problem of MCE training algorithm has been introduced in the literature so far. In this chapter, a generalized MCE (GMCE) training algorithm is proposed to solve this problem.

6.2 Generalized MCE (GMCE) Training Algorithm

MCE training algorithm provides a framework that enables transformation T and class parameters Λ to be optimized synchronously. However, it employs gradient descent method for optimization. This makes MCE training process a type of thorough searching process for local minimum. But global minimum is not guaranteed by this process. The results in the previous section show that the optimality of MCE training process largely depends on the initialization of T. A generalized MCE training algorithm is proposed in this section to mend the initialization problem of MCE training algorithm.

In GMCE training algorithm, the training process is regarded as two sequential training procedures: the first one is an initialization procedure, which will conduct a general search for the initial transformation matrix. The second procedure will conduct normal MCE training. Figure 6.3 compares GMCE training process with normal MCE training process.

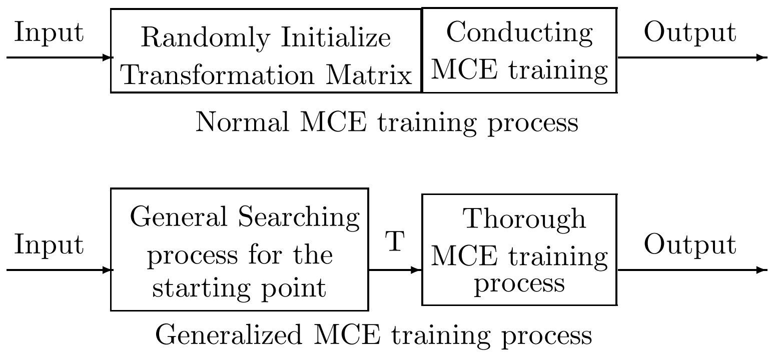

Input
Randomly Initialize Transformation Matrix
Conducting MCE training
Output
Normal MCE training process
Input
General Searching process for the starting point
T
Thorough MCE training process
Output
Generalized MCE training process

Figure 6.3: Comparison between normal MCE training process and generalized MCE training process.

The initialization procedure will provide the next MCE training procedure a suitable initial transformation matrix. Since this procedure is independent of the second procedure, any feature extraction criteria can be used in this procedure. In the following section, three different criteria, F-ratio, linear discriminant and principal component criterion, are employed and their corresponding performances in GMCE training are evaluated.

6.3 Criteria for Initialization Procedure

The best criterion for the initialization procedure of GMCE training algorithm is unknown. In this section, we employed three types of feature extraction criteria in this procedure. The three criteria are F-ratio, linear discriminant and principle component criteria. The evaluation of each criterion is given correspondingly.

6.3.1 F-Ratio Criterion

Since the methods of initializing transformation matrix introduced in Section 6.1 are equivalent to selecting features from the original observation vectors, it is natural to regard the initialization procedure as a feature selection process. Thus feature selection methods can be applied to the initial process. As introduced in Chapter 2, F-ratio is a very common feature selection method. F-ratio selects features by finding the largest ratio of *between-class* S_B and *within-class* covariance S_W.

$$F\text{–}ratio = \frac{S_B}{S_W} \tag{6.1}$$

The features that can keep this ratio largest will be selected until required number is selected. The other features will be discarded from the new feature vectors. The transformation matrix initialized by F-ratio is a reduced-rank identity matrix. Figure 6.4 shows the results after employing F-ratio method for initialization of the transformation matrix. The database is still Deterding Vowels data, so that the results are comparable to the results shown in Figure 6.2.

6.3.2 Linear Discriminant Criterion

Linear discriminant criterion is defined as the linear functions $T^T x$ for which the criterion function:

$$J(T) = \frac{T^T S_B T}{T^T S_W T} \tag{6.2}$$

is maximum. It can be shown that the solution of this function is that the columns of T are eigenvectors of matrix $S_W^{-1} S_B$.

The transformation matrix initialized by linear discriminant criterion, T_0 is a full rank matrix, of which the columns are eigenvectors of $S_W^{-1} S_B$ and ordered by the value of eigenvalues. In feature dimensionality reduction tasks, the rank of T_0 is reduced by discarding the trailing eigenvectors.

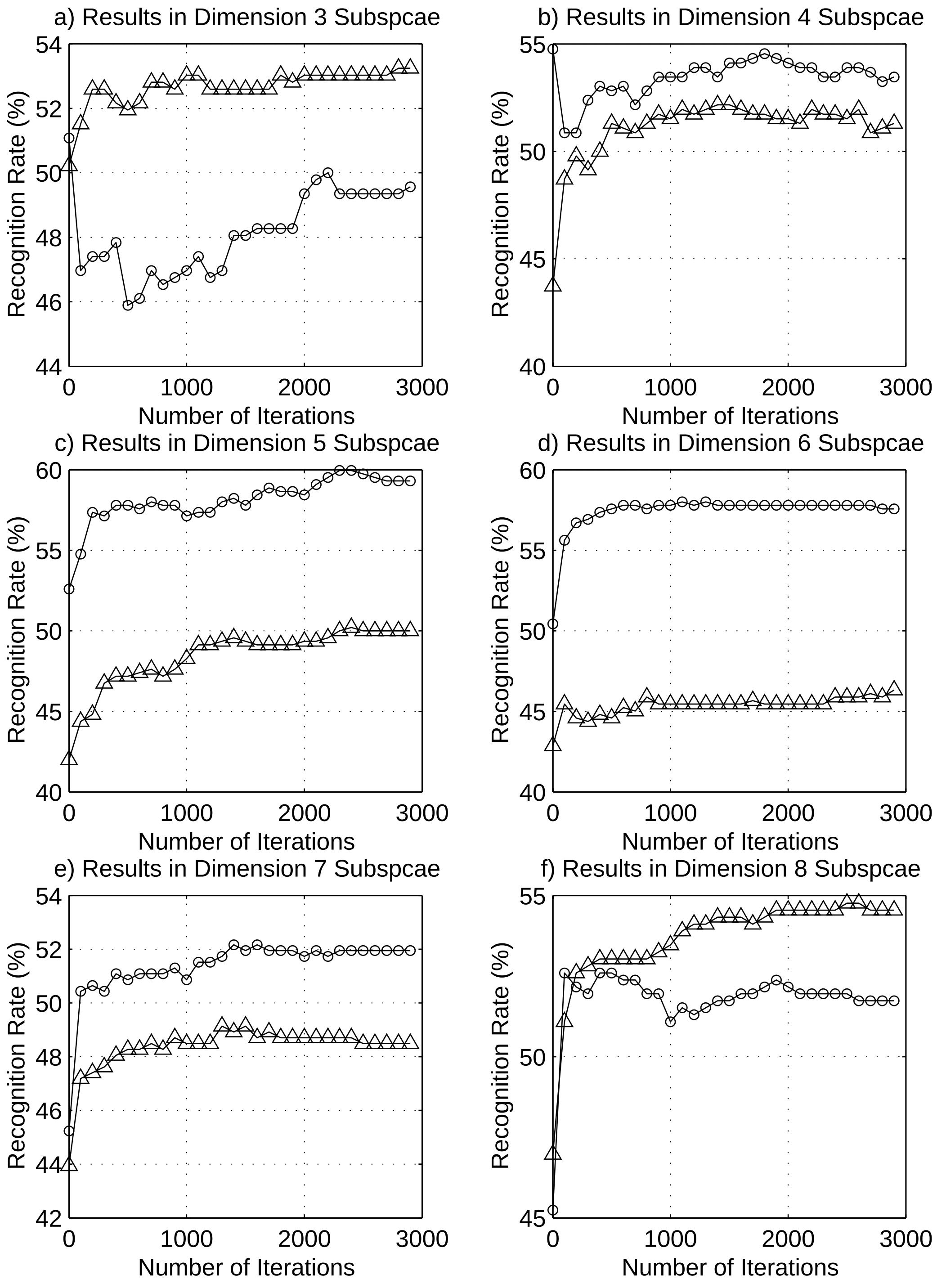


Figure 6.4: Results obtained by employing F-ratio method to initialize the transformation matrix on MCE training. (Deterding database : testing set) ◦ – Normal initialization method given in [76]; Δ – F-ratio initialization.

Figure 6.5 shows the recognition results of LDA, the alternative MCE training algorithm with the normal initialization method and the GMCE training algorithm with linear discriminant initialization criterion. The database used is Deterding Vowels database and the classifier is Mahalanobis distance classifier. In the figure, the GMCE training algorithm initialized with linear discriminant criterion is denoted as GMCE+LD. The alternative MCE training algorithm initialized the normal initialization method is denoted as MCE(alt).

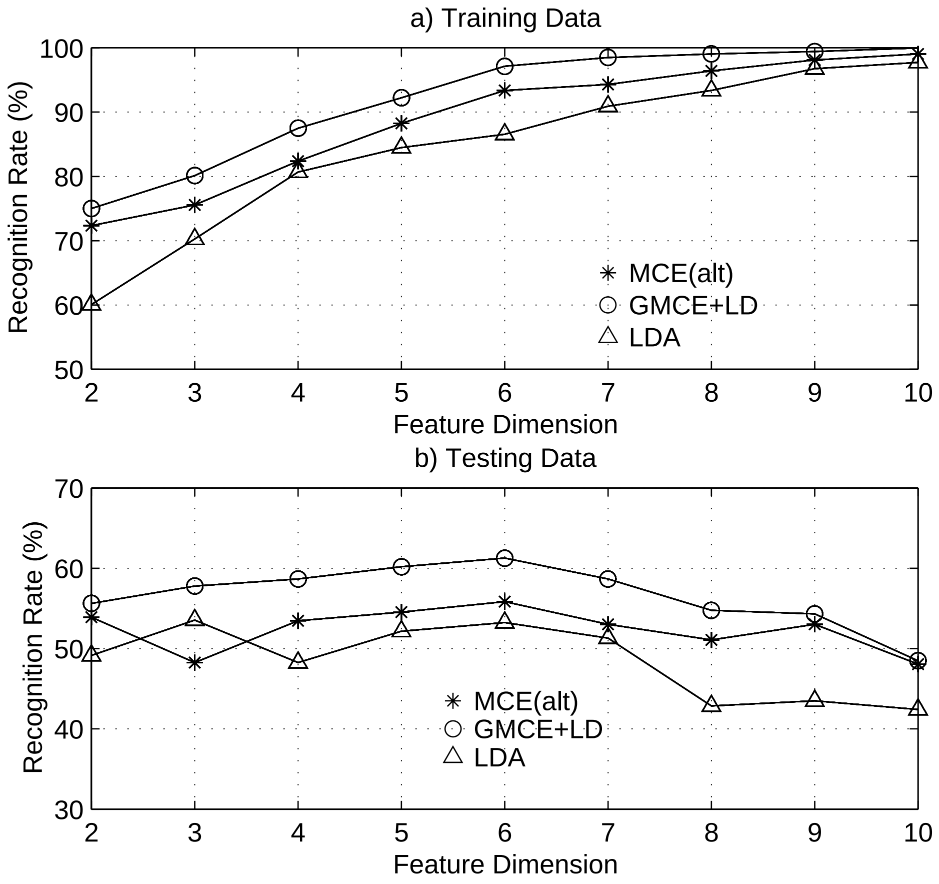


Figure 6.5: Comparison of the recognition rates of MCE(alt), GMCE+LD, LDA on Deterding Vowels database.

6.3.3 Principal Component Criterion

Principal component criterion searches for the directions onto which the global covariance matrix has the largest variate. Therefore the transformation matrix initialized by principal component criterion consists of the m leading eigenvectors of the global covariance matrix, where m is the required dimensionality of feature space.

Figure 6.6 shows the recognition results of PCA, the alternative MCE training algorithm with the normal initialization and the GMCE training algorithm with linear principal component criterion. The database used is Deterding Vowels database and the classifier is Mahalanobis distance classifier. In the figure, the GMCE training algorithm initialized with principal component criterion is denoted as GMCE+PC. The alternative MCE training algorithm initialized by the normal initialization method is denoted as MCE(alt).

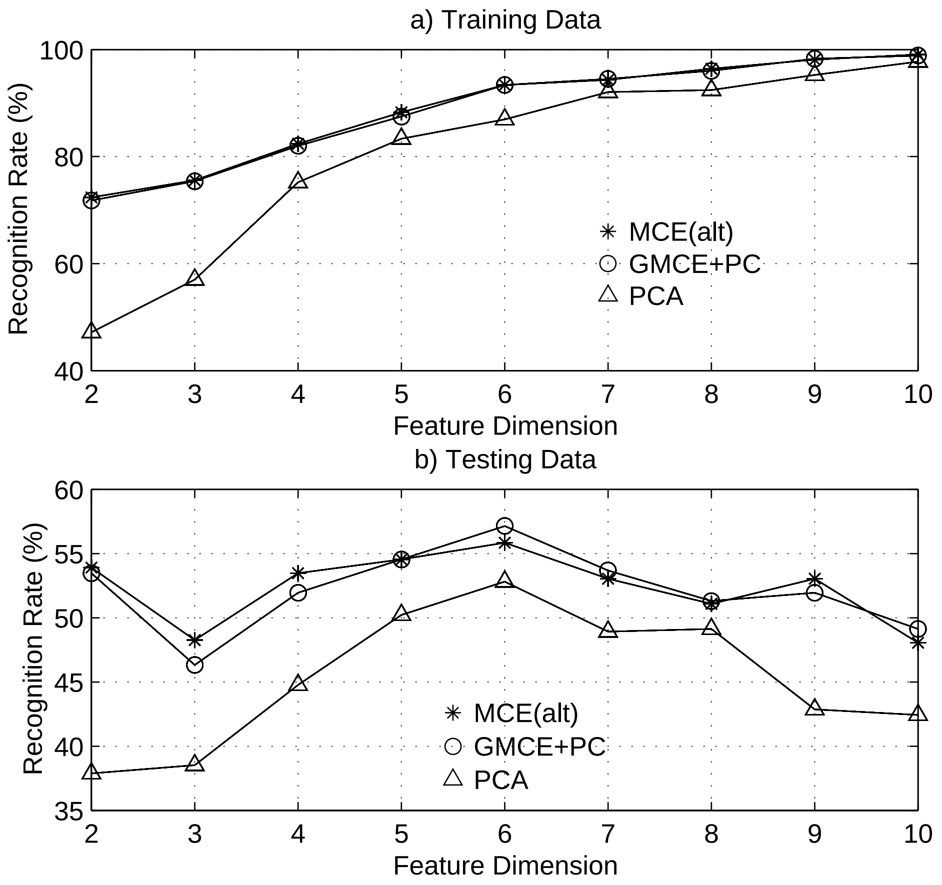


Figure 6.6: Comparison of the recognition rates of MCE(alt), GMCE+PC, PCA on Deterding Vowels database.

6.3.4 Evaluation on the Criteria

Figure 6.4 shows that the MCE training results obtained by F-ratio initialization criterion are better that the normal MCE training process only on dimension 3 and dimension 8. But its performances in other feature spaces (from dimension 4 to dimension 7) are poorer than those of normal MCE training process. This implies that F-ratio criterion is not stable in all dimensions and thus not suitable for the initialization procedure of GMCE training.

The results shown in Figure 6.5 are very encouraging. The performance of the GMCE training algorithm with linear discriminant initialization criterion is better than that of normal MCE training algorithm on all dimensions. The improvement of the performance of the GMCE training algorithm is consistent over dimensions.

Figure 6.6 shows that the performance of GMCE training with principal component initialization criterion is nearly the same as that of normal MCE training algorithm.

To further evaluate and compare the performances of the linear discriminant and principal component criteria, the two GMCE training algorithms are employed on the GLASS database. The corresponding recognition results are shown in Table 6.1.

DIM	MCE	GMCE+LD	GMCE+PC
2	77.9	81.0	82.8
3	83.4	82.2	84.0
4	80.4	82.8	83.4
5	80.4	84.7	82.8
6	82.8	84.1	82.2
7	83.4	82.8	82.8

Table 6.1: Results on GLASS data (in %).

Observations from the results on the two databases can be summarized as follows:

- Performances of GMCE training algorithm on both Deterding and GLASS databases are better than those of normal MCE training algorithm when linear discriminant criterion is used for the initialization of transformation matrix.

- The performance of GMCE training algorithm on GLASS database is better than that of normal MCE training algorithm when principal component criterion is used for the initialization of transformation matrix, while on the Deterding Vowels database, the two performances are nearly the same.

- A dramatic pattern can be observed from the results on Deterding and GLASS databases: the best performances of GMCE training algorithms on testing data do not appear on high dimensions, but on the dimensions that are around half of the full dimensionality.

6.4 Conclusion and Discussions

The results of GMCE training algorithms given in section 6.3 clearly show that a good initial estimation of the transformation matrix can improve the performance of MCE training algorithm. The proposed GMCE training algorithm is in fact a combination of general feature extraction algorithms and normal MCE training algorithms. The results on Deterding Vowels and GLASS databases show that the criterion for the initialization procedure is important to the success of GMCE training. The results also show that the linear discriminant criterion is better than the principal component and F-ratio criteria.

A clear pattern that can be observed from the results of both GMCE and MCE training algorithms on the testing set is that the best results on the testing data mostly appear on the dimensions that are around $50\% \sim 70\%$ of the original dimensionality. When the dimension is either lower or higher than this region, the performances of the algorithms start degrading. Similar patterns can also be observed from the performances of LDA and PCA. But they are not as clear as from the performances of GMCE and MCE training algorithms.

This pattern is very similar to the facts found in Chapter 5 and Brunzell and Eriksson's work [16]. It seems against our common understanding of the feature dimensionality reduction process. Our common understanding is that the performance of the classification algorithm is degraded with the decrease of the dimensionality because of the information loss with the reduction of features. This common understanding is true in the model training process because classes and their observations are known. Thus the information that discriminates classes is known so that feature dimensionality reduction algorithms can remove features in the order of the amount of discriminative information the features carry. In the testing process, however, the situation is different. The classes and discriminative information carried by features is unknown. Classification is largely dependent on the generalization of the class models obtained.

The results given in the previous section shows that feature dimensionality reduction does not necessarily lead to a degradation of performances of pattern classification systems. This implies that the generalization of class models is not linearly related to the dimensionality of features and it is possible that class models gain the largest generalization in a certain region of dimensionality. Thus the generalization of class models largely depends on the types of class information and their amount that are kept in the models. Generally speaking, the information included in features can be grouped into two types — one is discriminative information that discriminates the classes, and the other is confusing information that represents the similarity among classes. The generalization of class models relies on not only the amount of discriminative information but also the amount of confusing information kept in class models. The more the discriminative information and the less the confusing information kept in class models, the better the generalization of them. However, the main difficulty in linear feature extraction and dimensionality reduction is how to define or quantify the discriminative and confusing information properly. The discussion of this problem is beyond the scope of this thesis. We will leave it for our future research.

6.5 Summary of Chapter

This chapter first gives an example of how initial transformation matrix influences the MCE training process. Then a generalized MCE (GMCE) training algorithm is proposed. GMCE has a general initialization procedure searching for a suitable initial transformation matrix for the following MCE training procedure. F-ratio criterion, linear discriminant criterion and principal component criterion are used for the general initialization procedure. The results show that the linear discriminant criterion is more suitable for the initialization procedure.

Chapter 7

Support Vector Machine

7.1 Introduction

The Support Vector (SV) algorithm is a nonlinear generalization of the Generalized Portrait algorithm developed in Russia in 1960s by Vapnik and Lerner [96] and Vapnik and Chervonenkis [97]. SV algorithm is firmly grounded in the framework of statistical learning theory — VC theory, which improves the generalization ability of learning machines to unseen data [89]. The Support Vector Machine (SVM) was developed at AT& T Bell Laboratories [14, 23, 85, 99, 101]. Due to this industrial background, SVM had a sound character towards real-world applications. Initial work was focused on optical character recognition. Within a short period of time, SVM classifiers became competitive with the best available systems for optical character recognition and object recognition tasks [86, 87, 89]. SVM has now evolved into an active area of research.

SVM is a non-linear classification algorithm. It is a type of kernel method. Different from linear classification method, kernel method maps the original parameter vectors into a higher dimensional feature space through a non-linear kernel functions. The non-linear decision boundaries in parametric space may become non-linear in the feature space. One example, given by Burges [17], is that non-linear class distribution boundaries in parametric space can become linear in feature space, as shown in Figure 7.1. Decision planes are then pursuit in feature space. The projection of the linear decision planes in the parametric space is a non-linear decision boundaries. Thus SVM has advantages of handling the classes with complex distributions.

This chapter will discuss the formulation of SVM and design of SVM classifiers and evaluate the the performances of SVM classifiers in experiments based on Deterding Vowel database.

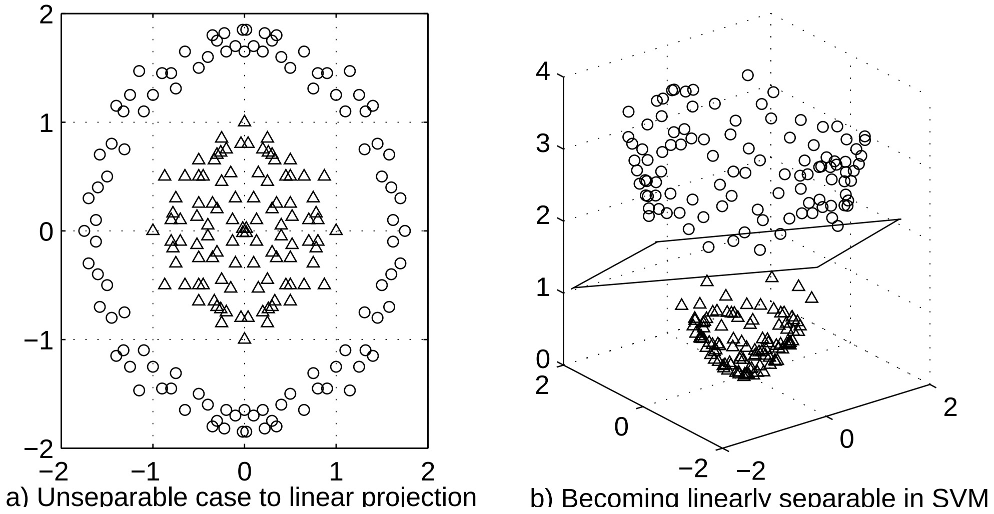


Figure 7.1: Unseparable case for conventional feature extraction methods but separable for SVM.

7.2 Formulation of SVM

7.2.1 Risk Minimization

Suppose we have a given set of training data $\mathcal{X} = \{(x_1, y_1), \cdots, (x_N, y_N)\} \subset \mathcal{R}^p \times \mathcal{R}$, where N is the total number of training data, $\mathcal{R}$ and $\mathcal{R}^p$ represent the real number space and p-dimensional real space, $x_i (i = 1, \ldots, N)$ is observation vector and $y_i (i = 1, \ldots, N)$ is the corresponding target of x_i. Assume that these training data have been drawn independently and identically distributed (iid) from some probability distribution $p(x, y)$. The goal of training is then to find a function f that minimizes the following risk function [98]:

$$R[f] = \int c(x, y, f(x)) dp(x, y) \tag{7.1}$$

where $c(x, y, f(x))$ denotes a cost function determining the penalty on estimation errors. In Eq. (7.1), $p(x, y)$ is unknown. A possible approximation of the risk would be to replace the integration with the empirical estimate. The empirical risk is given as follows:

$$R_{emp}[f] = \frac{1}{N} \sum_{i=1}^{N} c(x_i, y_i, f(x_i)) \tag{7.2}$$

The empirical risk function has the advantage of being easy to compute and a uniformly consistent hypothesis of classes with bounded complexity [98]. However, direct minimization of $R_{emp}[f]$ may lead to heavy overfitting, that is, poor generalization in the case of a very powerful class of models [90].

Hence, a capacity control term $\mathcal{T}(f)$ should be added to the empirical risk function, which leads to the regularized risk function:

$$R_{reg}[f] = R_{emp}[f] + \lambda \mathcal{T}(f) \tag{7.3}$$

where λ is the regularization constant to control the trade-off between model complexity and approximation in order to ensure a good generalization performance [12, 38, 90].

7.2.2 Cost function

The objective of SVM training is to find a function f, such that $f(x)$ is as close to y as possible. Suppose the estimation error is ξ, then $\xi = y - f(x)$. Cost function is chosen to determine how we penalize the estimate errors. There are mainly two considerations when choosing cost functions. One is that the cost function should not lead to difficult optimization problem nor be computationally expensive. The other consideration is that the cost function should keep the optimization problems convex programming problems.

The standard setting of a cost function in SVM is the so-called Vapnik's ϵ-insensitive cost function, which inherited from [99]. Given an estimate $f(x_i)$ and a measurement y_i, the estimation error ξ is penalized by $|\xi|_\epsilon = |y_i - f(x_i)|_\epsilon$ with:

$$c(\xi) = \begin{cases} 0 & for\ |\xi| < \epsilon \\ |\xi| - \epsilon & otherwise \end{cases} \tag{7.4}$$

where $\epsilon \geq 0$. The advantage of this cost function is that it leads to sparse decompositions and quadratic programming problems[91]. The restriction to $c(\xi) = |\xi|_\epsilon$, however, sometimes is too strong and can not lead to a good minimization of $R[f]$ [90]. Under the assumption that the samples were generated by a functional dependency $f(x_i)$ and additive noise ξ_i with density $p(\xi)$, the cost function can be chosen in a maximum likelihood sense as:

$$c(\xi) = -log(p(\xi)) \tag{7.5}$$

This would be desirable to extend the class of different cost functions for SVM regression. Table 7.1 gives some common density models and the corresponding cost function derived from Eq.(7.4).

