Speaker Verification Based on Probabilistic Neural Networks with A Priori Decision Thresholds

YIU, Kwok Kwong Michael

A dissertation[†] submitted in partial fulfillment of

the requirements for the degree of

Master of Philosophy

Department of Electronic and Information Engineering

The Hong Kong Polytechnic University

April 2000

[†]This research was supported by The Hong Kong Polytechnic University Grand A/C No. G-V557.

Abstract

Speaker Verification Based on Probabilistic Neural Networks with A Priori Decision Thresholds

Speaker verification is to verify the identity of a speaker based on his or her own voice. Typically, a speaker verification system requires one or more decision thresholds for making verification decisions: accepting the users and rejecting impostors. For the purpose of comparing the performance of different systems, researchers usually adjust the thresholds during verification in order to equalise the false acceptance rate and the false rejection rate. However, in real-world environment, the thresholds should be determined prior to verification.

In conventional approaches to speaker verification, a speaker model is constructed for each user, followed by a threshold determination procedure. While this two-step approach has been successful in many situations, it does not account for the interaction between the speaker models and the decision thresholds. In this dissertation, we integrate the speaker model construction and threshold determination procedures in a single framework by using probabilistic decision-based neural networks (PDBNNs). A PDBNN can be considered as a Gaussian mixture model (GMM) with trainable decision thresholds. GMMs have been widely used as speaker models because of their capability to model arbitrary density functions. However, GMMs have limitations as

they do not provide a proper mechanism for setting decision thresholds. By using the thresholding mechanism of PDBNNs, this dissertation aims to improve the robustness of speaker verification systems against intruder attacks.

This dissertation begins with detailed illustrations to compare the decision boundaries of PDBNNs with that of GMMs. The comparison is based on two pattern recognition tasks, namely the noisy XOR problem and the classification of two-dimensional vowel data. Experimental results show that the thresholding mechanism of PDBNNs is very effective in detecting data not belonging to any known classes. Based on this finding, the dissertation explains how the networks can be extended to speaker verification. Experimental evaluations based on 138 speakers of the YOHO corpus have been conducted. It is found that the error rate obtained by the PDBNNs is about half of that of Higgins et al. (a benchmark error rate for the YOHO corpus), suggesting that the discriminative training procedure of PDBNNs is able to improve the robustness of the speaker models. It is also found that the discriminative training procedure of PDBNNs is able to embed the background speakers characteristics in the speaker models, resulting in a substantial saving in computational resources during verification.

This work has also explored various channel compensation techniques for speaker verification over the public telephone network. A new channel compensation approach, which is based on the measurement of telephone handsets' frequency responses, is proposed. The capability of various channel compensation methods, such

as cepstral mean subtraction and signal bias removal, in reducing channel distortion is compared with that of the proposed approach. Results show that the proposed approach outperforms the conventional cepstral mean subtraction but is slightly inferior to signal bias removal.

ACKNOWLEDGMENTS

I would like to express sincere gratitude to various bodies from The Hong Kong polytechnic University, where I have the opportunity to study with. My major debt is to my Chief Supervisor Dr. M. W. Mak. Without his help, this study could not be completed. Besides, he also gave me many invaluable ideas and suggestions in writing my thesis. I would also like to thank Dr. C. K. Li, who is my second supervisor.

I thank S. H. Lin and S. Y. Kung for providing me the source code of PDBNNs. I also thank N. B. Karayiannis for giving me the 2-D vowel data.

I would also like to express my appreciation to all the members of the DSP Research Laboratory, past and present, especially for Dr. Cleve K.W. Ku, Dr. Clifford S.T. Choy, Dr. Steven T.F. Lo, Mr. Stephen Kwok, Mr. Manson Siu, Mr. W. L. Hui, Dr. Y. L. Chan, Dr. Steven S.O. Choy, Mr. W.C. Poon, Mr. Wilson Poon, Mr. Charles Ning, Mr. W.D. Zhang, Mr. C.B. Chow and Dr. C. T. Leung. The countless discussions with them have been proved to be fruitful and inspiring.

I would also like to thank all members of staff of the department of Electronic and Information Engineering and the clerical staff in the General Office. They have created a creative environment for me to work in.

Finally, it is my pleasure to acknowledge the Research and Postgraduate Studies

Office of The Hong Kong Polytechnic University for its generous support over the past two years.

Last but not least, I am indebted to my parents, my elder sister and brothers for their endless support and encouragement. Without them, this study would not have the chance to be completed.

STATEMENTS OF ORIGINALITY

We summarize below the major contributions of this thesis.

- This dissertation demonstrates that the thresholding mechanism of PDBNNs can produce bounded and locally conserved decision regions, which help to minimize the false acceptance rate of speaker verification systems.

- It is found that PDBNNs are better than VQ-based models in terms of speaker modeling capability, and that PDBNNs are more robust in rejecting impostors as compared to GMMs.

- This dissertation proposes a channel normalization algorithm based on the frequency responses of telephone handsets. The proposed algorithm reduces handset variability by adding an offset to the distorted speech in the cepstral domain.

- This dissertation demonstrates that the proposed channel normalization algorithm can improve the performance of telephone-based speaker verification systems. It is found that the proposed algorithm is superior to the conventional cepstral mean subtraction but is inferior to signal bias removal.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

This chapter introduces speaker recognition and the components of typical speaker recognition systems. In addition, the key issues in telephone-based speaker verification will be discussed. The key issues are centered upon three main areas: speaker-specific features, threshold determination, background noise and channel mismatches. Finally, an overview of the dissertation is provided.

1.1 Definition of Speaker Recognition

Speaker recognition is to recognize a person solely from his/her voice. It can be divided into speaker verification and speaker identification. The former is to determine whether the voice of the claimant matches the voice of the claimed identity, whereas the latter is to classify an unknown voice as uttered by one of the registered speakers. Since speaker verification involves a binary comparison, the performance (in terms of error probabilities) is independent of population size. On the other hand, the performance of a speaker identification system degrades with an increasing number of registered speakers.

Speaker recognition can also be divided into text-dependent and text-independent. In text-dependent systems, the same set of key words are used for enrollment and recognition. In text-independent systems, on the other hand, different phrases or sentences are used. Text-dependent systems require user cooperation and are typically based on the dynamic time warped template-matching technique to accumulate the distortion between the speech of the unknown speaker and the reference templates. They usually outperform text-independent systems because precise and reliable alignment between the unknown speech and reference templates can be made. However, text-independent systems are more appropriate for forensic and surveillance applications where pre-defined key words are not available and the users are usually not cooperative or not aware of the recognition task.

1.2 Components of Speaker Recognition Systems

Typically, a speaker recognition system is composed of a front-end feature extractor, a number of speaker models, and a decision unit. The feature extractor is to derive speaker-specific features from speech signals. These features are then characterized by the speaker models. To determine the identity of an unknown speaker, his/her voice is compared with all speaker models. Then, the decision unit selects the closest matched model. To verify a claimant, the distortion between the claimant's utterance and the model of the claimed identity is compared with a threshold, with the claimant being accepted (rejected) if the distortion is smaller (larger) than the threshold.

1.3 Key Issues in Telephone-based Speaker Verification

1.3.1 Speaker-Specific Features

Speech is assumed to be a quasi-stationary process, and therefore short-time spectral analysis can be performed on speech segments. The short-time spectra are then transformed into features vectors. Speech and speaker recognition systems commonly use linear prediction (LP) analysis [14] to extract the features. Several sets of features can be derived from the LP model. However, it has been shown that the cepstral coefficients are the most effective feature for speaker recognition [5].

1.3.2 Threshold Determination

Determination of decision thresholds is an important issue in speaker verification. In most speaker verification experiments, the decision threshold is set *a posteriori* to equalize the false acceptance rate (FAR) and the false rejection rate (FRR), resulting in an equal error rate (ERR). However, in real situations, decision thresholds have to be determined prior to verification. That is, decision thresholds should be determined during enrollment according to the cost of making false rejection and false acceptance.

1.3.3 Background Noise and Channel Mismatches

Robustness against noise and channel distortion is an important issue in speaker recognition. Although most systems are fairly reliable when the training and testing conditions are similar, their performance degrades rapidly under adverse conditions such as in the presence of background noise, channel interference, handset variation,

intersession variability and long-term variability of speaker's voice.

Telephone channels exhibit a bandpass filtering effect on speech signals, where different attenuations are exerted on different spectral bands. This effect can greatly affect the performance of speaker recognition systems. In particular, mismatches in handset types (e.g., carbon or electret) during enrollment and verification could result in serious performance degradation.

1.4 Overview of the Dissertation

This work aims to address two important issues in speaker verification: threshold determination and channel mismatch. For the former, we propose to determine the *a priori* decision thresholds by probabilistic decision-based neural networks (PDBNNs). For the latter, we propose to estimate the channel by measuring the frequency responses of telephone handsets. Hence, this work not only provides novel solutions to resolve some of the practical problems facing speaker verification researchers today, but also leads to a robust telephone-based speaker verification system.

The rest of this dissertation is organized as follows. In the next Chapter, the efforts to minimize the error rate of telephone-based speaker verification systems are reviewed. In Chapter 3, we consider four approaches to describe the probability density of data samples: parametric, non-parametric, mixture models and neural networks. Chapter 4 explains the structural properties and learning rules of GMMs and PDBNNs. The performance of PDBNNs and GMMs are then compared and their characteristics are highlighted in Chapter 5. In Chapter 6, VQ, PDBNNs and GMMs

are applied to speaker verification and their performance are compared. The techniques to create telephone speech corpora and several experiments that demonstrate the effects of handset variability on speaker recognition performance are presented in Chapter 7. Finally, we conclude the findings in Chapter 8.

Chapter 2

BACKGROUND

Efforts to minimize the error rate of telephone-based speaker verification systems have been centered upon five main areas: speaker modeling, decision strategies, channel mismatch equalization, background noise compensation, and joint compensation. Recent development of these efforts together with some new cepstral normalization techniques will be elaborated in this chapter.

2.1 Speaker Modeling

The choice of speaker models depends mainly on whether the verification is text-dependent or text-independent. In the former, it is possible to compare the claimant's utterance with that of the reference speaker by aligning the two utterance at equivalent points in time using dynamic time warping (DTW) techniques [14]. An alternative is to model the statistical variation in the spectral features. This is known as the hidden Markov modeling (HMM) technique which has been shown to outperform the DTW-based methods [51].

In text-independent speaker verification, aligning speech events at word or sentence level is impossible as the words or sentences cannot be easily predicted. As a

result, methods that look at long-term speech statistics [44] or consider individual spectral vectors to be independent of each other have been proposed. The latter includes vector quantization (VQ) [74], Gaussian mixture models (GMMs) [60], and neural networks [53], [37], [38], [40], [41]. Alternatively, input speech can be categorized as belonging to some broad phonetic classes based on an ergodic HMM and then a category-dependent classifier is used for verification [56], [18]. Interesting similarities can be found among these methods. For example, VQ can be regarded as a special case of single-state HMMs, and GMMs can be viewed as a special case of the single-state continuous ergodic HMMs.

2.2 Decision Strategies

Recent research has focused on the normalization of speaker scores to minimize error rates. Early work includes the likelihood ratio scoring proposed by Higgins, et al. [20] and the cohort normalized scoring by Rosenberg et al. [68]. Subsequent work based on likelihood normalization [45], [31], [32] and minimum verification training [70] also shows that including an impostor model not only improves speaker separability, but also allows the threshold to be easily set. Rosenberg and Parthasarathy [69] established some principles for constructing impostor models and showed that those with speech closest to the reference speaker's model perform the best. Their result, however, differs from that of [60] where a gender balanced, randomly selected impostor model is found to perform better, suggesting that more work is required in this area.

2.3 Channel Mismatch Equalization

Telephone speech is typically collected under different acoustic environments and communication channels, causing mismatches between speech gathered during enrollment and verification. There are three main schools of thought to address this problem: intraframe processing, interframe processing and affine transformation.

2.3.1 Intraframe Processing

This school of thought looks at the local spectral characteristics of a given frame of speech. It has led to some influential methods such as cepstral weighting [76], bandpass liftering [24], and more recently adaptive component weighting (ACW) [4] and pole-zero post-filtering [83]. Cepstral weighting emphasizes the reliable components in the feature vectors and suppresses the ones which are susceptible to noise and channel variation. ACW, on the other hand, emphasizes the formant regions, which are more resistance to environmental changes, and attenuates the broad bandwidth spectral components. The major difference between cepstral weighting and ACW is that cepstral weighting assumes that all speech frames are subject to the same distortion, while this assumption has been avoided in ACW. Pole-zero post-filtering is based on the concept of postfilters used in speech enhancement [59]. It can be considered as a special kind of cepstral weighting; however, the weighting procedure can be defined in terms of a transfer function. It has been shown that ACW performs significantly better than the cepstral weighting scheme in a telephone-based speaker identification task [4] and that the performance of ACW and pole-zero post-filtering are compara-

ble [83]. However, these experiments have limitations, as the speech corpora (KING and TIMIT with channel simulators) they used do not allow handset variability to be properly examined. Efforts must therefore be made to investigate whether these approaches are robust to handset variations.

2.3.2 Interframe Processing

This school of thought exploits the temporal variability of a sequence of feature vectors. It makes use of the fact that the temporal characteristics of communication channels are different from those of speech. Typical examples of this approach include cepstral mean subtraction [5], pole-filtered cepstral mean subtraction [50], delta cepstrum [14], and relative spectral (RASTA) processing [19].

In cepstral mean subtraction, the stationary channel effects are compensated by subtracting the long-term mean of the distorted cepstral vectors from each of the cepstral vectors, where the mean is an estimate of the channel cepstrum. Pole-filtered cepstral mean subtraction extends this idea one step further. Instead of subtracting the cepstral mean, this method subtracts from the distorted cepstral vectors the average of a cepstral sequence whose broad bandwidth components are further broadened. This is accomplished by moving the broad bandwidth poles radially away from the unit circle. It was shown that the average of these modified LP filters is a better estimate of the channel as compared to the cepstral mean [50]. The requirement of long-term averages, however, suggests that cepstral subtraction is not appropriate for real-time implementation. Delta cepstrum is a polynomial approximation of

the time-derivative of a cepstral sequence, which has been shown to be able to alleviate convolutional distortion [14]. However, delta cepstra do not perform well by themselves; they must be used in conjunction with other static features. In RASTA processing, a bandpass filter with a spectral zero at zero frequency is applied to a sequence of feature vectors. This has the effect of suppressing the slowly varying components and therefore minimizing convolutional distortion.

All of the above methods are similar in that they have a bandpass filtering effect on the time trajectory of the spectral components. However, their filtering characteristics are quite different. For example, the bandpass filter resulting from delta processing exhibits a sharp peak at a small range of frequencies, while that resulting from RASTA processing exhibits a broad pass band. Therefore, the former introduces more modification on the linguistic components of speech than the latter. While cepstral mean subtraction is nothing more than removing the dc component of the vector sequence, RASTA processing recursively removes the temporal average, making the current output dependent on its past.

Although the above methods have been successfully applied to reduce channel mismatches, they have limitations, as they assume that the channel can be approximated by a linear filter. Most telephone handsets, however exhibit energy dependent frequency responses [64] for which linear filtering may be a poor approximation. For example, Reynolds [61] found that current techniques can only compensate for part of the mismatches. Therefore, a more complex representation of handset characteristics is required.

2.3.3 Affine Transformation

This approach aims at making the classifiers more robust by compensating the distortion during the classification stage. This is typically achieved by an affine transformation [42] to bring the test data closer to the training data. More specifically, the test data $\vec{x}$ are transformed by

$$\vec{y} = A\vec{x} + \vec{b} \tag{2.1}$$

where A is the transformation matrix and b is the bias vector. The merit of this method is that speaker models can be trained on clean speech and operated on environmental distorted speech without any retraining. Additional computation, however, is required during verification to compute the transformation matrices. Interestingly, if cepstral coefficients are used as feature vectors, the cepstral mean subtraction, pole-filtered cepstral subtraction, and cepstral weighting are special cases of affine transformation [42]. For example, if A is an identity matrix, Equation (2.1) reduces to the cepstral mean subtraction, with $\vec{b}$ being the cepstral mean; if $\vec{b}$ is a zero vector and A contains the weighting factors, Equation (2.1) implements the cepstral weighting process.

2.4 Background Noise Compensation

It has been shown that about 40% of telephone conversations contain competing speech, music, or traffic noise [12]. This figure evidences the importance of background noise compensation in telephone-based speaker verification. Early approaches

to reducing background noise include spectral subtraction [8] and projection based distortion measure [43]. More recently, statistical-based methods such as noise integration model [67] and signal bias removal [57] have been proposed. The advantage of using statistical methods is that clean reference templates are no longer required. This property is particularly important to telephone-based applications as clean speech is usually not available. Despite the promise of these noise compensation methods in speech recognition, they have not been widely applied to speaker verification because convolutional distortion rather than additive noise is believed to be the major factor that degrades verification performance.

2.5 Joint Compensation

There have been several proposals aimed at addressing the problem of convolutional distortion and additive noise simultaneously. In addition to the affine transformation mentioned above, these proposals include stochastic pattern matching [72], parallel model combination [15], state-based compensation for continuous density hidden Markov models [3], and the maximum likelihood estimation of channels' autocorrelation function and noise [82]. Despite their success in telephone speech recognition, these methods have not been widely applied to speaker verification. This is because adapting a speaker model to new environments will affect its capability of recognizing speakers [6].

Recently, researchers in Carnegie Mellon University (CMU) introduced the idea of *joint* normalization for minimizing additive noise and spectral tilt [1] [2] [33] [34]

[47] [49] [58] [48]. Joint normalization means compensating the effects due to additive noise and convolutive channel simultaneously. Several successful algorithms have been proposed, including the code-word dependent cepstral normalization (CDCN), SNR-dependent cepstral normalization (SDCN), interpolated SNR-dependent cepstral normalization (ISDCN), fixed CDCN (FCDCN), and multiple fixed codeword-dependent cepstral normalization (MFCDCN). All of these algorithms operate in the cepstral domain and perform normalization by adding a compensation vector to the distorted cepstral vectors.

The SNR-Dependent Cepstral Normalization (SDCN) [1] algorithm requires a stereo database simultaneously recorded from the training and testing environments. The correction vector depends exclusively on the instantaneous SNR of the input. One drawback of SDCN is that it is not *environment-independent*, i.e., it cannot be applied to new microphones without re-training. Another drawback is that the normalization is based on long-term statistics.

Codeword-Dependent Cepstral Normalization (CDCN) applies the maximum likelihood approach to estimating additive noise and spectral tilt. Spectral compensation is performed via a minimum mean square estimator and is *codeword-dependent*. The capability of adapting to different acoustic environments makes CDCN *environment-independent*. It has been shown that CDCN is very robust in compensating the spectral distortion caused by different kinds of microphones. However, the algorithm is computationally intensive.

Interpolated SNR-Dependent Cepstral Normalization (ISDCN) [2] and Fixed CDCN

(FCDCN) are extensions of the SDCN and CDCN algorithms. In ISDCN, the correction vectors are a function of the instantaneous SNR of the noisy input and the environmental parameters (additive noise and spectral tilt). At low SNR, additive noise dominates the channel effect, whereas at high SNR, the spectral tilt has the strongest influence. ISDCN uses a sigmoid function to interpolate a noise vector at low SNR and an equalization vector at high SNR to form a correction vector. In FCDCN, the correction vector is a function of the chosen VQ-codeword and the SNR. FCDCN is computationally more efficient than CDCN, although it requires environment-dependent training. ISDCN is also computationally efficient and does not require environment-dependent data, but it is not as accurate as CDCN.

Many other algorithms have been evolved from CDCN and SDCN. All of them estimate an additive correction vector from the noisy speech in the cepstral domain. The correction vector can be determined either by directly comparing the speech recorded simultaneously from the training and testing environments (stereo database) or by estimating the correction vector based on the testing environment only.[1] Compensation vectors can also depend on the instantaneous SNR, vector quantization codeword identity or phoneme hypothesis. Compensation vectors can also be pre-computed for a number of environments; during recognition, an appropriate compensation vector is selected from the pre-computed vectors. When none of the pre-computed vectors can represent the testing environment, we can linearly weight the pre-computed vectors to obtain a compensation vector.

---

[1] From a practical point of view, compensation algorithms that do not require a priori knowledge about the testing environment are more attractive.

Moreno, et al. proposed a new family of environmental compensation algorithms called Multivariate Gaussian Based Cepstral Normalization (RATZ) [47]. RATZ assumes that the effects of additive noise and spectral tilt can be compensated by applying corrections to the means and variances of Gaussian mixture components. Experimental evaluations show that RATZ outperforms virtually all of the previous algorithms proposed by CMU. It was also shown that blind RATZ, which does not make explicit use of stereo training data, performs almost as well as stereo RATZ.

More recently, compensation methods based on Vector Taylor Series (VTS) [48] expansion and Vector Polynomial approximationS (VPS) [58] were proposed to handle the effects of linear filtering and additive noise. VTS uses a vector Taylor series to approximate the environment function, a function that describes the testing environment. It can be applied directly to the distorted speech vectors or to the statistics representing these vectors. VTS provides a unified treatment of the noise and channel problem without the need of stereo training data.

The VPS algorithm requires two-fold approximation. Firstly, the environment function is approximated by a generic piecewise polynomial with some constraints on its coefficients. Secondly, the Gaussian probability density function of noisy speech is approximated by another polynomial. The main point is that it is more important to approximate the means and variances of the noisy speech than to approximate the environment function. It was shown that VPS outperforms CDCN, RATZ and VTS at all SNRs.

2.5.1 Summary

In this chapter, the techniques to minimize the error rate of telephone-based speaker verification systems have been elaborated. These techniques center upon five main areas: speaker modeling, decision strategies, channel mismatch equalization, background noise compensation, and joint compensation. Although a lot of effort has been put into investigating these areas, many problems still remain to be solved. For example, there is currently no perfect method to determine the decision thresholds and existing channel compensation algorithms still rely on the linearity assumption of communication channels, which has been known to be inadequate. These limitations have motivated us to explore other possible solutions to these challenging problems.

Chapter 3

STATISTICAL PATTERN CLASSIFICATION

Statistical pattern classification is basically the assignment of an object or event to one of several known classes based on statistical methods. This is equivalent to partitioning the feature space into a number of regions, with one or more regions for each class. Each region may be contiguous or consists of several disjoint sub-regions. Typical pattern classification systems characterize the statistics of the feature vectors specific to the applications. Although perfect classification is not possible since overlap between classes is inevitable, we would like to minimize the probability of misclassification, or, if some errors are more costly than others, the overall risk.

Typical classification systems consist of two essential components: a feature extractor and a classifier. The feature extractor is to extract relevant features from the raw data. One of the important roles of feature extraction is to reduce the dimensionality of the raw data. The classifier uses the low-dimensional features to assign the unknown patterns to one of the known classes.

In this chapter, we briefly review the Bayes' Theorem, which is the fundamental concept behind many practical pattern classification systems. This is followed by the introduction of various approaches to modeling data distributions, which is the fundamental building block of most classifiers.

3.1 Bayes' Theorem

Bayes' Theorem [7], [13] is a fundamental concept of the statistical approach to pattern recognition. Bayes' Theorem estimates the posterior probability, $P(\omega_i|\vec{x})$, from the prior probability, $P(\omega_i)$, and the class conditional probability, $p(\vec{x}|\omega_i)$. More specifically,

$$P(\omega_i|\vec{x}) = \frac{p(\vec{x}|\omega_i)P(\omega_i)}{p(\vec{x})} \tag{3.1}$$

where

$$p(\vec{x}) = \sum_{i=1}^{S} p(\vec{x}|\omega_i)P(\omega_i) \tag{3.2}$$

where S is the total number of classes. New patterns are assigned to the class with the largest posterior probability.