7.2.3 Constructing SVM

Consider a two-class case. Suppose the two classes are ω_1 and ω_2 and we have a given set of training data $\mathcal{X} = \{x_1, \cdots, x_N\} \subset \mathcal{R}^p$. Training data are labeled by the following rule:

$$y_i = \begin{cases} +1 & if\ x_i \in \omega_1 \\ -1 & if\ x_i \in \omega_2 \end{cases} \tag{7.6}$$

Name	Cost Function	Density Model
ϵ-insensitive	$c(\xi) = \|\xi\|_\epsilon$	$p(\xi) = \frac{1}{2(1+\epsilon)} exp(-\|\xi\|_\epsilon)$
Laplacian	$c(\xi) = \|\xi\|$	$p(\xi) = \frac{1}{2} exp(-\|\xi\|)$
Gaussian	$c(\xi) = \xi^2/2$	$p(\xi) = \frac{1}{\sqrt{2\pi}} exp(-\xi^2/2)$
Polynomial	$c(\xi) = \frac{1}{p}\|\xi\|^p$	$p(\xi) = \frac{p}{2\Gamma(1/p)} exp(-\|\xi\|^p)$

Table 7.1: Common density models and corresponding cost functions.

The basic idea of SVM estimation is to project the input observation vectors non-linearly into a high dimensional feature space $\mathcal{F}$ and then compute a linear function in $\mathcal{F}$. The functions take the form:

$$f(x) = (w \cdot \Phi(x)) + b \tag{7.7}$$

with

$$\Phi : \mathcal{R}^p \to \mathcal{F} \quad and \quad w \in \mathcal{F} \tag{7.8}$$

where $(\cdot)$ denotes the dot product, $w = \{w_1, \cdots, w_p\}$ are weights to each $\Phi(x)$ and b is a linear constant. Ideally, all the data in these two classes satisfy the following constraints:

$$\begin{array}{ll} (w \cdot \Phi(x_i)) + b \geq +1 & for\ y_i = +1 \\ (w \cdot \Phi(x_i)) + b \leq -1 & for\ y_i = -1 \end{array} \tag{7.9}$$

These two inequations can be combined into one inequality:

$$y_i(w \cdot \Phi(x_i)) + b - 1 \geq 0 \quad \forall i \tag{7.10}$$

Consider the points $\Phi(x_i)$ in $\mathcal{F}$ for which the equality in (7.9) holds. These points lie on two hyperplanes $H_1 : (w \cdot \Phi(x_i)) + b = +1$ and $H_2 : (w \cdot \Phi(x_i)) + b = -1$. These two hyperplanes are parallel and no training points fall between them. The margin between the two planes is $\frac{2}{||w||}$. Therefore we can find a pair of hyperplanes with maximum margin by minimizing $||w||^2$ subject to (7.10)[17]. This problem can be written as a convex optimization problem:

$$\begin{array}{ll} minimize & \frac{1}{2}||w||^2 \\ subject\ to & y_i(w \cdot \Phi(x_i)) + b - 1 \geq 0 \quad \forall i \end{array} \tag{7.11}$$

where the first function is called *primal* objective function in convex optimization problems and the second function is the corresponding constraints. Naturally the capacity control term $\mathcal{T}(f)$ in Eq.(7.3) would be the primal function:

$$\mathcal{T}(f) = \frac{1}{2}||w||^2 \tag{7.12}$$

Consequently the regularized risk function becomes:

$$R_{reg}[f] = \frac{1}{N}\sum_{i=1}^{N} c(y_i - f(x_i)) + \frac{\lambda}{2}||w||^2 \tag{7.13}$$

7.2.4 Convex Programming Problem

First consider Eq. (7.11). It can be solved by constructing a Lagrange function from both the primal function and the corresponding constraints, by introducing dual variables. It has been proved that the Lagrange function has a saddle point at the optimal with respect to the primal and dual variables[89, 102]. Hence we introduce positive Lagrange multipliers $\alpha_i, i = 1, \cdots, N$, one for each constraints in Eq.(7.11). The Lagrangian is given by:

$$L_P = \frac{1}{2}||w||^2 - \sum_{i=1}^{N} \alpha_i y_i (x_i \cdot w + b) + \sum_{i=1}^{N} \alpha_i \tag{7.14}$$

L_P must be minimized with respect to w and b, which requires the gradient of L_P to vanish with respect to w and b. Hence the condition:

$$\frac{\partial L_P}{\partial w_s} = w_s - \sum_{i=1}^{N} \alpha_i y_i x_{is} = 0 \qquad s = 1, \cdots, p \tag{7.15}$$

$$\frac{\partial L_P}{\partial b} = -\sum_{i=1}^{N} \alpha_i y_i = 0 \tag{7.16}$$

where p is the dimension of space $\mathcal{F}$. Combine these conditions and other constraints on primal functions and Lagrange multipliers, we obtain the $Karush-Kuhn-Tucker$ (KKT) conditions. For the primal problems, the KKT conditions are stated as follows:

$$\frac{\partial L_P}{\partial w_s} = w_s - \sum_{i=1}^{N} \alpha_i y_i x_{is} = 0 \qquad s = 1, \cdots, p \tag{7.17}$$

$$\frac{\partial L_P}{\partial b} = -\sum_{i=1}^{N} \alpha_i y_i = 0 \tag{7.18}$$

$$y_i(w \cdot \Phi(x_i)) + b - 1 \geq 0 \quad \forall i \tag{7.19}$$

$$\alpha_i \geq 0 \quad \forall i \tag{7.20}$$

$$\alpha_i(y_i(w \cdot \Phi(x_i)) + b - 1) = 0 \quad \forall i \tag{7.21}$$

where w, b and α are the variables to be solved. The KKT conditions are necessary and sufficient for solving problems for SVMs since they are convex and the constraints are always linear[30]. There are several approaches to finding the solution of Eq.(7.17) $\sim$ (7.21). Among them, the primal-dual path following method is a popular and successful method. It will be discussed in the next section.

In most cases, the primal objective function in Eq. (7.11) is sufficient. However in some cases we allow for some estimation errors. Thus we achieve the regularized risk function expressed in Eq.(7.13) by introducing the penalized estimation errors into the primal function. Denoting the estimation error as $\xi = y_i - f(x_i)$, we can construct the convex optimization problem from Eq.(7.13):

$$\begin{array}{ll} \textit{minimize} & \frac{1}{2}||w||^2 + \frac{1}{N}\sum_{i=1}^{N} c(\xi) \\ \textit{subject to} & y_i(w \cdot \Phi(x_i)) + b - 1 \geq 0 \quad \forall i \end{array} \tag{7.22}$$

Then the Lagrange function is:

$$L_P = \frac{1}{2}||w||^2 + \frac{1}{N}\sum_{i=1}^{N} c(\xi) - \sum_{i=1}^{N} \alpha_i y_i (x_i \cdot w + b) + \sum_{i=1}^{N} \alpha_i \tag{7.23}$$

KKT conditions can be constructed consequently by the same process.

7.2.5 Dual Function

From KKT condition (7.17) and (7.18) we obtain:

$$w = \sum_{i=1}^{N} \alpha_i y_i \Phi(x_i) \tag{7.24}$$

and

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \tag{7.25}$$

Therefore,

$$f(x) = \sum_{i=1}^{N} \alpha_i y_i (\Phi(x_i) \cdot \Phi(x)) + b = \sum_{i=1}^{N} \alpha_i y_i k(x_i, x) + b \tag{7.26}$$

where $k(x_i, x)$ is a kernel function and defined as a dot product in the feature space:

$$k_{ij} = k(x_i, x_j) = (\Phi(x_i) \cdot \Phi(x_j)) \tag{7.27}$$

Substitute Eq.(7.24) and Eq.(7.25) into Eq.(7.14). This leads to the maximization of the dual function L_D:

$$L_D = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j k_{ij} + \sum_{i=1}^{N} \alpha_i \tag{7.28}$$

Writing the dual function incorporating the constraints, we obtain the dual optimization problem:

$$\begin{array}{ll} \textit{maximize} & -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j k_{ij} + \sum_{i=1}^{N} \alpha_i \\ \textit{subject to} & \sum_{i=1}^{N} \alpha_i y_i = 0 \\ & \alpha_i \geq 0 \quad \forall i \end{array} \tag{7.29}$$

Both the primal problem L_P and the dual problem L_D are constructed from the same objective function but with different constraints. The solution can be found by either minimizing L_P or maximizing L_D. Furthermore the solution by minimizing L_P with respect to w and b is bounded to the solution by maximizing L_D with respect to α. Since there is a Lagrange multiplier α_i for every feature vector, in the solution, the feature vectors with $\alpha_i > 0$ are called "support vectors" and lie on either hyperplane H_1 or H_2. These support vectors are critical to the SVM because they are the closest training vectors to the decision boundary(for the separable case). If all other training vectors were removed, the same separating hyperplane would be found [17].

7.3 Primal-Dual Path Following Method for Optimizing SVM

7.3.1 Primal-Dual Formulation

In order to be consistent with standard notation for quadratic optimization problem, Eq.(7.29) can be rewritten in minimization form and matrix notation as:

$$\begin{aligned} &\textit{minimize} && \tfrac{1}{2}\alpha^T D\alpha - \alpha \cdot \mathbf{1} \\ &\textit{subject to} && \alpha \cdot \mathbf{y} = 0 \\ & && \alpha \geq 0 \\ & && \alpha \leq \mathbf{C} \end{aligned} \tag{7.30}$$

where $\mathbf{1} = [1, 1, \cdots, 1]^T$, $\alpha = [\alpha_1, \cdots, \alpha_N]^T$, $\mathbf{y} = [y_1, \cdots, y_N]^T$, $\mathbf{C}$ is the upper bound of α and D is a $N \times N$ symmetric matrix with elements $D_{ij} = y_i y_j k_{ij}$. Since matrix D is positive semi-definite and the constraints are linear, the KKT conditions are necessary and sufficient for optimality [72, 73, 102]. Before setting up KKT conditions, we first add slack variables to remove all inequalities from Eq.(7.30). This yields:

$$\begin{aligned} &\textit{minimize} && \tfrac{1}{2}\alpha^T D\alpha - \alpha \cdot \mathbf{1} \\ &\textit{subject to} && \alpha \cdot \mathbf{y} = 0 \\ & && \alpha - g = 0 \\ & && \alpha + t = \mathbf{C} \end{aligned} \tag{7.31}$$

The KKT conditions are therefore:

$$\begin{array}{l} \nabla(\frac{1}{2}\alpha^T D\alpha - \alpha \cdot \mathbf{1}) + \mu\mathbf{y} - \Pi + \Upsilon = 0 \\ \Pi(\alpha - g) = 0 \\ \Upsilon(\alpha + t - \mathbf{C}) = 0 \\ \Pi \geq 0 \\ \Upsilon \geq 0 \\ \alpha \cdot \mathbf{y} = 0 \\ \alpha - g = 0 \\ \alpha + t = \mathbf{C} \end{array} \tag{7.32}$$

Then the Wolfe dual of Eq.(7.31) is:

$$\begin{array}{ll} \textit{maximize} & -\frac{1}{2}\alpha^T D\alpha + \mathbf{C}^T\Upsilon \\ & \\ \textit{subject to} & \nabla(\frac{1}{2}\alpha^T D\alpha - \alpha \cdot \mathbf{1}) + \mu\mathbf{y} - \Pi + \Upsilon = 0 \\ & \alpha \cdot \mathbf{y} = 0 \\ & \alpha - g = 0 \\ & \alpha + t = \mathbf{C} \end{array} \tag{7.33}$$

Moreover, since the set of primal and dual variables that is both feasible and satisfies the KKT conditions is the optimal solution[89, 102], we have: constraint $\times$ dual variable $= 0$, which is:

$$\begin{array}{ll} g_i\Pi_i = 0 & for\ all\ i \in [1, \ldots, n] \\ t_i\Upsilon_i = 0 & for\ all\ i \in [1, \ldots, n] \end{array} \tag{7.34}$$

An optimal solution to be found is that both the primal variable α and the dual variable μ satisfy the feasible conditions of Eq. (7.31) and Eq. (7.33) and KKT conditions of Eq. (7.34).

7.3.2 Iteration Strategy — Path-Following Method

We will use path-following method to solve Eq. (7.31) and Eq. (7.33) and KKT conditions of Eq. (7.34). In this method, we will not try to satisfy the KKT conditions as it is, but to solve the relaxed conditions for some $\delta > 0$ and then decrease δ while iterating, that is:

$$\begin{array}{ll} g_i\Pi_i = \delta & for\ all\ i \in [1, \ldots, n] \\ t_i\Upsilon_i = \delta & for\ all\ i \in [1, \ldots, n] \end{array} \tag{7.35}$$

This can be done by linearizing the above equations and solving them by a two-step predictor-corrector approach until the duality gap is small enough.

first we rewrite the primal and dual formulation and KKT conditions as:

$$\begin{array}{l}
(\alpha + \Delta\alpha)y = 0 \\
\alpha + \Delta\alpha - g - \Delta g = 0 \\
\alpha + \Delta\alpha + t + \Delta t = C \\
\frac{1}{2}\nabla(\alpha^T D\alpha) + \frac{1}{2}\nabla^2(\alpha^T D\alpha)\Delta\alpha + (\mu + \Delta\mu)\mathbf{y} - \Pi - \Delta\Pi + \Upsilon + \Delta\Upsilon = 1 \\
(g + \Delta g)(\Pi + \Delta\Pi) = \delta \\
(t + \Delta t)(\Upsilon + \Delta\Upsilon) = \delta
\end{array} \tag{7.36}$$

Then we solve Eq. (7.36) for the variables in Δ. We obtain:

$$\begin{array}{ll}
y\Delta\alpha & = \alpha y \\
\Delta\alpha - \Delta g & = g - \alpha \\
\Delta\alpha + \Delta t & = C - \alpha - t \\
\frac{1}{2}\nabla^2(\alpha^T D\alpha)\Delta\alpha + y\Delta\mu - \Delta\Pi + \Delta\Upsilon & = 1 - \frac{1}{2}\nabla(\alpha^T D\alpha) - y\mu + \Pi - \Upsilon \\
g^{-1}\Pi\Delta g + \Delta\Pi & = \delta g^{-1} - \Pi - g^{-1}\Delta g\Delta\Pi \\
t^{-1}\Upsilon\Delta t + \Delta\Upsilon & = \delta t^{-1} - \Upsilon - t^{-1}\Delta t\Delta\Upsilon
\end{array} \tag{7.37}$$

where $g^{-1} = [\frac{1}{g_1}, \cdots, \frac{1}{g_n}]$, $t^{-1} = [\frac{1}{t_1}, \cdots, \frac{1}{t_n}]$, $g^{-1}\Pi = [\frac{\Pi_1}{g_1}, \cdots, \frac{\Pi_n}{g_n}]$ and $t^{-1}\Upsilon = [\frac{\Upsilon_1}{t_1}, \cdots, \frac{\Upsilon_n}{t_n}]$. Before going further, we define:

$$\begin{array}{ll}
q_\Pi & := \delta g^{-1} - \Pi - g^{-1}\Delta g\Delta\Pi \\
q_\Upsilon & := \delta t^{-1} - \Upsilon - t^{-1}\Delta t\Delta\Upsilon
\end{array} \tag{7.38}$$

Solving Eq. (7.38) for $\Delta g, \Delta t, \Delta\Pi, \Delta\Upsilon$, we obtain:

$$\begin{array}{ll}
\Delta g & = g\Pi^{-1}(q_\Pi - \Delta\Pi) \\
\Delta t & = t\Upsilon^{-1}(q_\Upsilon - \Delta\Upsilon) \\
\Delta\Pi & = g^{-1}\Pi(g - \alpha - g\Pi^{-1}q_\Pi - \Delta\alpha) \\
\Delta\Upsilon & = t^{-1}\Upsilon(\Delta\alpha - C + \alpha + t + t\Upsilon^{-1}q_\Upsilon)
\end{array} \tag{7.39}$$

Again define:

$$\begin{array}{ll}
w & = 1 - \frac{1}{2}\nabla(\alpha^T D\alpha) - y\mu + \Pi - \Upsilon \\
\nu & = g^{-1}\Pi(g - \alpha - g\Pi^{-1}q_\Pi) \\
\tau & = t^{-1}\Upsilon(C - \alpha - t - t\Upsilon^{-1}q_\Upsilon)
\end{array} \tag{7.40}$$

Then a $reducedKKT - system$ can be formulated as:

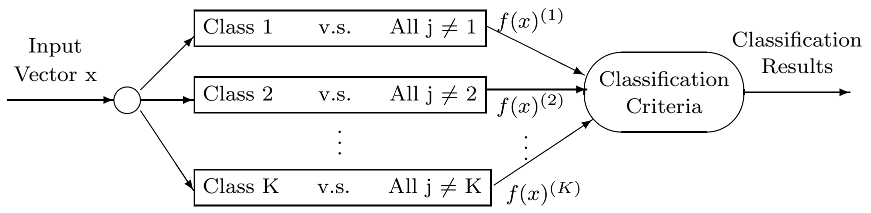
$$\begin{bmatrix} -(\frac{1}{2}\nabla^2(\alpha^T D\alpha) + g^{-1}\Pi + t^{-1}\Upsilon) & y \\ y & 0 \end{bmatrix} \begin{bmatrix} \Delta\alpha \\ \Delta\mu \end{bmatrix} = \begin{bmatrix} w - \nu - \tau \\ \alpha y \end{bmatrix} \tag{7.41}$$

In the predictor step Eq. (7.39) and (7.41) are solved with $\delta = 0$ and all Δ-terms on the R.H.S. are set to 0. In corrector step the values of Δ-terms are substituted back and solve Eq. (7.39) and (7.41) again. The values of these Δ-terms obtained by this iteration process are used to update the corresponding values of α, μ, g, t, Π and Υ. The above is a simplified SVM system, which has been used by a number of researchers [18, 32, 37, 72, 73, 74]. For a more complex SVM system, the conditions in Eq. (7.31) are relaxed in two aspects: 1) the linear function of α does not have to pass the origin and 2) the lower bound of α does not have to be restricted to 0. [89] and [102] have a detailed discussion on such a SVM system.

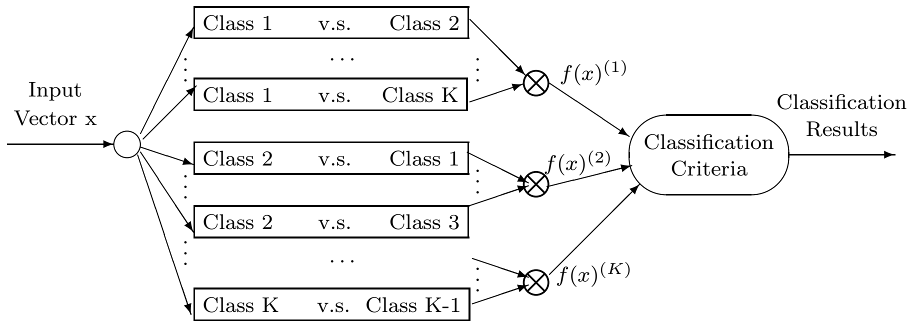
7.4 Results on Small Databases

7.4.1 Multi-classes Classes Classifier

SVM is a two-class based training algorithm. It is not applicable for multi-class cases. Therefore we have to expand the SVM classifier to multi-class classifiers. So far the best method of extending the two-class classifiers to multi-class problems is not clear [18]. Scholkopf and etc.[85] generally constructed a "one vs. all" classifier for each class. Clarkson and Moreno [18] proposed a construction of a "one vs. one" classifier for each pair of classes. The two types of classifiers are shown in Figure 7.2.



(a) Structure of "one vs. all" multi-class SVM Classifier



(b)Structure of "one vs. one" multi-class SVM classifier

Figure 7.2: Two types of multi-class SVM classifier.

These two types of classifiers have similar structures. Both of them have a feature extractor for each class. When an observation vector x enters the system, each extractor will generate an output $f^{(i)}(x), i = 1, \cdots, K$. The classifier then classifies x by the following classification criterion:

$$x \in Class\ i \quad if\ f^{(i)}(x) = \max_{for\, all\, j \in K} f^{(j)}(x) \tag{7.42}$$

The difference between "one vs. all" classifier and "one vs. one" classifier is that the latter has a more complicated sub-structure. The extractor of each class in the "one vs. one" classifier consists of $K-1$ sub-extractors, which combine the target class and all other classes in pairs. Each sub-extractor will generate a score $f^{(i,j)}(x)$ for an input vector. These scores are combined to generate the final output $f^{(i)}(x)$ for classification. So far the best way to combine $f^{(i,j)}(x)$ is not clear. A straight-forward way is to calculate $f^{(i)}(x)$ by $f^{(i)}(x) = \sum_{for\ all\ j \neq i} f^{(i,j)}(x)$. However it is clear that such an additive combination is easy to bring undesired information into $f^{(i)}(x)$. In this thesis, a statistical normalization method is used to calculate $f^{(i)}(x)$. In this method, we define $f^{(i)}(x)$ as the normalized mean of $f^{(i,j)}(x)$, which is:

$$f^{(i)}(x) = \frac{\mu_i}{\sigma_i} \tag{7.43}$$

where μ_i is the mean of $f^{(i,j)}(x)$, σ_i is the corresponding standard deviation.

7.4.2 Classification Results

Our classification experiments focus on the vowel classification tasks. Deterding Vowels database is used for classification. So far the kernel function $k(x_i, x_j)$ in Eq. (7.27) has not been defined. Rewrite Eq. (7.27) as:

$$k(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j) \tag{7.44}$$

where Φ is a map, which can be either linear or non-linear. In the linear case, a simple choice would be:

$$\begin{array}{ll} \Phi : \mathcal{R}^n \mapsto \mathcal{H} & \Phi(x) = x \\ k(x,y) = (x \cdot y) & \end{array} \tag{7.45}$$

Then the kernel function would be: $k(x_i, x_j) = x_i \cdot x_j$. In the non-linear case, the map $\Phi : \mathcal{R}^n \mapsto \mathcal{H}$ is chosen to map the feature points to a higher dimensional space $\mathcal{H}$. However explicitly computing the non-linear map $\Phi(x)$ is very difficult and computationally expensive. Since $\Phi(x)$ only appears in the feature space $\mathcal{R}^n$ while the computation is carried out in the higher dimensional space $\mathcal{H}$ and deals with the data in the form of $(\Phi(x_i) \cdot \Phi(x_j))$, we do not have to calculate $\Phi(x)$ explicitly. Instead, we compute the kernel function $k(x_i, x_j)$. There are many different definitions on the kernel function. Two popular forms are polynomial kernel and Gaussian radial basis function (RBF). Their definitions are:

$$\begin{array}{rl} Polynomial: & k(x,y) = (x \cdot y + 1)^p \\ RBF: & k(x,y) = e^{|x-y|^2/2\delta^2} \end{array} \tag{7.46}$$

Kernel	Classifier	Classification Rate (Training Data)	Classification Rate (Testing Data)
Linear	one vs. all	49.43%	40.91%
Linear	one vs. one	79.73%	53.03%
Polynomial	one vs. all	59.85%	42.42%
Polynomial	one vs. one	90.53%	55.63%
RBF	one vs. all	78.98%	51.95%
RBF	one vs. one	90.34%	58.01%

Table 7.2: Deterding Vowels database classification results.

Table 7.2 shows the classification results of using SVM in the Deterding Vowels database. In this classification task, linear, polynomial and RBF kernel functions are used. The degree of polynomial kernel function is 3. The construction of multi-class classifier uses both "one vs. all" and "one vs. one" classifiers. Some observations can be made from the results. These observations are:

- Linear kernel function does not work well at all in classification tasks. The corresponding performance is very poor.
- The performance of polynomial kernel function is comparable to that of RBF.
- The "one vs. one" multi-class classifier shows a better performance than "one vs. all" multi-class classifier no matter what type of kernel functions is used.
- The results of SVM with RBF kernel function and "one vs. one" multi-class classifier are comparable to those of MCE training algorithms shown in Chapter 5.

7.4.3 Conclusion

SVM extracts features by projecting the feature vectors with a map, which can be either linear map or non-linear kernel functions, into a higher dimensional space. In the higher dimensional space, SVM tries to represent the class by the samples which are the closest to the boundary rather than the conventional statistical class parameters, such as means and covariance. This enables the class models to have a more accurate class boundary than before. These features make SVM suitable for handling complex classes. In the experiments on Deterding Vowels database, the performance of SVM is comparable to that of linear feature extraction algorithms, such as LDA, PCA and MCE training algorithm.On the other hand, SVM has its limitations. The main limitation is that SVM is a two-class based algorithm.

Users have to construct a multi-class classifier over SVM for multi-class classification cases. However the best way to construct the multi-class classifier is not known yet.

7.5 Summary of Chapter

In this chapter, we discussed Support Vector Machine (SVM) for feature extraction and employed it on vowel classification tasks. The results show that SVM is comparable to linear feature extraction algorithms, such LDA, PCA and MCE training algorithms.