Bayes' Theorem allows the incorporation of "losses" associated with each misclassification. If $l(\alpha_j|\omega_i)$ is the loss incurred by misclassifying a pattern from class ω_i as belong to class α_j, then the *conditional risk* associated with classifying to class α_j is

$$R(\alpha_j|\vec{x}) = \sum_{i=1}^{S} l(\alpha_j|\omega_i)P(\omega_i|\vec{x}) \tag{3.3}$$

where S is the total number of classes.

The *Bayes decision rule* minimizes the overall risk by classifying to class α_j for which the *conditional risk* is minimum. The resulting minimum risk, referred to as *Bayes risk*, is the optimal performance that can be achieved.

In practice, we rarely have the complete knowledge about the classes in a pattern classification problem. Instead, we only have a limited number of sample patterns

from each class, which were generated by some underlying mechanisms of nature that we want to classify. Some means are therefore required to use these sparse, real-world patterns to design a classifier.

One approach to designing classifiers is to use the sample patterns to estimate the prior probabilities and class-conditional probabilities separately and then use Bayes' Theorem to determine the posterior probabilities. Another approach is to train a multi-layer neural network to output the posterior probabilities $P(\omega_i|\vec{x})$ directly [65].

In typical pattern classification problems, the prior probabilities can be obtained directly from the training data. However, the prior probabilities estimated from the training data may be different from those encountered in the testing environment. Corrective actions should be taken so as to account for the difference in prior probabilities between training and testing [36]. Estimation of the class-conditional densities is another difficult task. In practice, the number of available training samples is always too small, or they are too costly to obtain. This problem becomes very serious when the dimensionality of the sample patterns is large. This phenomenon is known as the *curse of dimensionality*.

Probability densities can be estimated from labelled data (supervised) as well as from unlabeled data (unsupervised). The distinction is that the training data are labelled with class information for supervised training, whereas class information is unknown for unsupervised training.

In this chapter, we consider three approaches to estimating the probability density function from training data. They are referred to as parametric methods, non-

parametric methods, mixture models, and neural networks.

3.2 Parametric methods

In parametric methods [7], [13], the probability density $p(\vec{x})$ is represented by a specific functional form with a fixed number of adjustable parameters. The parameters are optimized to give the best fit to the training data. The most commonly used parametric model is the *Gaussian* distribution. The general multivariate Gaussian density in d dimensions can be written as

$$p(\vec{x}) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left\{-\frac{1}{2}(\vec{x}-\vec{\mu})^T\Sigma^{-1}(\vec{x}-\vec{\mu})\right\} \tag{3.4}$$

where $\vec{\mu}$ and Σ are the mean and covariance of the Gaussian distribution.

Two principle techniques, *maximum likelihood* and *Bayesian inference*, are commonly used to determine the parameters of Gaussian distributions. The maximum likelihood method treats the model parameters as fixed but unknown quantities. It finds the optimal parameters by maximizing the likelihood function, $\mathcal{L}(\theta)$, derived from the training data, $\chi \equiv \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n\}$, i.e.,

$$\mathcal{L}(\theta) \equiv p(\chi|\theta) = \prod_{n=1}^{N} p(\vec{x}_n|\theta) \tag{3.5}$$

where $\mathcal{L}(\theta)$ represents the likelihood of θ given the training data set χ. Intuitively, maximizing $\mathcal{L}(\theta)$ is equivalent to choosing the value of θ that in some sense best agrees with the observed data.

In Bayesian inference, the parameters are viewed as random variables described by a prior probability distribution. Observation of new samples converts the initially

prior distribution to a posterior distribution by Bayes' Theorem and causes it to peak around the true values of the parameters. The solution is an integral over all possible values of the parameters, weighted by their posterior distribution. This process is known as *Bayesian learning*.

3.3 Non-parametric methods

In most statistical pattern recognition applications, the data density are multimodal, and therefore parametric forms become inadequate. In such cases, non-parametric methods [7], [13] can be used. These methods do not assume a particular functional form; rather, the form of the density is determined entirely from the data.

The simplest heuristic form of non-parametric density estimation is the histogram approach. A histogram is obtained by dividing the input space into a number of bins, and by approximating the density at each input value by the fraction of the patterns falling inside the corresponding bin. Two serious problems make this approach unsuitable for practical applications. First, it is difficult to choose the bin width that controls the smoothness of the estimated density. Second, if the approach is generalized to higher dimensions, an impractically large number of data patterns is required because of the "curse of dimensionality."

In practical non-parametric density estimation problems, there are two basic approaches that we can adopt: the kernel-based approach and the K-nearest-neighbour approach. In the kernel-based approach, a *kernel function* (also known as the *Parzen window* [54]) corresponding to a hypercube is centered on each of the data

points, $\vec{x}$. The probability density at point $\vec{x}$ is the summation of all kernel functions of all data points fallen into the hypercube of point $\vec{x}$, and then divided by the total number of data points, N, and the volume of hypercube, V. This is similar to the histogram approach, except that the hypercubes' locations are determined by the data points instead of by the predefined bins. Nevertheless, we still need to choose an appropriate value for the width of the hypercube. The kernel-based method also has scaling problems because the number of variables in the model grows rapidly with the number of training data points.

Unlike the kernel-based approach, the K-nearest-neighbour approach uses hyperspheres of variable volume, V. The radius of the sphere is allowed to grow until it contains precisely K data points. The probability density at point $\vec{x}$ is K/NV, where N is the total number of data points and V is the volume of the hypersphere. Again, we have to determine the smoothing parameter K. Another disadvantage of the K-nearest-neighbour technique is the requirement of excessive memory storage and processing power because all of the training data must be available for evaluating the density.

3.4 Mixture models

Both parametric and non-parametric approaches have merits and limitations. In particular, the commonly used density functions of the parametric approach rarely fit the densities actually encountered in practice, although the approach allows the density functions to be evaluated quickly. Non-parametric methods do not assume the

form of the density functions; rather the functions only depend on the data. However, the non-parametric methods require all of the data points to be stored, making the evaluation of density functions very slow.

The semi-parametric [7], [10], [77] method combines the advantages of both parametric and non-parametric methods. It does not restrict the density functions to a specific functional form and the model size only grows with the complexity of the problem. However, its disadvantage is that estimating the model parameters is computational intensive.

Mixture model is a particular form of density functions that can be trained by semi-parametric methods. The density function of a mixture model is formed from a linear combination of basis density functions. The number of basis functions M is typically much less than the number of training data N. An important property of mixture models is that they can approximate any continuous densities to an arbitrary degree of accuracy provided that the model has sufficient parameters and that the model parameters are chosen correctly.

The training method of mixture models is based on maximum likelihood and involves non-linear optimization and re-estimation (such as the EM algorithm [77]). A typical example is the Gaussian Mixture Model that will be discussed in later chapters. Various procedures have been developed for determining the parameters of a Gaussian mixture model from a set of data. Most of them are based on maximizing the likelihood of the parameters given a data set or on minimizing an error function.

3.5 Neural Networks

There are several neural network approaches to statistical pattern classification [7]. For example, the probabilistic neural network (PNN), proposed by Specht [75], is a four-layer feed-forward network with a number of general Gaussian kernels, or Parzen windows. It has been shown that PNNs can realize an optimal classifier by approximating the general homoscedastic Gaussian mixtures. It has also been shown that the outputs of multilayer perceptrons can be interpreted as the posterior probabilities of some classes [71].

3.5.1 Probabilistic Neural Networks

Probabilistic Neural Networks [75] are based on the concepts of statistical pattern recognition. A PNN uses Parzen windows to estimate the probability distributions of a given class. By using an exponential Parzen window as the activation function, a PNN can learn nonlinear decision boundaries that approach the Bayes optimal. The multivariate estimates can be expressed as

$$f_A(\vec{x}) = \frac{1}{(2\pi)^{\frac{p}{2}}\sigma^p}\frac{1}{m} \times \sum_{i=1}^{m} \exp\left[-\frac{(\vec{x}-\vec{x}_{Ai})^T(\vec{x}-\vec{x}_{Ai})}{2\sigma^2}\right] \tag{3.6}$$

where p is the dimensionality of the measurement space, m is the total number of training patterns in class A, $\vec{x}_{Ai}$ is the ith training pattern from class A and σ is a smoothing parameter.

A PNN consists of four layers, namely input layer, pattern layer, summation layer and output layer. The input layer is used to distribute the input patterns, $\vec{x}$, to the

pattern layers. The pattern layer consists of m units, one for each training patterns, e.g., $\vec{x}_{Ai}$ for the ith pattern from Class A. The weights of each pattern unit are set to one of the training patterns, $\vec{x}_{Ai}$. The output of each pattern unit is the dot product of an input pattern and the unit's weight vector. An exponential activation function is applied to the dot product. The output activation level is then passed to a summation unit. The summation unit simply sums the inputs from the pattern units of the same class. The output unit (also called decision unit) are a two-input neuron producing a binary output signal. The output unit has only a single weight, C,

$$C = -\frac{h_B l_B}{h_A l_A} \cdot \frac{n_A}{n_B} \tag{3.7}$$

where h_k is the prior probability that a feature vector $\vec{x}$ belongs to Class k ($k = A$ or B), l_k is the loss associated with a wrong decision and n_k is the number of training patterns from Class k.

PNNs can be used for classification, function mapping, and estimation of posterior probabilities. The most important advantage of PNNs is that training is simple and instantaneous. Sparse samples are adequate for good performance. PNNs can also work with time-varying statistics because old patterns can be overwritten by new ones. One of the main disadvantages of PNNs is that the size of the pattern layer can be very large since all of the training data must be stored in that layer.

3.5.2 Radial Basis Function Networks

Another type of feed-forward network is the radial basis function (RBF) networks [46]. Instead of using the dot product of input vectors and weight vectors, the hidden

units of RBF networks evaluate the *distance* between the input vectors and the weight vectors. RBF networks can be classified into normalized [46] and non-normalized [55]. For normalized RBF networks, the network output is defined as [46]

$$y_k(\vec{x}) = \frac{\sum_{j=1}^{M} \omega_{kj}\phi_j(\vec{x})}{\sum_{j=1}^{M} \phi_j(\vec{x})} \tag{3.8}$$

whereas for non-normalized RBF network [55], the output is

$$y_k(\vec{x}) = \sum_{j=1}^{M} \omega_{kj}\phi_j(\vec{x}) \tag{3.9}$$

where $\phi_j(\vec{x})$ are the basis functions. For the case of Gaussian basis functions, we have

$$\phi_j(\vec{x}) = \exp\left(-\frac{\|\vec{x} - \vec{\mu}_j\|^2}{2\sigma_j^2}\right) \tag{3.10}$$

where μ_j and σ_j^2 are the mean and variance of the j-th basis function.

An RBF network has a three-layer architecture (input, hidden, and output layers). The input layer distributes the input patterns, $\vec{x}$, to the hidden layers. Each hidden unit is actually a Gaussian basis function represented by a center $\vec{\mu}_j$ and a width parameter σ_j. The number M of basis functions is typically much less than the number N of training data, and the weight matrix, $\{\omega_{kj}\}$, is determined during training. The output of hidden unit, ϕ_j, is weighted by ω_{kj} and is connected to output unit k. A bias parameter, ω_{k0}, can be included in the linear sum by introducing an extra basis function with output being fixed at 1.0.

The three-layer architecture with linear output units immediately leads to a two-stage training procedure. First, the parameters ($\vec{\mu}_j$, σ_j) of the basis functions in the hidden layer are determined by fast unsupervised methods (such as the k-means

clustering algorithm and the k-nearest neighbour heuristic). However, the number of clusters, k, has to be predetermined to achieve optimal results. Second, the weights of the output layer are determined by the least mean square algorithm or by the least squares techniques. Because of this two-stage training approach, RBF networks have a shorter training time as compared to the backpropagation (BP) networks.

Similar to BP networks, RBF networks can approximate a function to an arbitrary degree of accuracy if sufficient basis functions are available. Beside classification, RBF networks have also been applied to function approximation [17], [16], [55] and noisy interpolation [78].

3.5.3 Summary

Statistical pattern classification is basically the assignment of an object or event to one of several classes. Bayes' Theorem is a fundamental concept of the statistical approach to pattern recognition. To make the best optimal decision, unknown patterns are assigned to the class with the largest posterior probability.

In this chapter, we have considered four approaches to describe the probability density of data samples: parametric, non-parametric, mixture models and neural networks.

Both parametric and non-parametric methods have merits and limitations. For example, the density functions of the parametric approach rarely fit the densities actually encountered in practice, although they are computationally efficient. Non-parametric methods do not assume the form of the density functions. However, they

suffer from the drawback of requiring all of the data points to be stored, which can make density evaluation very slow. The semi-parametric method combines the advantages of both parametric and non-parametric methods. It does not restrict the form of the density function, and the model size grows with the problem complexity only.

Neural networks can be considered as statistical pattern classifiers. In particular, the probabilistic neural networks (PNNs) and radial basis function (RBF) networks have been widely applied to solve practical classification problems.

One common property of them mentioned in this chapter is that all of the methods use Gaussian functions as their kernel. Although Gaussian functions provide a powerful tool for approximating multi-dimensional probability density functions, they have difficulty in detecting novel data. To resolve this problem, Roberts and Tarassenko [66] proposed a robust method for novelty detection. The method uses a Gaussian mixture model together with a decision threshold to reject unknown data. Other proposals include the Probabilistic Decision-Based Neural Networks suggested by Lin, Kung and Lin [28], where a decision threshold for each class is used to reject patterns not belonging to any known classes. These networks will be explained in detail in the next chapter.

Chapter 4

PROBABILISTIC DECISION-BASED NEURAL NETWORKS

Although Gaussian mixture models are commonly applied to pattern classification, they have limitations, as patterns not belonging to any known classes will be wrongly classified to one of the known classes. A possible solution to address this problem is to use the probabilistic decision-based neural networks (PDBNN) proposed by Lin, Kung and Lin [28]. This chapter explains the structural properties and learning rules of GMMs and PDBNNs. The differences between these two models are highlighted.

4.1 Gaussian Mixture Models

Gaussian mixture models (GMMs) are one of the semi-parametric techniques for estimating probability density functions (pdf). The output of a Gaussian mixture model is the weighted sum of R component densities, as shown in Fig. 4.1. Given a set of N independent and identically distributed patterns $\mathcal{X}_i = \{x(t); t = 1, 2, \ldots, N\}$ associated with class ω_i, we assume that the class likelihood function $p(x(t)|\omega_i)$ for class ω_i is a mixture of Gaussian distributions, i.e.,

$$p(x(t)|\omega_i) = \sum_{r=1}^{R} P(\Theta_{r|i}|\omega_i) p(x(t)|\omega_i, \Theta_{r|i}) \tag{4.1}$$

where $\Theta_{r|i}$ represents the parameters of the rth mixture component, R is the total number of mixture components, $p(x(t)|\omega_i, \Theta_{r|i}) \equiv \mathcal{N}(\mu_{r|i}, \Sigma_{r|i})$ is the probability density function of the rth component and $P(\Theta_{r|i}|\omega_i)$ is the prior probability (also called mixture coefficients) of the rth component. Typically, $\mathcal{N}(\mu_{r|i}, \Sigma_{r|i})$ is a Gaussian distribution with mean $\mu_{r|i}$ and covariance $\Sigma_{r|i}$.

It is assumed that a parametric family of mixture densities of the form (4.1) is specified and that $P(\Theta_{r|i}|\omega_i)$ and $p(x(t)|\omega_i, \Theta_{r|i})$ for $r = 1, \ldots, R$ are the "true" parameter values to be estimated. It is natural to regard $p(x(t)|\omega_i)$ in (4.1) as modeling a population consisting of a mixture of R component populations with associated component densities $p(x(t)|\omega_i, \Theta_{r|i})$ and mixing proportions $P(\Theta_{r|i}|\omega_i)$. The data set $\{x_k\}_{k=1}^N$ is assumed to be consisted of N independent, unlabeled observations, i.e., a set of N independent, identically distributed random variables with density $p(x(t)|\omega_i)$.

The training of GMMs can be formulated as a maximum likelihood problem where the mean vectors $\{\mu_{r|i}\}$, covariance matrices $\{\Sigma_{r|i}\}$, and mixture coefficients $\{P(\Theta_{r|i}|\omega_i)\}$ are typically estimated by the EM algorithm [40], [41]. More specifically, the parameters of a GMM are estimated iteratively by [1]

$$\begin{aligned}
\mu_{r|i}^{(j+1)} &= \frac{\sum_{t=1}^{N} P^{(j)}(\Theta_{r|i}|x(t))x(t)}{\sum_{t=1}^{N} P^{(j)}(\Theta_{r|i}|x(t))}, \\
\Sigma_{r|i}^{(j+1)} &= \frac{\sum_{t=1}^{N} P^{(j)}(\Theta_{r|i}|x(t))[x(t) - \mu_{r|i}^{(j)}][x(t) - \mu_{r|i}^{(j)}]^T}{\sum_{t=1}^{N} P^{(j)}(\Theta_{r|i}|x(t))}, \text{ and} \\
P^{(j+1)}(\Theta_{r|i}) &= \frac{\sum_{t=1}^{N} P^{(j)}(\Theta_{r|i}|x(t))}{N},
\end{aligned} \tag{4.2}$$

where j denotes the iteration index and $P^{(j)}(\Theta_{r|i}|x(t))$ is the posterior probability

[1] To simplify the notation, we have dropped ω_i in Equations (4.2) – (4.4).

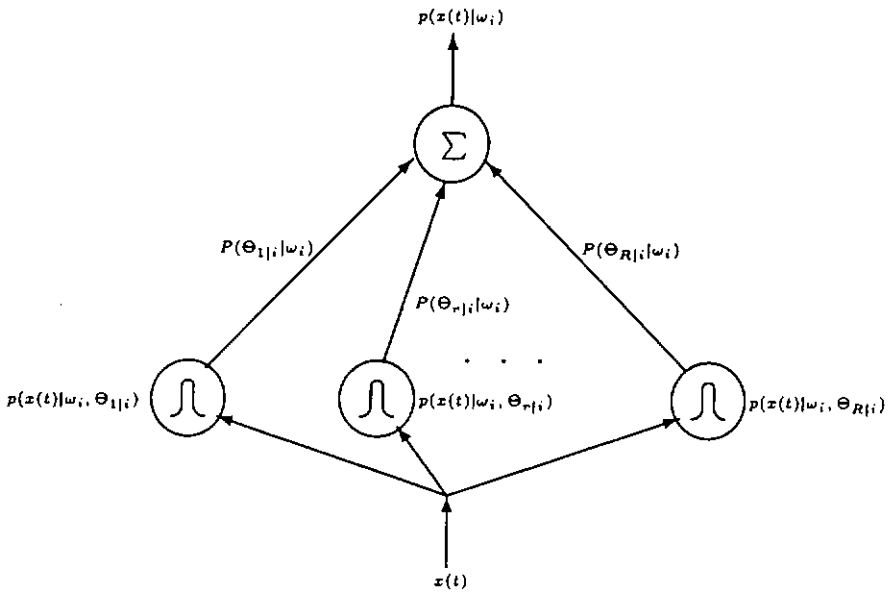


Figure 4.1: Architecture of a GMM.

for the rth mixture ($r = 1, \ldots, R$). The latter can be obtained by Bayes' theorem, yielding

$$P^{(j)}(\Theta_{r|i}|x(t)) = \frac{P^{(j)}(\Theta_{r|i})p^{(j)}(x(t)|\Theta_{r|i})}{\sum_k P^{(j)}(\Theta_{k|i})p^{(j)}(x(t)|\Theta_{k|i})} \tag{4.3}$$

in which

$$p^{(j)}(x(t)|\Theta_{r|i}) = \frac{1}{(2\pi)^{\frac{I}{2}}|\hat{\Sigma}^{(j)}_{r|i}|^{\frac{1}{2}}} \exp\left\{-\frac{1}{2}(x(t) - \mu^{(j)}_{r|i})^T(\Sigma^{(j)}_{r|i})^{-1}(x(t) - \mu^{(j)}_{r|i})\right\} \tag{4.4}$$

where I is the input dimension.

4.2 Probabilistic Decision-Based Neural Networks

4.2.1 Decision-Based Neural Networks

Decision based neural networks (DBNNs) were originally proposed by Kung and Taur [27] for robust pattern classification. One unique feature of DBNNs is that they adopt a modular network structure. In other words, a DBNN is composed of a number of small sub-networks, with each sub-network representing one class. Learning in DBNNs is based on a decision-based learning rule where the teacher only tells the correctness of classification for each training pattern. The weights of the network will be updated whenever misclassification occurs. *Reinforced learning* is applied to the subnet corresponding to the correct class so that the weight vector is updated in a direction parallel to the gradient of the discriminant function, whereas *anti-reinforced learning* is applied to the (unduly) winning subnet to move the weight vector along the opposite direction. This has the effect of increasing the chance of classifying the same pattern correctly in the future.

4.2.2 Probabilistic Decision-Based Neural Networks

PDBNNs [28] are a probabilistic variant of their predecessor, DBNNs [26]. Like DBNNs, PDBNNs employ a modular network structure as shown in Fig. 4.2. However, unlike DBNNs, they follow a probabilistic constraint. The subnet discriminant functions of a PDBNN are designed to model some log-likelihood functions of the

$$k = \begin{cases} \text{argmax}_j\{\phi(x(t), \mathbf{w}_j)\} & \text{if } \phi(x(t), \mathbf{w}_j) - T_j > 0 \quad \forall j = 1, \ldots, L \\ \text{unknown class} & \text{if } \phi(x(t), \mathbf{w}_j) - T_j \leq 0 \quad \forall j = 1, \ldots, L \end{cases}$$

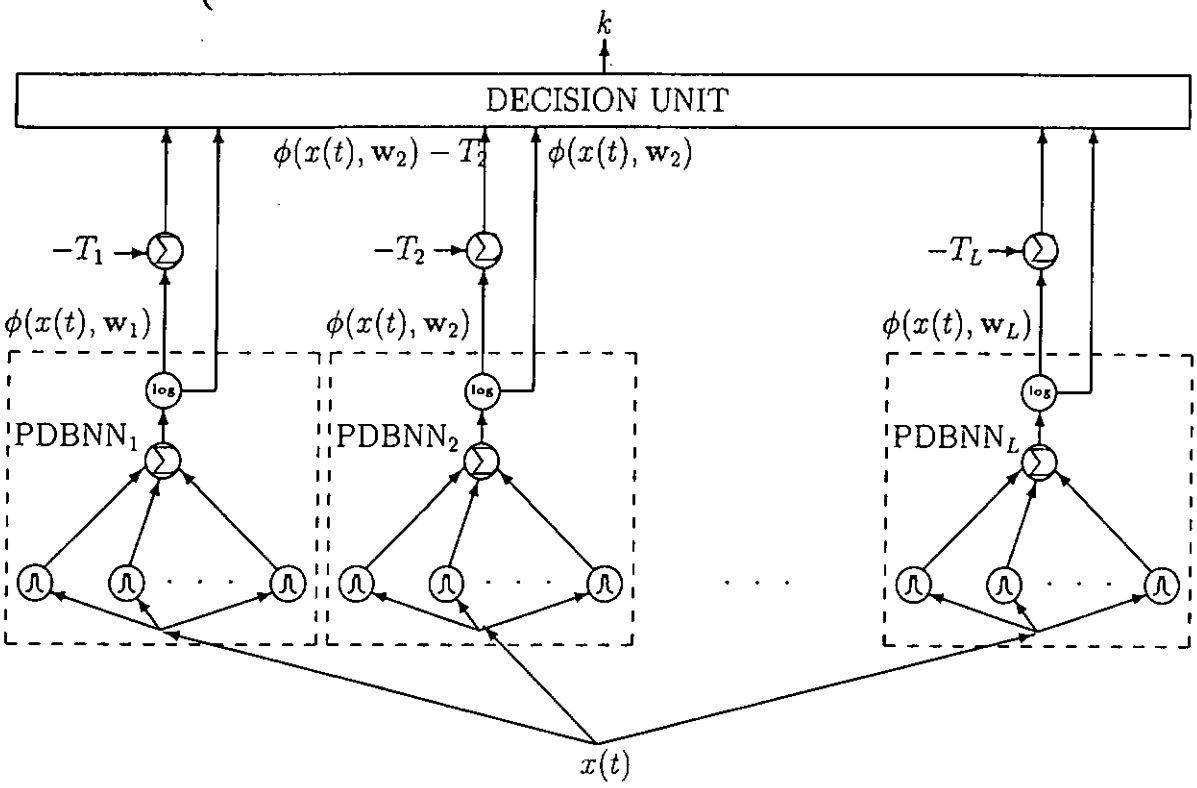


Figure 4.2: Structure of a PDBNN. Each class is modeled by a subnet. The subnet discriminant functions are designed to model the log-likelihood functions as given by Equation (4.5).

form

$$\begin{aligned} \phi(x(t), \mathbf{w}_i) &= \log p(x(t)|\omega_i) \\ &= \log \left[\sum_{r=1}^{R} P(\Theta_{r|i}|\omega_i) p(x(t)|\omega_i, \Theta_{r|i}) \right] \end{aligned} \tag{4.5}$$

where $\mathbf{w}_i \equiv \left\{ \mu_{r|i}, \Sigma_{r|i}, P(\Theta_{r|i}|\omega_i), T_i \right\}$ and T_i is the decision threshold of the subnet.