Chapter 8

Reduced-Dimensional SVM

8.1 Introduction

The core of SVM is to map observation vectors into a high dimensional feature space $\mathcal{H}$ by a non-linear map $\Phi(x)$:

$$\Phi : \mathcal{R}^n \mapsto \mathcal{H} \tag{8.1}$$

SVM then uses convex programming technique to optimize the objective function in the feature space $\mathcal{H}$. The objective function is ensured to be a convex programming problem by a kernel function $k(x, y)$, which is defined as a dot product of $\Phi(x)$:

$$k(x, y) = \Phi(x) \cdot \Phi(y) \tag{8.2}$$

Thus the optimization problem in SVM becomes a quadratic programming problem as expressed in Eq. (7.30). Rewrite the objective function of Eq. (7.30) as follows:

$$\frac{1}{2}\alpha^T D\alpha - \alpha \cdot \mathbf{1} \tag{8.3}$$

where $\mathbf{1} = [1, 1, \cdots, 1]^T_{1\times N}$, $\alpha = [\alpha_1, \cdots, \alpha_N]^T$, D is a $N \times N$ symmetric matrix with elements $D_{ij} = y_i y_j k_{ij}$ and N is the number of observation vectors. The quadratic term in Eq. (8.3) shows that each observation vector becomes a dimension of $\mathcal{H}$ after mapping. The total number of elements in D is N^2, which means that there are at least N^2 times kernel computation in one SVM training. In some large speech databases, such as TIMIT and Resource Management databases, the number of observations is more than 10,000. Suppose the dimensionality of observation vectors is 40 and the kernel function is a linear kernel, then 40 times multiplication and 39 times addition are needed to finish a kernel computation. Altogether more than 100 Mbytes memory, 4 $\times 10^9$ times multiplication and 3.9 $\times 10^9$ times addition are needed to employ SVM training on these databases. This will be a large burden for any computing systems.

Osuna [72, 73, 74] and Joachims [52] proposed a method to reduce the consumption on computational resources of SVM by bringing the concept "active set" into SVM training. In this method, the observation vectors are divided into two sets. One is active and the other is non-active. Only the observation vectors in active set participate in SVM training. This method can effectively reduce the number of N. However, in many cases, N has to be large enough to ensure the robustness of training and the generalization of models.

Apart from the problem of computational burden, another problem with SVM is that it is a two-class based feature extraction and classification method. In multi-class cases, the SVM classifier has to be constructed based on two-class SVM models as discussed in Chapter 7. However, a two-class SVM model trained on a certain pair of classes may be completely not applicable to other classes. Thus unexpected errors may be brought into the multi-class SVM classifier.

In this chapter we propose a reduced-dimensional SVM algorithm (RDSVM) both as a supplementary to Osuna and Joachims' method and to reduce the possibility of errors of two-class SVM models entering the multi-class SVM classifier.

8.2 Reduced-Dimensional SVM

The basic idea of RDSVM to reduce computational burden is that the total number of computations of SVM can be reduced by reducing the number of computations in kernel functions, since the number of observation vectors N can not be reduced to a very low level in many cases. An effective way of reducing the number of computations in kernel functions is to reduce the dimensionality of observation vectors.

Since SVM is an original two-class algorithm, it is hardly possible to modify the whole algorithm into a multi-class algorithm. In practice, however, we can reduce the negative effects of its two-class originality in the multi-class SVM classifier. For example, one of the major problems encountered by multi-class SVM classifier is in unseparable cases. When two classes overlap with each other, SVM model is unable to handle the overlapping area. While using the discriminative learning techniques discussed in previous chapters, the overlapping area can be reduced to its minimum. These examples and facts naturally lead us to the consideration of combining the advantages of discriminative learning and SVM together.

RDSVM is in fact a combination of discriminative learning and SVM algorithms. It has a two-layer structure. The first layer conducts discriminative learning, of which the objective is to reduce the dimensionality of feature space and obtain the largest discriminants of classes. The second layer conducts SVM training in the reduced-dimensional feature space, which is provided by the first layer. Thus the kernel functions will be calculated as follows:

$$k(x^{'}, y^{'}) = \Phi(x^{'}) \cdot \Phi(y^{'}) = \Phi(T^T x) \cdot \Phi(T^T y) = k(T^T x, T^T y) \tag{8.4}$$

where $x^{'}$ and $y^{'}$ are feature vectors in the reduced-dimensional feature space, x and y are observation vectors and T is the transformation optimized by the first layer. Figure 8.1 shows the structure of RDSVM.

The GMCE training algorithm with linear discriminant initialization criterion is selected for the discriminant learning in the first layer in the proposed RDSVM algorithm, since the GMCE training algorithm has shown the best performance among other feature extraction and dimensionality reduction algorithms in the classification tasks discussed in Chapter 6.

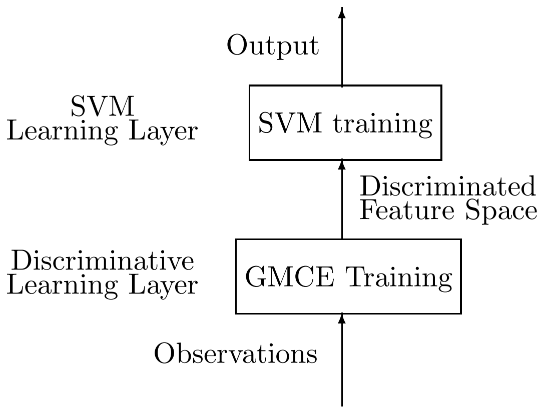


Figure 8.1: Reduced-dimensional SVM.

8.3 Experiment Result on Deterding Vowels Database

As with previous chapters, RDSVM is applied to Deterding Vowels database. The feature dimensions used in the experiment are from 2 to 10 (full dimension). Figure 8.2 gives a comparison of the results of GMCE training algorithm, LDA, SVM and RDSVM. Since SVM can only be operated in the observation space, i.e. dimension 10, its results are presented as dots on dimension 10. Observations from the performance of RDSVM can be drawn as follows:

- Compared to SVM, the performance of RDSVM on dimension 10 is improved on training data, while on testing data, RDSVM's performance on dimension 10 remains the same as that of SVM.
- Both SVM and RDSVM have better performances on dimension 10 than discriminative learning algorithms, i.e. GMCE training algorithm and LDA.

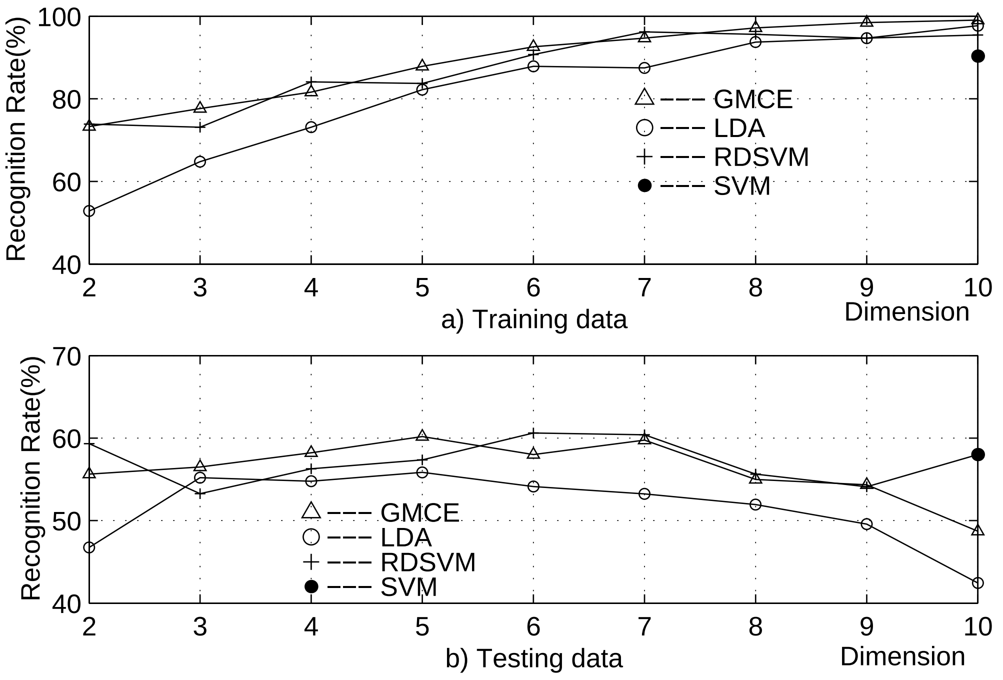


Figure 8.2: Results of reduced-dimensional SVM on Deterding Vowels database.

- The performance of RDSVM is very close to that of GMCE training algorithm on training data and is better than that of LDA except that the performance curve of RDSVM over dimensions is not as smooth as that of GMCE training algorithm.

- On testing data, RDSVM performs slightly poorer than GMCE training algorithm in low dimensional feature spaces (dimension 3 $\sim$ dimension 5), while on high dimensional feature spaces (from dimension 6 to dimension 9), RDSVM has a slightly better performance than GMCE training algorithm. On dimension 2 and dimension 10, the very low and full dimensional feature spaces, RDSVM performs much better than GMCE training algorithm.

- The overall performance of RDSVM is dramatically better than that of LDA on both training and testing data.

- The highest recognition rate on testing data does not appear on the full dimension (10) but on dimension 6. This is similar to the patterns found in Chapter 6.

8.4 Conclusion

The results given in Section 8.3 show that the performance of SVM on training data is poorer than that of GMCE training algorithm and LDA. This implies that SVM has a poorer fitness to the training data than discriminative learning algorithms. At the same time, the performance of RDSVM is improved on training data. This shows that the discriminative learning layer in RDSVM does help to reduce the negative effects of the defects of SVM.

Another conclusion that can be drawn from the results is that the performance of RDSVM in the reduced-dimensional feature spaces is comparable to that of GMCE training algorithm, which is so far the best linear feature extraction and dimensionality reduction algorithm discussed in this thesis. The performance curve of RDSVM over dimensions, however, is not as smooth as that of GMCE training algorithm. A possible reason may be that the database used is small and can not provide enough number of training data to SVM training. In the following chapters, a vowel recognition experiments on large databases will be designed and the corresponding results will give a clear answer to this problem.

8.5 Summary of Chapter

In this chapter, a RDSVM algorithm is proposed to mend the defects with SVM. The proposed RDSVM is a combination of discriminative learning algorithm and SVM. It is applied on Deterding vowels database and corresponding conclusion is drawn.

Chapter 9

Experiments on TIMIT Database

9.1 Introduction

In previous chapters, we investigated major independent feature extraction algorithms, such as LDA, PCA, integrated feature extraction and classification algorithms, i.e. MCE training algorithm, and non-linear classification method, i.e. SVM. We have proposed the use of MCE training algorithm for joint feature extraction and classification, the alternative MCE and GMCE training algorithm and RDSVM. Their performances on some small databases, such as Deterding Vowels database and D. German's GLASS database, are shown and corresponding evaluation are made. Some results are fairly encouraging. However, because of the small scale of these database, it is very hard to make accurate evaluation from the performances of these algorithms. The major drawbacks of these small databases center on their limited number and low dimensionality of parameter vectors. Therefore, in this chapter, experiments on a large database are designed to evaluate the performances of the algorithms investigated or proposed in previous chapters.

9.2 TIMIT Database

The database selected for the experiment is TIMIT database. TIMIT database is a well-known large scale speech database. It is based on the TIMIT corpus of read speech, which was designed to provide speech data for the acquisition of acoustic-phonetic knowledge and for the development and evaluation of automatic speech recognition systems. TIMIT database resulted from joint efforts of several sites under sponsorship from the Defense Advanced Research Projects Agency - Information Science and Technology Office (DARPA-ISTO). Text corpus design was a joint effort of the

Region Code	Dialect Region	Male	Female	Total
dr1	New England	31(63%)	18(27%)	49(8%)
dr2	Northern	71(70%)	31(30%)	102(16%)
dr3	North Midland	79(67%)	23(23%)	102(16%)
dr4	South Midland	69(69%)	31(31%)	100(16%)
dr5	Southern	62(63%)	36(37%)	98(16%)
dr6	New York City	30(65%)	16(35%)	46(7%)
dr7	Western	74(74%)	26(26%)	100(16%)
dr8	Army Brat (moved around)	22(67%)	11(33%)	33(5%)

Table 9.1: Dialect distribution of speakers in Timit database.

Massachusetts Institute of Technology (MIT), Stanford Research Institute (SRI), and Texas Instruments (TI). The speech was recorded at TI, transcribed at MIT, and was maintained, verified, and prepared for CD-ROM production by the National Institute of Standards and Technology (NIST). A brief description of the TIMIT Speech Corpus is contained in the file "readme.doc" on the TIMIT database CD-ROM. Additional information including the referenced material and some relevant reprints of articles may be found in the printed documentation which is also available from NIST (NIST# PB91-100354).

TIMIT database contains a total of 6300 sentences, 10 sentences spoken by each of 630 speakers from 8 major dialect regions of the United States. Table 9.1 shows the number of speakers from the 8 dialect regions, broken down by gender. The percentages are given in parentheses. A speaker's dialect region is the geographical area of the U.S. where they lived during their childhood years. The geographical areas correspond with recognized dialect regions in U.S. (Language Files, Ohio State University Linguistics Dept., 1982), with the exception of the Western region (dr7) in which dialect boundaries are not known with any confidence and dialect region 8 where the speakers moved around a lot during their childhood [95].

The text material in the TIMIT database prompts (found in the included file "prompts.doc" on CD-ROM) consists of 2 dialect "shibboleth" sentences designed at SRI, 450 phonetically-compact sentences designed at MIT, and 1890 phonetically-diverse sentences selected at TI. The dialect sentences (the SA sentences) were meant to expose the dialectal variants of the speakers and were read by all 630 speakers. The phonetically-compact sentences were designed to provide a good coverage of pairs of phones, with extra occurrences of phonetic contexts thought to be either difficult or of particular interest. Each speaker read 5 of these sentences (the SX sentences) and each text was spoken by 7 different speakers. The phonetically-diverse sentences (the SI sentences) were selected from existing text sources - the Brown Corpus (Kuchera and Francis, 1967) and the Playwrights Dialog (Hultzen, et al.,

Sentence Type	Sentences	Speakers	Total	Sentences per Speaker
Dialect (SA)	2	630	1260	2
Compact (SX)	450	7	3150	5
Diverse (SI)	1890	1	1890	3
Total	2342	638	6300	10

Table 9.2: TIMIT speech material.

1964) - so as to add diversity in sentence types and phonetic contexts. The selection criteria maximized the variety of allophonic contexts found in the texts. Each speaker read three of these sentences, with each sentence being read only by a single speaker. Table 9.2 summarizes the speech material in TIMIT database.

The above data of TIMIT database come from [95].

9.3 Vowel Classification

9.3.1 Vowels Selection

Our classification task is set for vowel classification. The vowels used in the classification task are selected from the vowels,semi-vowels and nasals that are listed in the phoneme document "phoncode.doc" in TIMIT database. There are altogether 20 vowels, 7 semi-vowels and 7 nasals listed in this file. Table 9.3 $\sim$ 9.5 list these phonemes:

20 vowels are listed in Table 9.3. All of them, except $\mathbf{ux}, \mathbf{axr}$ and $\mathbf{ax-h}$, are selected for the vowel recognition experiment. The reason that $\mathbf{ux}, \mathbf{axr}$ and $\mathbf{ax-h}$ are not selected is that these three vowels are very similar to some other vowels listed in Table 9.3, such as **ux** is close to **uh**, **axr** is close to **ax** and $\mathbf{ax-h}$ is close to **er**.

Although nasals and semi-vowels are not vowels, some nasals and semi-vowels listed in Table 9.4 and 9.5 have significant vowel characteristics. Therefore, some of them are also selected for our experiments. Among the 7 nasals listed in Table 9.4, **em** and **en** are the two that are very close to vowels. Since they are similar to each other, only **en** is selected for the experiment. There are 7 semi-vowels listed in Table 9.5. Among them **el** has the most notable vowel characteristics and thus is selected for the recognition experiments. Table 9.6 lists all the selected phonemes for the vowels recognition experiment.

9.3.2 Vowels Sampling

The speech signals are stored in two major sets in TIMIT database — "train" and "test", which are to be used for training and testing purposes, respectively. The speech data in each set are further separated into 8 subsets,

Vowels	Example Words	POSSIBLE PHONETIC TRANSCRIPTION
iy	beet	bcl b IY tcl t
ih	bit	bcl b IH tcl t
eh	bet	bcl b EH tcl t
ey	bait	bcl b EY tcl t
ae	bat	bcl b AE tcl t
aa	bott	bcl b AA tcl t
aw	bout	bcl b AW tcl t
ay	bite	bcl b AY tcl t
ah	but	bcl b AH tcl t
ao	bought	bcl b AO tcl t
oy	boy	bcl b OY
ow	boat	bcl b OW tcl t
uh	book	bcl b UH kcl k
uw	boot	bcl b UW tcl t
ux	toot	tcl t UX tcl t
er	bird	bcl b ER dcl d
ax	about	AX bcl b aw tcl t
ix	debit	dcl d eh bcl b IX tcl t
axr	butter	bcl b ah dx AXR
ax-h	suspect	s AX-H s pcl p eh kcl k tcl t

Table 9.3: Vowels list in TIMIT database.

Nasals	Example Words	POSSIBLE PHONETIC TRANSCRIPTION
m	mom	M aa M
n	noon	N uw N
ng	sing	s ih NG
em	bottom	b aa tcl t EM
en	button	b ah q EN
eng	washington	w aa sh ENG tcl t ax n
nx	winner	w ih NX axr

Table 9.4: Nasals list in TIMIT database.

Semi-vowels	Example Words	POSSIBLE PHONETIC TRANSCRIPTION
l	lay	L ey
r	ray	R ey
w	way	W ey
y	yacht	Y aa tcl t
hh	hay	HH ey
hv	ahead	ax HV eh dcl d
el	bottle	bcl b aa tcl t EL

Table 9.5: Semi-vowels list in TIMIT database.

Type	Phonemes
Vowel	aa, ae, ah, ao, aw, ax, ay, eh, er, ey, ih, ix, iy, ow, oy, uh, uw
Nasal	en
Semi-vowel	el

Table 9.6: Selected phonemes for the vowel recognition experiment.

dr1 $\sim$ dr8, according the speakers' dialect regions. As mentioned in Section 9.2, TIMIT database contains a total of 6300 sentences. Each occurrence of a sentence is recorded in a speech data file ($*$ .wav). The center 4K samples of each selected vowels' segment is picked out for the experiments. The segments picked from the train set are used for training purposes. The segments from the test set are used for testing.

Phonemes	dr1	dr2	dr3	dr4	dr5	dr6	dr7	dr8
aa	249	541	448	458	425	221	563	124
ae	346	665	650	560	616	328	645	177
ah	176	313	364	299	355	169	352	103
ao	217	445	443	478	477	206	475	148
aw	60	126	116	121	94	65	117	30
ax	121	207	251	239	242	100	185	54
ay	198	395	398	328	343	181	395	144
eh	297	591	594	544	581	276	592	145
el	77	145	135	131	140	64	151	43
en	55	97	108	77	79	53	113	29
er	129	384	363	294	281	140	328	105
ey	189	346	371	332	354	178	407	96
ih	316	697	716	709	744	324	747	222
ix	292	583	650	622	589	319	692	161
iy	463	1089	1046	993	1034	460	1033	348
ow	183	336	355	305	327	176	359	90
oy	64	118	126	77	82	54	126	37
uh	40	57	66	69	65	49	67	25
uw	63	106	75	56	75	50	76	16
total	3535	7241	7275	6692	6903	3413	7423	2097

Table 9.7: Number of selected phonemes in training dataset.

The vowel recognition experiments are carried out on all the 8 sub-directories of data stored both in the train and test sets (dr1 $\sim$ dr8). Since each sub-directory in the train and test sets has different number of sentences and occurrences of the sentences, the number of vowels' segments is different as well. Table 9.7 and 9.8 record the number of segments of the 19 selected vowels that are picked out from these sub-directories.

Phonemes	dr1	dr2	dr3	dr4	dr5	dr6	dr7	dr8
aa	77	176	168	190	191	79	172	66
ae	79	214	229	261	234	97	199	94
ah	59	136	123	118	140	71	139	39
ao	74	168	181	226	202	67	147	77
aw	10	40	30	40	42	21	25	8
ax	41	89	83	115	103	25	65	30
ay	56	131	134	164	148	52	117	49
eh	83	225	230	252	216	84	181	90
el	22	42	40	70	64	14	43	32
en	8	34	32	41	31	15	28	21
er	47	135	128	149	103	48	109	76
ey	54	116	116	159	128	56	120	53
ih	104	239	214	326	248	92	199	100
ix	97	201	196	247	249	101	217	96
iy	168	381	384	490	422	155	354	160
ow	57	116	108	142	145	49	105	54
oy	17	49	45	42	34	17	40	19
uh	15	21	33	29	34	16	29	14
uw	15	37	28	25	25	7	12	10
total	1083	2550	2502	3086	2759	1066	2301	1088

Table 9.8: Number of selected phonemes in testing dataset.

9.3.3 Speech Features

In TIMIT database, the speech signals are stored in wave files. Each wave file records an occurrence of a sentence. A corresponding label file is given to label out all the phonemes appearing in the sentence and mark out their starting time and ending time. In our experiment, we use these label files to pick out all the segments of the 19 selected phonemes from the database. Then a feature vector is extracted from each segment for the classification tasks.

The feature vectors contain 1 energy coefficient and 20 Mel-Frequency Cepstral Coefficients (MFCCs). The features are extracted by the following steps: The center 4K samples are taken out from the vowel segment since many vowels last longer than 256 msec (The sampling frequency is 16 KHz in TIMIT database). If the length of a segment is less than 4K, 0s are added to the end of the sequence to make the length of the sequence to 4K. A Hamming window is applied to the sequence and the power spectrum is calculated. The power spectrum of the speech signals is then correlated with a triangular filter bank. The filter bank has 20 filters and is designed to give approximately equal resolution on a mel-scale. The mel-scale is defined by:

$$Mel(f) = 2595 \log_{10}(1 + \frac{f}{100}) \tag{9.1}$$

The center frequencies of each filter used in our experiments are:{100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1150, 1320, 1520, 1750, 2000, 2300, 2640, 3040, 3500, 4000}. The filtered power spectrum magnitude coefficients are accumulated in each filter band to obtain the 20 mel-scale filterbank parameters. These mel-scale parameters are transformed into MFCCs in the last step by the Discrete Cosine Transform (DCT). The DCT is defined as:

$$c_i = \sqrt{\frac{2}{N}} \sum_{j=1}^{N} m_j cos(\frac{\pi i}{N}(j - 0.5)) \tag{9.2}$$

MFCCs have been used by speech recognition applications. They give good discrimination and lend themselves to a number of manipulations. In particular, the effect of inserting a transmission channel on the input speech is to multiply the speech spectrum by the channel transfer function. In the log cepstral domain, this multiplication becomes a simple addition which can be removed by subtracting the cepstral mean from all input vectors. Nevertheless, this simple technique is very effective in practice where it compensates for long-term spectral effects as those caused by different microphones and audio channels [112].

9.4 Experiment Setup

The vowel recognition experiment is to examine the performances of the feature extraction and classification algorithms investigated or proposed in previous chapters. It includes two major parts – speaker dependent and speaker independent. The class models of these algorithms are first trained and tested on the 8 sub-directories (dr1 $\sim$ dr8) separately in speaker dependent part. Then in speaker independent part, the algorithms are tested on the whole database. The algorithms evaluated in the experiment are listed as follows:

- **LDA** and **PCA** are used for both feature extraction and feature dimensionality reduction. Their results will be used as references for MCE and GMCE training algorithms, SVM and RDSVM because both LDA and PCA are popular algorithms for feature extraction and feature dimensionality reduction. The classifiers used after both LDA and PCA training are Mahalanobis distance classifiers.

- **MCE** and **GMCE** training algorithms are used for both feature extraction and feature dimensionality reduction. Their performances will be compared to those of LDA and PCA. The classifiers used in both MCE and GMCE training algorithms are also Mahalanobis distance classifiers so that the results of MCE and GMCE training algorithms are comparable to those of LDA and PCA.

- **SVM** is used for feature extraction. Feature dimension used in SVM is full dimension because SVM is not suitable for feature dimensionality reduction. The SVM classifier used is "one vs. one" multi-class classifier.

- **RDSVM** is used for feature extraction both in full dimensional feature space and reduced-dimensional feature space. The classifier used is "one vs. one" multi-class classifier.

9.5 Results Analysis

This section shows the results of a vowel recognition experiment and makes corresponding analysis. The vowel recognition experiment consists of two sub-experiments. One is a speaker dependent experiment and the other is a speaker independent experiment. In the speaker dependent experiment, the experiment results are organized in three groups, as shown in the following:

- Comparison between PCA, LDA, MCE and SVM – This group has two major tasks. One is to analyse the performances of independent and integrated feature extraction and classification algorithms. The other is to analyse the performances of linear and non-linear classification algorithms. In the first task, the performances of PCA and LDA, as independent feature extraction algorithms, are compared to that of MCE training algorithm, an integrated feature extraction and classification algorithm. The classifier used in PCA, LDA and MCE is minimum distance classifier, which is a typical linear classifier, while SVM is a non-linear classifier. Therefore in the second task, the performances of LDA, PCA MCE and SVM are also compared and analysed.

- Analysis of GMCE training algorithm – This group analyses the performances of two types of GMCE training algorithms. One employs LD initialization criterion and the other employs PC initialization criterion. Their performances are compared to those of LDA, PCA and MCE training algorithm.

- Analysis of RDSVM – This group investigates the performance of RDSVM. The performance of RDSVM is compared to those of LDA, GMCE training algorithm and SVM.

In the speaker independent experiment, similar analysis is conducted on the above three groups, i.e. comparison between independent and integrated feature extraction and classification algorithms, between linear and non-linear classifiers, analysis of GMCE training algorithm and RDSVM. Apart from this, the performances of the algorithms in the speaker independent experiment are compared to those in the speaker dependent experiment. The performance of each algorithm in the speaker dependent experiment is represented by the average performances and maximum-minimum value area over the 8 sub-directories.