Learning in PDBNNs is divided into two phases: locally unsupervised (LU) and globally supervised (GS). In the LU learning phase, PDBNNs adopt the expectation-

maximization (EM) algorithm [11] to maximize the likelihood function

$$\begin{aligned} l(\mathbf{w}_i; \mathcal{X}_i) &= \sum_{t=1}^{N} \log p(x(t)|\omega_i) \\ &= \sum_{t=1}^{N} \log \left[\sum_{r=1}^{R} P(\Theta_{r|i}|\omega_i) p(x(t)|\omega_i, \Theta_{r|i}) \right] \end{aligned} \tag{4.6}$$

with respect to the parameters $\mu_{r|i}$, $\Sigma_{r|i}$, and $P(\Theta_{r|i}|\omega_i)$, where $\mathcal{X}_i = \{x(t); t = 1, 2, \ldots, N\}$ denotes the set of N independent and identically distributed training patterns. The EM algorithm achieves this goal via two steps: expectation (E) step and maximization (M) step. The former formulates a so-called complete-data likelihood by introducing a set of missing variables, and the latter maximizes the function in each iteration. Specifically, at iteration j of the E-step, we compute the posterior probability $P^{(j)}(\Theta_{r|i}|x(t), \omega_i)$ for cluster r $(r = 1, \ldots, R)$:

$$\begin{aligned} P^{(j)}(\Theta_{r|i}|x(t), \omega_i) &= \frac{P^{(j)}(\Theta_{r|i}|\omega_i) p^{(j)}(x(t)|\omega_i, \Theta_{r|i})}{\sum_k P^{(j)}(\Theta_{k|i}|\omega_i) p^{(j)}(x(t)|\omega_i, \Theta_{k|i})} \\ &= h_{r|i}^{(j)}(t). \end{aligned} \tag{4.7}$$

Then in the M-step, the complete-data likelihood function [28] is maximized, resulting in

$$\begin{aligned} P^{(j+1)}(\Theta_{r|i}|\omega_i) &= (1/N) \sum_{t=1}^{N} h_{r|i}^{(j)}(t), \\ \mu_{r|i}^{(j+1)} &= \left(1/\sum_{t=1}^{N} h_{r|i}^{(j)}(t) \right) \sum_{t=1}^{N} h_{r|i}^{(j)}(t) x(t), \text{ and} \\ \Sigma_{r|i}^{(j+1)} &= \left(1/\sum_{t=1}^{N} h_{r|i}^{(j)}(t) \right) \\ &\quad \times \sum_{t=1}^{N} h_{r|i}^{(j)}(t) [x(t) - \mu_{r|i}^{(j)}][x(t) - \mu_{r|i}^{(j)}]^T. \end{aligned} \tag{4.8}$$

Note that Equation (4.8) is the same as Equation (4.2), the learning algorithm of GMMs.

In the globally supervised (GS) training phase, target values are utilized to fine-tune the decision boundaries. Specifically, when a training pattern is misclassified to the ith class, reinforced and/or anti-reinforced learning are applied to update the mean vectors and covariance matrices of subnet i. Thus we have,

$$\begin{aligned}
\mu_{r|i}^{(j+1)} &= \mu_{r|i}^{(j)} + \eta_\mu \sum_{t,x(t)\in\mathcal{D}_2^i} h_{r|i}^{(j)} \Sigma_{r|i}^{-1(j)} \left[x(t) - \mu_{r|i}^{(j)}\right] \\
&\quad -\eta_\mu \sum_{t,x(t)\in\mathcal{D}_3^i} h_{r|i}^{(j)} \Sigma_{r|i}^{-1(j)} \left[x(t) - \mu_{r|i}^{(j)}\right] \\
\Sigma_{r|i}^{(j+1)} &= \Sigma_{r|i}^{(j)} + \frac{1}{2}\eta_\sigma \sum_{t,x(t)\in\mathcal{D}_2^i} h_{r|i}^{(j)} \left(H_{r|i}^{(j)}(t) - \Sigma_{r|i}^{-1(j)}\right) \\
&\quad -\frac{1}{2}\eta_\sigma \sum_{t,x(t)\in\mathcal{D}_3^i} h_{r|i}^{(j)} \left(H_{r|i}^{(j)}(t) - \Sigma_{r|i}^{-1(j)}\right)
\end{aligned} \tag{4.9}$$

where $H_{r|i}^{(j)}(t) = \Sigma_{r|i}^{-1(j)} \left[x(t) - \mu_{r|i}^{(j)}\right] \left[x(t) - \mu_{r|i}^{(j)}\right]^T \Sigma_{r|i}^{-1(j)}$, and η_μ and η_σ are user-assigned (positive) learning rates. The false rejection set $\mathcal{D}_2^i$ and the false acceptance set $\mathcal{D}_3^i$ are defined as follows:

- $\mathcal{D}_2^i = \{x(t); x(t) \in \omega_i,\ x(t)$ is misclassified to another class $\omega_j\}$
- $\mathcal{D}_3^i = \{x(t); x(t) \notin \omega_i,\ x(t)$ is classified to $\omega_i\}$

The intermediate parameter $h_{r|i}^{(j)}(t)$ is computed as in Equation (4.7). At the end of the jth epoch, the conditional prior probability $P^{(j+1)}(\Theta_{r|i}|\omega_i)$ is updated according to

$$P^{(j+1)}(\Theta_{r|i}|\omega_i) = (1/N) \sum_{t=1}^{N} h_{r|i}^{(j)}(t). \tag{4.10}$$

An adaptive learning rule is employed to train the threshold T_i of subnet i. Specifically, the threshold T_i at iteration j is updated according to

$$T_i^{(j+1)} = \begin{cases} T_i^{(j)} - \eta_t l'(T_i^{(j)} - \phi(x(t), \mathbf{w}_i)) & \text{if } x(t) \in \omega_i \quad \text{(reinforced learning)} \\ T_i^{(j)} + \eta_t l'(\phi(x(t), \mathbf{w}_i) - T_i^{(j)}) & \text{if } x(t) \notin \omega_i \quad \text{(anti-reinforced learning)} \end{cases} \tag{4.11}$$

where η_t is a positive learning parameter, $l(d) = \frac{1}{1+e^{-d}}$ is a penalty function, and $l'(d)$ is the derivative of the penalty function.

4.3 Summary

In this chapter, the structural properties and learning rules of GMMs and PDBNNs are explained. A Gaussian mixture model is the weighted sum of R component densities. The training of GMMs can be formulated as a maximum likelihood problem.

PDBNNs employ a modular network structure. Each sub-network represents one class and consists of a GMM and a decision threshold. The subnet discriminant functions of a PDBNN follow a probabilistic constraint, and they are designed to model some log-likelihood functions. Learning in PDBNNs is divided into two phases: locally unsupervised (LU) and globally supervised (GS). In the LU learning phase, PDBNNs adopt the expectation-maximization (EM) algorithm to maximize the likelihood function. In the globally supervised (GS) training phase, target values are utilized to fine-tune the decision boundaries and decision thresholds.

Chapter 5

APPLICATION TO NOISY XOR AND 2-D VOWEL PROBLEMS

Probabilistic decision-based neural networks (PDBNNs), as described in the previous chapter, can be considered as a special form of Gaussian mixture models (GMMs) with trainable decision thresholds. This chapter is to provide detailed illustrations to compare the recognition accuracy and decision boundaries of PDBNNs with that of GMMs through two pattern recognition tasks, namely the noisy XOR problem and the classification of two-dimensional vowel data. This chapter highlights the strengths of PDBNNs by demonstrating that their thresholding mechanism is very effective in detecting data not belonging to any known classes. The original PDBNNs use elliptical basis functions with diagonal covariance matrices, which may be inappropriate for modeling feature vectors with correlated components. This chapter overcomes this limitation by using full covariance matrices, and shows that the matrices are effective in characterizing non-spherical clusters.

In the next section, the methodology of applying PDBNNs and GMMs for pattern classification is explained. Experimental procedures and results are provided in Sections 5.2 and 5.3, where two problem sets, namely the noisy XOR problem

and the classification of two-dimensional (2-D) vowel data, are used to evaluate the performance of PDBNNs and GMMs. Finally, summaries are given in Section 5.4.

5.1 Methodology

We have used two problem sets, namely the noisy XOR problem and the classification of two-dimensional (2-D) vowel data, to evaluate the performance of PDBNNs and GMMs. The K-means algorithm was used to determine the initial positions of the function centers for both PDBNNs and GMMs. Then, the covariance matrices in the noisy XOR problem were initialized by sample covariance [13] in the case of one center per cluster or by the K-nearest neighbors algorithm ($K = 2$) in the case of two and four centers per cluster. For the classification of 2-D vowel data, the covariance matrices were initialized by sample covariance. The EM algorithm was subsequently used to estimate the mean vectors, covariance matrices, and prior probabilities. In the globally supervised (GS) training phase of the PDBNNs, reinforced and anti-reinforced learning were applied to fine-tune the decision boundaries and the decision thresholds.

In order to avoid over-training during the GS learning phase, the network parameters and the classification accuracy on the training set were recorded after every epoch. When the classification accuracy ceased to increase for T_m epochs, training was terminated and the network parameters producing the maximum classification accuracy were retained. T_m was set such that the maximum classification accuracy can be reliably determined. A large value of T_m can improve the reliability of the max-

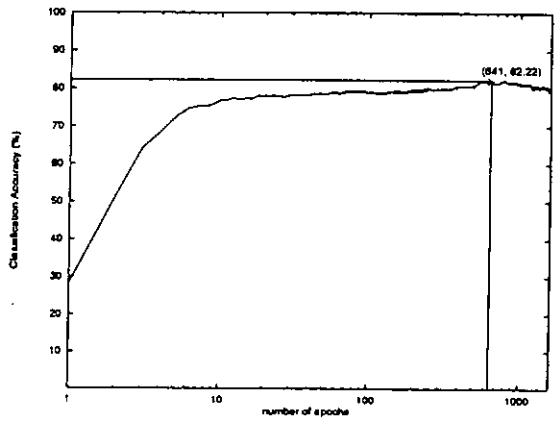


Figure 5.1: Variation of classification accuracy in the course of globally supervised learning. This curve is based on the 2-D vowel problem, with 2 centers per class.

imum classification accuracy, but it could also prolong the training time excessively. While a small value of T_m could reduce the computation time, small fluctuation in classification accuracy during GS training could result in a premature termination of the training. We found empirically that $T_m = 1000$ gives a good compromise between these two constraints. Figure 5.1 illustrates the variation of the classification accuracy during the GS learning phase in the 2-D vowel problem. Note that the accuracy continues to increase up to a maximum and then starts to decrease gradually, suggesting that setting $T_m = 1000$ is appropriate.

The performance of GMMs and PDBNNs was evaluated. The scores of the discriminant functions $\phi(x(t), \mathbf{w}_i)$ corresponding to individual classes were compared. The sub-network with the maximum score was claimed to be the winner. A recognition decision is regarded as "correct" when the winner is associated with the correct class and its score is greater than its decision threshold. When the winner is not as-

sociated with the correct class and its score is larger than its threshold, the unknown pattern is regarded as "incorrectly recognized." On the other hand, a false acceptance is encountered when the winner is not associated with the correct class but its score is greater than its threshold. When the scores of all sub-networks are smaller than their corresponding thresholds, the input pattern was regarded as "unclassifiable." More formally, a speech pattern $\vec{x}$ was considered as being correctly recognized (CR), incorrectly recognized (IR), unclassifiable (UC), or falsely accepted (FA) according to the following criteria:

- Correctly Recognized (CR):

  $\vec{x} \in C_k$ s.t. $\phi_k(\vec{x}) > T_k$ and $\phi_k(\vec{x}) > \phi_j(\vec{x})\ \forall j \neq k$

- Incorrectly Recognized (IR):

  $\vec{x} \in C_k\ \exists j \neq k$ s.t. $\phi_j(\vec{x}) > T_j$ and $\phi_j(\vec{x}) > \phi_k(\vec{x})$

- Unclassifiable (UC):

  $\vec{x} \in C_k$ but $\phi_k(\vec{x}) < T_k\ \forall k$

- Falsely Accepted (FA):

  $\vec{x} \notin$ any known classes but $\phi_k(\vec{x}) > T_k$ and $\phi_k(\vec{x}) > \phi_j(\vec{x})\ \forall j \neq k$

In the above criteria, T_k and $\phi_k(\vec{x})(= \phi(\vec{x}, \mathbf{w}_k))$ denote, respectively, the threshold and the discriminant function's output corresponding to class C_k.

For the GMMs, classification decisions were solely based on their outputs, with the largest one being selected as the recognized class. This is equivalent to assigning

a zero decision threshold to all GMMs.

5.2 *Noisy XOR Problem*

In the neural computing literature, much effort has been spent on solving the exclusive-OR (XOR) problem. An interesting property of the XOR problem is that it is the simplest logic function which is not linearly separable. As a result, single-layer networks are not able to solve this problem. The XOR problem with bipolar inputs has four training patterns: $\{(-1,\ -1)^T; 0\}$, $\{(-1,\ 1)^T; 1\}$,$\{(1,\ -1)^T; 1\}$, and $\{(1,\ 1)^T; 0\}$, where T denotes vector transpose. The patterns with the same inputs (e.g., $(1,\ 1)^T$) are classified to one class (class 0), while those with different inputs (e.g., $(-1,\ 1)^T$) are classified to another class (class 1). This problem can be extended to n dimensions, resulting in an n-bit parity problem.

The noisy exclusive-OR problem can be considered as an extension of the conventional XOR problem. Instead of using four training patterns, the noisy XOR problem investigated in this work utilizes 2000 training patterns generated from four Gaussian distributions with mean vectors and covariance matrices as shown in Table 5.1. Each distribution is responsible for generating 500 random samples. The networks were trained to classify the patterns into two classes. The patterns generated by the distributions with mean vectors $(1,\ 1)^T$ and $(-1,\ -1)^T$ are classified to one class (class 0), whereas those generated by distributions with mean vectors $(1,\ -1)^T$ and $(-1,\ 1)^T$ are classified to another class (class 1). Therefore, the noisy XOR problem can be considered as an extension of the XOR problem with the inputs being corrupted by

noise.

The objective of this experiment is to compare the decision boundaries formed by the PDBNNs and the GMMs through the noisy XOR problem. The training set consists of four Gaussian clusters (2 clusters per class). The number of center(s) per cluster was set to one, two, and four. In the experiments, we set the PDBNNs' learning rates as follows: $\eta_\mu = 0.001$, $\eta_\sigma = 0.00001$, and $\eta_t = 0.0001$. As these parameters interacted with each other nonlinearly, their values were found empirically.

Figures 5.2 and 5.3 show the test data, decision boundaries, function centers, and contours of constant basis function outputs formed by the PDBNNs and GMMs with various numbers of function centers. Table 5.2 summarizes the performance of the PDBNNs and the GMMs, respectively. The figures were based on 2000 test vectors drawn from the same population as the training set.

It is evident from Figure 5.3 that some of the decision boundaries of the GMMs extend to infinity in the input space. On the other hand, the decision boundaries of the PDBNNs are confined to the regions with a large among of test data. This is because the decision boundaries formed by the GMMs are purely dependent on the outputs of the mixture models, i.e., they are formed by the points in the input space where the two mixture outputs are equal. No matter how far away from the origin, there should be input vectors producing equal outputs in the two mixture models. As a result, some of the decision boundaries of the GMMs extend to infinity. The PDBNNs, however, use decision thresholds to reject data that produce low outputs in all subnets. Consequently, the PDBNNs' decision boundaries enclose most of the

test data. Of particular interest is that the GMMs are only able to classify the data as either Class 1 or Class 2, while the PDBNNs are able to create decision regions where data are neither classified as Class 1 nor Class 2.

Although Table 5.2 shows that the recognition accuracy of the PDBNNs is lower than that of the GMMs, its chance of incorrectly recognizing a pattern is lower. Basically, a PDBNN can be considered as a GMM with trainable decision thresholds. The thresholds provide a trade-off between the recognition accuracy and the ability to reject data not belonging to any known classes. PDBNNs are, therefore, appropriate for classification problems where false acceptance must be minimized.

Table 5.3 compares the prior probabilities (mixture components) of PDBNNs and GMMs. It shows that the GS learning of PDBNNs is able to set the prior probabilities $P(\Theta_{r|i}|\omega_i)$ of some components to zero. This is equivalent to removing the redundant function centers without affecting the decision boundaries. Evidence can also be found in Figures 5.2(b) and 5.2(c) where the shape of the decision boundaries depends mainly on some of the Gaussian components, suggesting that the other components have insignificant contribution to the mixture distribution. On the contrary, all mixture coefficients in the GMMs are non-zero, as shown in Table 5.3. Therefore, we cannot remove any mixture components in the GMMs without affecting the decision boundaries.

The noisy XOR problem we investigated utilizes 2000 training patterns generated from four Gaussian distributions with mean vectors and covariance matrices shown in Table 5.1. As each cluster is generated by one gaussian function, the number of

Class 1		Class 2	
mean vector	cov. matrix	mean vector	cov. matrix
$\begin{pmatrix} -1 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 0.9 & 0.2 \\ 0.2 & 0.6 \end{pmatrix}$	$\begin{pmatrix} -1 \\ -1 \end{pmatrix}$	$\begin{pmatrix} 0.5 & -0.2 \\ -0.2 & 1.0 \end{pmatrix}$
$\begin{pmatrix} 1 \\ -1 \end{pmatrix}$	$\begin{pmatrix} 0.7 & 0.0 \\ 0.0 & 0.7 \end{pmatrix}$	$\begin{pmatrix} 1 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 0.7 & 0.1 \\ 0.1 & 0.5 \end{pmatrix}$

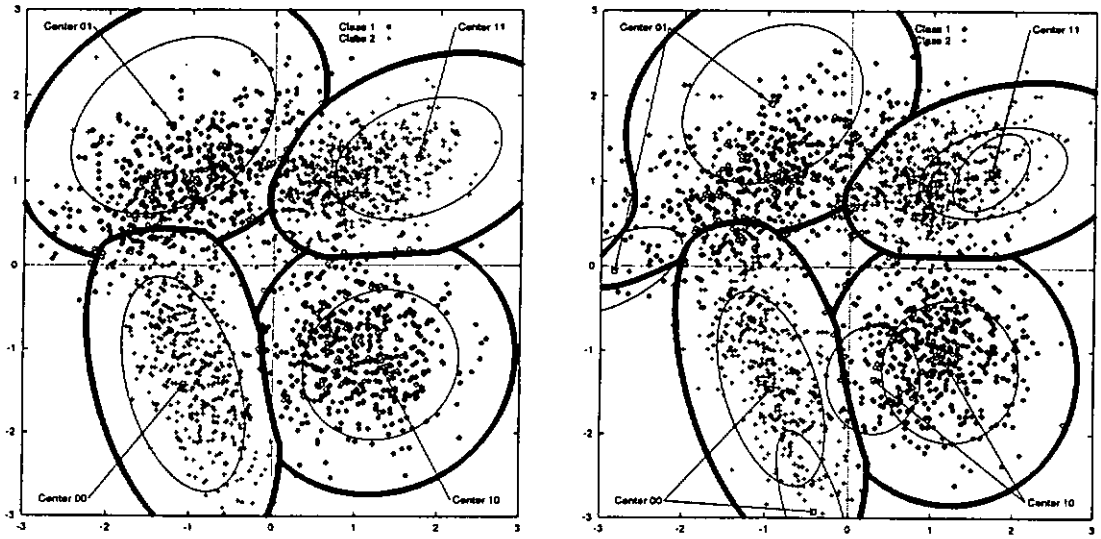
Table 5.1: Mean vectors and covariance matrices of the four clusters in the noisy XOR problem. Each cluster contains 500 samples.

	Number of Center(s) per Cluster for PDBNNs						Number of Center(s) per Cluster for GMMs					
	1		2		4		1		2		4	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
CR %	89.72	89.97	89.89	89.64	88.93	88.64	92.08	92.36	92.34	92.15	92.37	91.85
IR %	7.73	7.48	7.52	7.43	7.18	7.22	7.92	7.64	7.66	7.85	7.63	8.15
UC %	2.55	2.55	2.59	2.93	3.89	4.14	N/A	N/A	N/A	N/A	N/A	N/A

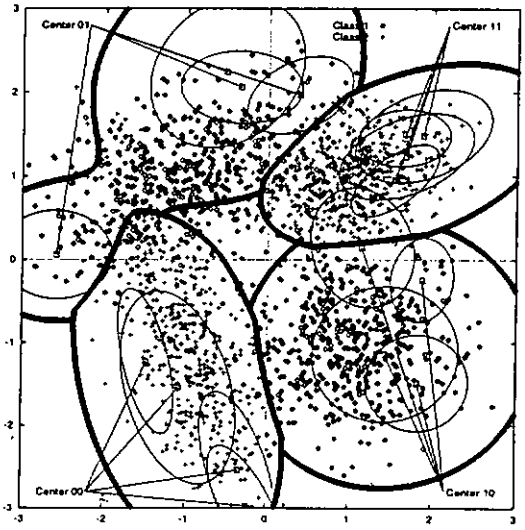
Table 5.2: Performance of PDBNNs and GMMs with 1, 2 and 4 centers based on the noisy XOR problem. (CR – Correctly Recognized, IR – Incorrectly Recognized, UC – Unclassifiable.)

Component r	PDBNNs		GMMs	
	$P(\Theta_{r\|1}\|\omega_1)$	$P(\Theta_{r\|2}\|\omega_2)$	$P(\Theta_{r\|1}\|\omega_1)$	$P(\Theta_{r\|2}\|\omega_2)$
1	0.3692	0.0000	0.1075	0.2203
2	0.0000	0.0015	0.0461	0.0797
3	0.1248	0.0003	0.1625	0.1594
4	0.0001	0.4945	0.1855	0.0398
5	0.0000	0.5036	0.0976	0.0919
6	0.0000	0.0000	0.1163	0.1684
7	0.0000	0.0000	0.1390	0.0954
8	0.5058	0.0000	0.1455	0.1452

Table 5.3: The mixture components $P(\Theta_{r|i}|\omega_i)$ of PDBNNs and GMMs in the noisy XOR problem with 8 centers per class.