In the eighth sub-directory *dr8*, accidentally, both the *global* and *within* covariance matrices of the vowels' features have two very close eigenvalues. This brings significant difficulties to the application of PCA and LDA in high dimensional feature spaces. It has some negative effects on MCE training in high dimensional spaces as well. Since this problem is irrelevant to the objective of our experiment and has little influence on the overall outcome of the experiment, the four algorithms' results *dr8* in high dimensional spaces (Dimension 16 to Dimension 21) are neglected. Therefore in this chapter only the results from Dimension 3 to Dimension 15 will be shown in all the figures corresponding to set sub-directory dr8.

9.5.1 Speaker Dependent Experiment

Comparison between PCA, LDA, MCE and SVM

In this group, we compare the performances of the four existing feature extraction algorithms — PCA, LDA, MCE training algorithm and SVM. The results of these four algorithms on the 8 sub-directories, *dr1* $\sim$ *dr8*, are given in Figure 9.1 $\sim$ Figure 9.8, respectively. The MCE training algorithms used in this and the following experiments are alternative MCE training algorithm and noted as MCE in the figures. Feature dimensionality reduction is conducted in LDA, PCA and MCE training. The minimum dimension used is 3 and the maximum is the full dimension (21). The horizontal axis of the figures are dimensions. The vertical axis of the figures is the recognition rates. This group includes two tasks. The first involves an analysis of the performances of independent and integrated feature extraction and classification algorithms. The performances of independent feature extraction algorithms, i.e. PCA and LDA are compared to that of the integrated feature extraction and classification algorithm, i.e. MCE training algorithm in the figures on all dimensions. The second task involves a comparison between linear and non-linear classifiers. SVM, as a non-linear classifier, is compared to the linear classifier, which is minimum distance classifier based on Mahalanobis distance measure and employed in LDA, PCA and MCE training. However, SVM uses parameter vectors directly and is unable to conduct feature extraction and dimensionality reduction. Thus it has a single classification result on each sub-directory of TIMIT database and its results appear as single points on dimension 21 in each figure.

1. Results on /dr1

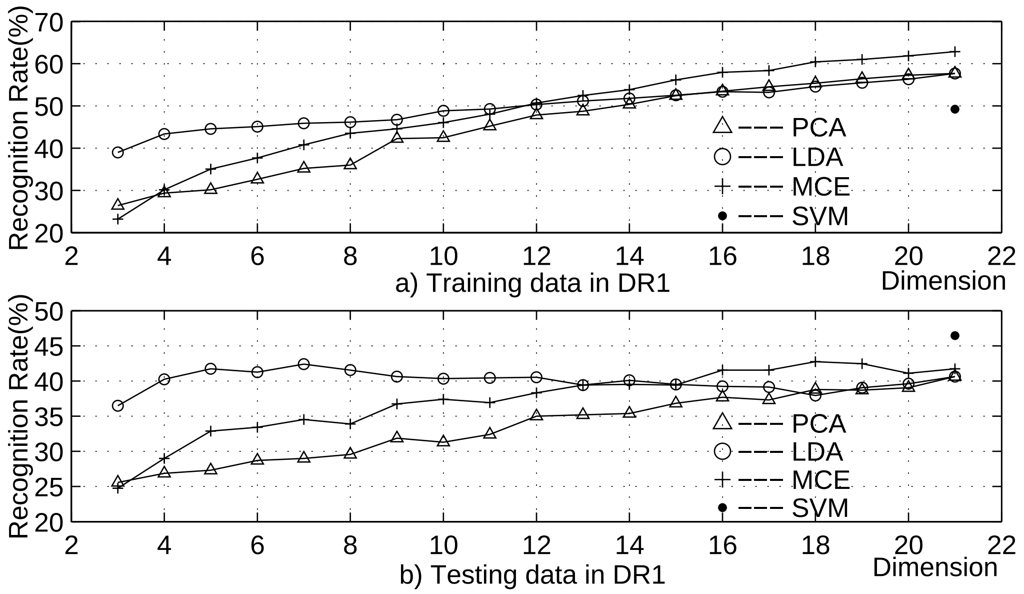


Figure 9.1: Results of LDA, PCA, MCE and SVM on DR1

2. Results on /dr2

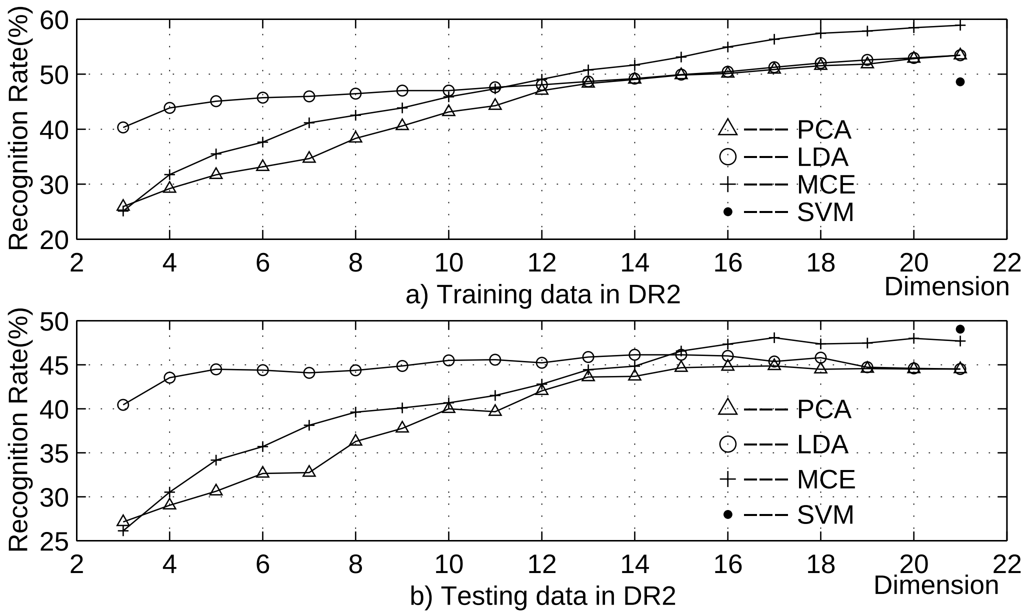


Figure 9.2: Results of LDA, PCA, MCE and SVM on DR2

3. *Results on /dr3*

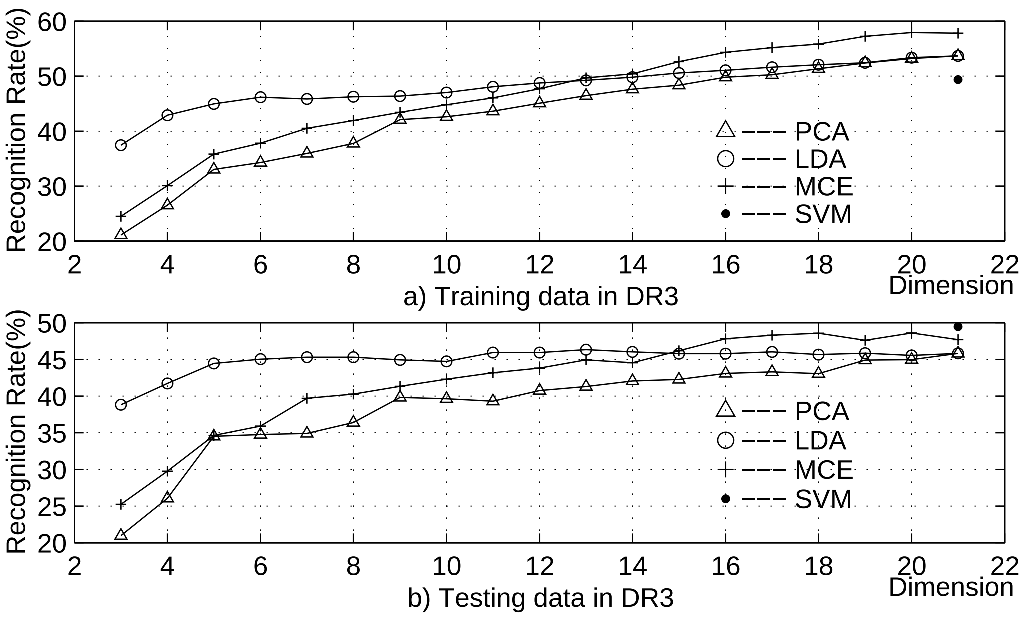


Figure 9.3: Results of LDA, PCA, MCE and SVM on DR3

4. *Results on /dr4*

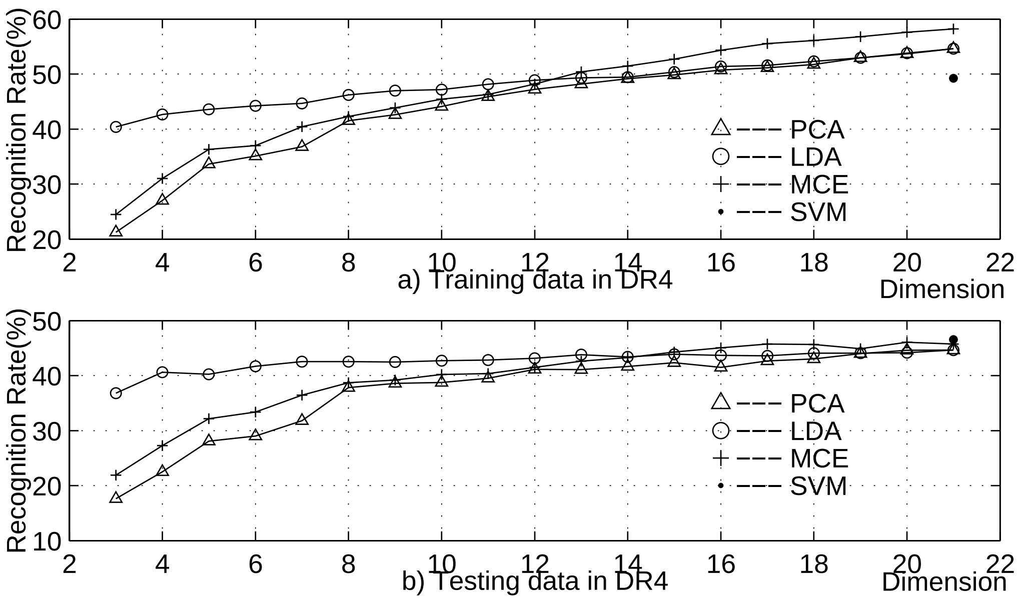


Figure 9.4: Results of LDA, PCA, MCE and SVM on DR4

5. Results on /dr5

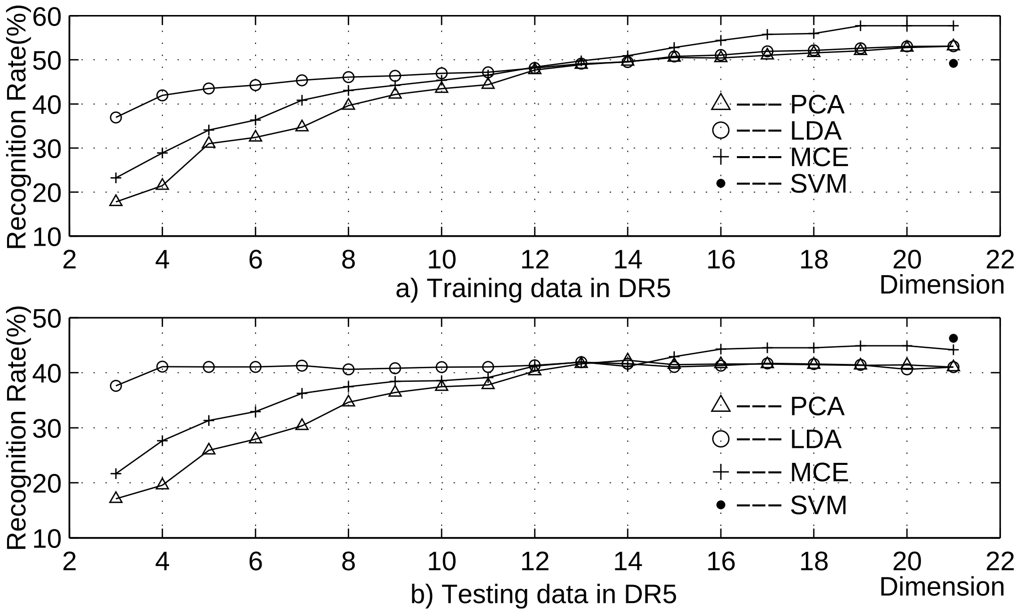


Figure 9.5: Results of LDA, PCA, MCE and SVM on DR5

6. Results on /dr6

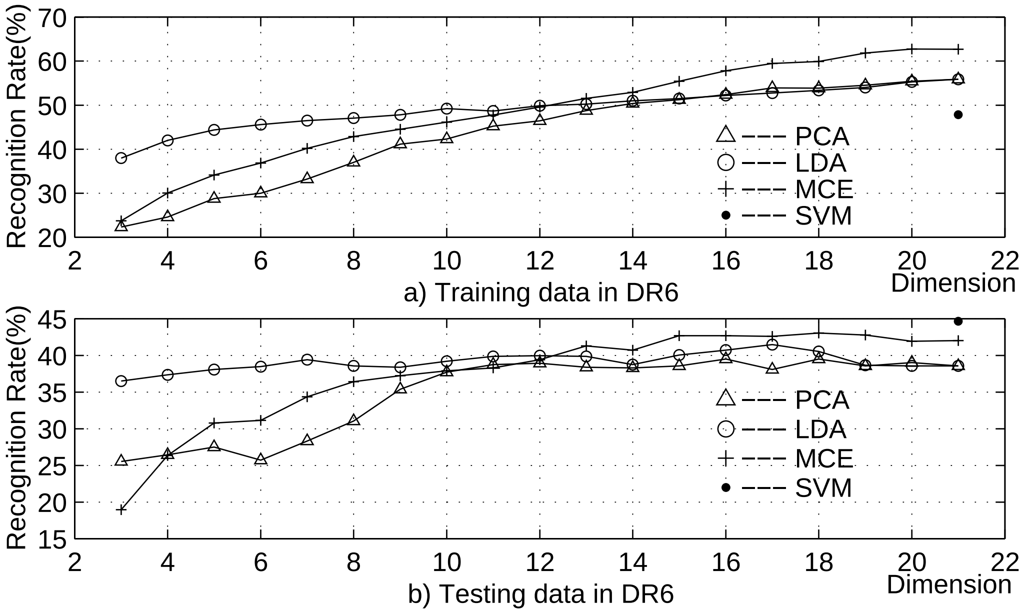


Figure 9.6: Results of LDA, PCA, MCE and SVM on DR6

7. Results on /dr7

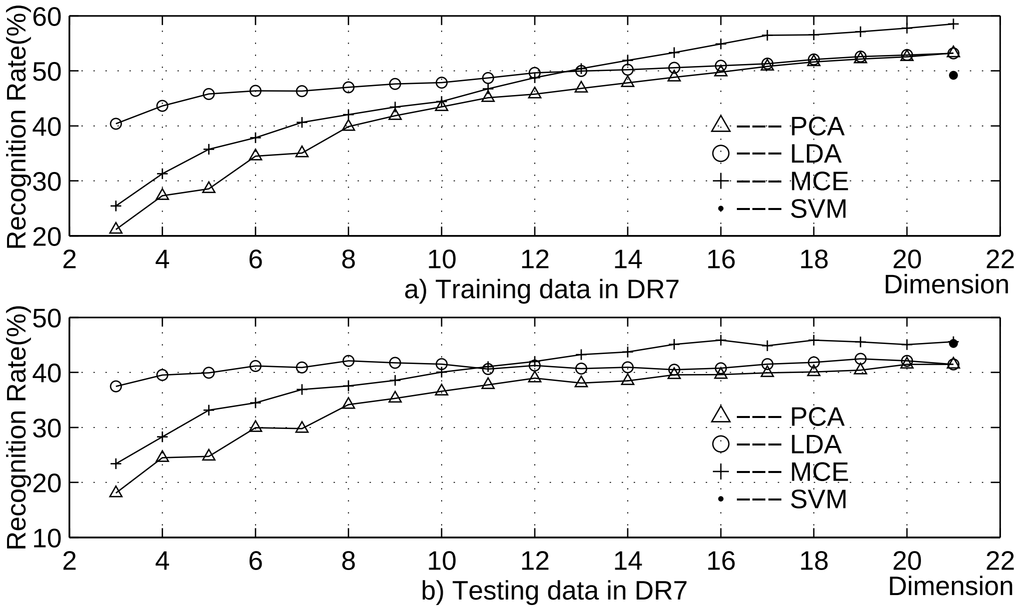


Figure 9.7: Results of LDA, PCA, MCE and SVM on DR7

8. Results on /dr8

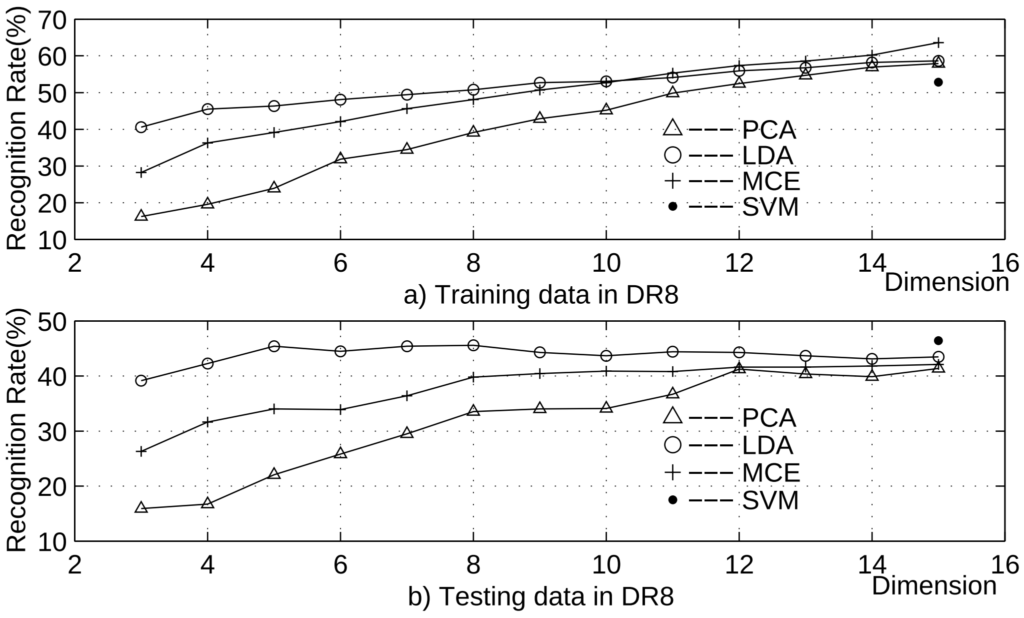


Figure 9.8: Results of LDA, PCA, MCE and SVM on DR8

Observations from Figure 9.1 $\sim$ Figure 9.8 can be summarized as follows:

- Most figures show that in low-dimensional feature spaces (Dimension 3 $\sim$ Dimension 12) on the training data, LDA performs better than PCA and MCE training algorithm. On the testing data, LDA performs better than PCA and MCE training algorithm on low dimensions (from dimension 3 to dimension 15).
- MCE training algorithm performs better than LDA and PCA in high-dimensional feature spaces (Dimension 3 $\sim$ Dimension 12) on training data. On the testing data, MCE training algorithm performs better than PCA and LDA on high dimensions (from dimension 16 to dimension 21).
- PCA has the poorest performance in low dimensional feature spaces (Dimension 3 $\sim$ Dimension 13 on the training data and Dimension 3 $\sim$ Dimension 15 on testing data). While in high dimensional features spaces (Dimension 14 $\sim$ Dimension 21 on the training data and Dimension 16 $\sim$ Dimension 21 on testing data), the performances of PCA are close to those of LDA.
- In low dimensional feature spaces (Dimension 3 $\sim$ Dimension 13 on the training data and Dimension 3 $\sim$ Dimension 15 on testing data) the performances of MCE training algorithm are between those of LDA and PCA.
- In most figures, the result curves of LDA are very flat over dimensions. Those of PCA and MCE training algorithm drop rapidly with the decrease of dimensions. The result curves on testing data in each figure are not as smooth as those on training data.
- The performances of SVM on training data are poorer than those of LDA, PCA and MCE training algorithm, which use linear minimum distance classifier.
- SVM performs much better than LDA, PCA and MCE training algorithm on testing data.

Analysis of GMCE Training Algorithm

In this section, the performance of GMCE training algorithm is investigated. Two types of GMCE are used. One is GMCE training algorithm with linear discriminant (LD) initialization criterion, which is denoted as GMCE+LD. The other one is with principal component (PC) initialization criterion and is denoted as GMCE+PC.

GMCE with LD Initialization Criterion

1. Results on /dr1

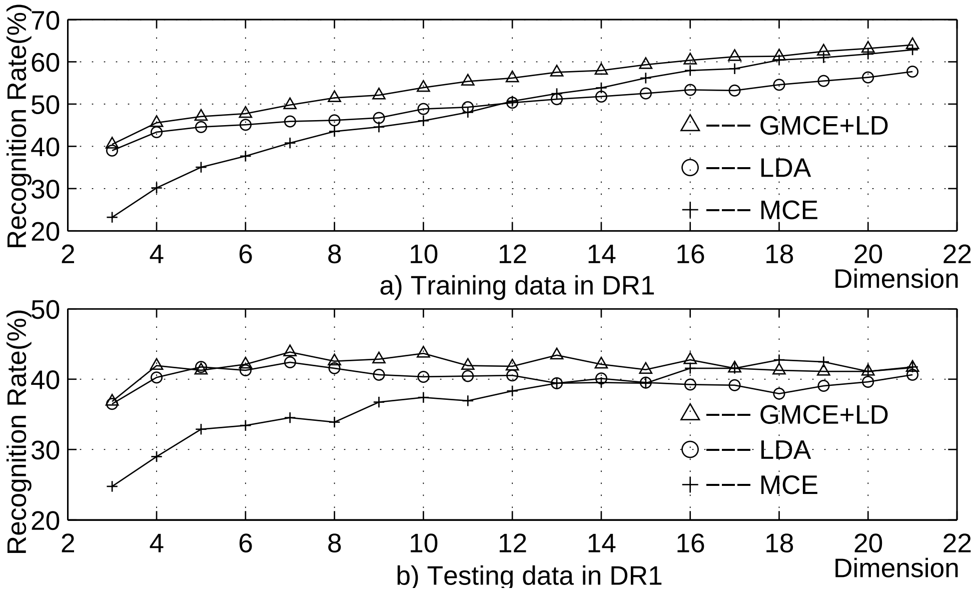


Figure 9.9: Results of GMCE+LD, MCE and LDA on DR1

2. Results on /dr2

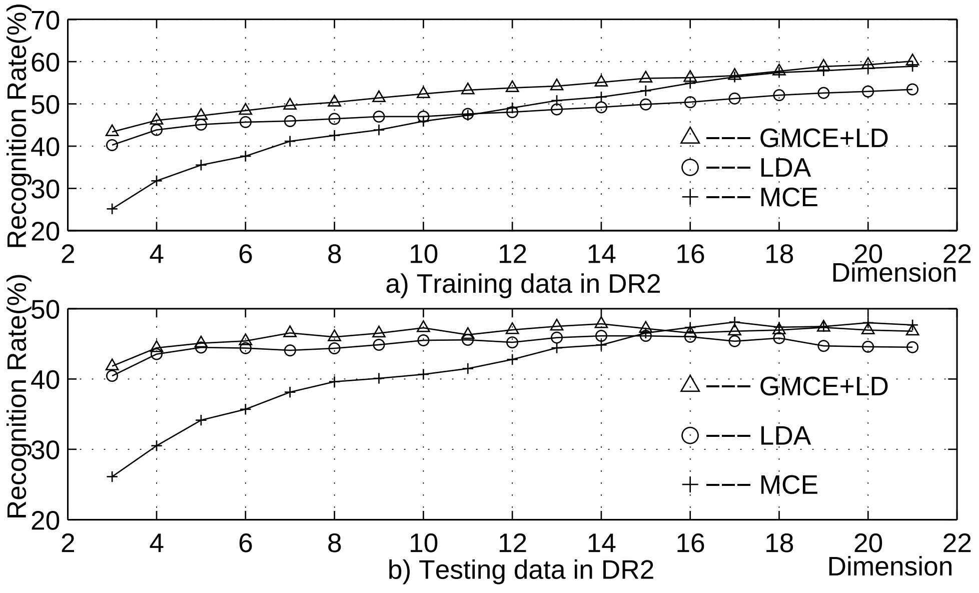


Figure 9.10: Results of GMCE+LD, MCE and LDA on DR2

3. Results on /dr3

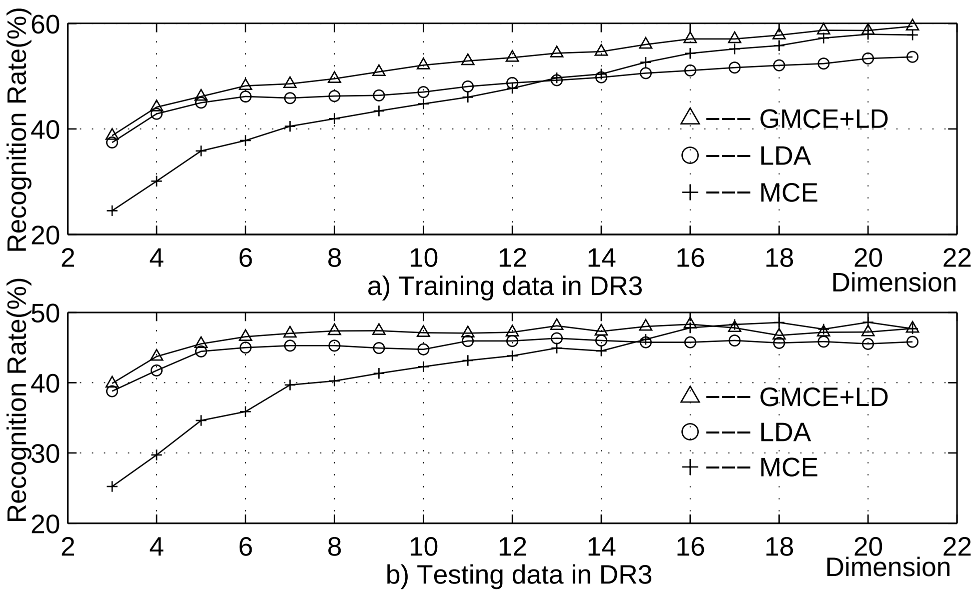


Figure 9.11: Results of GMCE+LD, MCE and LDA on DR3

4. Results on /dr4

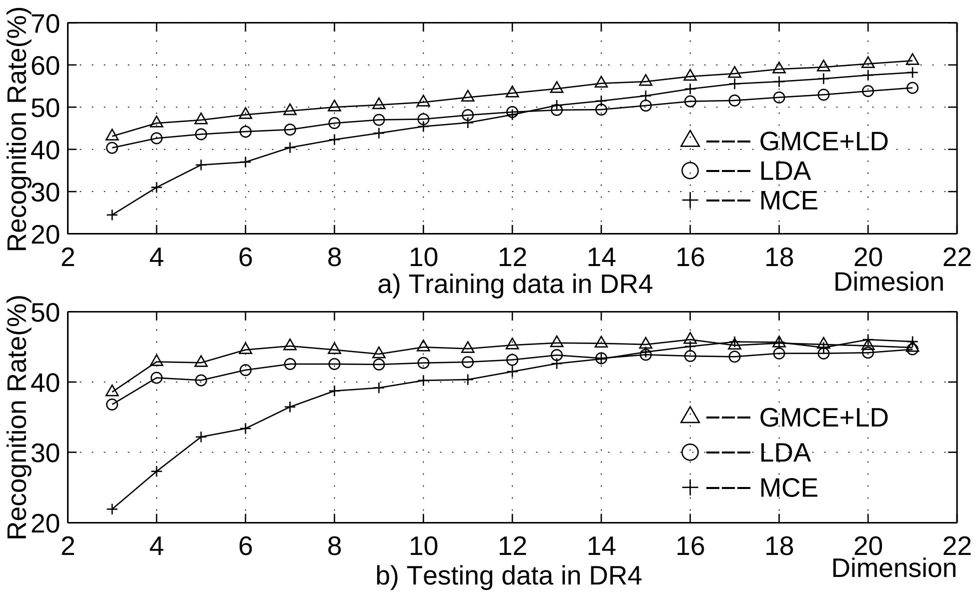


Figure 9.12: Results of GMCE+LD, MCE and LDA on DR4

5. Results on /dr5

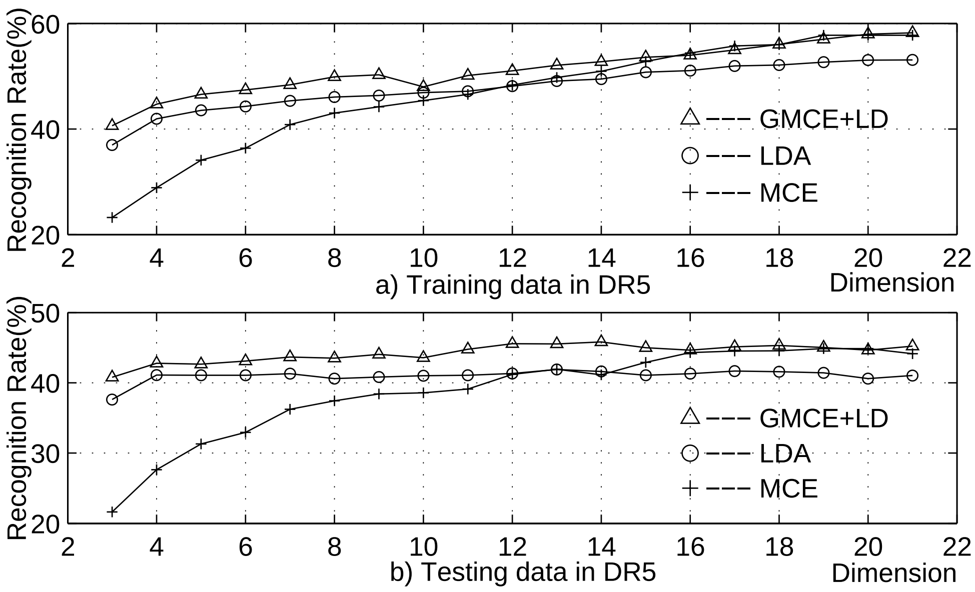


Figure 9.13: Results of GMCE+LD, MCE and LDA on DR5

6. Results on /dr6

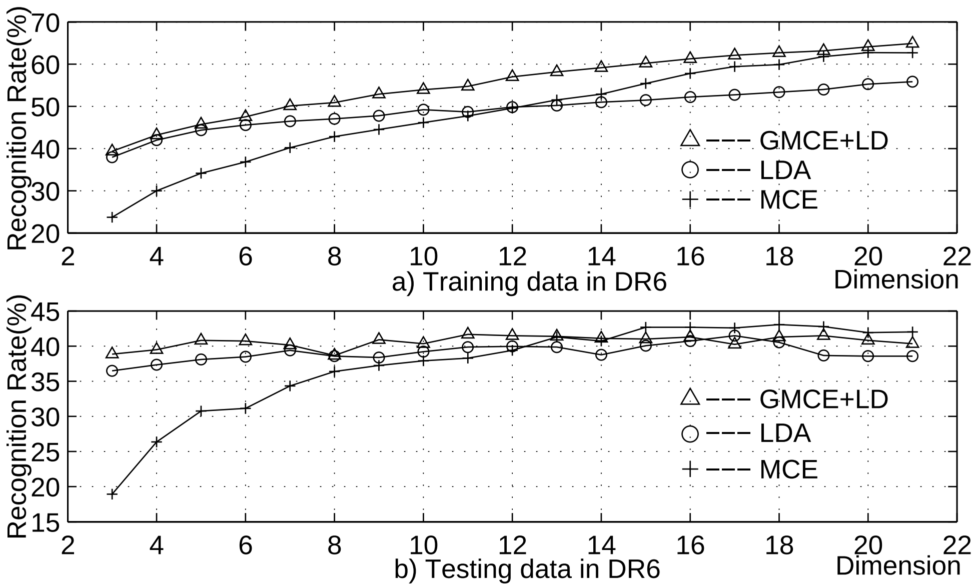


Figure 9.14: Results of GMCE+LD, MCE and LDA on DR6

7. Results on /dr7

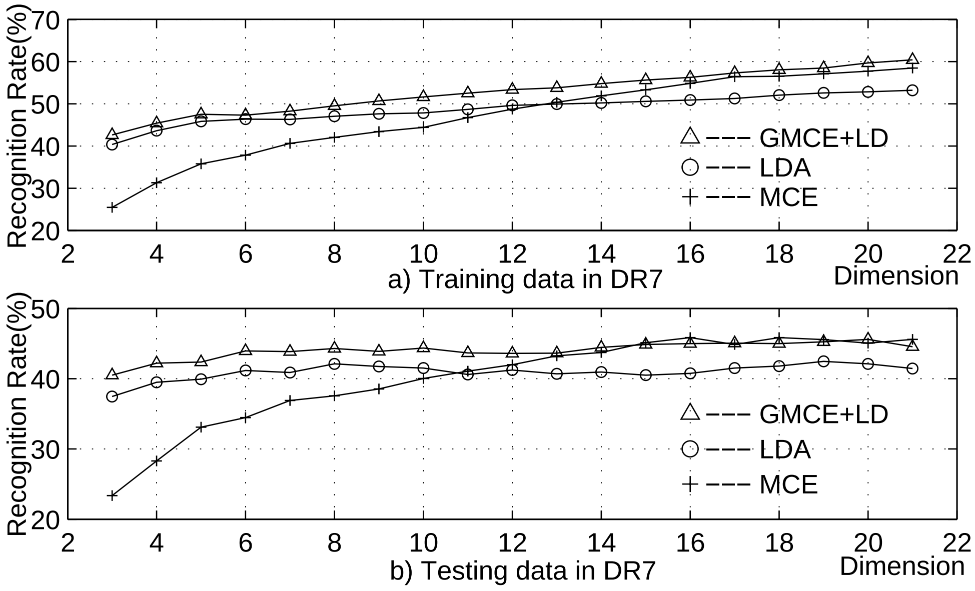


Figure 9.15: Results of GMCE+LD, MCE and LDA on DR7

8. Results on /dr8

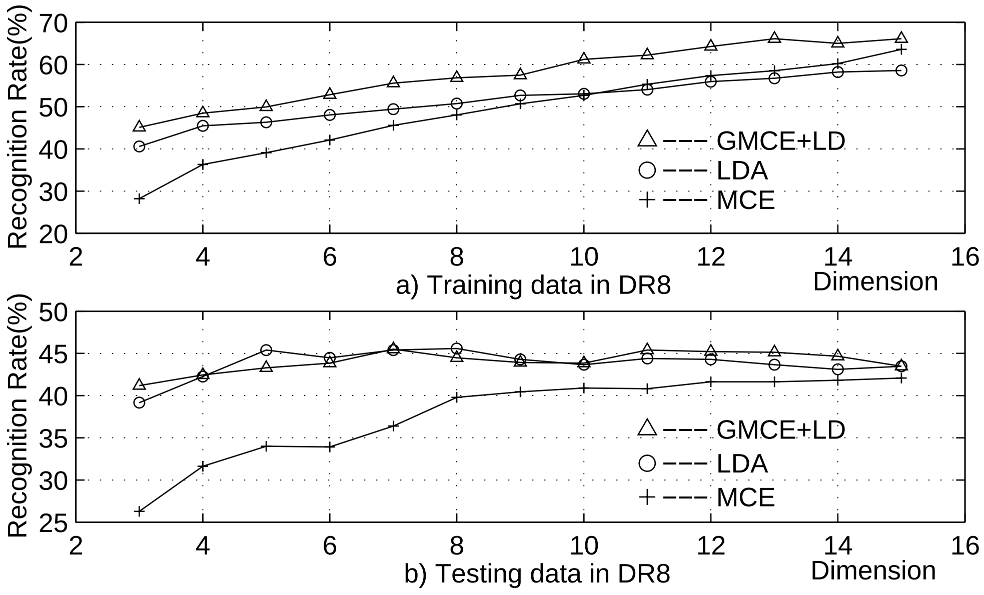


Figure 9.16: Results of GMCE+LD, MCE and LDA on DR8

GMCE with PC Initialization Criterion

1. Results on /dr1

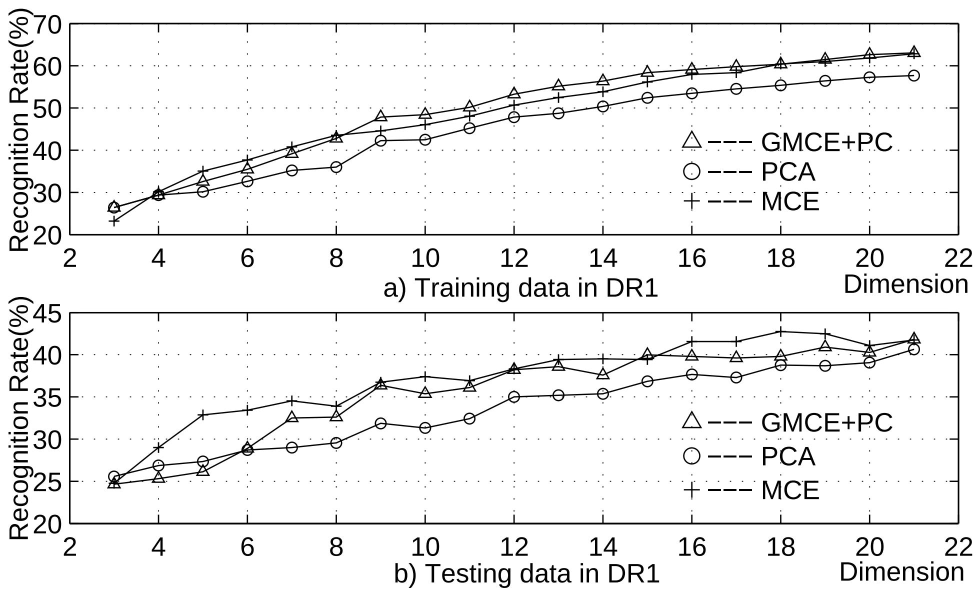


Figure 9.17: Results of GMCE+PC, MCE and PCA on DR1

2. Results on /dr2

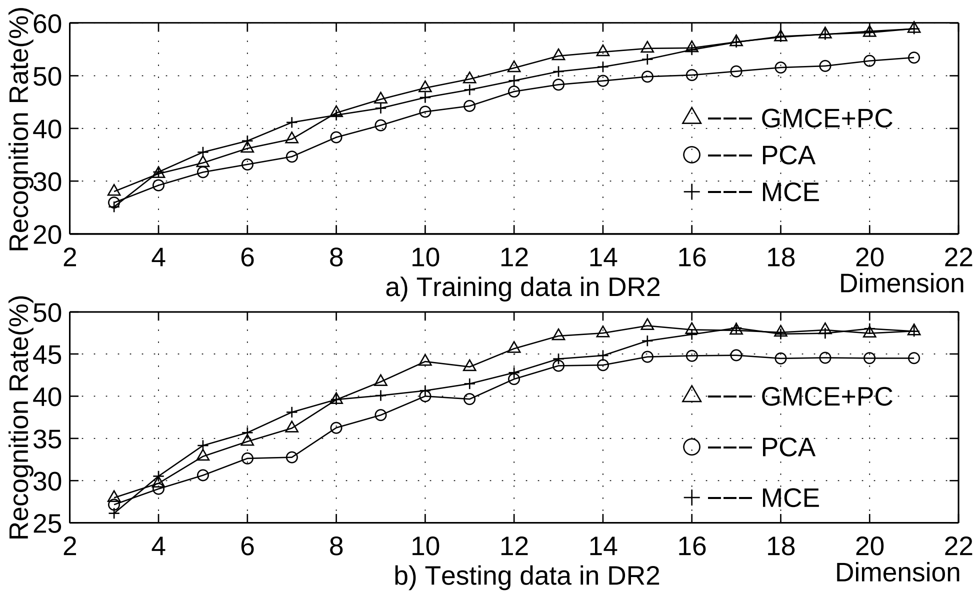


Figure 9.18: Results of GMCE+PC, MCE and PCA on DR2

3. Results on /dr3

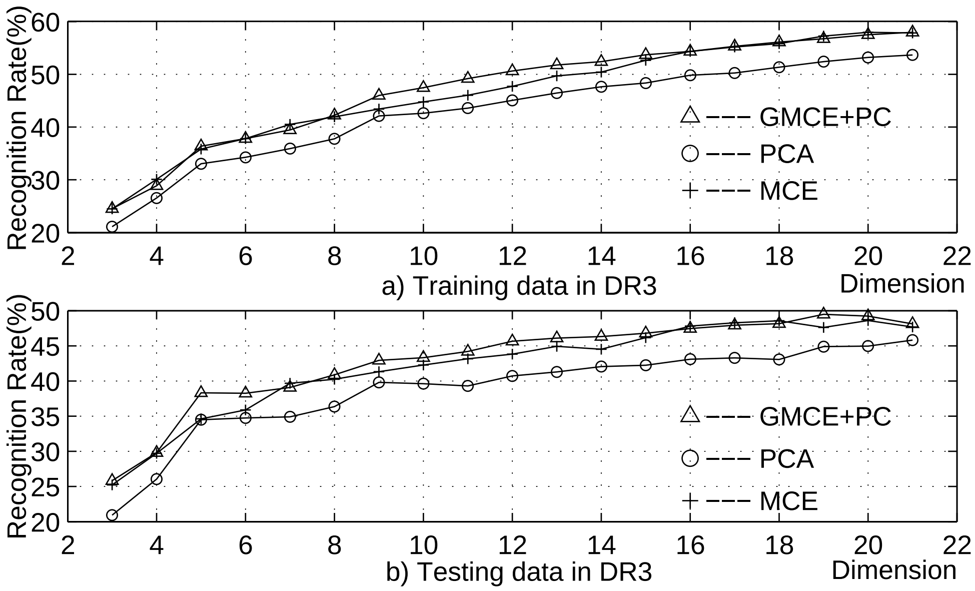


Figure 9.19: Results of GMCE+PC, MCE and PCA on DR3

4. Results on /dr4

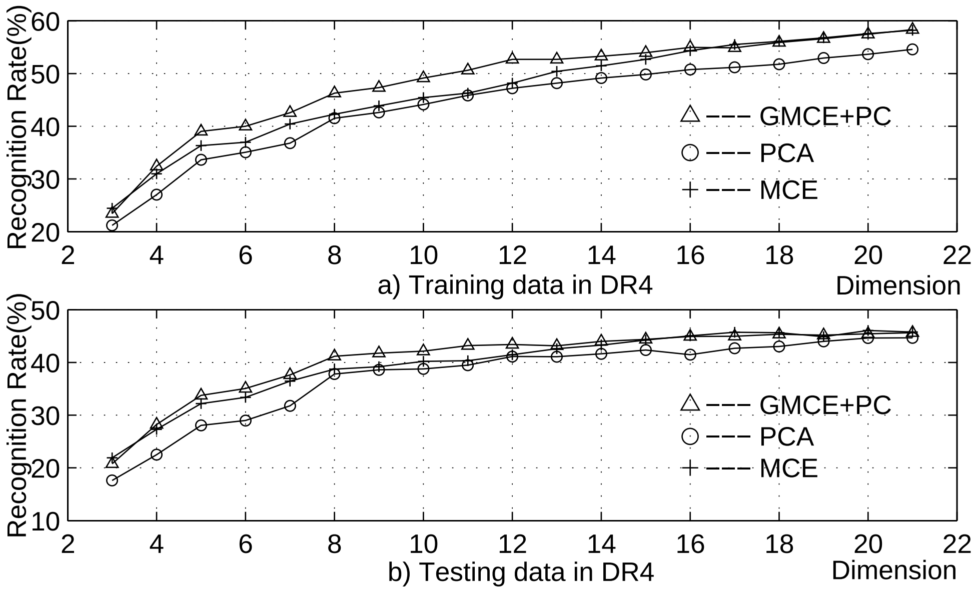


Figure 9.20: Results of GMCE+PC, MCE and PCA on DR4

5. Results on /dr5

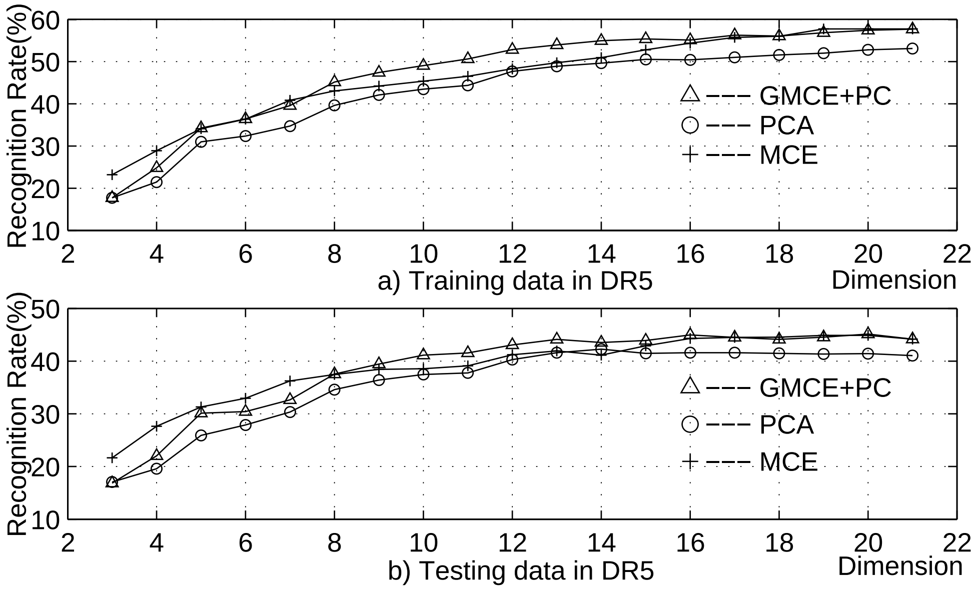


Figure 9.21: Results of GMCE+PC, MCE and PCA on DR5

6. Results on /dr6

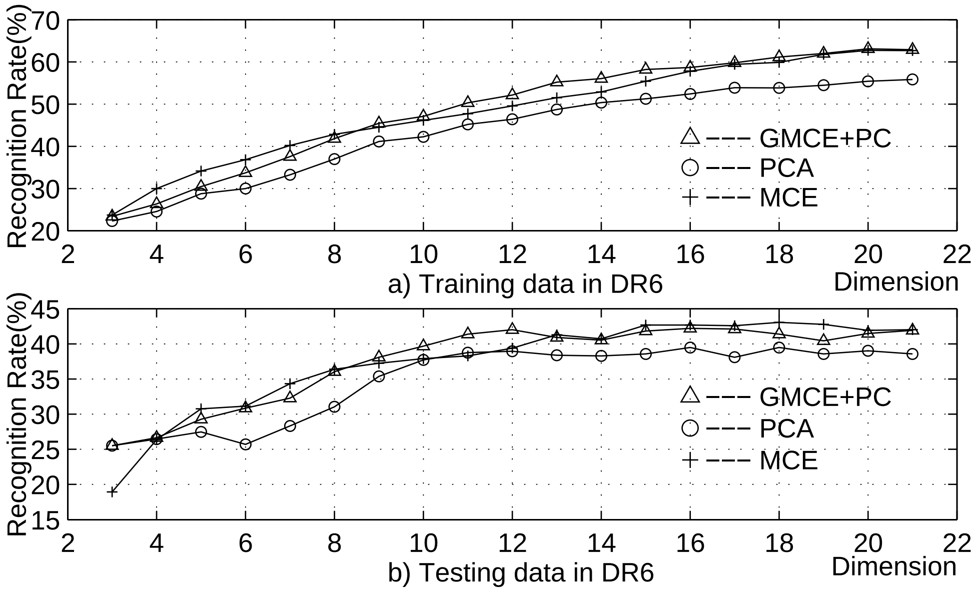


Figure 9.22: Results of GMCE+PC, MCE and PCA on DR6

7. Results on /dr7

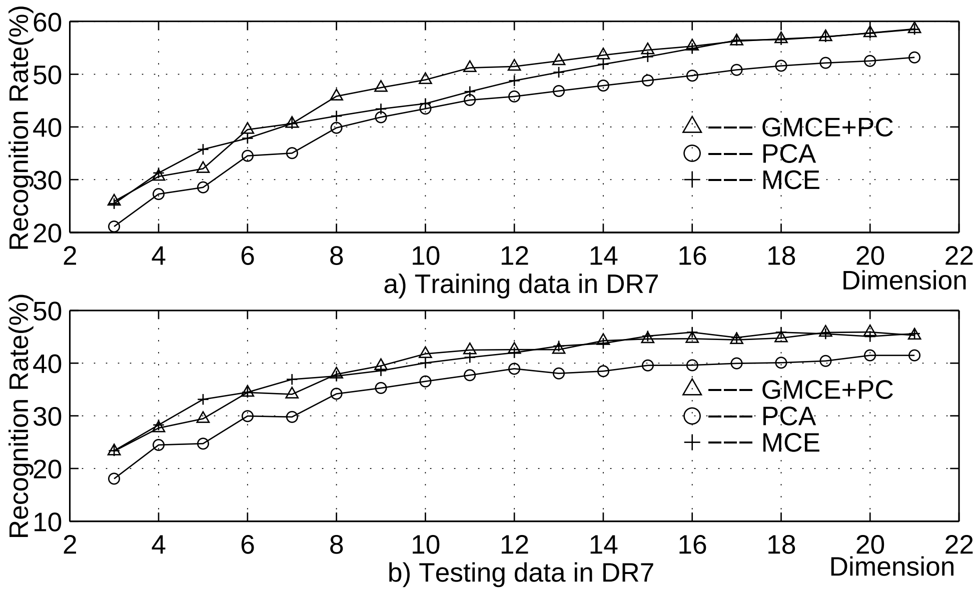


Figure 9.23: Results of GMCE+PC, MCE and PCA on DR7

8. Results on /dr8

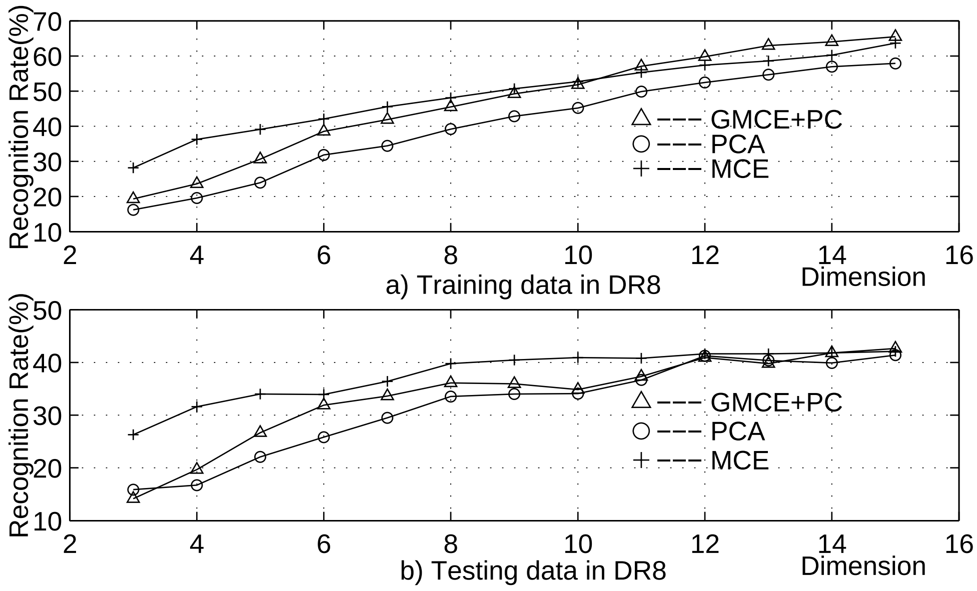


Figure 9.24: Results of GMCE+PC, MCE and PCA on DR8

Observations from the results of GMCE training algorithm with LD initialization criterion can be summarized as follows:

- With LD initialization criterion, the performances of GMCE training algorithm are better than both LDA and MCE training algorithm over all the dimensions.
- In high dimensional sub-spaces (Dimension 15 $\sim$ Dimension 21), the performances of GMCE+LD are slightly better than those of MCE training algorithm, which has better performances than LDA.
- In medium dimensional sub-spaces (Dimension 7 $\sim$ Dimension 15), GMCE+LD has significantly better performances than both LDA and MCE training algorithm.
- In low dimensional sub-spaces (Dimension 3 $\sim$ Dimension 7), the performances of GMCE+LD are slightly better than those of LDA but dramatically better than those of MCE training algorithm.
- Similar observations can be summarized from the results on all the eight TIMIT sub-directories.