(a) (b)



(c)

Figure 5.2: Decision boundaries, function centers, and contours of constant basis function outputs (thin ellipses) produced by PDBNNs with (a) 1 center per cluster, (b) 2 centers per cluster, and (c) 4 centers per cluster. Center 00, Center 01, Center 10, and Center 11 represent the four groups of centers corresponding to the four clusters.

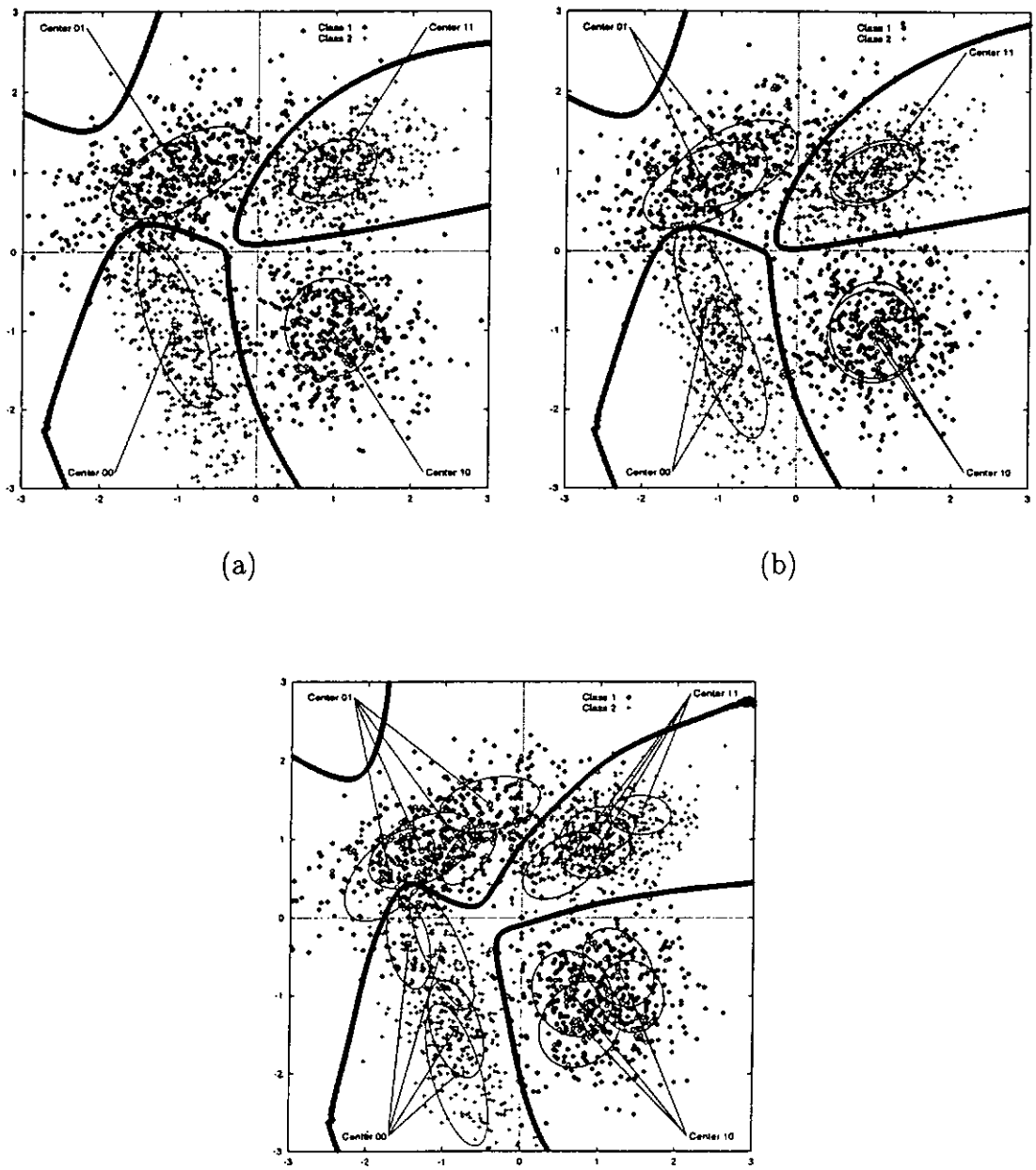


Figure 5.3: Decision boundaries, function centers, and contours of constant basis function outputs (thin ellipses) produced by Gaussian mixture models with (a) 1 center per cluster, (b) 2 centers per cluster, and (c) 4 centers per cluster. Center 00, Center 01, Center 10, and Center 11 represent the four groups of centers corresponding to the four clusters.

centers for each cluster should theoretically be equal to one. If more centers per cluster are used, the number of free parameters in the model will also increase. It is possible to achieve perfect separation of the training data by using all the training samples as cluster centers. However, such approach will *over-fit* the training data, which results in poor representation of the underlying density function (i.e., poor generalization).

The number of centers per cluster required to adequately represent a given data set is application dependent. In the case of the noisy XOR problem, since each cluster is generated by one gaussian function, the number of centers for each cluster in the model should be equal to one. The modelling accuracy with respect to the variation of the number of centers per cluster is shown in Table 5.2. It shows that the generalization performance of both PDBNNs and GMMs decreases as the number of centers per cluster increases (although it is not significant). As illustrated in Figures 5.2 and 5.3, the PDBNNs' decision boundaries are less sensitive to the number of centers per cluster. This is because after the globally supervised training, the prior probabilities $P(\Theta_{r|i}|\omega_i)$ of some components in the PDBNNs becomes very small, as shown in Table 5.3. This is equivalent to removing the redundant function centers without affecting the decision boundaries. For the GMMs, all of the mixture coefficients are non-zero. This greatly affects the smoothness of the decision boundaries.

5.3 Two-Dimensional Vowel Data

In this experiment, the performance of the PDBNNs and GMMs was compared using a set of two-dimensional (2-D) vowel data [30], [52], [25]. They were obtained by

computing the first and second formants ($F1$ and $F2$) of ten vowels spoken by 67 speakers. The resulting feature vectors were divided into a training set with 338 vectors and a test set with 333 vectors. These data sets are particularly suitable for accessing the performance of pattern classifiers as they contain overlapped and non-spherical clusters.

In the original PDBNNs, the learning rates for optimizing the threshold values are identical for both reinforced and anti-reinforced learning. In this experiment, however, the learning rate η_t for reinforced learning was set to 0.05 while that for anti-reinforced learning was set to 0.0. As a result, the thresholds were optimized by reinforced learning only. This arrangement was found to yield lower threshold values and higher recognition accuracy. The learning rates η_μ and η_σ were set to 0.5 and 150, respectively. They are identical for both reinforced and anti-reinforced learning.

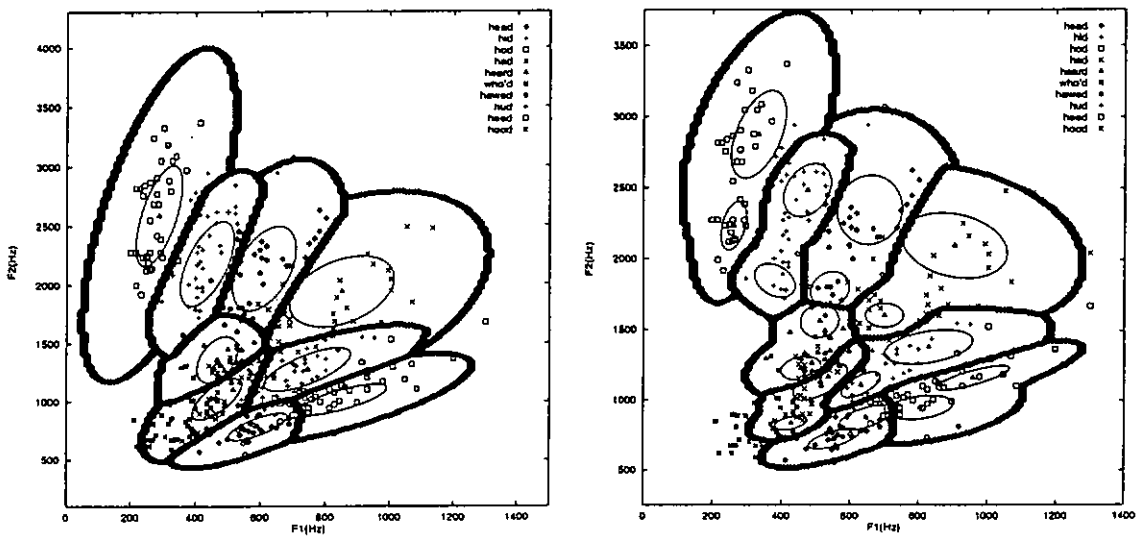
Ten vowel classes in the data set were divided into two subsets: unknown set and target set. The vowel /u/ in the word "who'd" was chosen arbitrarily as the unknown set. The remaining vowels were chosen as the target set.[1] This arrangement enables us to evaluate the robustness of the PDBNNs and GMMs in detecting data not belonging to any known classes. Each of the nine vowels in the target set was modeled by a GMM and a PDBNN.

Figures 5.4 and 5.5 show the training data, decision boundaries, contours of constant basis function outputs, and function centers formed by the PDBNNs and GMMs with various numbers of function centers. Table 5.4 summarizes their performance.

[1] As a result, 305 out of 338 vectors in the training set were used for training.

It can be seen from Figure 5.5 that some of the GMMs' decision regions are unbounded, while those formed by the PDBNNs are bounded. This is because the thresholding mechanism in PDBNNs rejects data in regions where the network outputs are lower than the decision thresholds. Table 5.4 shows that the recognition accuracy of PDBNNs is comparable to that of the GMMs. Although the GMMs are able to classify most of the data, they fail to identify the data in the unknown class (*) and blindly classify them to one of the known classes, resulting in 100% false acceptance rate. On the other hand, Figure 5.4 clearly shows that the PDBNNs are able to create decision regions where unknown data are rejected, resulting in a lower false acceptance rate. Figure 5.5(b) also shows that there is an artifact in the GMMs' decision boundaries. The triangular region (around coordinate (100,1000)) is incorrectly classified to the vowel class / ∧ / (in the word "hud") whose function centers are located at (632,1108) and (830,1381).

In [25], Karayiannis and Mi used the same set of data to evaluate the performance of what they called the Growing Radial Basis Neural Networks (GRBNNs). As they used 10 classes instead of 9, it may be difficult to compare the recognition accuracy obtained in their experiments with that of ours. However, it is interesting to compare the decision boundaries formed by the PDBNNs with those by GRBNNs. We can see from Figures 5.6 (Reprinted from [25]) that the vowel /i/ in the word "heed" is not enclosed by the decision boundaries. This means that the decision regions corresponding to /i/ extends to infinity in the F1–F2 space. On the other hand, this situation has been prevented by the thresholding mechanism of PDBNNs, as shown



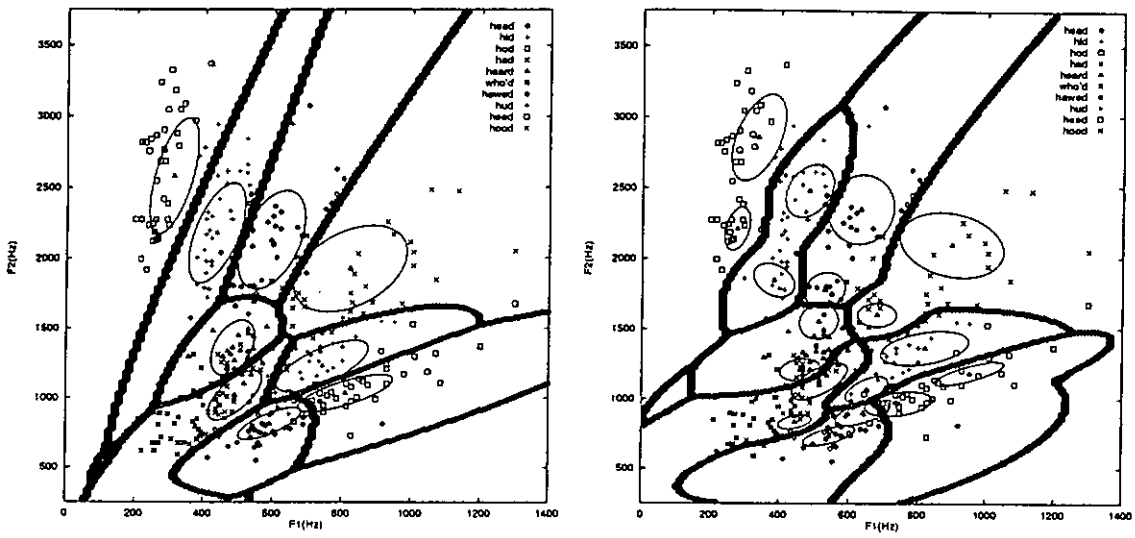
(a) (b)

Figure 5.4: Decision boundaries, function centers, and contours of constant basis function outputs (thin ellipses) produced by PDBNNs with (a) 1 center per class and (b) 2 centers per class.

in Figure 5.4(b).

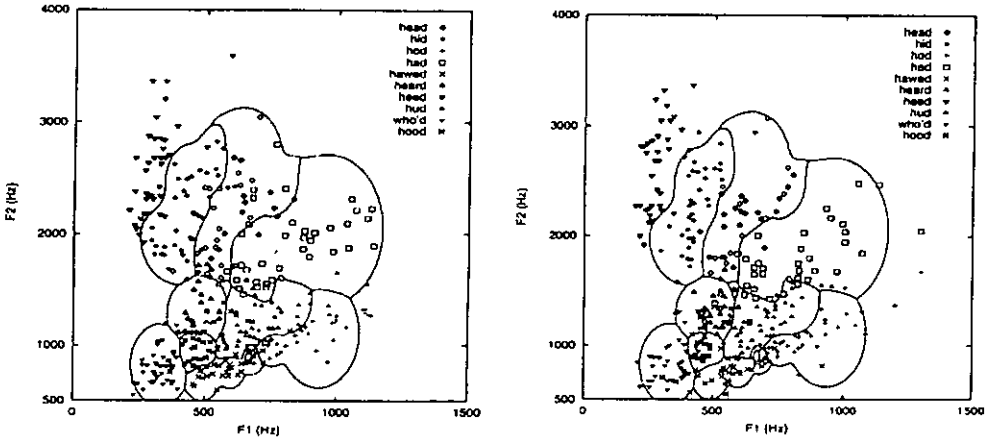
5.4 Summary

Basically, a PDBNNs can be considered as a GMMs with trainable decision thresholds. PDBNNs and GMMs are similar in that their architecture and training algorithms have a great deal of commonality. For example, both PDBNNs and GMMs contain a mixture of gaussian densities. Their major difference, however, lies in their parameter update mechanisms. For instance, PDBNNs have a globally supervised training phase in which the mean vectors and covariance matrices are adjusted and a decision threshold is determined, whereas there is no such training phase in GMMs. An



(a) (b)

Figure 5.5: Decision boundaries, function centers, and contours of constant basis function outputs (thin ellipses) produced by Gaussian mixture models with (a) 1 center per class and (b) 2 centers per class.



(a) (b)

Figure 5.6: Classification of the 2-D vowel data produced by the GRBF network: (a) the training data set and (b) the testing data set. (Reprinted from Figure 4 of [25]).

	Number of Center(s) per Cluster for PDBNNs				Number of Center(s) per Cluster for GMMs			
	1		2		1		2	
	Train	Test	Train	Test	Train	Test	Train	Test
CR %	75.68	77.97	80.34	78.14	77.77	78.39	79.40	79.02
IR %	23.42	19.60	19.36	19.63	22.23	21.61	20.60	20.98
UC %	0.90	2.43	0.30	2.24	N/A	N/A	N/A	N/A
FA %	66.67	63.64	45.15	54.24	100.00	100.00	100.00	100.00

Table 5.4: Performance of PDBNNs and GMMS with 1 and 2 centers based on the 2-D vowel data set. (CR – Correctly Recognized, IR – Incorrectly Recognized, UC – Unclassifiable, FA – Falsely Accepted.)

illustrated example is given in the noisy XOR problem in Section 5.2.

Based on two pattern recognition tasks, the performance of PDBNNs and GMMs have been compared and their characteristics have been highlighted. The locally unsupervised learning of PDBNNs and GMMs are essentially the same. However, PDBNNs have a globally supervised learning phase through which the decision boundaries and thresholds are adjusted whenever misclassification occurs. They employ a modular network structure with each class represented by one subnet. The discriminant function of each subnet is compared with that of others and with its decision threshold for making classification decisions.

In the noisy XOR problem, it was found that the globally supervised learning rule of PDBNNs is able to remove redundant function centers by gradually decreasing the corresponding prior probabilities to zero. This leads to a smaller network. In the case of GMMs, all components have contributions to the outputs; therefore, no redundant centers can be removed without affecting the decision boundaries.

With the thresholding mechanism, the performance of PDBNNs can be divided

into recognition accuracy, incorrectly recognized rate, unclassifiable rate, and false acceptance rate. It was found that large decision thresholds result in low recognition accuracy, and vice versa for small thresholds. The thresholds also produce bounded and locally conserved decision regions. This effectively minimizes the chance of falsely accepting unknown patterns. Hence, PDBNNs are suitable for classification applications where the minimization of false acceptance rate is an important issue.

Chapter 6

APPLICATION TO SPEAKER VERIFICATION

PDBNNs were used to implement a hierarchical face recognition system in [28] and have achieved excellent performance (97.75% recognition, 2.25% false rejection, 0% misclassification and 0% false acceptance). The characteristics of PDBNNs' decision boundaries have been investigated in our previous study [80] and in Chapter 5. We have also demonstrated in [80] and [79] that the thresholding mechanism of PDBNNs is very effective in detecting data not belonging to any known classes. In light of this finding, this chapter applies PDBNNs to speaker verification and attempts to improve the robustness of a speaker verification system against intruder attacks by using the PDBNNs' thresholding mechanism.

The chapter is organized as follows. Sections 6.1 and 6.2 outline the speakers models and speech corpus. The enrollment and verification procedures are explained in Sections 6.3 and 6.5. A threshold determination method is introduced in Section 6.4. This is followed by a performance comparison in Section 6.6 and a concluding remark in Section 6.7.

6.1 Speaker Models

In this chapter, each speaker is modeled by a PDBNN consisting of elliptical basis functions with full covariance matrices.[1] Each speaker model is trained to maximise a log-likelihood function using the utterances of the corresponding speaker. This is followed by discriminative training where the model is fine-tuned and a decision threshold is estimated in order to differentiate the voice of the speaker from that of 45 randomly selected speakers (denoted as anti-speakers). Unlike the speaker model proposed in [20] where a set of ratio speakers must be determined during verification, the discriminative training procedure of PDBNNs embeds the characteristics of the background speakers in the speaker model during enrollment. Hence, a PDBNN does not require a set of cohort or background speakers during verification, resulting in a substantial saving in computational resources.

Experimental evaluations based on 138 speakers of the YOHO corpus show that the equal error rate obtained by the PDBNNs is about half of that of Higgins et al. [20] (0.89% vs. 1.8%). This suggests that the discriminative training of PDBNNs is able to improve the robustness of the speaker models.

Experiments using PDBNNs, vector quantization (VQ) [29] and Gaussian mixture models (GMMs) as speaker models have been carried out. Each PDBNN consists of eight elliptical basis functions with full covariance matrices. The K-means algorithm was used to determine the initial positions of the function centers. Then, the co-

[1] It was found in a related study that elliptical basis functions with full covariance matrices achieve a higher recognition performance as compared to those with diagonal covariance matrices, provided that sufficient training data is available [39].

variance matrices were initialized by the K-nearest neighbors algorithm ($K = 1$). In other words, all off-diagonal elements were zero and the diagonal elements (being equal) of each matrix were set to the average Euclidean distance between the corresponding center and its K nearest centers. The EM algorithm was subsequently used to estimate the mean vectors, covariance matrices, and prior probabilities (or mixture coefficients). In the globally supervised (GS) training phase of the PDBNNs, reinforced and anti-reinforced learning were applied to fine-tune the decision boundaries and the decision thresholds.

In the original PDBNNs, the mean vectors and covariance matrices in Equation (4.9) are updated in batch mode (see Equation (4.9)). However, in this work, sequential mode has been used, as follows:

$$\begin{aligned} \mu_{r|i}^{(j+1)} &= \mu_{r|i}^{(j)} + q(t)\eta_\mu h_{r|i}^{(j)} \Sigma_{r|i}^{-1(j)} \left(x(t) - \mu_{r|i}^{(j)}\right) \\ \Sigma_{r|i}^{(j+1)} &= \Sigma_{r|i}^{(j)} + \frac{1}{2} q(t)\eta_\sigma h_{r|i}^{(j)} \left(H_{r|i}^{(j)}(t) - \Sigma_{r|i}^{-1(j)}\right), \end{aligned} \tag{6.1}$$

where $q(t) = 1$ when $x(t) \in \mathcal{D}_2^i$ and $q(t) = -1$ when $x(t) \in \mathcal{D}_3^i$. We have conducted a pilot investigation and found that the sequential mode achieves faster convergence as compared to the batch mode. The potential advantage of the sequential mode is that its convergence does not depend heavily on the amount of duplicated information in the training set. For example, if we replicate the original training set to double the number of training patterns, the batch mode will roughly take two times longer to converge, although the amount of information contained in the training data remains unchanged. The sequential mode, on the contrary, updates the weights after each

pattern presentation. Therefore, duplicating and concatenating the training data will not affect the overall convergence of the sequential mode. Another potential advantage of the sequential mode is that in "on-line" training, the new pattern upon arrival is used to compute the weight adjustment and can be discarded immediately. This can potentially save a lot of local storage. In addition, the order of presenting the training patterns can be randomized for each epoch in the sequential mode. The randomization makes the search in weight space stochastic, hence reducing the possibility of being trapped in the local minima of the error surface.

For the Gaussian mixture speaker models, each GMM consists of eight Gaussian functions with full covariance matrices. The Gaussian functions are divided into two classes. Four Gaussian functions represent the speaker class and the remaining four represent the background speaker (or anti-speaker) class. Each class was trained independently using a method identical to the locally unsupervised training of the PDBNNs.

For the VQ speaker models, speaker-specific codebooks were generated by clustering the training patterns using the classical LBG algorithm [29]. Codebook size of 8, 16, 32, 64, and 128 were used.

Speech signals are almost always represented by a sequence of temporally related, short-time feature vectors. To capture the temporal relation embedded in a vector sequence, GMMs, PDBNNs and VQ require dynamic features such as delta cepstra or delta-delta cepstra. This is because these models are originally designed for capturing static features, and they consider the feature vectors to be independent of each

other. Therefore, whenever static features are used, the order of presentation becomes unimportant. In other words, training can be performed by randomly presenting the speech patterns to the model as long as the speech patterns are derived from the utterances of the same speaker. Thus, the data independency assumption of these models, as mentioned in Section 4.1, can be satisfied.

6.2 Speech Corpus

In this work, all of the 138 speakers (108 male, 30 female) in the YOHO corpus have been used for experimental evaluations. For each speaker in the corpus, there are four enrollment sessions. Each of these sessions contains 24 utterances. Likewise, there are ten verification sessions for each speaker, each of them containing four utterances. Each utterance is composed of three 2-digit numbers (e.g., 34-52-67). All sessions were recorded in an office environment using a high quality telephone handset and sampled at 8kHz, 16 bits per sample.