Observations from the results of GMCE training algorithm with PC initialization criterion can be summarized as follows:

- With PC initialization criterion, the general performances of GMCE training algorithm are not significantly improved.
- In high dimensional sub-spaces (Dimension 15 $\sim$ Dimension 21), the performances of GMCE+PC are either very close or equal to those of MCE training algorithm.
- In medium dimensional sub-spaces (Dimension 7 $\sim$ Dimension 15), GMCE+PC has slightly better performances than MCE training algorithm.
- In low dimensional sub-spaces (Dimension 3 $\sim$ Dimension 7), the performances of GMCE+PC are poorer than those of MCE training algorithm but better than those of PCA.
- Similar observations can be summarized from the results on all the eight TIMIT sub-directories.

Analysis of RDSVM

In this section, we will investigate the performance of RDSVM on TIMIT database. The results of RDSVM are compared to those of SVM, GMCE training algorithm with LD initialization criterion and LDA.

1. *Results on /dr1*

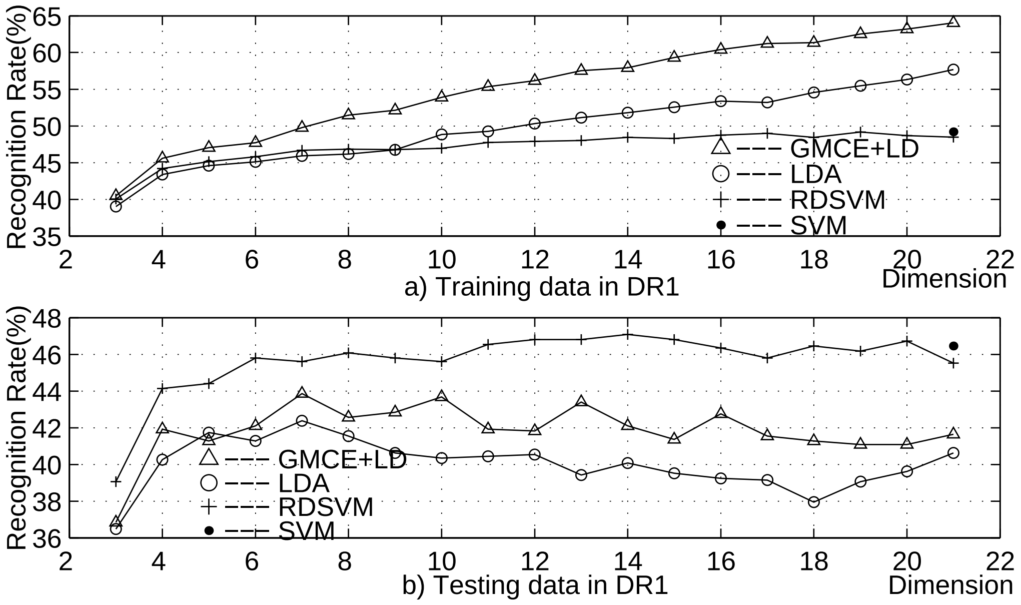


Figure 9.25: Results of RDSVM on DR1

2. *Results on /dr2*

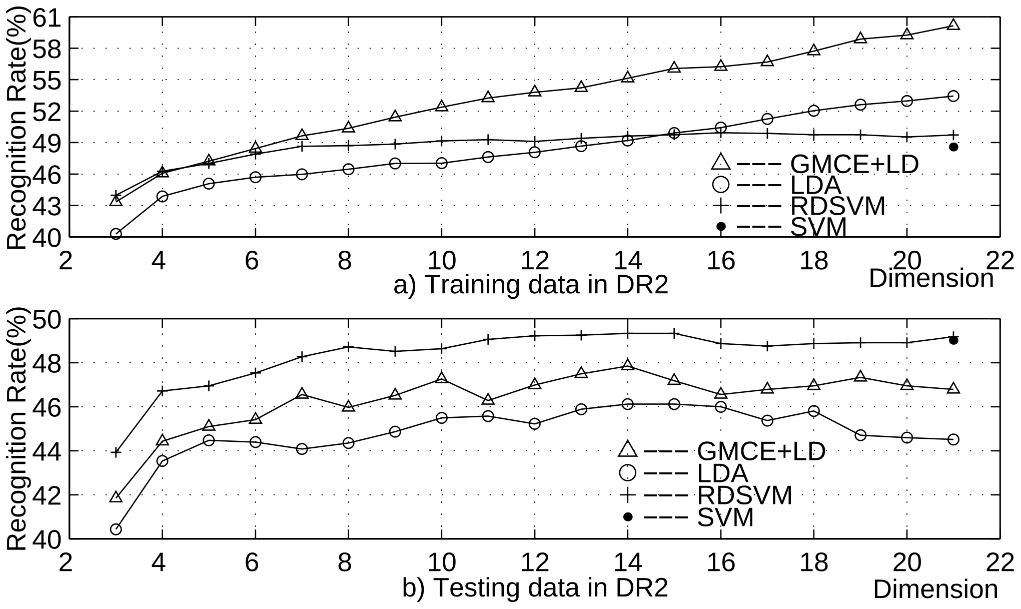


Figure 9.26: Results of RDSVM on DR2

3. Results on /dr3

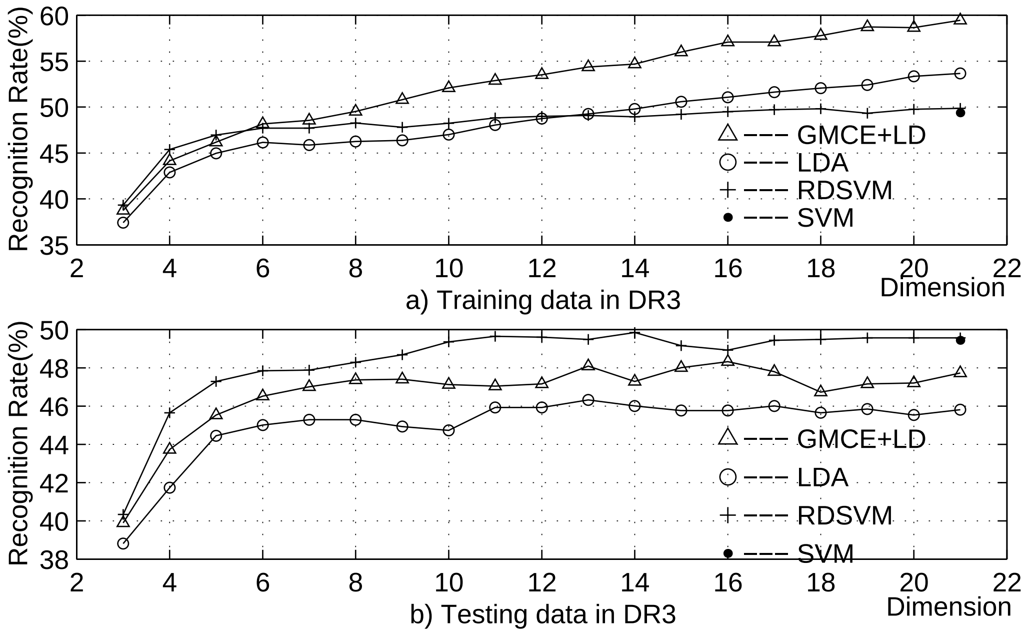


Figure 9.27: Results of RDSVM on DR3

4. Results on /dr4

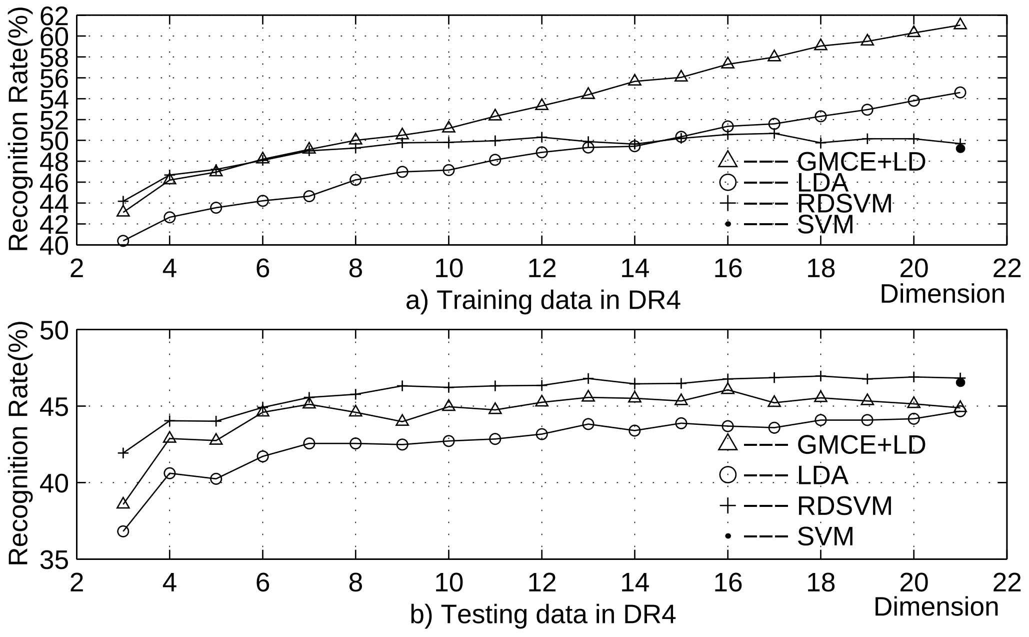


Figure 9.28: Results of RDSVM on DR4

5. Results on /dr5

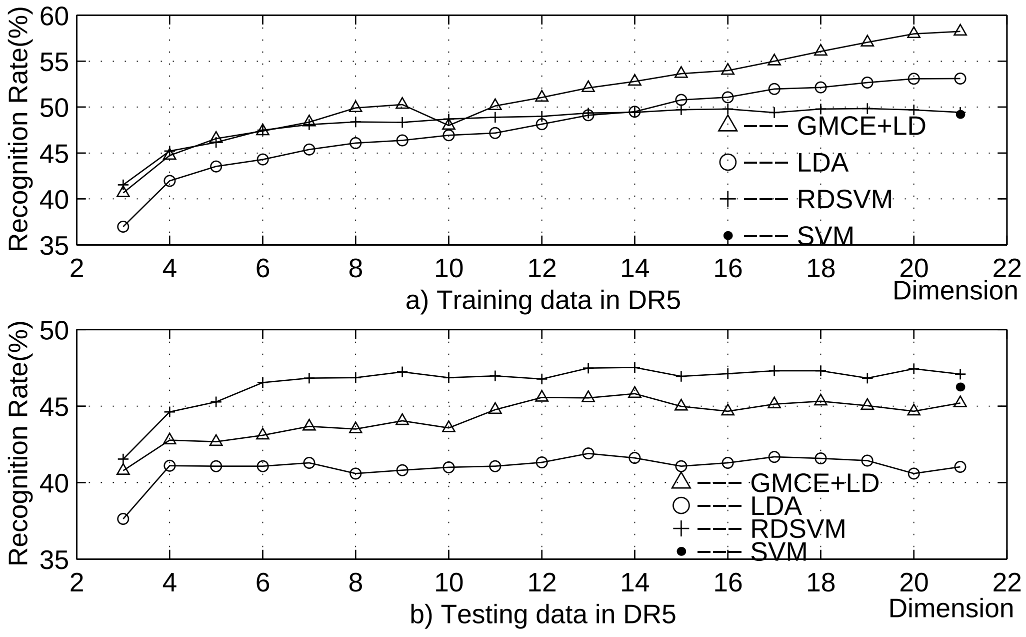


Figure 9.29: Results of RDSVM on DR5

6. Results on /dr6

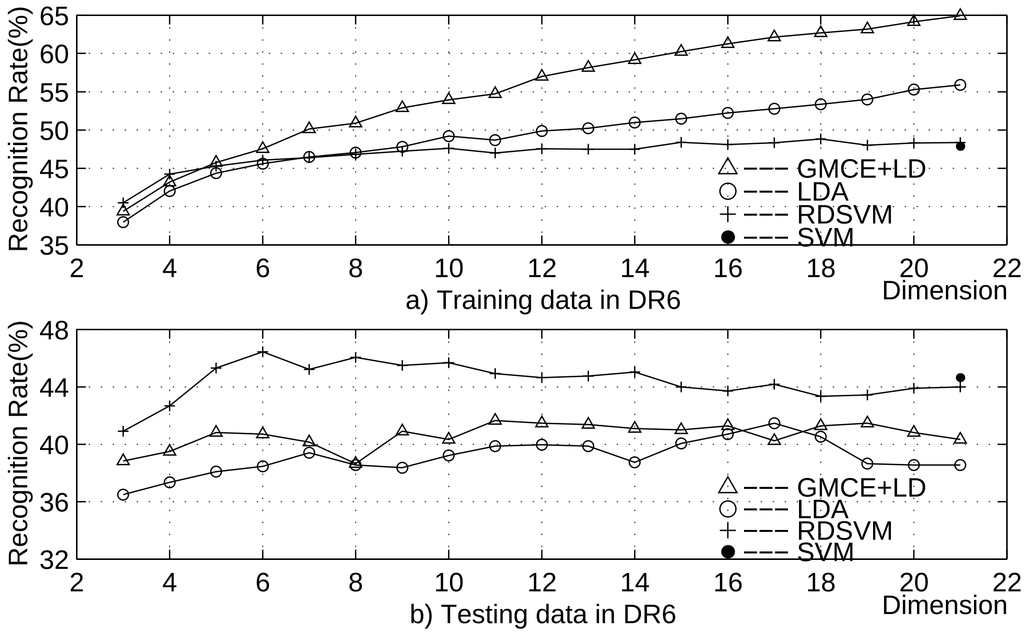


Figure 9.30: Results of RDSVM on DR6

7. Results on /dr7

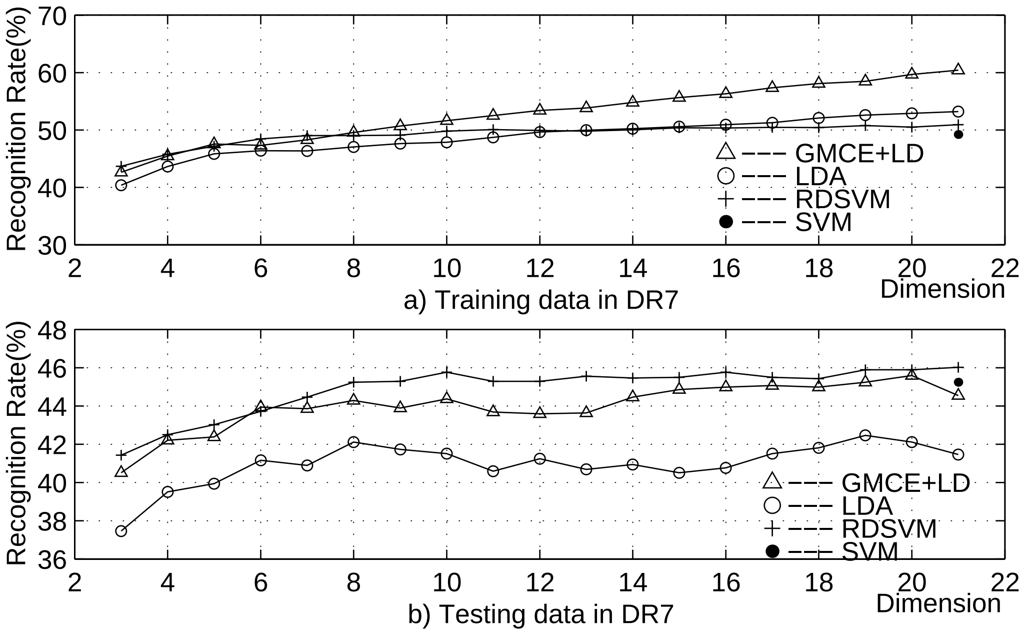


Figure 9.31: Results of RDSVM on DR7

8. Results on /dr8

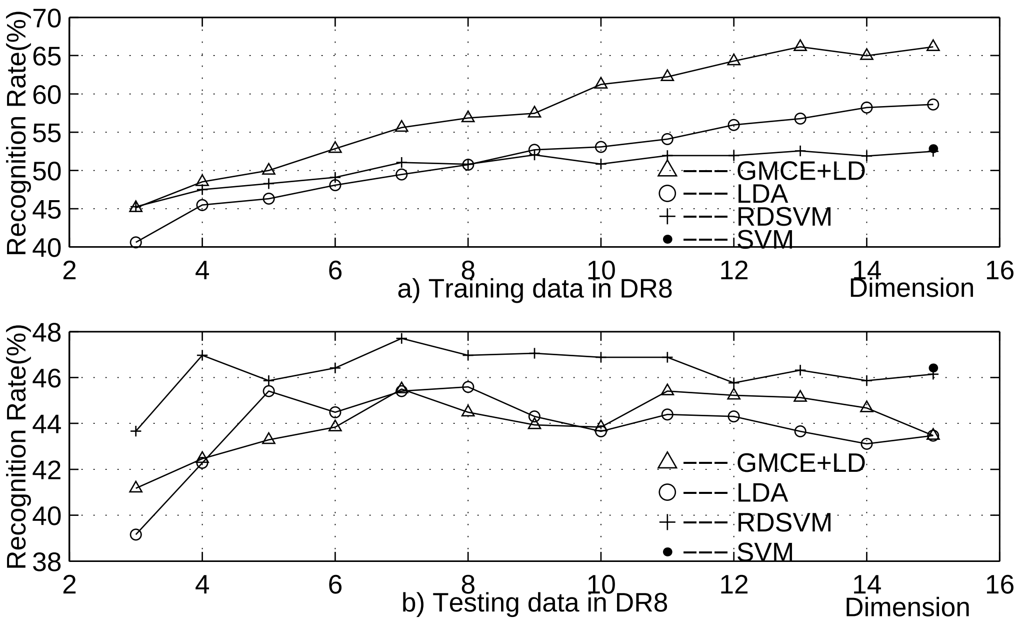


Figure 9.32: Results of RDSVM on DR8

Observations from the results of RDSVM can be summarized as follows:

- Compared to SVM, the performance of RDSVM on full-dimensional feature space is improved on training data. RDSVM's performance on testing data (on full-dimensional feature space) is also improved on some sub-directories and remains the same on the rest.

- The performance of RDSVM on training data is poorer than that of both GMCE+LD and LDA in both medium and high dimensional feature spaces (dimension 12 $\sim$ dimension 21). In very low dimensional feature spaces (dimension 3 and dimension 4) RDSVM performs better on training data than LDA and GMCE+LD. On the rest dimensions, the performance of RDSVM is between that of GMCE+LD and LDA.

- The general performance of RDSVM on testing data is much better than that of both GMCE+LD and LDA on all dimensions. In some sub-directories, the recognition rates of RDSVM are over 5 percent ahead of those of GMCE+LD on average on all dimensions.

- The performance of RDSVM is very stable throughout all the dimensions on both training and testing data. The performance of RDSVM usually starts degrading only when the feature dimension is less than 5.

- The performance curves of RDSVM on the training data of all the eight sub-directories are fairly smooth as those of GMCE+LD and LDA. But The performance curves of RDSVM on the testing data are not as smooth as those on the training data.

9.5.2 Speaker Independent Experiment

Analysis on Feature Extraction and Classification Algorithms

In this section, the performance of feature extraction and classification algorithms, i.e. LDA, PCA, MCE, GMCE+LD, GMCE+PC, SVM and RDSVM are analysed in the speaker independent experiment. Figure 9.33 compares the performance of independent (LDA and PCA) and integrated (MCE) feature extraction and classification algorithms and that of linear (minimum distance classifier) and non-linear SVM) classifiers. Figure 9.34 compares the performance of LDA, MCE and GMCE with LD initialization criterion. Figure 9.35 compares the performance of PCA, MCE and GMCE with PC initialization criterion. Figure 9.36 compares the performance of LDA, GMCE with LD initialization criterion, SVM and RDSVM.

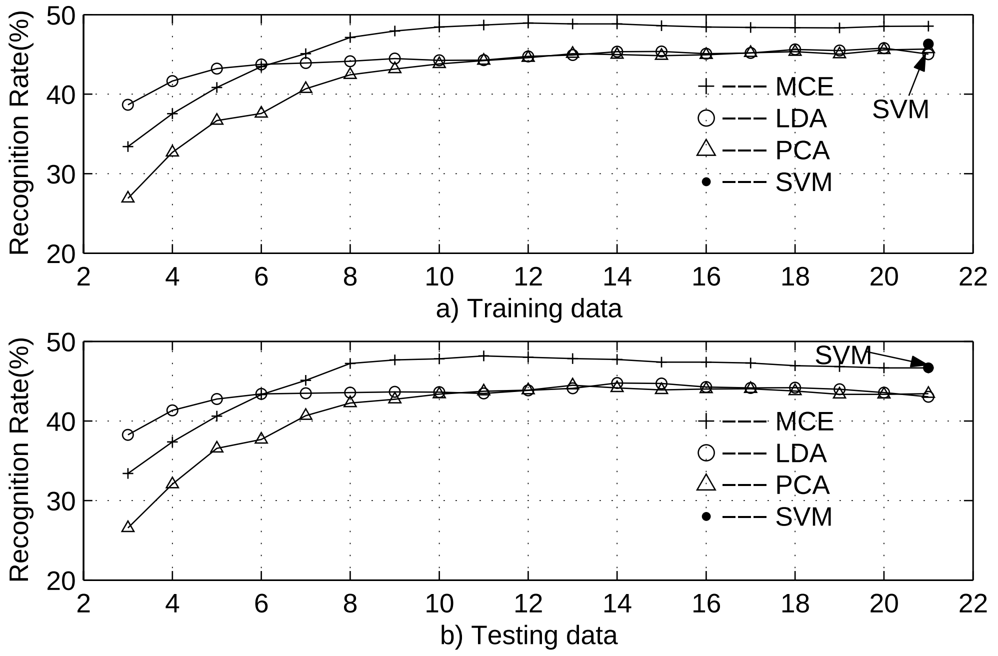


Figure 9.33: Results of LDA, PCA, MCE and SVM in speaker independent experiment.

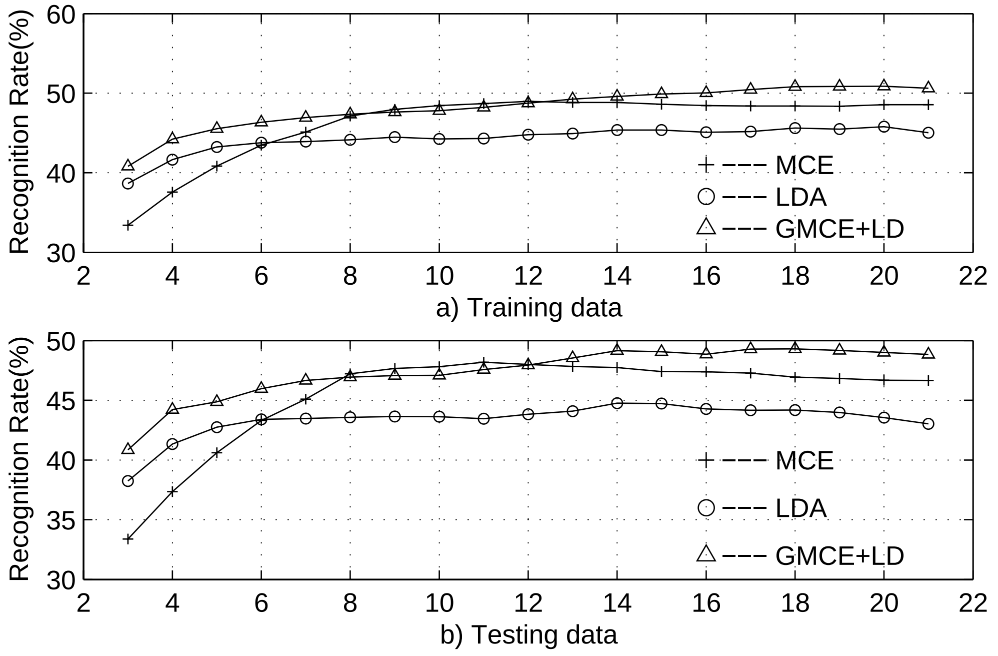


Figure 9.34: Results of LDA, MCE and GMCE+LD in speaker independent experiment.

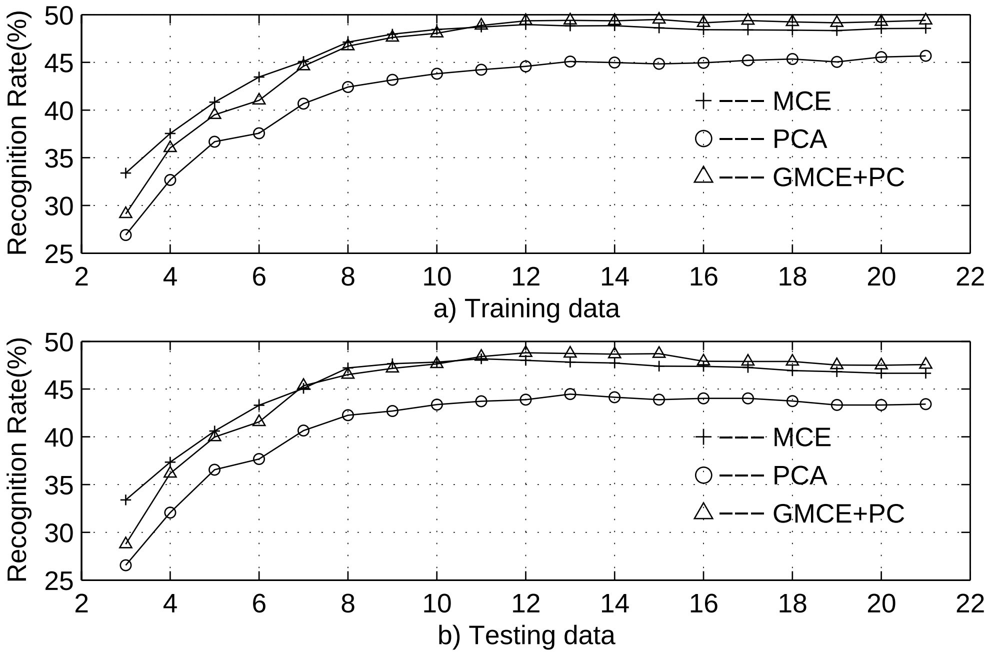


Figure 9.35: Results of PCA, MCE and GMCE+PC in speaker independent experiment.

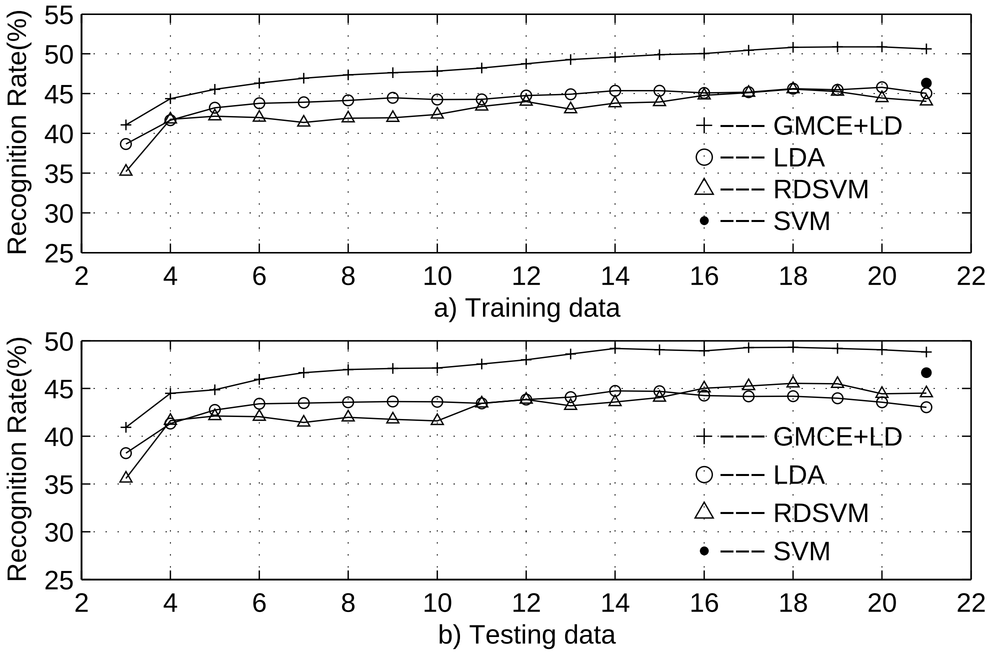


Figure 9.36: Results of LDA, GMCE+LD, SVM and RDSVM in speaker independent experiment.

Observations from the recognition results can be summarized as follows:

- The performances of all the feature extraction algorithms, including LDA, PCA and MCE training algorithm, are very stable over dimensions in the speaker independent experiment. No significant degrading on the performances can be observed until the dimensionality is reduced to be very low (< 8).

- The performances of PCA and LDA are nearly identical on high dimensions (≥ 10). But the performance of PCA degrades quickly on low dimensions (< 10), while the performance of LDA is much more stable on these dimensions.

- MCE training algorithm performs significantly better LDA and PCA on most dimensions (dimension 7 $\sim$ dimension 21). Its performance is poorer than that of LDA only on very low dimensions (dimension 3 to 5).

- The performance of SVM is slightly better than those of LDA and PCA on training data, but poorer than that of MCE. On testing data, the performance of SVM is almost identical to that of MCE, which is better than those of LDA and PCA.

- GMCE with LD initialization criterion performs better than LDA and MCE on all dimensions. The performance of GMCE with PC initialization criterion is very close to that of MCE training algorithm on most dimensions.

- The performance of RDSVM is close to that of LDA but poorer than that of GMCE on all dimensions.

- The overall performance of SVM and RDSVM, which are non-linear classifiers, are poorer that those of linear classifiers in the speaker independent experiment.

- All the algorithms have closer performances on training and testing data in the speaker independent experiment than in speaker dependent experiment. The performance curves in the speaker independent experiment are smoother that those in the speaker dependent experiment.

Speaker Independent Properties of the Algorithms

Figure 9.37 to Figure 9.42 and Table 9.9 compares the performances of the algorithms investigated in the speaker dependent and independent experiments. The performances of the algorithms in the speaker dependent experiment is represented by their average performances and maximum-minimum value area on the 8 sub-directories.

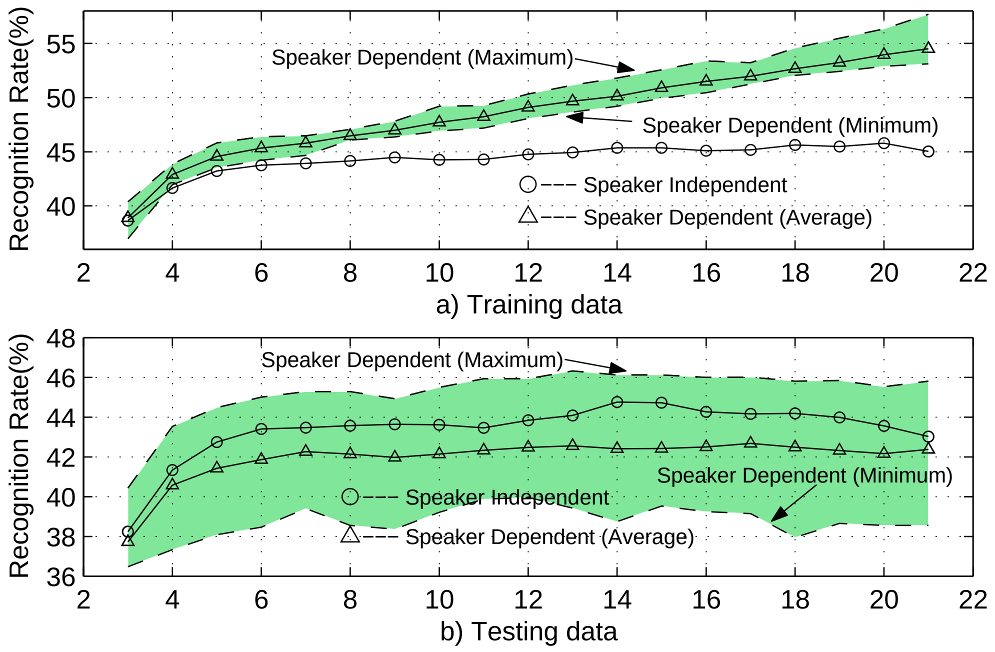


Figure 9.37: The performances of LDA in speaker dependent and independent experiments.

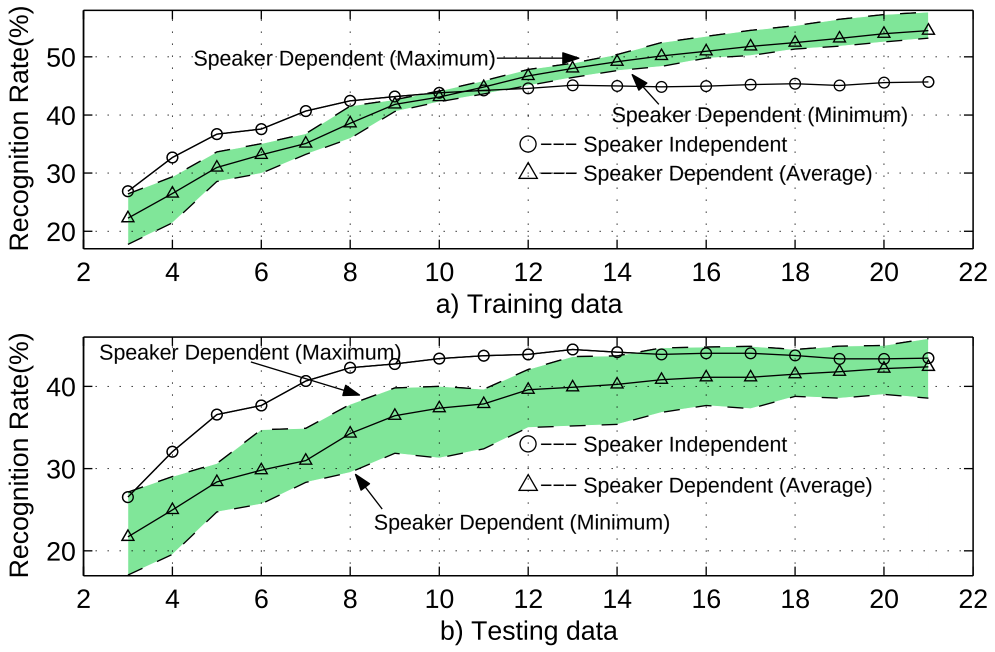


Figure 9.38: The performances of PCA in speaker dependent and independent experiments.

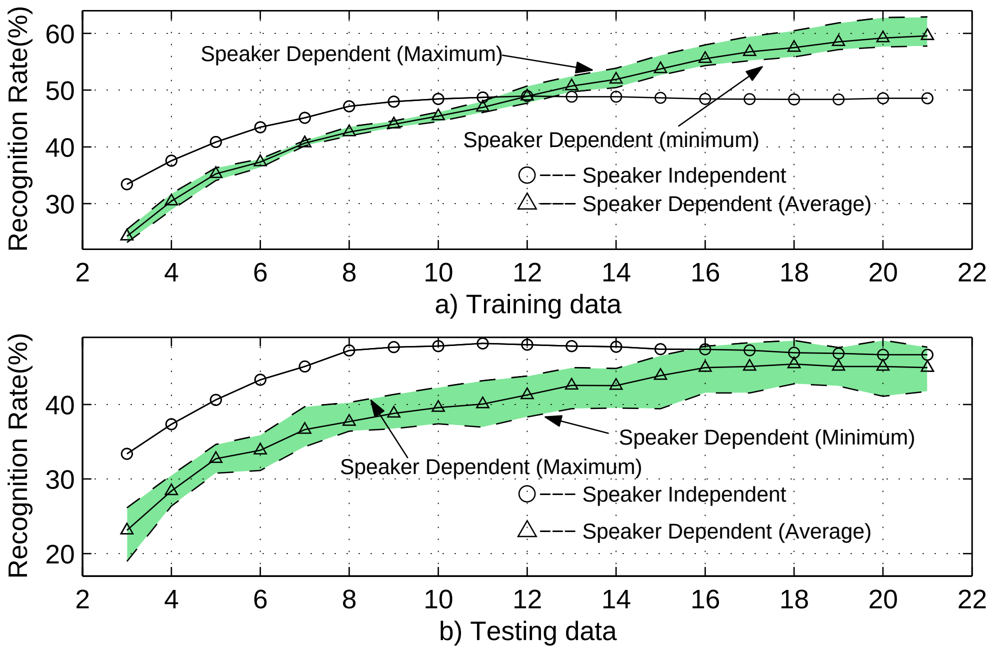


Figure 9.39: The performances of MCE in speaker dependent and independent experiments.

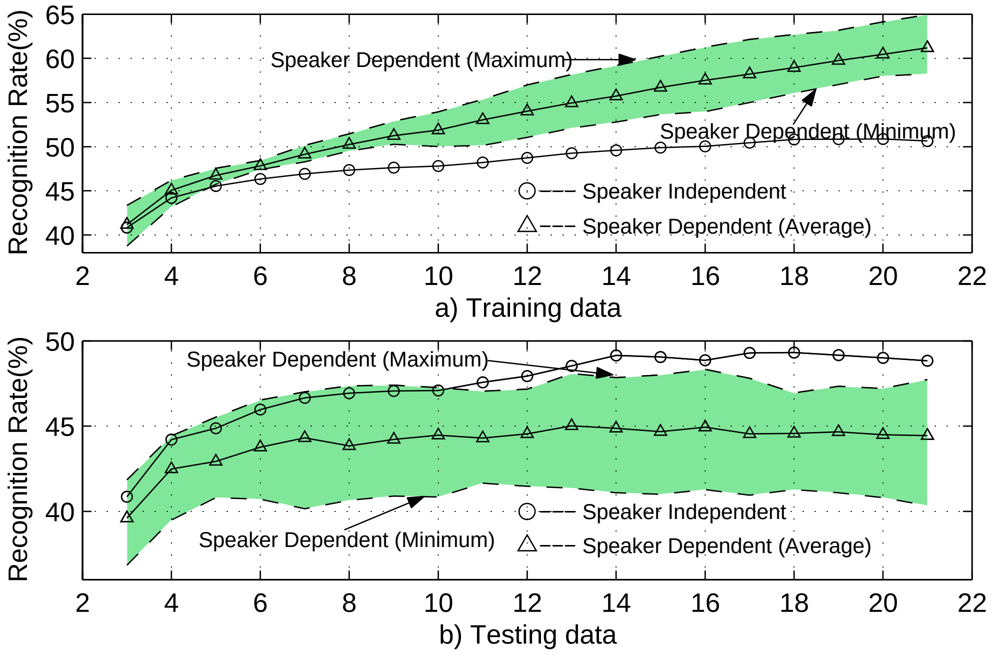


Figure 9.40: The performances of GMCE+LD in speaker dependent and independent experiments.

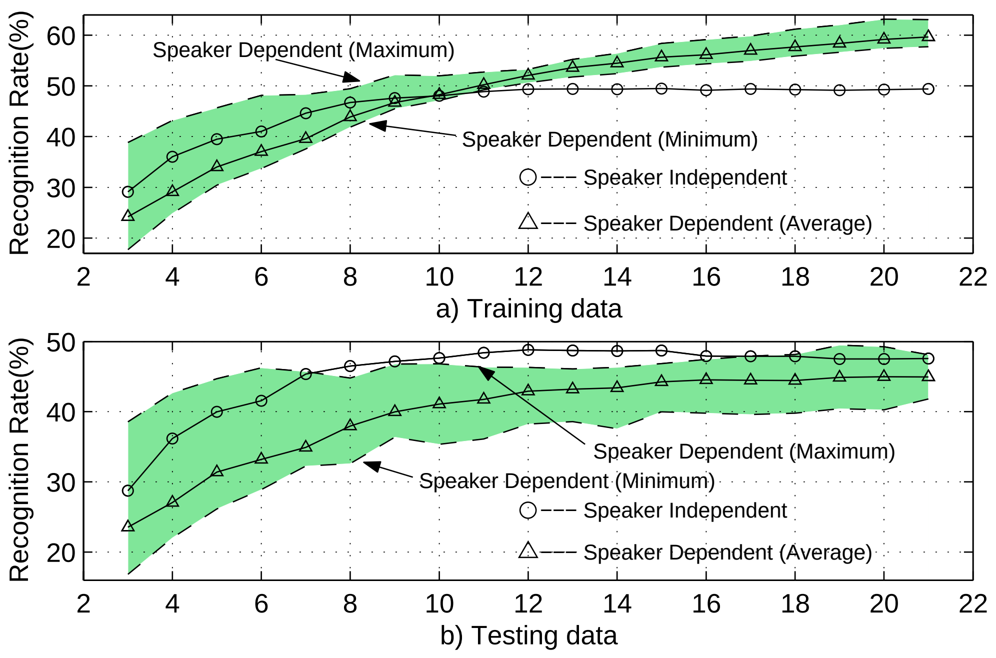


Figure 9.41: The performances of GMCE+PC in speaker dependent and independent experiments.

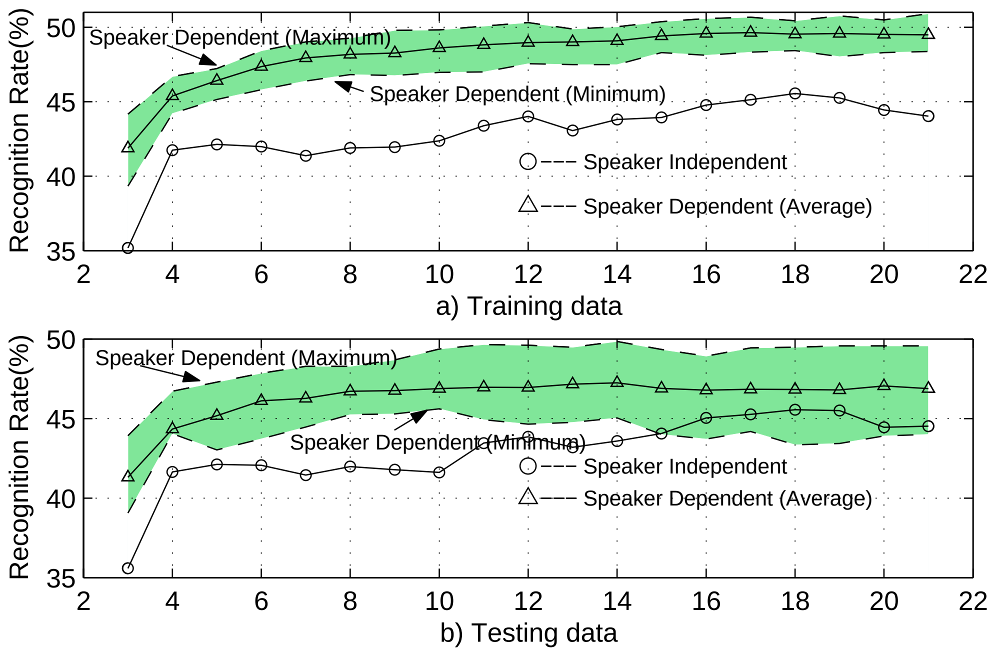


Figure 9.42: The performances of RDSVM in speaker dependent and independent experiments.

Data	Speaker Dependent			Speaker
Set	Maximum	Average	Minimum	Independent
Training	52.84	49.43	47.85	46.32
Testing	49.44	46.75	44.65	46.67

Table 9.9: The performances of SVM in speaker dependent and independent experiments.

Observations from these figures and table can be summarized as follows:

- The performance of LDA on training data in the speaker independent experiment is poorer than that in the speaker dependent experiment, especially on high dimensions. On testing data, the performance of LDA in the speaker independent experiment is between the average and the maximum performances of LDA in the speaker dependent experiment.

- The performance curve of PCA on training data in the speaker independent experiment is very flat. Hence PCA has better performances in the speaker independent experiment than in the speaker dependent experiment on low dimensions (Dimension 3 $\sim$ 10) and poorer performances on high dimensions. On testing data, the performance of PCA in the speaker independent experiment is much better than PCA's maximum performance in the speaker dependent experiment on low dimensions (4 $\sim$ 13) and they are very close on the rest dimensions.

- The performances of MCE in the speaker independent and dependent experiments have similar patterns to those of PCA: On training data, MCE's performance in the speaker independent experiment is better than that in the speaker dependent experiment on low dimensions (3 $\sim$ 11), but poorer on high dimensions. On testing data, the performance of MCE in the speaker independent experiment is much better than MCE's maximum performance in the speaker dependent experiment on most dimensions (Dimension 3 $\sim$ 16) and close on the rest dimensions.

- The performance of GMCE with LD initialization criterion on training data in the speaker independent experiments is poorer than that in the speaker dependent experiment, while on testing data, GMCE+LD's performance in the speaker independent experiment is better than it's best performance in the speaker dependent experiment on high dimensions (14 $\sim$ 21) and quite close on the rest dimensions (3 $\sim$ 13).

- On training data, the performance of GMCE with PC initialization criterion is better than GMCE+PC's average performance in the speaker dependent experiment on low dimensions (Dimension 3 $\sim$ 9), but poorer on the rest dimensions. On testing data, GMCE+PC's performance in the speaker independent experiment is better than its maximum performance in the speaker dependent experiment on some dimensions (8 $\sim$ 16) and between the maximum and average on the rest dimensions.

- The performance of SVM on training data in the speaker independent experiment is poorer than that in the speaker dependent experiment. On testing data, its performance in the speaker independent experiment is slightly lower than the average performance in the speaker dependent experiment.

- The overall performance of RDSVM in the speaker independent experiment is poorer than that in the speaker dependent experiment.

9.6 Conclusion

In this chapter, we investigated the feature extraction and classification algorithms discussed in Chapter 3, 4, 5, 6, 7 and 8 both in the speaker dependent and independent experiments. Six algorithms are involved in the investigation. They are LDA, PCA, MCE, GMCE training algorithms, SVM and RDSVM. From the observation of the experiment's results, the following conclusion can be drawn:

- *Feature Extraction* – The results of feature extraction algorithms, i.e. LDA, PCA, MCE and GMCE training algorithms, in TIMIT experiment show that integrated feature extraction and classification algorithms, i.e. MCE and GMCE training algorithms have generally better performances than independent feature extraction and classification algorithms, i.e. LDA and PCA. LDA and PCA have similar performances. But LDA is more stable in low-dimensional feature spaces, while PCA has better speaker independent properties. MCE training algorithm performs better than LDA and PCA in high-dimensional feature spaces. But its performance degrades rapidly with the decrease of feature dimensionality. GMCE training algorithm integrates the advantages of both LDA and MCE and has the best performances over all dimensions. The experiment results show that LD initialization criterion is better than PC initialization criterion. Both MCE and GMCE training have fairly good speaker independent properties.

- *Classification* – The experiment results show that SVM and RDSVM, as non-linear classification algorithms, have better generalization properties than those of linear classification algorithms, such as distance classifier. However, the results in the speaker independent experiment show that SVM and RDSVM do not have good speaker independent properties as linear classification algorithms do.

- *Speaker Dependent and Independent* – The performances of all these algorithms in speaker dependent experiments are generally better than those in speaker independent experiments. This is because the pronouncation variations in speaker dependent experiments are less than that in speaker independent experiments.

9.7 Summary of Chapter

In this chapter we first introduced the database used in our vowel classification experiment — TIMIT database and the selection of vowels. The setup of the experiment is also given. We then investigated six feature extraction and classification algorithms, i.e. LDA, PCA, MCE and GMCE training algorithms, SVM and RDSVM in speaker dependent and independent experiments. The analysis over the experiment's results is carried out and corresponding conclusion is drawn.

Chapter 10

Conclusion

In this thesis, we have discussed independent and integrated feature extraction and classification algorithms, which include LDA, PCA and MCE training algorithm, and a non-linear classification algorithm, i.e. SVM. New algorithms are proposed to mend the drawbacks of existing algorithms. The proposed algorithms are: alternative MCE training algorithm, GMCE training algorithm and RDSVM. All the algorithms concerned are evaluated on several small databases including Deterding vowels database, GLASS database, and et al. In Chapter 9, an experiment on a large database (TIMIT) is designed and conducted to evaluate and compare the performances of all the algorithms mentioned above. In the following sections, we will describe the conclusions drawn from the results of these evaluations.

10.1 Independent and Integrated Feature Extraction and Classification Methods

Independent feature extraction and classification method conducts feature extraction and classification separately. LDA and PCA are the two popular independent feature extraction algorithms. Integrated feature extraction and classification method conducts feature extraction and classification jointly. In this thesis, MCE training algorithm is applied to integrated feature extraction and classification. An alternative MCE and GMCE training algorithms are proposed to improve the performance of MCE training algorithm in the integrated tasks.