6.3 Enrollment

The enrollment process for constructing the PDBNN speaker models involves two steps: locally unsupervised (LU) training and globally supervised (GS) training. In the LU training phase, all utterances from the enrollment sessions of each speaker in the YOHO corpus [22] were used to train his/her speaker model. Then, in the GS training phase, the speaker's enrollment utterances and the utterances from all enrollment sessions of 45 randomly selected anti-speakers (none of them was the

speaker) were used to fine-tune the speaker model and to determine a speaker-specific decision threshold (see Section 6.4 below). Therefore, the PDBNN is to differentiate the feature vectors derived from two classes: the speaker class and the anti-speaker class.

Instead of using all the feature vectors derived from the anti-speakers for GS training, a subset of feature vectors was used. The subset was obtained by sampling the anti-speakers' feature vectors. This idea is based on our previous investigation into threshold determination where a large set of pseudo-impostor is sampled during enrollment to estimate the decision thresholds [81]. This sampling strategy was found to be able to provide sufficient feature vectors to model the impostor population without posing an excessive computational burden on the enrollment process.

For the Gaussian mixture speaker models, all utterances from the enrollment sessions of each speaker in the YOHO corpus were used to train his/her four Gaussian functions (speaker class). The utterances from all enrollment sessions of 45 randomly selected anti-speakers (same anti-speakers set as in the case of PDBNNs) were used to train the other four Gaussian functions (background class).

For the VQ speaker models, all utterances from the enrollment sessions of each speaker in the corpus were used to train his or her codebook, and no discriminative training and anti-speakers' feature vectors were required.

6.4 *Threshold Determination*

The procedures for determining the decision thresholds of PDBNNs, VQ speaker models and GMMs are different. For the PDBNNs, the thresholds were trained during the GS training phase as follows. The enrollment utterances of the speakers and the anti-speakers were applied to the speaker models, and the false acceptance rate (FAR) and false rejection rate (FRR) were recorded after every epoch. The error rates were determined by the ratio of the number of incorrect classifications to the total number of classifications.[2] The threshold was initialized to some arbitrary high value (which depends on the maximum score of the training vectors) so that the initial FRR was greater than the initial FAR. The GS training increases FAR and decreases FRR gradually. Once the FAR was greater than the FRR, training was terminated and the resulting network was used as a speaker model.

The above strategy will theoretically produce speaker models with equal chance of rejecting true speakers or accepting impostors, i.e., FAR being equal to FRR. In the case where either of these errors is more important than the other, the termination criterion can be adjusted accordingly. The equal error rate (FAR and FRR being identical) provides a reasonable performance indicator for speaker verification systems. This is because without prior knowledge about the consequence of false acceptance and false rejection, striking a balance between these two error rates seems to be a good choice.

The original PDBNN termination criterion is implicitly based on classification

[2]See Section 6.5 for the procedure of determining FARs and FRRs.

accuracy. However, this approach can lead to undesirably low threshold values. This is because the system will blindly accept impostor patterns and cause a high FAR. On the other hand, if the termination criterion is based on the minimization of FAR only, it is possible to achieve a highly secure system but may cause inconvenience to the true speakers. Hence, the above termination criterion strikes a balance between high FAR and high FRR and was used in our experiments.

In practice, it is difficult to find a threshold such that FAR and FRR are exactly the same. During GS training, the decision boundaries vary with the changing model parameters. As a result, the FAR and FRR curves (variations of FAR and FRR against decision threshold) are also varying. This will introduce a corresponding change in the equal error threshold. Hence, FAR and FRR can vary substantially from epoch to epoch. Fig. 6.1 shows typical variations of FAR, FRR and threshold against the number of epochs. Evidently, the FAR increases and FRR decreases substantially from epoch 35 to epoch 41. As a result, we are not able to find a threshold such that FAR and FRR are exactly identical. As we terminate the GS training once the FAR is greater than the FRR, the threshold obtained will bias towards favoring a lower FRR. For example, the GS training in Fig. 6.1 was terminated at epoch 39, resulting in a lower FRR.

Equal error thresholds can be obtained if we evaluate the FAR and FRR after every pattern presentation in the sequential mode of GS training in Equation (6.1).[3] However, this will introduce excessive computation overhead during training. This is

[3]This has the effect of smoothing the FAR and FRR curves.

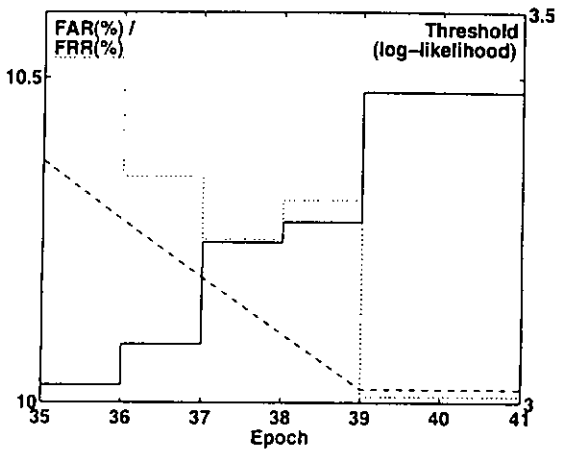


Figure 6.1: The variations of FAR (solid), FRR (dotted) and threshold (dashed) during enrollment.

because after each pattern presentation, we need to feed all the training patterns into the network in order to compute the instantaneous FAR and FRR. This process is computationally intensive when the number of training patterns is large.

The learning rates for the mean vectors $\mu_{r|i}$ and covariance matrices $\Sigma_{r|i}$ in Equation (6.1) were set to 1×10^{-7} and 1×10^{-10}, respectively. They are identical for both reinforced and anti-reinforced learning. As these parameters interact with each other nonlinearly, their values were found empirically. In the original PDBNNs, the learning rates for optimizing the thresholds are identical for both reinforced and anti-reinforced learning; in this work, however, the learning rate η_t in Equation (4.11) for reinforced learning was set to 0.0001 while that for anti-reinforced learning was set to zero. As a result, the thresholds were optimized by reinforced learning only. This arrangement was found to achieve faster convergence as compared to the case where the reinforced and anti-reinforced learning have identical learning rate. This is illustrated

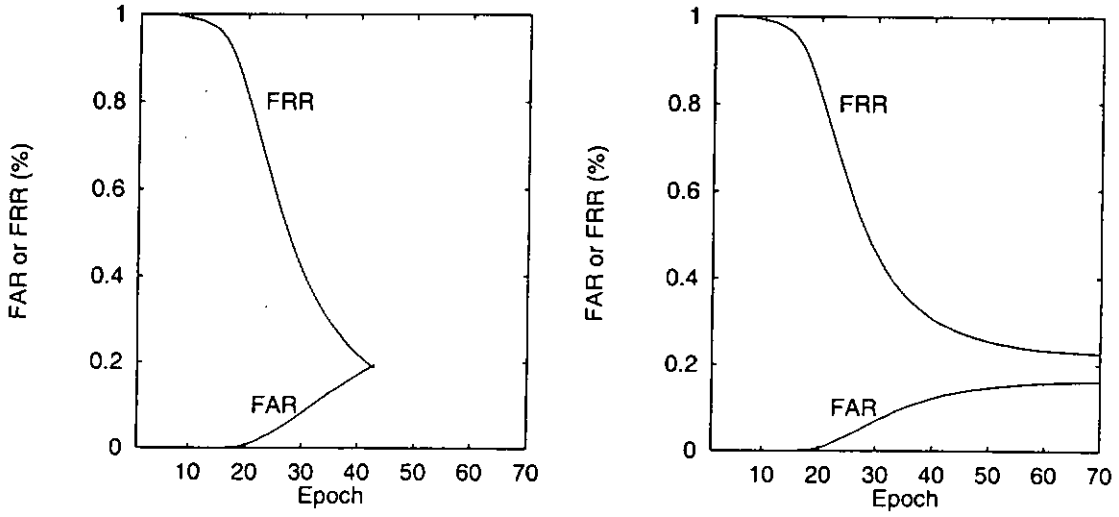


Figure 6.2: The variations of FAR and FRR (during enrollment) based on (a) reinforced learning only and (b) both reinforced and anti-reinforced learning.

in Fig. 6.2 where the variations of FAR and FRR against the number of epochs are illustrated for (a) reinforced learning only and (b) both reinforced and anti-reinforced learning. Evidently, learning in Figure 6.2 (a) was terminated after epoch 43, while that in Figure 6.2(b) took a lot more epoches. It was also found that when both reinforced and anti-reinforced learning were used, the training process might oscillate around a fixed error rate and therefore never met the termination criterion.

For the Gaussian mixture speaker models, the utterances from all enrollment sessions of 45 randomly selected anti-speakers were used for threshold determination. Sampling was not necessary because the thresholds for the GMMs were determined heuristically, which is much faster than the globally supervised training of PDBNNs. The threshold for each model was adjusted until the FAR and FRR obtained during

enrollment were equal. The resulting threshold is referred to as equal error threshold.

For the VQ speaker models, the utterances from all enrollment sessions of 45 randomly selected anti-speakers were used for threshold determination. Again, sampling was not necessary because classification using VQ (when the background speaker or cohort models were not used) is much faster than PDBNNs. The threshold was adjusted until the FAR obtained during enrollment falls below a pre-defined level. In this work, we set this level to 5%. The reason behind this approach is that the FAR curves were derived from a large number of pseudo-impostors; hence, they are more reliable and a better resolution can be obtained. In addition, the FAR curves can be used to predict the robustness of the verification system against impostor attacks [81].

6.5 Verification Procedures

During verification, the feature vectors derived from the utterances of a claimant were concatenated to form a test sequence $\mathcal{T} = \{\vec{x}_t, t = 1, \ldots, T\}$ where T is the total number of patterns in $\mathcal{T}$. The sequence was then divided into a number of overlapping segments, with each segment containing T_s ($T_s = 300$) consecutive vectors. Each of these segments was treated as an independent verification trial. Note that this approach is similar to that of [63], [39], where each segment is considered to be independent. The advantage of this approach is that the number of independent trials (both genuine and impostor) can be increased by reducing the segment size.

For the t-th segment, $\mathcal{T}_t$, the average log-likelihood (also called the score)

$$z_t = \frac{1}{T_s} \sum_{\vec{x} \in \mathcal{T}_t} \phi(\vec{x}, \mathbf{w}) \tag{6.2}$$

was computed, where $\phi(\vec{x}, \mathbf{w})$ is the log-likelihood function in Equation (4.5). Verification decisions were based on the criterion:

$$\text{if } z_t \begin{cases} > \zeta & \text{accept the claimant} \\ \leq \zeta & \text{reject the claimant} \end{cases} \tag{6.3}$$

where ζ is a speaker-specific *a priori* threshold controlling the false rejection and false acceptance rates (see Section 6.4 above). A verification decision was made for each segment, with the error rate (either FAR or FRR) being the proportion of incorrect verification decisions to the total number of decisions. More specifically, after every verification decision, a window covering 300 consecutive cepstral vectors was shifted by three vector positions in the sequence and the verification procedure was repeated. The above steps were repeated for the test utterances of each speaker and of all impostors in the population. The verification performance was then computed as the average of the error rates over all speakers in the population.

For the Gaussian mixture speaker models, the verification decision for a given segment of claimant's feature vectors $\vec{x} \in \mathcal{T}_t$ was based on the difference between the log-likelihood output of the speaker model and that of the background speaker model. More specifically, decisions were based on the criterion:

$$\text{if } d_t \begin{cases} > \xi & \text{accept the claimant} \\ \leq \xi & \text{reject the claimant} \end{cases} \tag{6.4}$$

where ξ is a decision threshold and d_t is a normalized score, i.e.,

$$d_t = \frac{1}{T_s} \sum_{\vec{x} \in \mathcal{T}_t} \left\{ \log p_S(\vec{x}|\Lambda_S) - \log p_B(\vec{x}|\Lambda_B) \right\}, \tag{6.5}$$

where $p_S(\vec{x}|\Lambda_S)$ and $p_B(\vec{x}|\Lambda_B)$ are respectively the probability density functions (GMM's outputs) corresponding to the speaker class and background speaker class, and Λ_S and Λ_B are the corresponding GMMs.

For the VQ speaker models, each of the verification decisions was based on the average Euclidean distance between a given segment of claimant's feature vectors $\vec{x}$ and his/her VQ codebook $\{\vec{c}_k\}_{k=1}^{K}$. More specifically, decisions were based on the criterion:

$$\text{if } d_t \begin{cases} < \xi & \text{accept the claimant} \\ \geq \xi & \text{reject the claimant} \end{cases} \tag{6.6}$$

where $d_t = \frac{1}{T_s} \sum_{\vec{x} \in \mathcal{T}_t} \min_{k=1}^{K} \|\vec{x} - \vec{c}_k\|$ is the score, ξ is a decision threshold, and K is the number of code vectors in the codebook.

Although likelihood ratio and cohort scoring normalization have been shown to improve speaker verification performance. Classification using VQ might be slow when the background speakers or the cohort models were used. Computation burden becomes more serious when the number of speaker models increases. This is because the cohort model's score should be obtained from the mean score of a set of utterance-and-speaker-dependent reference models, which must be determined during verification. As a result, for each utterance, the scores of all speaker models have to be computed in order to find the reference models. This process is computationally intensive.

Verification was performed using each speaker in the YOHO corpus as a claimant, with 45 impostors being randomly selected from the remaining speakers (excluding the anti-speakers and the claimant) and rotating through all the speakers. The claimant's and impostors' utterances from the corresponding verification sessions in the YOHO corpus were concatenated to form two different test sequences: a claimant sequence and an impostor sequence. Verification decisions were made according to Equations (6.3), (6.4), and (6.6) with the segment length T_s in Equations (6.2) and (6.5) being set to 300. After every genuine trial (using the claimant sequence), a window covering 300 vectors was shifted forward along the vector sequence by three vector positions. A similar procedure was used in the impostor trials.

LP-derived cepstral coefficients were used as acoustic features. For each utterance, the silent regions were removed by a silent detection algorithm based on the energy and zero crossing rate of the signal. The remaining signals were pre-emphasized by a filter with transfer function $1-0.95z^{-1}$. Twelfth-order LP-derived cepstral coefficients were computed using a 28 ms Hamming window at a frame rate of 14 ms.

6.6 Results and Discussions

Table 6.1 summarizes the average FAR, FRR, and EER obtained by the PDBNN, VQ and Gaussian mixture speaker models. The EERs were obtained by adjusting the thresholds during verification to equalize the FAR and FRR. All results are based on the average of 138 speakers in the YOHO corpus. The results, in particular the EER, demonstrate the superiority of the PDBNNs over the VQ speaker models. For

example, Table 6.1 shows that the smallest EER obtained by the VQ models is 1.04%, which is greater than 0.89%, the EER of PDBNNs. Bear in mind that the number of parameters in the VQ models with 128 centers is larger than that in PDBNNs with 8 centers (1537 vs. 729), the results clearly suggest that the PDBNN training procedure is capable of producing better speaker models.

Speaker Model		Enrollment		Verification		
		FAR %	FRR %	FAR %	FRR %	EER %
PDBNNs	8 centers	0.98	0.01	0.70	5.72	**0.89**
GMMs	8 centers	1.32	0.00	1.12	2.70	**0.60**
VQs	8 centers	5.86	1.69	4.65	9.80	**4.04**
	16 centers	5.69	0.17	4.45	4.61	**2.50**
	32 centers	5.43	0.02	4.31	2.74	**1.72**
	64 centers	5.84	0.01	4.60	2.05	**1.42**
	128 centers	5.30	0.01	4.15	1.56	**1.04**

Table 6.1: Average error rates obtained by different speaker models. The pre-defined FAR for VQ was set to 5%.

Table 6.1 also shows that the smallest equal error rate (EER) obtained by the PDBNNs is about half of that of Higgins et al. [20] (0.89% vs. 1.80%) where background speaker models were used during verification. This suggests that the idea of embedding the background speakers' characteristics in the speaker models during enrollment is promising.

Table 6.1 also indicates that the FAR obtained by the PDBNNs during enrollment is much greater than the FRR. Recall from Section 6.4 that we need to reduce the computation overhead by introducing a termination criterion that biases the threshold towards lower FRRs (see also Fig. 6.1). In addition, one should bear in mind that the

interpretations of FAR and FRR in PDBNN training are different from that in speaker verification. In PDBNN training, the FAR and FRR are evaluated in a pattern-by-pattern basis, whereas the FAR and FRR are evaluated in a segment-by-segment basis during verification. Hence, an equal error termination criterion does not necessarily lead to equal error during verification.

It is also interesting to note that the FARs corresponding to enrollment and verification differ by a small amount, whereas the FRRs in verification are much greater than those in enrollment. This is because the FAR curves (variation of FAR with respect to the threshold) were obtained by presenting the utterances of 45 impostors to the speaker models, whereas the FRR curves (variation of FRR with respect to the threshold) were obtained by using the claimant's utterances only. As a result, the number of feature vectors for deriving the FAR curves is considerably larger than that for deriving the FRR curves. This makes the FAR curves more predictable, as shown in Fig. 6.3 where the FAR and FRR curves corresponding to the GMMs, VQ models and the PDBNNs are plotted. Fig. 6.3 shows that the FAR curves corresponding to enrollment and verification almost overlap one another. On the other hand, the FRR curves corresponding to enrollment and verification exhibit a large displacement, suggesting that reliable thresholds are difficult to estimate solely from the FRR curves, due to insufficient true-speaker utterances. While it may be difficult to obtain a large training set that completely characterizes the true speaker (because it requires a lot more enrollment sessions), it is relatively easy to acquire a large amount of anti- or

background speaker data to train the background model.[4] As a result, the background model (or the embedded anti-speaker model in the case of PDBNNs) can be a good representation of the impostors. This explains the high predictability of the FAR curves, as shown in Fig. 6.3.

Fig. 6.4 plots the FRRs against the FARs of 138 speakers. Evidently, most of the PDBNNs have a very low FAR and a relatively high FRR. This confirms the result in Table 6.1. On the other hand, most VQ models have an FAR around 5-10% because the *a priori* thresholds were obtained by setting the pre-defined FAR to 5%.

Fig. 6.5 plots the *a posteriori* equal error thresholds against the *a priori* thresholds found by the PDBNN, VQ and Gaussian mixture speaker models. The equal error thresholds were found by adjusting the thresholds until FAR is equal to FRR. Fig. 6.5(a) shows that the *a priori* thresholds of most VQ speaker models are greater than the *a posteriori* equal error thresholds. While this feature makes the verification system friendly to users, it may make the system vulnerable to impostor attacks, causing a high FAR for some speakers as shown in Fig. 6.4(a). The PDBNN speaker models, on the other hand, provide better security, as shown in Fig. 6.4(b), since on average the FAR is only one-eighth of the FRR.

Table 6.1 also shows that the EER obtained by the GMMs is 0.60%, which is lower than the EER achieved by the PDBNNs. This is because the globally supervised training of PDBNNs alters the maximum likelihood solution. During the globally supervised training of PDBNNs, the decision threshold as well as the speaker models

[4]This can be achieved by pooling the utterances of other speakers in the population, without increasing the number of enrollment sessions.

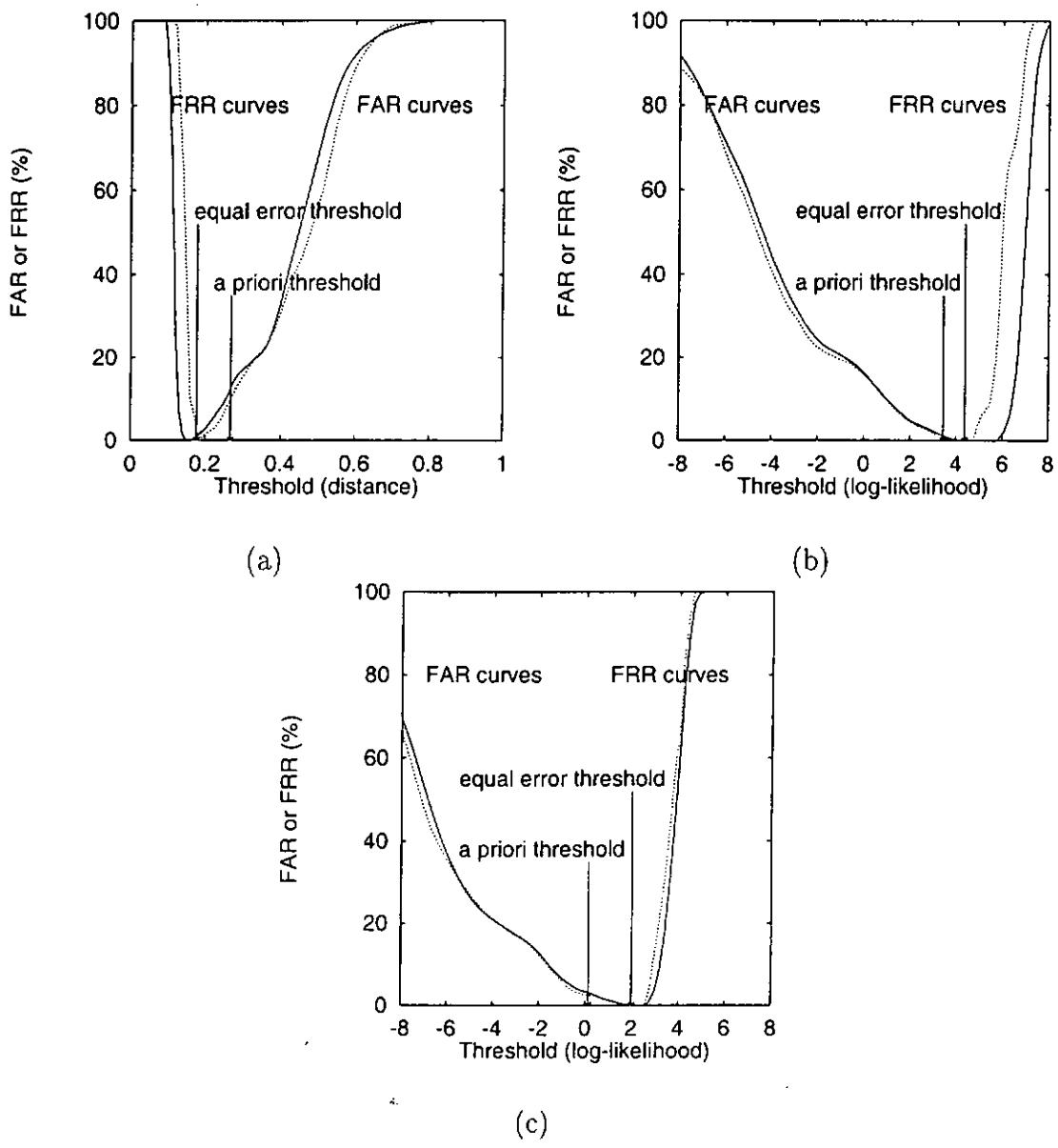


Figure 6.3: FAR and FRR versus threshold. The curves are based on the enrollment (solid) and verification (dots) sessions of speaker 192 using (a) a VQ speaker model with 128 code vectors and (b) a PDBNN with 8 centers, and (c) a GMM with 8 centers. The arrows indicate the values of the *a priori* thresholds and equal error thresholds.

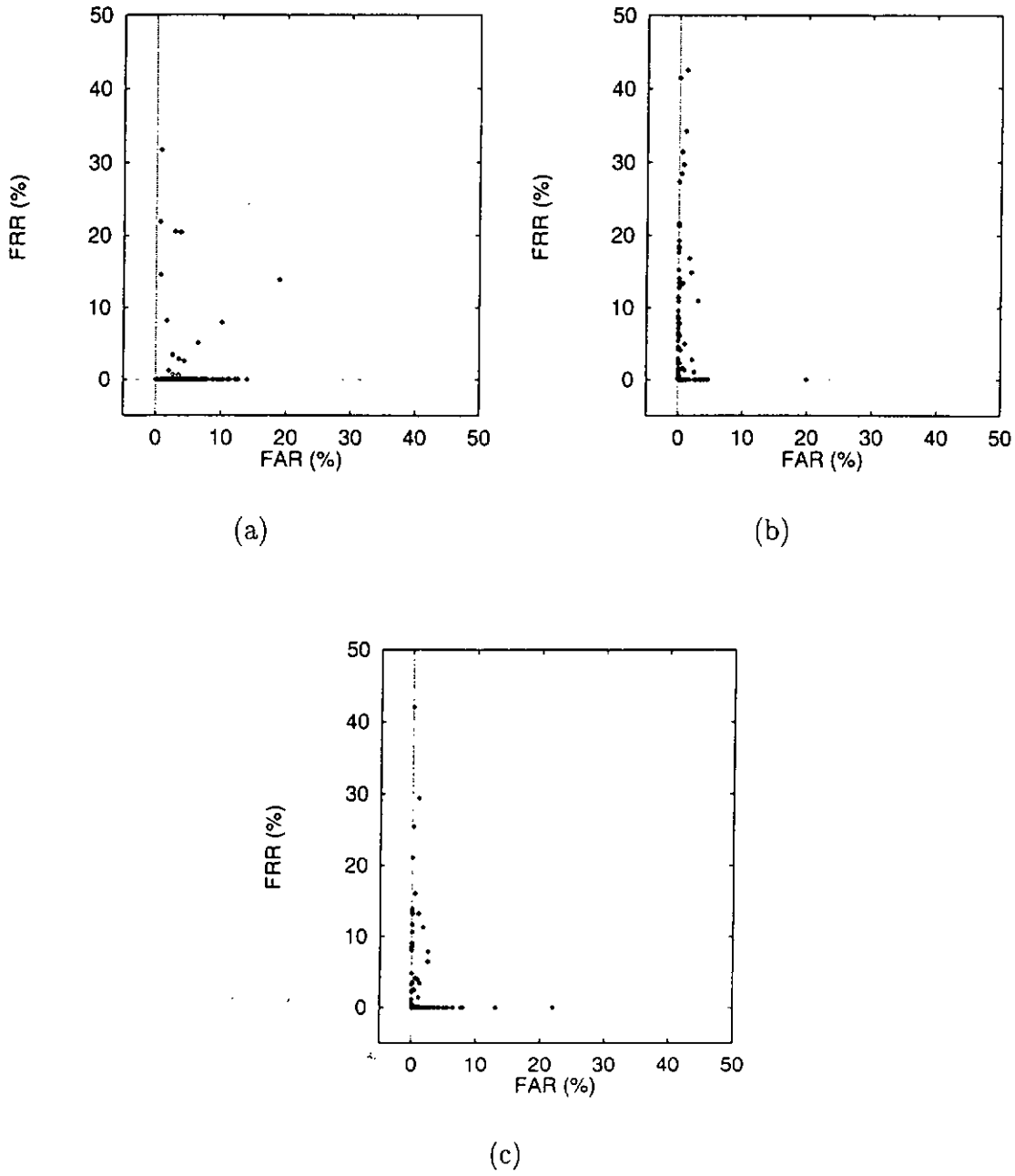


Figure 6.4: FRRs versus FARs (during verification) of 138 speakers based on (a) VQ speaker models with 128 code vectors, (b) PDBNNs with 8 centers and (c) GMMs with 8 centers.

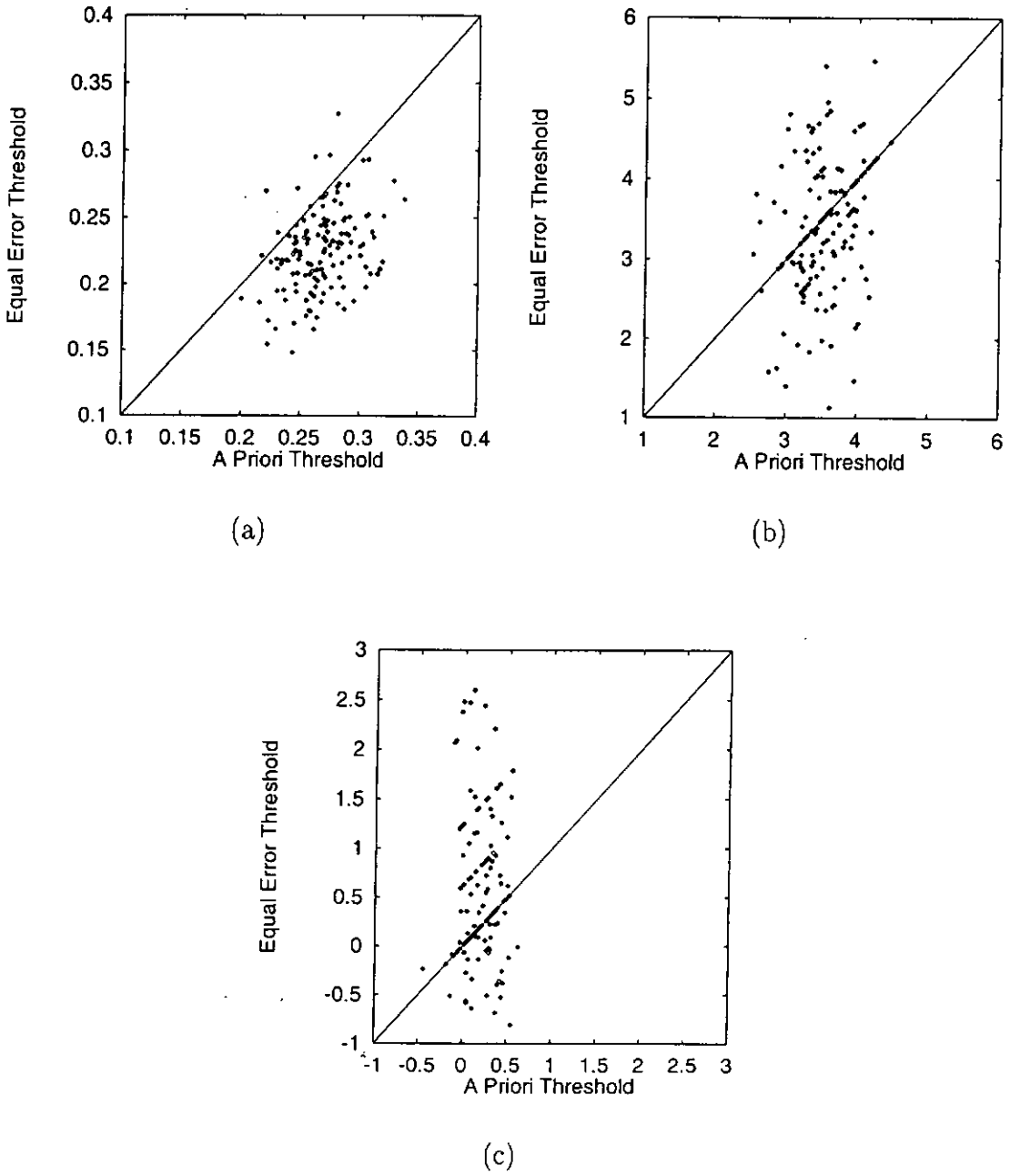


Figure 6.5: *A posteriori* equal error thresholds versus *a priori* thresholds found by (a) VQ speaker models with 128 code vectors, (b) PDBNNs with 8 centers and (c) GMMs with 8 centers.

are altered for each pattern presentation; whereas there is no such adjustment in the Gaussian mixture speaker models. Another reason is that each Gaussian mixture speaker model consists of two parts, with one part modeling the speaker class and the other part modeling the background speaker class. Based on this notion, Equation (6.4) can be rewritten as

$$\text{if } d_{t,S} \begin{cases} > \xi + d_{t,B} & \text{accept the claimant} \\ \leq \xi + d_{t,B} & \text{reject the claimant} \end{cases} \tag{6.7}$$

where $d_{t,S} = \frac{1}{T_s} \sum_{\vec{x} \in \mathcal{T}_t} \{\log p_S(\vec{x}|\Lambda_S)\}$ and $d_{t,B} = \frac{1}{T_s} \sum_{\vec{x} \in \mathcal{T}_t} \{\log p_B(\vec{x}|\Lambda_B)\}$. By comparing Equations (6.7) with (6.4), the background speaker class in the GMMs acts as a varying offset to the decision threshold. Therefore, the effective thresholds ($\xi + d_{t,B}$) of the Gaussian mixture speaker models are dependent on how close the unknown utterance is to the background speakers. The effective threshold will become very large if the unknown utterance is very close to the background speakers, resulting in a higher chance of rejecting the claimant. On the other hands, the background speaker's characteristics in PDBNNs are embedded in the speaker models. Recall from Section 6.4 that the PDBNNs are trained to differentiate the true speakers from the background speakers. No Gaussian functions are reserved for modeling the background speakers, only the true speaker's Gaussian functions are adjusted to reflect the background speaker characteristics. As a result, there is no varying offset to the thresholds in the case of PDBNNs. These constant thresholds make the PDBNNs less robust when the claimant's patterns are not sufficiently close to the speaker patterns in the feature space, thus causing a high FRR during verification as shown in Table

6.1.

Note that the above comparison is based on EERs. Bear in mind that EERs are independent of the a priori thresholds (thresholds determined during enrollment); rather they are dependent on the speaker models and the test data. This means that the EERs indicate the potential capability of the speaker models, and this capability will become achievable once the appropriate thresholds have been found. Therefore, comparing the achievable EERs of PDBNNs and GMMs will give us some insights into their recognition capability.

In order to measure the computational complexity, we have re-run the speaker verification simulations based on a subset of YOHO (i.e., 10 speakers in the population). The computation time during enrollment and verification for VQ, GMMs and PDBNNs are measured, and the results are shown in Table 6.2. The results show that PDBNNs take the longest time for enrollment and verification. GMMs take less time as compared to PDBNNs in both enrollment and verification, but they are slower than VQ. VQ takes the shortest training and verification time. The main reason is that VQ is based on Euclidean distance whereas the other two models are based on Mahalanobis distance.

Speaker Model		Enrollment (sec.)	Verification (sec.)
PDBNNs	8 centers	26864.70	1763.37
GMMs	8 centers	7308.42	1028.76
VQ	8 centers	1788.55	821.78
	128 centers	3542.92	942.68

Table 6.2: The computation time (in seconds) for enrollment and verification.

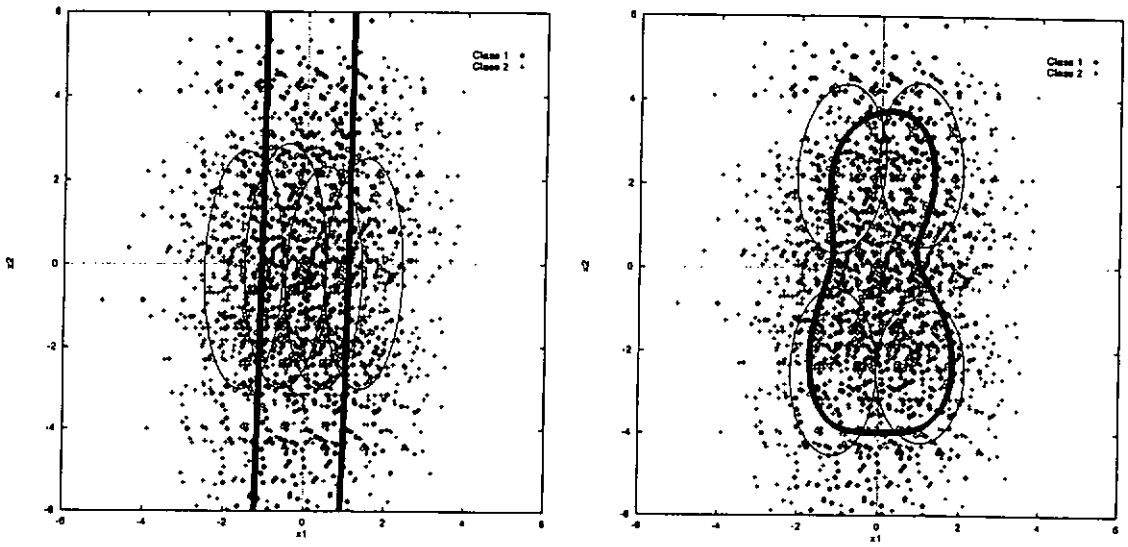
Class 1		Class 2	
mean vector	cov. matrix	mean vector	cov. matrix
$\begin{pmatrix} -0.5 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 1.0 & 0 \\ 0 & 1.0 \end{pmatrix}$	$\begin{pmatrix} -1.5 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 1.0 & 0 \\ 0 & 1.0 \end{pmatrix}$
$\begin{pmatrix} 0.5 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 1.0 & 0 \\ 0 & 1.0 \end{pmatrix}$	$\begin{pmatrix} 1.5 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 1.0 & 0 \\ 0 & 1.0 \end{pmatrix}$

Table 6.3: Mean vectors and covariance matrices of the four Gaussian clusters in the two-class problem. Each cluster contains 500 samples.

	PDBNN		GMM	
	Train	Test	Train	Test
EER(%)	36.70	40.30	32.30	34.90

Table 6.4: Performance of the PDBNN and GMM in the two-class problem.

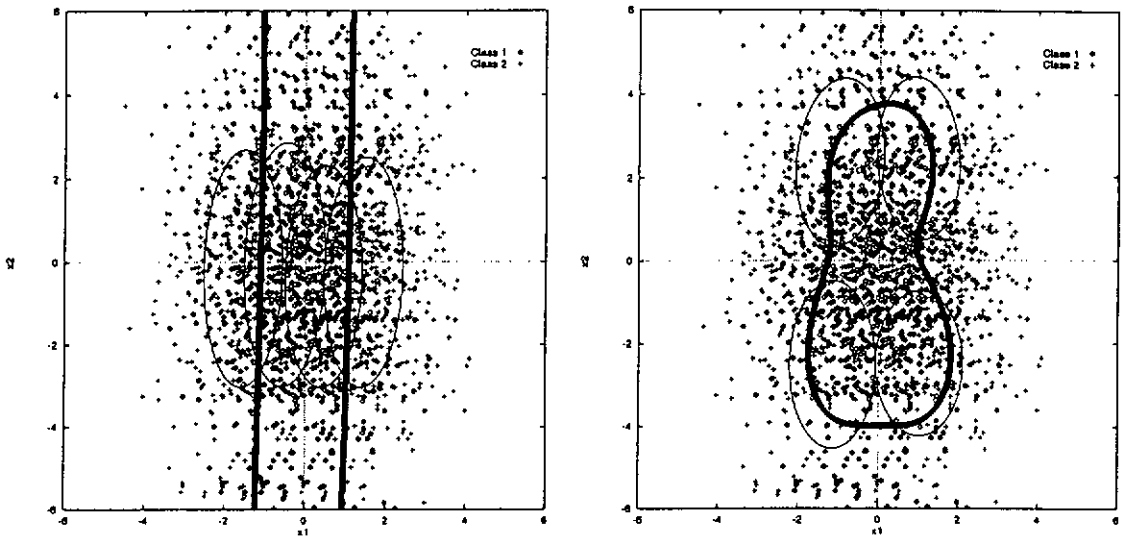
To illustrate the difference between the PDBNN and GMM speaker models, a two-class hypothetical problem is investigated. The problem utilizes 2000 training patterns generated from four Gaussian distributions with mean vectors and covariance matrices shown in Table 6.3. Class 1 and Class 2 are to simulate the speaker and impostor classes, respectively. A PDBNN and a GMM with 4 function centers were trained to classify the patterns into two classes—similar to the enrollment procedure in the speaker verification experiments. For the GMM, two Gaussian functions were trained to recognized the patterns from Class 1, while the other two were trained to recognize the patterns from Class 2. For the PDBNN, all of the four Gaussian functions were trained to recognize the Class 1 patterns, while the Class 2 patterns were used in the globally supervised training. In this experiment, the training methods, learning rate and verification methods used were identical to the speaker verification experiments described previously.



(a) (b)

Figure 6.6: Decision boundaries, function centers, and contours of constant basis function outputs (thin ellipses) produced by (a) a Gaussian mixture model and (b) a PDBNN based on the training data set.

Table 6.4 compares the performance of the PDBNN and GMM. The figures were based on 2000 test vectors generated by four Gaussian distributions with means and variances identical to those of the training data. The EERs were obtained by adjusting the thresholds during verification (*a posteriori*) to equalize the FAR (of Class 2) and FRR (of Class 1). Figures 6.6 and 6.7 show the patterns, decision boundaries, function centers, and contours of constant basis function outputs formed by the PDBNN and the GMM during training and testing. The decision boundaries are based on the equal error thresholds obtained from the corresponding data set. Therefore, they indicate the position of equal chance of misclassifying Class 1 and Class 2, i.e., FAR being equal to FRR.



(a) (b)

Figure 6.7: Decision boundaries, function centers, and contours of constant basis function outputs (thin ellipses) produced by (a) a Gaussian mixture model and (b) a PDBNN based on the testing data set.

As shown in Figures 6.6 and 6.7, the PDBNN's decision boundary encloses a small region in the input space. This is because recognition in PDBNNs is based on the difference between the log-likelihood function $\phi(\vec{x};\omega)$ and the decision threshold. As $\phi(\vec{x};\omega)$ is the logarithm of a Gaussian mixture, its value will tend to negative infinity when $\vec{x}$ tends to infinity. Applying a constant threshold is equivalent to cutting the log-likelihood function along a plane parallel to the $x_1 - x_2$ axis in Figures 6.6 and 6.7. The value of the threshold determines the cutting point. Therefore, the decision boundaries encloses a smaller region when the decision threshold is large, or vice versa for small decision thresholds.

On the other hand, recognition in GMMs is based on the difference between the

probability density functions of two classes and a decision threshold, as indicated by Equation (6.7). When $\vec{x}$ tends to infinity, both density functions will output a value very close to zero. However, depending on the location of $\vec{x}$, one of the density function will produce a value slightly higher than the other. As long as the difference is larger than the decision threshold (which could be zero in this hypothetical problem), the unknown data point will be classified to Class 1, no matter how far it is away from the origin. Therefore, the GMM's decision boundary extends to infinity.

As indicated by the EER during testing, the PDBNN performs slightly poorer as compared to the GMM in this two-class problem (see Table 6.4). This agrees with our results in the speaker verification experiments (see Table 6.1). Although PDBNNs are inferior to GMMs in terms of EERs, they can be a very robust speaker model in terms of rejecting impostors. This argument is supported by the low FAR in the speaker verification experiments and by the decision boundaries in the hypothetical problem. For example, Table 6.1 shows that the FAR obtained by the PDBNNs is 0.70%, which is the lowest among all speaker models, and in the hypothetical problem, the threshold mechanism results in a locally conserved, enclosed decision boundary, which can help reject unknown data.

6.7 Summary

This chapter applies VQ, PDBNNs and GMMs to speaker verification and attempts to compare their performance based on FARs, FRRs, and ERRs. The YOHO corpus has been used for experimental evaluations. Each speaker is modeled by a PDBNN

consisting of eight elliptical basis functions with full covariance matrices. For the Gaussian mixture speaker models, each GMM consists of eight Gaussian functions with full covariance matrices. The Gaussian functions are divided into speaker class and background speaker class. For the VQ speaker models, speaker-specific codebooks were generated by clustering the training patterns using the classical LBG algorithm. Codebook sizes of 8, 16, 32, 64, and 128 were used.

The enrollment, verification and threshold determination procedures have been explained. Experiments using PDBNNs, vector quantization (VQ) and Gaussian mixture models (GMMs) as speaker models have been carried out. Experimental evaluations suggest that the smallest equal error rate (EER) obtained by the PDBNNs is about half of that of Higgins et al. [20] (0.89% vs. 1.80%). The EERs also demonstrate the superiority of the PDBNNs over the VQ speaker models.

On the other hand, in terms of EER, the PDBNNs perform slightly poorer as compared to the GMMs in the speaker verification experiments (0.89% vs. 0.60%). However, the PDBNNs are more robust in rejecting impostors. This is indicated by the low FAR that the PDBNNs have achieved in the speaker verification experiments and by their locally conserved decision boundaries in the hypothetical problem.

Chapter 7

SPEAKER VERIFICATION USING TELEPHONE SPEECH

Although speaker recognition based on clean speech has reached a high level of performance, severe performance degradation is still very common in practical, mismatched conditions. This is one of the major obstacles to the commercialization of speaker recognition technologies. One example of "mismatched conditions" is handset mismatch (or transducer mismatch). For automatic speaker recognition over the telephone, handset mismatch occurs when the recognizer is trained with speech recorded from a wide-band microphone and tested with speech recorded from a narrow-band handset.

Several successful compensation techniques, including cepstral mean subtraction [14], delta cepstrum [73], RASTA filtering [19] and signal bias removal [57], have been proposed to compensate the channel and handset mismatches. Although these channel compensation techniques have been throughoutly evaluated in the literature, they are typically evaluated separately using different experimental setting and speech corpora. Therefore, a throughout comparison among these techniques is necessary.

In this chapter, we describe the procedure of acquiring a telephone corpus from the

YOHO corpus and present an empirical study of the effects of handset variability on text-independent speaker recognition performance. Then, the procedure of measuring the frequency responses of telephone handsets is explained. Based on the frequency responses, channel cepstra are derived to normalize the telephone speech. Finally, evaluation using the telephone corpus is described and comparison with traditional channel normalization techniques is presented.

7.1 TYOHO Corpus

The YOHO corpus [22] was collected by ITT for government secure access applications. It features multiple speakers, inter-session variability, combination lock phrase syntax, high-quality telephone speech (3.8kHz/clean), and no telephone channel effect. These features make YOHO ideal for speaker verification research. The telephone YOHO (TYOHO) corpora that we constructed were produced by playing the clean YOHO corpus directly through different telephone handsets. These corpora are different in that their spectral characteristics were distorted by different handset transducers. The strategy used is similar to the generation of narrow-band TIMIT (NTIMIT) [23], cellular TIMIT (CTIMIT) [9] and handset TIMIT (HTIMIT) [62].

There are telephone speech corpora (e.g., LLHDB) for speaker verification research. They provide a good platform for evaluation and comparison of different compensation techniques. However, none of them provides enough information on the characteristics of the recording handsets. For example, the LLHDB corpus does not provide information about the frequency responses of the telephone handsets, and

the handsets used for recording are difficult to find or not available from the market nowadays. As the method proposed in this dissertation is based on the measurement of handset frequency responses, a new corpus was constructed.

The training and testing sessions of all speakers in the clean YOHO corpus were played back through a "mouth simulator" (B&K Type 4227) in an ordinary office with a sound pressure level of 50–60dB (linear weighting). The mouth simulator is designed to replicate the sound field generated by the human mouth, as specified in IEEE standard No. 269-1992 [21]. The handset and the mouth simulator were mounted on a telephone test head (B&K Type 4602) that allows accurate positioning of telephone handsets relative to the mouth simulator. This arrangement enables measurements in standardized speaking positions (LRGP, HATS, REF and AEN) to be performed. In this work, the mouth simulator was fitted to the telephone test head with the LRGP positioning jig.