Both independent and integrated feature extraction and classification algorithms are investigated on some popular small and large scale databases. The performances of these algorithms are compared and analysed in the feature spaces with different dimensionality. The results show that LDA and PCA have similar performances in high-dimensional feature spaces. PCA, however, is more sensitive to the feature dimensionality reduction than LDA. The performance of PCA in low-dimensional feature spaces degrades faster than that of LDA. MCE training algorithm has better performance than both LDA and PCA in high-dimensional feature spaces. An alternative MCE training algorithm is proposed to further improve the performance of MCE training algorithm. But the experiments show that the performance of MCE training algorithm is highly dependent on the initialization of the transformation matrix in the integrated feature extraction and classification tasks. This problem leads to a rapid degradation on MCE's performance in low-dimensional feature spaces. This thesis thus proposes a GMCE training algorithm to mend this shortcoming of MCE training algorithm. The experiment results show that GMCE training algorithm has very stable performances even in very low-dimensional feature spaces and its overall performance is the best among all the feature extraction and classification algorithms investigated.

The speaker independent property of these algorithms is also investigated. The performances of these algorithms in both speaker dependent and independent experiments show that all the algorithms lose fitness to training data to some extent in the speaker independent experiment. But their generalization properties do not degrade. Some are even better. More specifically, LDA and GMCE training algorithm lose fitness to training data significantly on all dimensions. PCA and MCE training algorithm lose fitness only on high dimensions, but have better fitness on low dimensions. The generalization properties of LDA does not change significantly in the speaker independent experiment. Its performance in the speaker independent experiment is better than the average level of its performance in the speaker dependent experiment, but poorer than the maximum level. PCA, MCE and GMCE training algorithms have better generalization properties in the speaker independent experiment. Their corresponding performances are better than the maximum level of their performances in the speaker dependent experiment on most dimensions and very close on the rest dimensions.

10.2 Linear and Non-linear Classification Methods

Most conventional classification algorithms, such as minimum distance, likelihood and Bayesian classifier, are linear classification methods. The decision boundaries they generate are linear. SVM is a recently developed non-linear classification algorithm. It maps the parameter vectors onto a high dimensional feature space through a non-linear mapping and pursuits linear decision boundaries in the feature space. Thus the decision boundaries in the parametric spaces become non-linear.

This thesis has investigates the performance of SVM and compared it to the performance of a popular linear classification algorithm – minimum distance classifier based on Mahalanobis distance measure. A RDSVM algorithm is proposed to adopt features extraction into SVM training. The experiment results show that both SVM and RDSVM have better generalization properties than linear algorithms, but their fitness to training data is not as good as that of linear algorithms. The performance of RDSVM is very stable over all dimensions in feature dimensionality reduction tasks. However, the results in the speaker independent experiment show that SVM and RDSVM do not have good speaker independent properties as linear classifiers do.

10.3 Future Work

In this thesis, we have investigated independent and integrated feature extraction and classification algorithms, i.e. LDA, PCA and MCE training algorithm and a non-linear classification algorithms, i.e. SVM. Three algorithms, the alternative MCE, GMCE and RDSVM, are proposed to mend the drawbacks of existing algorithms. All the algorithms concerned show both merits and shortcomings in pattern recognition experiments. The experiment results also show that the merits of some algorithms are complementary. For example, MCE training algorithm is suitable for thorough optimization, while LDA is suitable for generalized optimization. SVM has good generalization properties but is unable to select features, while feature extraction algorithms is able to do that. The proposed algorithms in this thesis, such as the GMCE training algorithm and RDSVM, are based on and combine the complementary merits of different algorithms. The experiment results show that they are significantly more effective than their individual predecessors.

As discussed in Chapter 1, feature extraction is necessary because the parameter vectors are often not suitable for pattern classification. Current speech parameters used in speech recognition, such as MFCCs, are based on the knowledge of physical mechanism of speech and no class discriminantions is considered. Thus it would be interesting to integrate parameter and feature process extraction together. This is based on the idea that the speech signals are integrations of various information, such as speakers' characters, speakers' mood, dialect characters, phoneme characters and et. al. Current parameter extraction methods, such as LPC and MFCC extractors, pack all the information into speech parameters without discrimination. If feature extraction is embeded into parameter extraction process, the speech features will be more effective and efficient. A. Biem and et. al. [9, 10, 11] has done some work in this area by adopting discriminative feature extraction into filter bank design. However, integration of parameter and feature extraction is a very wide area and far more work needs to be done.

Another interesting area is the application of the feature extraction and classification techniques investigated in continuous speech recognition. For example, Hidden Markov Model (HMM) is the most popular technique in continuous speech recognition. However, HMM uses the maximum likelihood criterion, which requires the search of all possible path of and speech input sequence. This search becomes extremely low efficient when the speech sequence is very long. MCE criterion requires much less search than maximum likelihood. Thus it is possible reduce the number search paths by embeding the MCE criterion into HMM and improve the efficiency of HMM.

Finally, speech signals are time-variant and have countless variations. Linear classifiers have significant difficulties in discript the distributions of speech classes. Non-linear classifiers have advantages in dealing with such distributions. Thus it would be interesting to apply SVM classifiers to speech recognition. A possible way would be to embed SVM into HMM framework, since HMM provides an ideal framework for continuous speech recognition.

Bibliography

[1] S.Aeberhard, O. de Vel and D.Coomans, "Comparative Analysis of Statistical Pattern Recognition Methods in High Dimensional Settings", *Pattern Recognition*, 27(8), pp. 1065-1077, 1994.

[2] M. Allerhand, *Knowledge-Based Speech Pattern Recognition*, Kogan Page Ltd, London, 1987.

[3] H.Almuallim and T.G.Dietterich, "Efficient algorithms for identifying relevant features", Proceedings of the 9th Canadian Conference on Artificial Intelligence, pages 38–45, Vancouver, BC,1992.

[4] H.C.Andrews, *Introduction to Mathematical Techniques in Pattern Recognition*, Wiley-Interscience, a Division of John Wiley & Sons Inc., New York, 1972.

[5] T.W.Anderson, "Asymptotic theory for principal component analysis", *Ann. Statist. Section*, 3, pp 77-95, 1963.

[6] S.P.Banks, *Signal Processing, Image Processing and Pattern Recognition*, Prentice Hall, New York, 1990.

[7] R.E.Bellman, *Dynamic Programming*, Princeton University Press, 1957.

[8] A.Biem and S.Katagiri, "Feature Extraction Based on Minimum Classification Error/Generalized Probabilistic Descent Method", *Proceedings of IEEE International Conference of Acoustics, Speech and Signal Processing*, Vol. 2, pp. 275-278, 1993.

[9] A.Biem and S.Katagiri, "Filter Bank Design Based on Discriminative Feature Extraction", *Proceedings of IEEE International Conference of Acoustics, Speech and Signal Processing*, Vol. 1, pp. 485-488, 1994.

[10] A.Biem, E.McDermott, S.Katagiri, "A Discriminative Filter Bank Model for Speech Recognition", *Proceedings of Eurospeech*, 1995.

[11] A.Biem, "Discriminative Feature Extraction Applied to Speech Recognition", *PhD Thesis*, The University of Paris, pp. 119-121, 1997.

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

[13] E.L.Bocchieri and J.G.Wilpon, “Discriminative analysis for feature reduction in automatic speech recognition”, *Proceedings of IEEE International Conference of Acoustics, Speech and Signal Processing*, vol.1, pp. 501-504, 1992.

[14] B.E.Boser, I.M.Guyon and V.Vapnik, “A training algorithm for optimal margin classifiers”, In Haussler, D., editro, *5th Annual ACM Workshop on COLT*, pp.144-152, Pittsburgh, PA, 1992.

[15] L.Breiman, J.H.Friedman, R.A.Olshen and C.J.Stone, *Classification and Regression Trees*, Wadsworth Inc., Belmont, California, 1984.

[16] H.Brunzell and J.Eriksson, “feature Reduction for Classification of Multidimensional Data”, *Pattern Recognition*, 33, pp. 1741-1748, 2000.

[17] C.J.C.Burges, “A tutorial on support vector machines for pattern recognition”, *Data Mining and Knowledge Discovery*, 2(2):955-974, 1998.

[18] P.Clarkson and P.J.Moreno, “On the use of Support Vector Machines for Phonetic Classification”. *proceedings of ICCASP ’99.*, 1999.

[19] N.A.Campbell, “Shrunken Estimators in Discriminant and Canonical Variate Analysis”, *Applied Statistics*, Vol. 29, No. 1, pp. 5-14, 1980.

[20] P.C.Chang, S.H.Chen and B.H.Juang, “Discriminative Analysis of Distortion Sequences in Speech Recognition”, *Proceedings of IEEE International Conference of Acoustics, Speech and Signal Processing*, Vol. 1, pp. 549-552, 1991.

[21] W.Chou, B.H.Juang and C.H.Lee, “Segmental GPD Training of HMM based Speech Recognizer”, *Proceedings of IEEE International Conference of Acoustics, Speech and Signal Processing*, Vol. 1, pp. 473-476, 1992.

[22] W.Chou, “Minimum Error Rate Training for Designing Tree-Structured Probability Density Function”, *Proceedings of IEEE International Conference of Acoustics, Speech and Signal Processing*, pp. 1507-1510, 1997.

[23] C.Cortes and V.Vapnik, Support vector networks, *Machine Learning*, 20, pp.273-297, 1995.

[24] G.W.Cottrell, "Principal Componants Analysis of Images via Back Propagation", *SPIE Proceedings in Visual Communication and Image Processing*, Vol. 1001, pp. 1070-1077, 1988.

[25] B.N.Datta, *Numerical Linear Algebra and Applications*, Brooks/Cole Publishing Company and An International Thomson Publishing Company, New York, 1995.

[26] P.Devijver and J.Kittler, *Pattern Recognition: A Statistical Approach*, Prentice-Hall, Englewood Cliffs, New Jersy, 1982.

[27] R.P.W.Dubi and E.Backer, "Discriminant analysis in a non-probabilistic context based on fuzzy labels", in *Pattern Recognition and Artificial Intelligence*, Edited by Gelsema, E.S. and Kanal, L.N., Elsevier Science Publishers B.V., pp 229-235, 1988.

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

[29] R.O.Duda, P.E.Hart and D.G.Stork *Pattern Classification*, John Wiley & Sons Press, Secong Edition, New York, 2001.

[30] R.Fletcher, *Practical Methods for Optimization*, John-Wiley and Sons, and edition, 1987.

[31] B.N.Flury, "Common principal components in k groups", *Journal of American Statistical Association*, 79, pp892-898.

[32] N.Freitas, M.Milo, P.Clarkson, M.Niranjan and A.Gee, "Sequential Support Vector Machines", *IEEE International Workshop on Neural Networks for Signal Processing (NNSP99).* Winsconsin, USA. 1999.

[33] H.P.Friedman and J.Rubin, "On Some Invariant Criteria for Grouping Data", *American Statistical Association Journal*, pp. 1159-1178, 1967.

[34] K.Fu, *VLSI for Pattern Recognition and Image Processing*, Springer-Verlag, New York, 1984.

[35] K.Fukunaga and D.R.Olsen, "An Algorithm for Finding Intrinsic Dimensionality of Data", *IEEE Transactions on Computers*, C-20(2), pp. 176-183, 1971.

[36] K.Fukunaga, *Introduction to Statistical pattern Recognition*, Second Edition, Academic Press, Inc., San Diego, 1990.

[37] A.Ganapathiraju, J.Hamaker and J.Picone, "Support Vector Machines for Speech Recognition", *Proceedings ICSLP*, Sydney, Australia, 1998.

[38] F.Girosi, M.Jones and T.Poggio, "Priors, stabilizers and basis functions: from regularization to radial, tensor and additive splines", A.I. Memo No.1430, MIT, 1993.

[39] M.A.Girshick, "On the sampling theory of roots of determinantal equations", *Ann. Math. Statist.*, 10, pp 203-224, 1939.

[40] R.C.Gonzalez and M.G.Thomason, *Syntactic Pattern Recognition, An Introduction*, Addison-Wesley Publishing Company, Reading, Massachusetts, 1978.

[41] J.C.Gower, "Some distance properties of latent root and vector methods used in multivariate analysis", *Biometrika*, 53, pp 325-338, 1966.

[42] T.J.Hastie, R.Tibshirani, "Flexible Discriminant Analysis by Optimal Scoring", *AT&T Bell Labs Technical Report*, December, 1993.

[43] T.J.Hastie, A.buja and R.Tibshirani, "Penalized Discriminant Analysis", *AT&T Bell Labs Technical Report*, December, 1993.

[44] T.J.Hastie and R.Tibshirani, "Nonparametric regression and classification part II – nonparametric classification", in *From Statistics to Neural Networks - Theory and Pattern Recognition Applications*, Edited by Cherkassky, V., Friedman, J.H. and Wechsler, H., NATO ASL Series, pp 70-82, 1993.

[45] T.J.Hastie and R.Tibshirani, "Nonparametric regression and classification part I – nonparametric regression", in *From Statistics to Neural Networks - Theory and Pattern Recognition Applications*, Edited by Cherkassky, V., Friedman, J.H. and Wechsler, H., NATO ASL Series, pp 62-69, 1993.

[46] T.J.Hastie, R.Tibshirani and A.Buja "Flexible Discriminant and Mixture Models", *Proceedings of Neural Networks and Statistics Conference*, Edinburgh, Oxford University Press, 1995.

[47] T.J.Hastie, R.Tibshirani, "Discriminant Analysis by Gaussian Mixtures", *AT&T Bell Labs Technical Report*, December, 1994.

[48] M.Hotelling, "Analysis of a Complex of Statistical Variables into Principal Components", *Journal of Educational Psychology*, 24, pp. 498-520, 1933.

[49] A.K.Jain, "Advances statistic pattern recognition", in *Pattern Recognition Theory and Applications*, Edited by P.A.Devijver and Kittler, NATO ASI Series, Springer-Verlag, New York, 1986.

[50] A.K.Jain and M.D.Ramaswami, “Classifier design with parzen window”, in *Pattern Recognition and Artificial Intelligence*, Edited by E.S.Gelsema and L.N.Kanal, Elsevier Science Publishers B.V., New York, 1988.

[51] A.Jain, *Fundmentals of Digital Image Processing*, Prentice-Hall, Englewood Cliffs, New Jersy, 1989.

[52] T.Joachims, “Making large-scale SVM learning practical”, in Scholkopf, B., Burges, C.J.C. and Smola, A.J., editors, *Advancess in Kernel Methods - Support Vector Learning*, MIT Press, Cambirdge, USA, 1998.

[53] I.T.Jolliffe, *Principal component analysis*, Springer-Verlag, New York, 1986.

[54] B.H.Juang and S.Katagiri, “Discriminative Learning for Minimum Error Classification”,*IEEE Transactions on Signal Processing*, Vol. 40, No. 12, December, 1992.

[55] N.Kambhatla, “Local Models and Gaussian Mixture Models for Statistal Data Processing”, *PhD Thesis*, Oregon Graduate Institute of Science and Technology, 1996.

[56] S.Katagiri, C.H.Lee and B.H.Juang, “A Generalized Probabilistic Descent Method”, *Proceedings of the Acoustic Society of Japan, Fall Meeting*, pp. 141-142, 1990.

[57] I.Komori and S.Katagiri, “GPD Training of Dynamic Programming-based Speech Recognizer”, *Journal of Acoustical Society of Japan(E)*, Vol. 13, No. 6, pp. 341-349, 1992.

[58] W.J. Krzanowski, “Principal component analysis in the presence of group structure”, *Applied Statistics*, 33, pp164-168, 1984.

[59] N.Kumar and A.G.Andreou, “A generalization of Linear Discriminant Analysis in Maximum Likelihood Framework”, *Proceedings of the Joint Statistical Meeting, Statistical Computing section*, Chicago, Aug 4-8, 1996.

[60] N. Kumar and A.G. Andreou, “On generalizations of linear discriminant analysis”, Technical Report, JHU/ECE-9607, Johns Hopkins University, 1996.

[61] T.K.Leen, “Dynamics of Learning in Linear Feature-Discovery Networks”, *Network: Computation in Neural Systems*, Vol. 2, pp. 85-105, 1991.

[62] C.J.Leggetter, *Improved acoustic modelling for HMMs using linear transformations*, PhD Thesis, Unversity of Cambridge, 1995.

[63] C.S.Liu, C.H.Lee, W.Chou and B.H.Juang, "A Study on Minimum Error Discriminative Training for Speaker Recognition", *Journal of Acoustical Society of America*, Vol. 97, No. 1, pp. 637-648, Jan. 1995.

[64] K.V.Mardia, J.T.Kent and J.M.Bibby, *Multivariate Analysis*, Academic Press, Harcourt Brace & Co., New York, 1979.

[65] E.McDermott and S.Katagiri, "Prototype-Based Minimum Classification Error/Generalized Probabilistic Descent for Various Speech Units", *Computer Speech and Language*, Vol. 8, No. 8, pp. 351-368, 1994.

[66] E.McDermott and S.Katagiri, "String-Level MCE for Continuous Phoneme Recognition", *Procceedings of Eurospeech'97*, Vol. 1, pp. 123-126, 1997.

[67] E.McDermott and S.Katagiri, "Prototype-Based Discriminative Training for Various Speech Units", *International Conference on Acoustics, Speech and Signal Processing*, Vol. 1, pp. 417-420, 1992.

[68] S. Mika, G. Ratsch, J. Weston, B. Scholkopf and K.-R. Muller. "Fisher Discriminant Analysis with Kernels", *Proceedings of IEEE Neural Networks for Signal Processing Workshop*, 1999.

[69] S.Mika, B.Scholkopf, A.Smola, K.-R.Muller, M.Scholz, and G.Ratsch, "Kernel PCA and de-noising in feature spaces", in *Advances in Neural Information Processing Systems*, 1999.

[70] H.Niemann, *Pattern Analysis*, Springer series in information sciences, Springer-Verlag, Berlin, 1981.

[71] E.Oja, *Subspace Methods of Pattern Recognition*, John Wiley and Sons Inc., New York, 1983.

[72] E.E.Osuna, R.Freund and F.Girosi, "Support vector machine: training and applications", A.I. Memo No. 1602, C.B.C.L. Paper No. 144, MIT, 1997.

[73] E.E.Osuna, R.Freund and F.Girosi, "Training support vector machine: an application to face detection", *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 130-136, 1997.

[74] E.E.Osuna, R.Freund and F.Girosi, "An improved training algorithm for support vector machines", *IEEE Workshop on Neural Networks for Signal Processing*, pp. 24-26, Amelia Island, FL, USA, September, 1997.

[75] K.K.Paliwal, “Dimensionality Reduction of the Enhanced Feature Set for the HMM-Based Speech Recognizer”, *Digital Signal Processing*, No. 2, pp. 157-173, 1992.

[76] K.K.Paliwal, M.Bacchiani and Y.Sagisaka, “Simultaneous Design of Feature Extractor and Pattern Classifier Using the Minimum Classification Error Training Algorithm”, *Proceedings of IEEE Workshop on Neural Networks for Signal Processing*, Boston, USA, pp. 67-76, September, 1995.

[77] K.Pearson, “On lines and planes of closet fit to systems of points in space”, *Phil. Mag.*, No.6, Vol 2, pp559-572, 1901.

[78] W.L.Poston and D.J.Marchette, “Recursive Dimensionality Reduction Using Fisher’s Linear Discriminant”, *Pattern Recognition*, Vol. 31, No. 7, pp. 881-888, 1998.

[79] D.Rainton and S.Sagayama, “Minimun Error Classification Training of HMMs-Implementation Details and Experimental Results”, *Journal of Acoustical Society of Japan(E)*, Vol. 13, No. 6, pp. 379-387, 1992.

[80] C.R. Rao, “The use and interpretation of principal component analysis in applied research”, *Sankhya A*, 26, pp 329-358, 1964.

[81] T. Robinson, Dynamic Error Propogation Networks, PhD Thesis, Cambridge University Engineering Department, February 1989.

[82] V.Roth and V.Steinhage, “Nonlinear discriminant analysis using kernel functions”, Technical Report, Nr IAI-TR-99-7, ISSN 0944-8535, University Bonn, 1999.

[83] F.E.Shaudys and T.K.Leen, “Feature selection for improved classification”, *International Conference on Neural Networks*, Baltimore, 1992.

[84] M.Scherf and W.Brauer, “Feature selection by means of a feature weighting approach”, *Rechnical Report No. FKI-221-97*, Forschungsberichte Kunstliche Intelligenz, Institut fur Informatik, Technische Universitat Munchen, 1997.

[85] B.Scholkopf, C.Gurges and V.Vapnik, “Extracting support data for a given task”, *Proceedings of First International Conference on Knowledge Discovery and Data Mining*, Menlo Park, 1995.

[86] B.Scholkopf, C.Gurges and V.Vapnik, “Incorporating invariances in support vector learning machines”, *International Conference on Artificial Neural Networks – ICANN’96*, pp. 47-52, Berlin, 1996.

[87] B.Scholkopf, P.Bartlett, A.Smola and R.Williamson, "Support vector regression with automatic accuracy control", *Proceedings of 8th International Conference on Artificial Neural Netwoks*, Perspectives in Neural Computing, pp.111-116, Berlin, 1998.

[88] B.Scholkopf, A.Smola and K.-R.Muller, "Nonlinear component analysis as a kernel eigenvalue problem", *Neural Computaton*, 10:1299-1319, 1998.

[89] A.J.Smola and B.Scholkopf, "A tutorial on support vector regression", *NeuroCOLT2 Technical Report Series NC2-TR-1998-030*, ESPRIT working group on Neural and Computational Learning Theory "NeuroCOLT 2", 1998.

[90] A.J.Smola, B.Scholkopf and K.Muller, "General cost functions ofr support vector regression", In Downs, T., Frean, M. and Gallagher, M., editors, *Proceedings of the Ninth Australian Conference on Neural Networks*, pp.79-83, Brisbane, Australia, 1998.

[91] A.J.Smola, B.Scholkopf, "On a kernael-based method for pattern recognition, regression, approximation and operator inversion", *Algorithmica*, 1998.

[92] R.A.Sukkar and J.G.Wilpon, "A Two-pass Classifier for Utterance Rejection in Keyword Spooting", *Proceedings of IEEE International Conference of Acoustics, Speech and Signal Processing*, Vol. 2, pp. 451-454, 1993.

[93] D.X.Sun, "Feature Dimension Reduction Using Reduced-Rank Maximum Likelihood Estimation For Hidden Markov Model", *Proceedings of Internation Conference on Spoken Language Processing*, Philadelphia, USA, pp. 244-247, 1996.

[94] B. Tian and M.R. Azimi-Sadjadi, "Comparison of two different PNN training approaches for satellite cloud data classification", IEEE Transactions on Neural Networks, vol 12, no. 1, pp. 164-168, 2001.

[95] "The DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus (TIMIT)", [On-line], Available at URL: http://www.ldc.upenn.edu/readme_ files/timit.readme.html

[96] V.Vapnik and A.Lerner, "Pattern recognition using generalized portrait method", *Automation and Remote Control*, 24, 1963.

[97] V.Vapnik and A.Chervonenkis, "A note on class of perceptrons", *Automation and Remote Control*, 25, 1964.

[98] V.Vapnik, *Estimation of Dependences Based on Empirical Data*, Springer-Verlag, Berlin, 1982.

[99] V.Vapnik, *The Nature of Statistical Learning Theory*, Springer, N.Y., 1995.

[100] V.Vapnik, *Statistical Learning Theory*, Wiley, N.Y., 1998.

[101] V.Vapnik, S.Golowich and A.J.Smola, "Support vector method for function approximation, regression estimation, and signal processing", In Mozer, M., Jordan, M. and Petsche, T., editors, *Advances in Neural Information Processing Systems 9*, pp. 281-187, Cambridge, MA, 1997, MIT Press.

[102] R.J.Vanderbei, "LOQO: An interior point code for quadratic programming", *Optimization Methods and Software*, vol. 11, pp. 451-484,1999.

[103] X.Wang and K.Paliwal, "A modified minimum classification error training algorithm for dimensionality reduction", *Journal of VLSI Signal Processing Systems*, vol 32, pp. 19-28, April 2002.

[104] X.Wang and K.Paliwal, "Feature extraction and dimensionality reduction algorithms and their application in vowel recognition", Submitted to *Pattern Recognition*, April 2002.

[105] X.Wang and K.Paliwal, "Discriminative learning and informative learning in pattern recognition", *9th International Conference on Neural Information Processing*, Singapore, November 2002.

[106] X.Wang and K.Paliwal, "Feature extraction for integrated pattern recognition systems", *Fourth Workshop on Signal Processing and Applications*, Brisbane, Australia, December 2002.

[107] X.Wang and K.Paliwal, "Generalized minimum classification error training algorithm for dimensionality reduction", *Microelectronic Engineering Research Conference 2001*, Brisbane, Australia, 2001.

[108] X.Wang and K.Paliwal, "Using minimum classification error training in dimensionality reduction", *Proceedings of the 2000 IEEE Workshop on Neural Networks for Signal Processing X*, pp. 338-345, Sydney, 2000.

[109] X.Wang, K.Paliwal and J. Chen, "Extension of minimum classification error training algorithm", *Microelectronic Engineering Research Conference 1999*, Brisbane, Australia, 1999.

[110] J.Werner, *Optimization Theory and Application*, Friedr. Vieweg & Sohn, Braunschweig/Weisbaden, 1984.

[111] J.Yang and G.A.Dumont, "Classification of Acoustical Emmission Signals via Hebbian Feature Extraction", *IEEE proceedings of the IJCNN*, Piscataway, New Jersy, Vol. 1, pp. 113-118, 1991.

[112] S.Young, D.Kershaw, J.Odell, D.Ollason, V.Valtchev and P.Woodland, "The HTK Book (for version 2.2)", Entropic, 1999.