Figures 7.1 and 7.2 show the experimental setup used to collect the TYOHO corpora. Two computers (master and slave) with identical sound cards were used during the recording process. Both sound cards sampled the speech at 8kHz, with 16 bits per sample. The master computer's sound card was connected to the mouth simulator via a telephone interface (B&K Type 5906/WH3028). The telephone handset was plugged directly into the telephone interface which can be configured to either Send or Receive mode and have five artificial line-sections for simulating telephone lines. To avoid any unwanted degradations caused by the telephone network, the telephone interface was configured to Send mode with no line being selected. Before

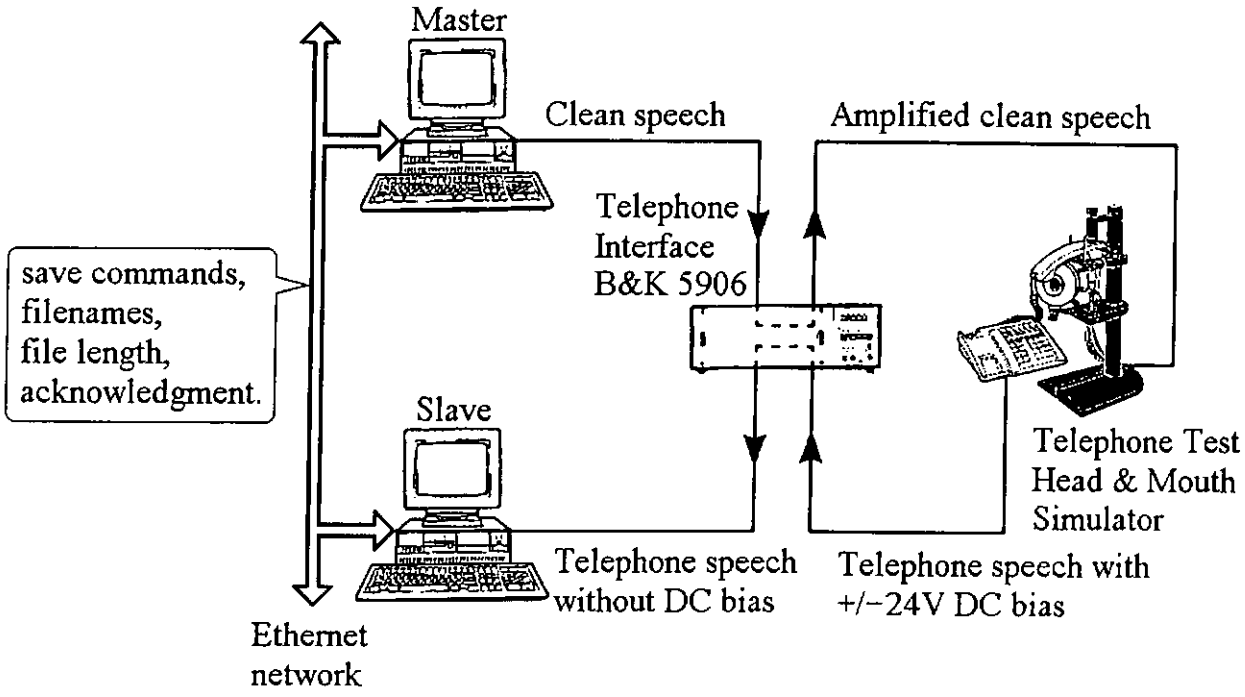


Figure 7.1: Experimental setup for collecting the TYOHO corpora.

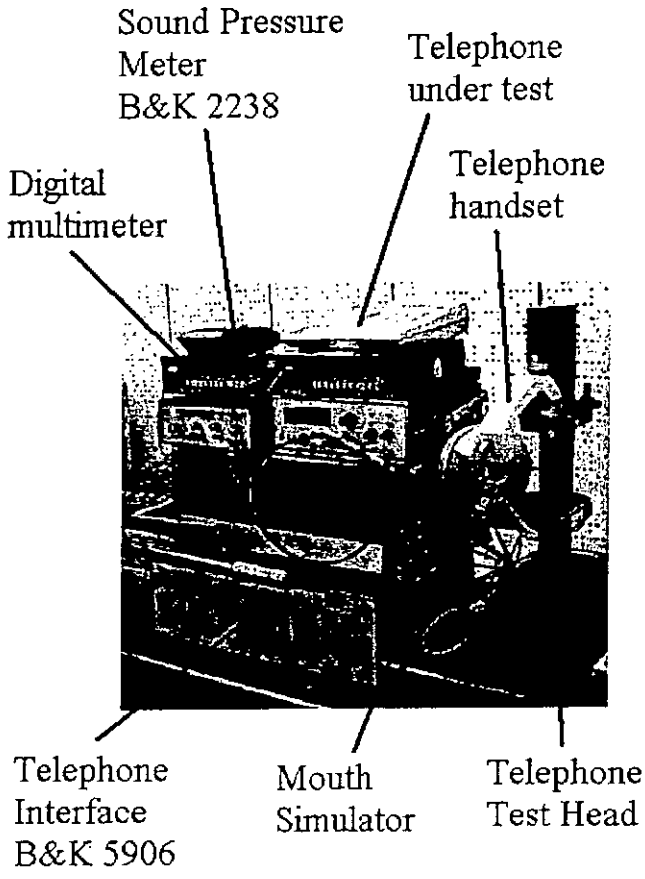


Figure 7.2: Experimental setup for collecting the TYOHO corpora.

Handset	Bland Name	Model Number
1	PHILIPS	unknown
2	CTI	unknown
3	ACL	ACL-503 Handsfree speakerphone

Table 7.1: Bland name and model number of handsets used in the corpora.

playing back a sound file, the master computer issued a save command and sent the filename of the sound file to the slave computer. The slave computer was initialized to wait for the save command. The telephone speech signal is digitized at 8kHz and stored on the hard disk of the slave computer. Once the sound file has been saved, the slave computer issues an acknowledgment packet to notify the master computer that it is ready for the next sound file. This master/slave arrangement ensures that all TYOHO utterances have the same duration as compared to the corresponding YOHO utterances. Zero crossing rate and instantaneous energy were used to create a phonetic transcriptions file labelling speech and non-speech segments. The resulting files are organized into a directory structure identical to that of the YOHO corpus.

Three telephone handsets have been used (see Table 7.1), resulting in three telephone YOHO corpora. The playback sound level of the master computer was fixed and the recording level of the slave computer was adjusted to provide a large dynamic range without peak clipping. Figure 7.3 shows the frequency spectra and the LP spectra of a voiced frame extracted from the YOHO and TYOHO corpora, and Figure 7.4 plots the smooth spectra on a single graph. Evidently, different handsets introduced different degree of distortion to the clean speech.

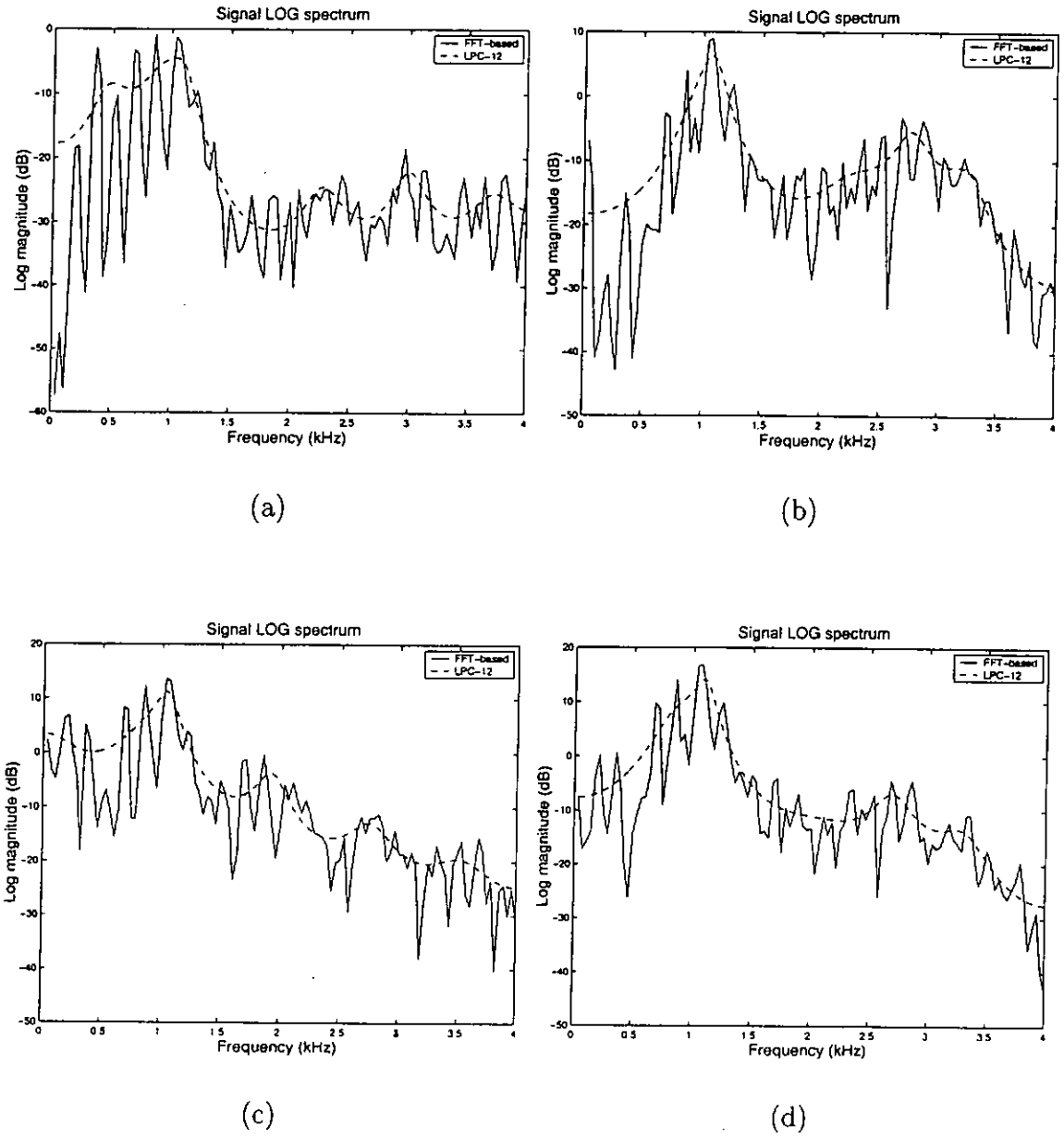


Figure 7.3: The FFT-based spectra and smooth spectra of a voiced frame extracted from (a) the clean YOHO corpus, (b) the T1 telephone corpus, (c) the T2 telephone corpus, and (d) the T3 telephone corpus.

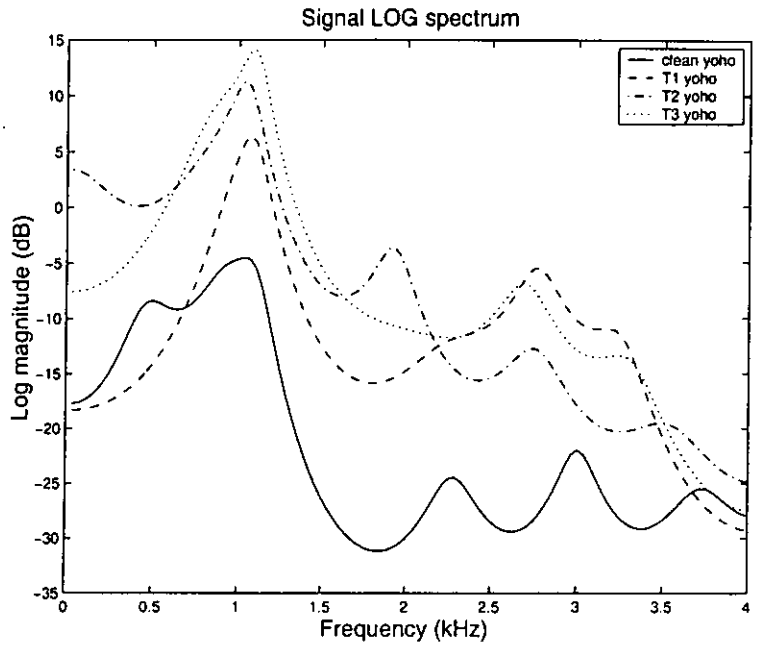


Figure 7.4: The smooth spectra of a voiced frame extracted from the clean YOHO and telephone YOHO corpora.

7.2 Channel Model

To measure the frequency responses of telephone handsets, the same equipment set up (mouth simulator, telephone test head and telephone interface) described in Section 7.1 was used. The master computer generated sinusoidal signals of frequency ranging from 100Hz to 4000Hz in steps of 100Hz. Each of these sinusoidal signals was generated one at a time and was amplified by the telephone interface before playing on the mouth simulator. A sound level meter (B&K 2238) was used to measure the sound pressure level at the mouth reference point (25mm in front of the mouth simulator's lip ring). The sound pressure level was maintained at a constant level across all frequencies of interest. Each of the frequency tones was picked up by the tele-

phone handset, and a digital multimeter (DVM) connecting directly to the telephone interface was used to measure the root mean square values of the handset's output (without the ± 24V DC offset).

Prior to measuring the frequency response of the telephone handset, the mouth simulator should be calibrated using the sound pressure meter. The goal of the calibration is to set up a constant sound pressure level at the reference point over the frequency range of interest. This was achieved by adjusting the amplitude of the sine wave generated by the server computer and by recording the sound pressure level at the reference point using the sound pressure meter with the handset being dismounted. The sine wave's amplitude was adjusted manually until a stable and desirable sound pressure level was obtained. The sound pressure levels that have been used include 75dB, 80dB, 85dB and 90dB. For each of these sound pressure levels, measurement was carried out in the frequency range 100Hz to 4000Hz in steps of 100Hz. Figure 7.5 shows the frequency responses of three handsets at different sound pressure levels using linear weighting.

After the input amplitude has been determined for a particular frequency and sound pressure level, the telephone handset was then mounted on the telephone test head and measurement was performed. A single frequency tone was picked up by the telephone handset and the corresponding root mean square voltage on the DVM was recorded. As a result, we obtained a set of frequency response curves for each handset. Each curve was measured at a particular sound pressure level and consists of 40 discrete points.

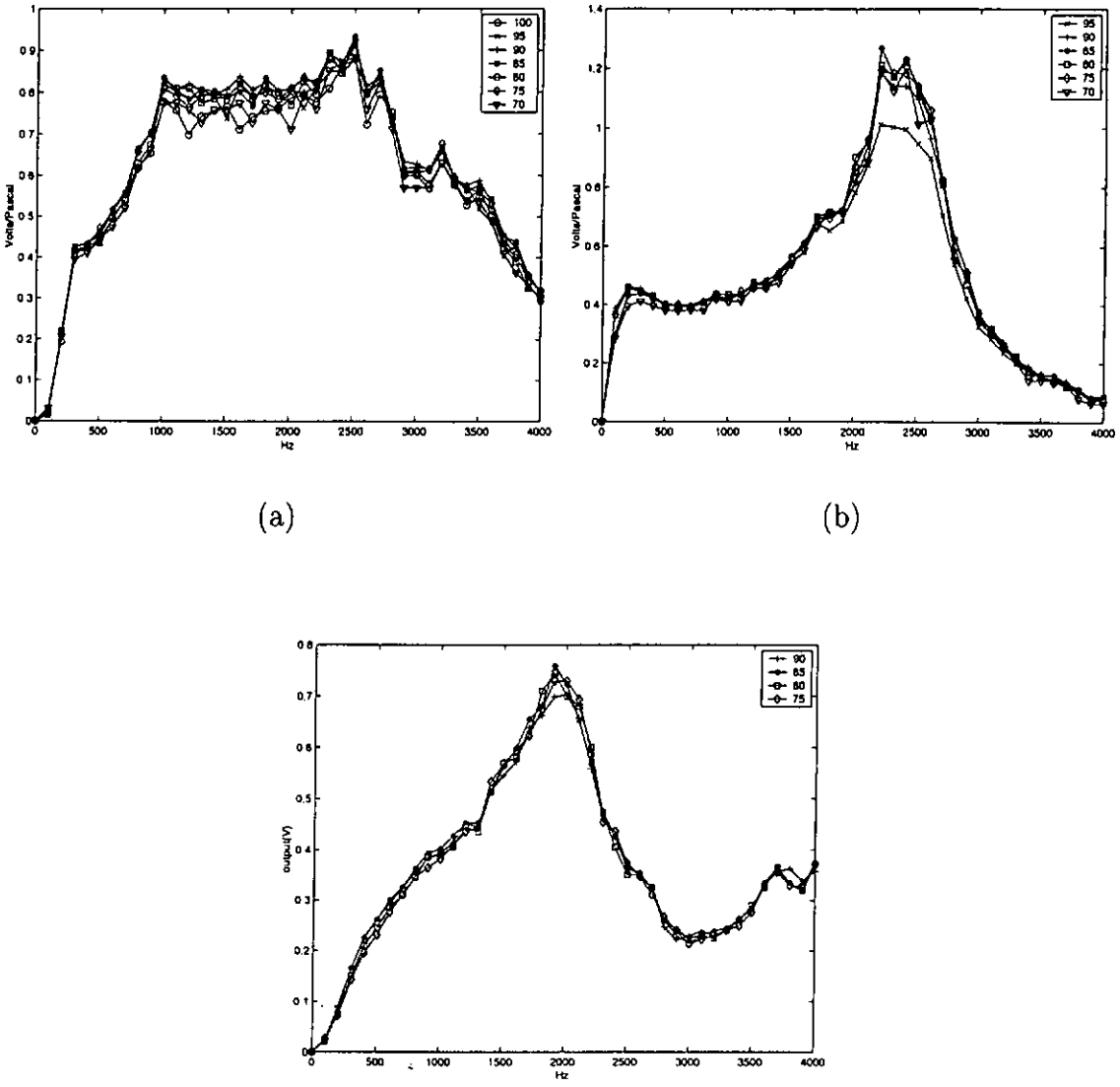


Figure 7.5: Frequency responses of (a) Handset 1, (b) Handset 2, and (c) Handset 3 at different sound pressure levels using linear weighting.

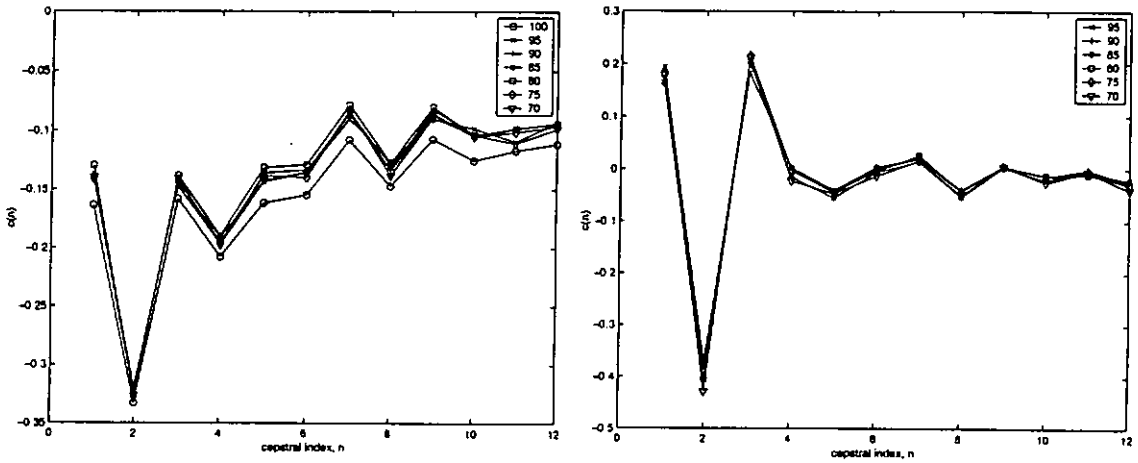
The 40 discrete points were interpolated to produce a frequency response with 128 discrete points. The frequency response was then mirrored to produce a 256-point series. The channel cepstrum was computed by applying inverse discrete Fourier transform on the log magnitude of the 256-point series. The channel cepstrum is derived from the first twelve coefficients of the transformed series excluding the dc component. Figure 7.6 shows the cepstra of three handsets at different sound pressure levels. Evidently, different handsets have different cepstra, resulting in different distortion to the clean speech.

7.3 Speaker Verification Experiments using Telephone Speech

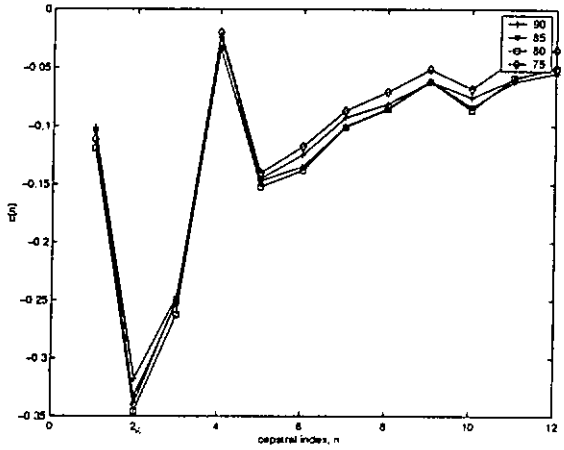
In the experiment, a speaker verification system with VQ speaker models has been used. The speakers models consist of 128 function centers and were trained with the training sessions of the clean YOHO corpus. VQ models were used because of their short training time. For each registered speaker, a speaker-specific codebook was generated by clustering his/her voice patterns using the classical LBG algorithm [29].

Verification was performed using the testing sessions of the clean YOHO corpus and the telephone YOHO corpora described in Section 7.1. The aim is to compare the speaker verification performance under "matched" and "mismatched" conditions.

Equal error rate (EER) is used as a performance index to compare the verification performance among different channel normalization techniques. As the speaker models remain fixed once they were trained, (i.e., their discriminating capability remains unchanged), EER can be used to determine the degree of feature overlapping between



(a) (b)



(c)

Figure 7.6: Cepstra of handsets at different sound pressure levels. (a) Handset 1. (b) Handset 2. (c) Handset 3.

the true speaker and the impostors. The EER is determined by adjusting the decision threshold such that the false rejection curve of the speaker's test utterance crosses the false acceptance curve of 45 random selected impostors.

In telephone-based speaker verification, the acquired speech signal, $y(t)$, is often a distorted version of the clean signal, $x(t)$. There are two types of distortions: convolutive and additive. Therefore, the acquired signal $y(t)$ can be expressed as

$$y(t) = x(t) * h(t) + n(t) \tag{7.1}$$

where h, n, and $*$ represent the impulse response of the channel, the additive noise, and the convolution operators, respectively.

For environment with high signal-to-noise ratio, the additive noise can be neglected and the convolutive channel noise dominates the verification system performance. In this work, a novel channel normalization technique, which is based on the channel cepstra obtained in Section 7.2, is proposed. Evaluation using the telephone YOHO corpora and comparison with other normalization techniques such as cepstral mean subtraction (CMS) and signal bias removal (SBR) has been performed.

7.3.1 Cepstral Mean Subtraction (CMS)

In CMS [14], the channel cepstrum, $\bar{c}$, is estimated as the average cepstral vector of the distorted utterance, i.e.,

$$\bar{c} = \frac{1}{T}\sum_{t=1}^{T} y_t \tag{7.2}$$

where $\{Y = y_1, y_2, \cdots, y_t, \cdots, y_T\}$ are the distorted cepstral vectors. The clean signal is recovered by subtracting the average from the distorted signal, i.e.,

$$\hat{x}_t = y_t - \bar{c} \quad \forall t. \tag{7.3}$$

Although this simple technique has been widely used in many speech/speaker recognition systems and has achieved a reasonably good performance, the underlying assumption that the mean cepstrum of clean speech is zero is not always correct [35].

7.3.2 Signal Bias Removal (SBR)

In SBR [57], the channel is represented by an additive bias term b. This bias term is estimated from the distorted cepstral vectors using the following maximum likelihood formulation:

$$p(Y|\bar{b}, \Lambda) = \max_{b,i} \left[\prod_t p(y_t - b|\lambda_i) \right] \tag{7.4}$$

where $\{Y = y_1, y_2, \cdots, y_t, \cdots, y_T\}$ is an observation sequence of T frames and $\Lambda = \{\lambda_i, i = 1, 2, \cdots, M\}$ are the hidden Markov models of M different speech units. For a given Λ, the maximum likelihood bias estimator, $\bar{b}$, is the one that achieves Equation (7.4). This results in a two-step iterative procedure. Given a set of centroids μ_i of distorted speech, the procedure begins with estimating the channel bias b from each utterance of T frames, i.e.,

$$b \simeq \bar{b} = \frac{1}{T} \sum_{t=1}^{T} (y_t - \mu_{i^*}) \tag{7.5}$$

where μ_{i^*} is the nearest neighbor according to a distance criterion which is consistent with the probability distribution of the distorted signal y_t:

$$i^* = \arg\min_i \|y_t - \mu_i\|. \tag{7.6}$$

The estimated bias, $\bar{b}$, is then subtracted from the distorted signal to recover the undistorted cepstrum, i.e.,

$$\tilde{x}_t = y_t - \bar{b} \quad \forall t. \tag{7.7}$$

These two-step procedure results in a maximization of Equation (7.4). The procedure is iterated until a local optimal solution for b is reached.

7.3.3 Channel Cepstrum

The channel cepstrum $\bar{c}$ is derived from the direct measurement of the telephone handset's frequency response $H(\omega)$. It is computed by truncating the inverse Fourier transform of the log magnitude of the interpolated frequency response, $\widehat{H(\omega)}$, i.e.,

$$\bar{c} = \text{truncated } \{ \text{ IFFT}\{\log|\widehat{H(\omega)}|\}\} \tag{7.8}$$

After truncating and removing the dc component, a twelve-th order channel cepstrum is resulted.

Similar to CMS, the channel cepstrum is subtracted from the distorted signal to recover the source signal, i.e.,

$$\tilde{x}_t = y_t - \bar{c} \quad \forall t. \tag{7.9}$$

We have measured the frequency responses of different handsets at sound pressure

levels of 75dB, 80dB, 85dB and 90dB. The final channel cepstrum of each handset is calculated by averaging the cepstra at these sound pressure levels.

Figure 7.7 shows the smooth spectra of a voiced frame recovered from different compensation techniques (no processing, CMS, SBR, and our proposed channel normalization). Note that the spectra are shifted upward to avoid overlapping and to provide clear illustrations. The figure shows that the three techniques do a reasonably good job in compensating the low frequency part of the distorted speech. However, all of the recovered spectra exhibit a spectral tilt at the high frequency region. The degree of the spectral tilt is comparable to the unprocessed telephone speech, but it is higher than that of the clean speech. This result suggests that there is plenty of room for improving the compensation techniques, and that better results may be obtained by focusing on the high frequency region of the recovered spectra.

7.4 Results and Analysis

Table 7.2 compares the equal error rates (EERs) obtained by different channel normalization techniques. It shows that for the clean YOHO corpus, CMS deteriorates the verification performance in the matched condition since the equal error rate is about tenfold higher than that when CMS is not applied. This result suggests that CMS can remove speaker-specific information, although its objective is to remove convolutive distortion. The high EER corresponding to the telephone speech evidences the mismatched conditions created by the handsets. Although CMS has been widely used, Table 7.2 shows that it is less effective in normalizing the telephone speech as com-

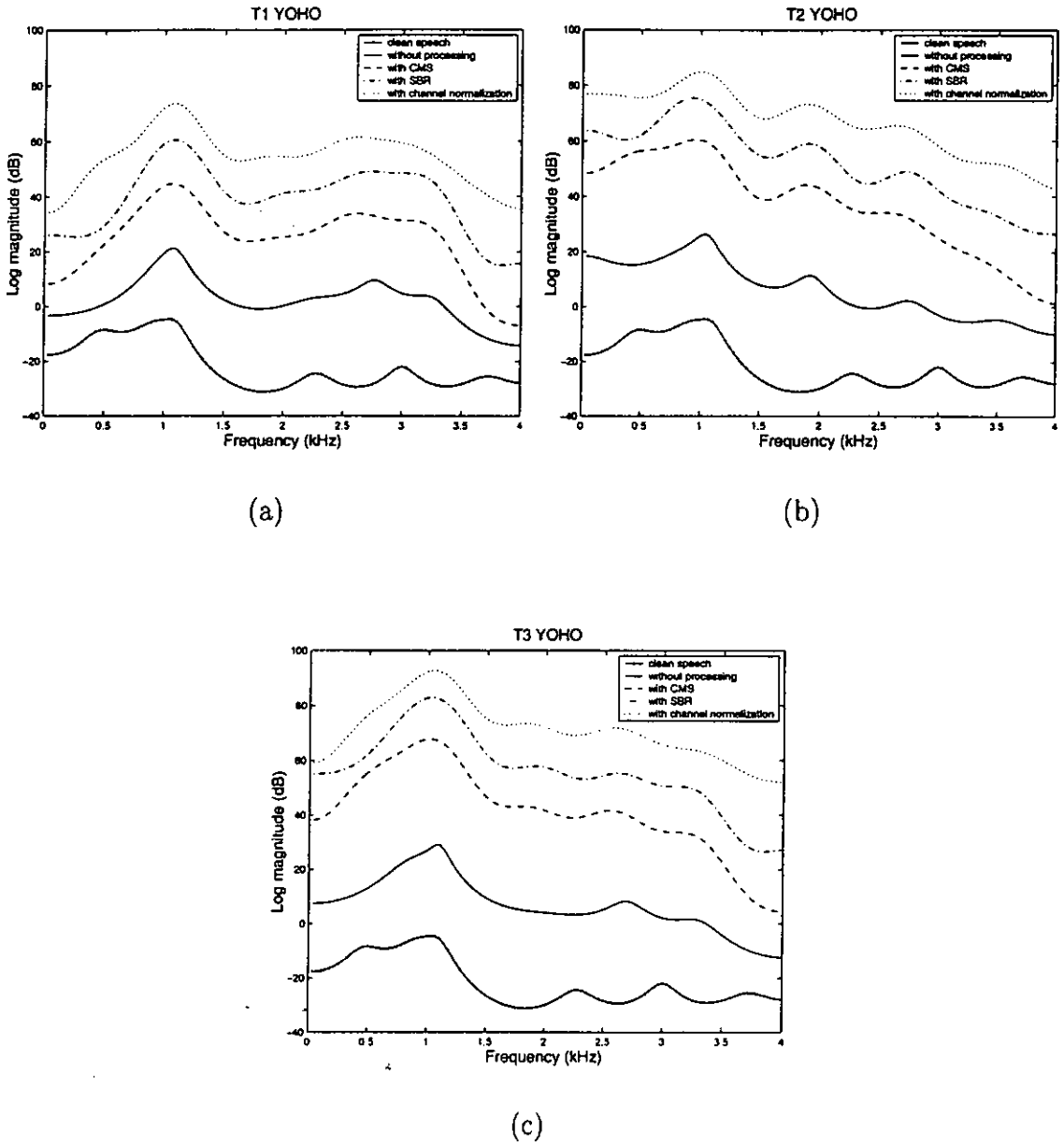


Figure 7.7: The smooth spectra of a voiced frame recovered from different compensation techniques (no processing, CMS, SBR, and our proposal channel normalization). The voice frame is extracted from (a) the T1 telephone corpus, (b) the T2 telephone corpus, and (c) the T3 telephone corpus.

pared to our proposed channel cepstrum and SBR. In particular, it deteriorates the system performance in the case of T3 YOHO. Our channel cepstrum normalization technique is superior to CMS in all cases.

We also compare our technique with a recently proposed technique, signal bias removal (SBR). Table 7.2 shows that SBR is superior to our proposed technique. This may be due to the fact that SBR uses a maximum likelihood formulation to estimate a bias term for each utterance. In our case, a single bias term is applied to all utterances. This suggests that performance improvement can be expected if we combine the channel cepstrum with the maximum likelihood formulation in an appropriate manner. For an example, we can assumes that the bias term has the form

$$b = \lambda \vec{s} + \bar{c} \tag{7.10}$$

where the channel cepstrum $\bar{c}$ is based on the direct measurement of the telephone handset's frequency response, $\vec{s}$ is a shift vector with the same magnitude at all dimension in the feature space and λ is a scalar determined by an estimation technique similar to SBR. The bias represents a linear shift of the channel cepstrum and is determined for each utterance.

We have also applied the three channel cepstra in a blind manner. For example, the channel cepstrum corresponding to Handset 1 was applied to compensate the speech obtained from Handsets 2 and 3. Table 7.3 shows the equal error rate obtained when we blindly used one of the channel cepstra (e.g., Handset 1) to compensate the channel effect due to other handsets (e.g., Handsets 2 and 3). Surprisingly, using the

Channel Normalization Method	Equal Error Rate			
	Clean Yoho	T1 Yoho	T2 Yoho	T3 Yoho
No compensation	1.16	28.77	29.92	25.82
CMS	10.85	21.93	29.09	26.51
Channel cepstrum	-	19.74	23.65	25.25
SBR	1.32	14.42	21.22	20.09

Table 7.2: Equal error rates obtained by different channel compensation techniques.

Channel Normalization Method	Equal Error Rate		
	T1 Yoho	T2 Yoho	T3 Yoho
T1 channel cepstrum	19.74	32.01	23.59
T2 channel cepstrum	27.61	23.65	25.13
T3 channel cepstrum	22.64	33.56	25.25

Table 7.3: Equal error rates obtained by applying different channel cepstra to compensate the channel effect caused by the three telephone handsets.

channel cepstrum to compensate the speech distorted by the corresponding channel does not necessarily produce the lowest error rate. For example, compensating T1 YOHO by T3 channel cepstrum gives the best performance. This suggests that a handset classifier should be used instead of blindly using a channel cepstrum. The handset classifier should be able to select the best channel cepstrum from a pool of known channel cepstra according to the observed telephone speech. The pool of known channel cepstra is derived from the frequency responses of a large set of telephone handsets. In addition to handset classifier, we can also linearly combine the pool of channel cepstra according to the incoming telephone speech.

7.5 Summary

In this chapter, we have explained the techniques to create telephone speech corpora and presented several experiments that demonstrate the effects of handset variability on text-independent speaker recognition performance. The telephone corpora were recorded by playing the clean YOHO corpus directly through different telephone handsets. The channel spectra of the telephone handsets are measured using B&K's mouth simulator, telephone test head and telephone interface. Channel cepstra are derived from the channel spectra to normalize the telephone speech in the cepstral domain.

Performance evaluations based on the telephone YOHO corpora have been performed. Comparison with traditional channel normalization techniques indicates that our channel cepstrum normalization method is superior to the conventional CMS. However, it is still inferior to SBR. Suggestions are given to further improve the performance of the proposed technique.

Chapter 8

CONCLUSIONS

In this work, we have addressed two important issues in speaker verification: threshold determination and channel mismatch. In this chapter, we conclude our findings based on our simulation results.

We have investigated the techniques that minimize the error rate of telephone-based speaker verification systems. For example, in order to determine the *a priori* decision thresholds and to avoid impostor attacks, we have proposed to use probabilistic decision-based neural networks (PDBNNs) as speaker models. As PDBNNs can be considered as GMMs with trainable decision thresholds, we have compared the structural properties and learning rules of PDBNNs with that of GMMs. The performance and characteristics of PDBNNs and GMMs have been evaluated based on the noisy XOR and 2-D vowel problems. Speaker verification experiments using VQs, PDBNNs and GMMs as speaker models have been carried out in order to evaluate their performances.

For the channel mismatch problem, we have explained the techniques to create telephone corpora, and we have measured the characteristics of different handsets in order to derive the channel cepstra. We have also proposed and evaluated a new channel normalization method and compared its performance with that of other al-

gorithms.

Here, we summarize our major findings of this work.

- With the thresholding mechanism, the performance of PDBNNs can be divided into recognition accuracy, incorrectly recognized rate, unclassifiable rate, and false acceptance rate. It was found that large decision thresholds result in low recognition accuracy, and vice versa for small thresholds.

- The thresholding mechanism of PDBNNs produces bounded and locally conserved decision regions. This effectively minimizes the chance of falsely accepting unknown patterns. On the other hand, the GMMs' decision regions are unbounded. Hence, PDBNNs are suitable for classification applications where the minimization of false acceptance rate is an important issue.

- The use of PDBNNs as speaker models for speaker verification is promising. Experimental evaluations suggest that the smallest equal error rate (EER) obtained by the PDBNNs is about half of that of Higgins et al. [20] (0.89% vs. 1.80%). The EERs also demonstrate the superiority of the PDBNNs over the VQ speaker models.

- The PDBNNs are more robust in rejecting impostors as compared to GMMs. This is indicated by the low FAR that the PDBNNs have achieved in the speaker verification experiments and by their locally conserved decision boundaries in the hypothetical problem.

- The major source of degradation encountered in telephone-based speaker verification is caused by handset variability. Handset variability, however, can be reduced by adding an offset to the distorted speech in the cepstral domain.

- Channel cepstra can be derived from the frequency response of telephone handsets. Comparison with conventional channel normalization techniques indicates that our channel normalization method is superior to the cepstral mean subtraction. However, it is still inferior to signal bias removal.

Chapter 9

FUTURE WORK

While this work has shown that PDBNNs provide a promising solution to speaker modeling, the problem of channel mismatches remains largely unsolved. Here, we propose some approaches to resolving the channel mismatch problem in future work.

Methods based on cepstral subtraction assume that the channel is linear. This assumption, however, is known to be invalid because most telephone handsets exhibit energy dependent frequency responses. Therefore, we propose to develop a handset classifier based on the energy dependent frequency responses of telephone handsets so as to avoid the reliance on this assumption. Fig. 9.1 depicts the procedure of creating such classifier. The frequency responses at G different energy levels of M telephone handsets with various microphone types are measured according to IEEE Standard 269-1992 [21]. The frequency responses are then converted to cepstral coefficients $\{\vec{c}_{k,g}\}$ where k and g represent the handset type and energy level, respectively. Distorted speech is produced by playing clean speech to each of the M handsets at different energy levels. A feedforward neural network [7] is trained to classify the distorted speech (cepstrum) as coming from one of the handsets. Therefore, the training set is composed of $\{\vec{x}_k, g; \vec{z}_k\}$ where $\vec{x}_k$ $(k = 1, \ldots, M)$ denotes the distorted cepstrum originated from handset k, g the energy level, and $\vec{z}_k$ the desired output vectors with

the kth component being equal to 1 and the others equal to 0.

Fig. 9.2 depicts the operation of the classifier. The neural network classifies the telephone speech as coming from one of the known channels. Assuming that the channel characteristics remain unchanged within an utterance, we can obtain the channel cepstrum in an utterance-by-utterance basis as follows. A cepstral sequence and an energy sequence, $\{\vec{x}(t), g(t); t = 1, \ldots, T\}$, corresponding to the current utterance is fed to the neural network. Then, the network's outputs $y_k(t), k = 1, \ldots, M$ and $t = 1, \ldots, T$, are averaged to yield the average outputs

$$\bar{y}_k = \frac{1}{T} \sum_{t=1}^{T} y_k(t). \tag{9.1}$$

Note that $\sum_k \bar{y}_k \simeq 1$ as $y_k(t) \simeq P(C_k|\vec{x}(t), g)$. As the network outputs can be considered as the *a posteriori* probabilities of the channels given the telephone speech and its energy [65], the estimated channel can be expressed as a linear combination of the known channels. More specifically, the channel cepstrum is approximated by

$$\vec{c}_{\text{chan}}(g) = \sum_{k=1}^{M} \bar{y}_k \vec{c}_{k,g}. \tag{9.2}$$

If the winner-takes-all approach is used, we have

$$\vec{c}_{\text{chan}}(g) = \vec{c}_{i,g} \tag{9.3}$$

where $i = \text{argmax}_k\ \bar{y}_k$. Then, the channel effect of the current frame of speech is removed by subtracting the energy dependent channel cepstrum from the distorted speech:

$$\vec{s} = \vec{x} - \vec{c}_{\text{chan}}(g). \tag{9.4}$$

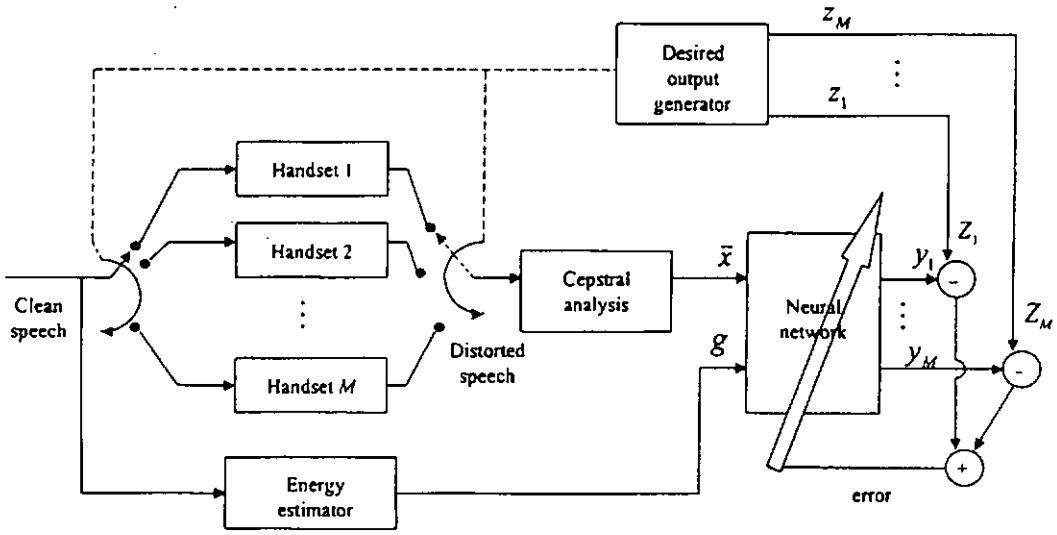


Figure 9.1: Creating the channel classifier

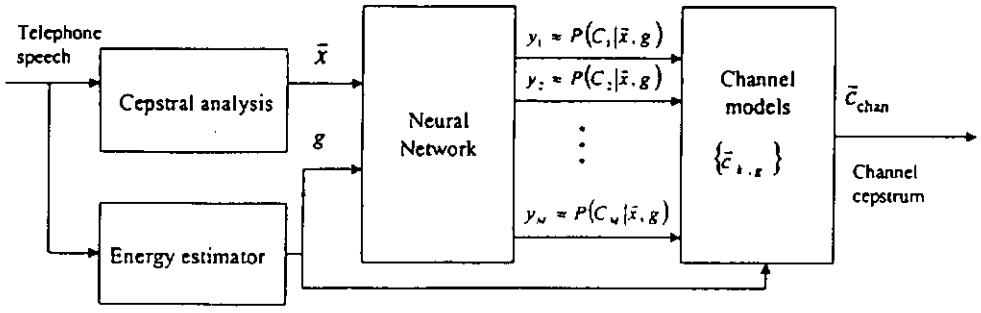


Figure 9.2: Estimating the energy dependent channel cepstrum

BIBLIOGRAPHY

[1] A. Acero and R. M. Stern. Environmental robustness in automatic speech recognition. In *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, pages 849–852, 1990.

[2] A. Acero and R. M. Stern. Robust speech recognition by normalization of the acoustic space. In *ICASSP-91*, pages 893–896, 1991.

[3] Mohamed Afify, Yifan Gong, and Jean-Paul Haton. A general joint additive and convolutive bias compensation approach applied to noisy lombard speech recognition. *IEEE Trans. on Speech and Audio Processing*, 6(6):524–538, 1998.

[4] K. T. Assaleh and R. J. Mammone. New LP-derived features for speaker identification. *IEEE Trans. on Speech and Audio Processing*, 2(4):630–638, 1994.

[5] B. S. Atal. Effectiveness of linear prediction characteristics of the speech wave for automatic speaker identification and verification. *J. Acoust. Soc. Amer.*, 55(6):1304–1312, June 1974.

[6] F. Beaufays and M. Weintraub. Model transformation for robust speaker recognition from telephone data. In *ICASSP97*, volume 2, pages 1063–1066, 1997.

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

[8] S. F. Boll. Suppression of acoustic noise in speech using spectral substraction. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-27(2):113–120, April 1979.

[9] K. L. Brown and E. B. George. CTIMIT: a speech corpus for the cellular environment with applications to automatic speech recognition. In *ICASSP-95*, volume 1, pages 105–108, 1995.

[10] J. Ćwik and J. Koronacki. Probability density estimation using a Gaussian clustering algorithm. *Neural Computing and Applications*, 4:149–160, 1996.

[11] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. of Royal Statistical Soc., Ser. B.*, 39(1):1–38, 1977.

[12] K. M. Dobroth, B. L. Zeigler, and D. Karis. Future directions for audio interface research: characteristics of human-human order-entry conversations. In *Proc. Am. Voice Input/Output Soc.*, September 1989.

[13] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* Wiley, New York, 1973.

[14] S. Furui. Cepstral analysis technique for automatic speaker verification. *IEEE Trans. on Acoustics, Speech and Signal Processing*, ASSP-29(2):254–272, April 1981.

[15] M. J. F. Gales and S. J. Young. Robust speech recognition in additive and convolutional noise using parallel model combination. In *Computer Speech and Language*, volume 9, pages 289–307, 1995.

[16] F. Girosi. Some extensions of radial basis functions and their applications in artificial intelligence. *Computers Math. Applic.*, 24(12):61–80, 1992.

[17] F. Girosi and T. Poggio. Networks and the best approximation property. *Biological Cybernetics*, 63:169–176, 1990.

[18] S. K. Gupta and M. Savic. Text-independent speaker verification based on broad phonetic segmentation of speech. *Digital Signal Processing*, (2):69–79, 1992.

[19] H. Hermansky and N. Morgan. RASTA processing of speech. *IEEE Transactions on Speech and Audio Processing*, 2(4):578–589, Oct 1994.

[20] A. Higgins, L. Bahler, and J. Porter. Speaker verification using randomized phrase prompting. *Digital Signal Processing*, 1:89–106, 1991.

[21] IEEE. IEEE standard methods for measuring transmission performance of analog and digital telephone sets. *IEEE Std. 269-1992*, 1993.

[22] Jr. J. P. Campbell. Testing with the YOHO CD-ROM voice verification corpus. In *ICASSP'95*, volume 1, pages 341–344, 1995.

[23] C. Jankowski, A. Kalyanswamy, S. Basson, and J. Spitz. NTIMIT: A phonetically balanced, continuous speech, telephone bandwidth speech database. In *ICASSP90*, pages 109–112, 1990.

[24] B. H. Juang, L. R. Rabiner, and J. G. Wilpon. On the use of bandpass liftering in speech recognition. *IEEE Trans. on Acoustic, Speech and Signal Processing*, ASSP-35(7):947–954, 1987.

[25] N. B. Karayiannis and G. W. Mi. Growing radial basis neural networks: Merging supervised and unsupervised learning with network growth techniques. *IEEE Trans. on Neural Networks*, 8(6):1492–1506, 1997.

[26] S. Y. Kung. *Digital Neural Networks*. Prentice Hall, New Jersey, 1993.

[27] S. Y. Kung and J. S. Taur. Decision-based neural networks with signal/image classification applications. *IEEE Trans. on Neural Networks*, 6:170–181, 1995.

[28] S. H. Lin, S. Y. Kung, and L. J. Lin. Face recognition/detection by probabilistic decision-based neural network. *IEEE Trans. on Neural Networks, Special Issue on Biometric Identification*, 8(1):114–132, 1997.

[29] Y. Linde, A. Buzo, and R. M. Gray. An algorithm for vector quantizer design. *IEEE Trans. on Communications*, 28(1):84–95, 1980.

[30] R. P. Lippmann. Pattern classification using neural networks. *IEEE Communications Magazine*, 27:47–50,59–64, 1989.

[31] C. S. Liu, C. H. Lee, B. H. Juang, and A. E. Rosenberg. Speaker recognition based on minimum error discriminative training. In *Proc. ICASSP'94*, volume 1, pages 325–328, 1994.

[32] C. S. Liu, H. C. Wang, and C. H. Lee. Speaker verification using normalized log-likelihood score. *IEEE Trans on Speech and Audio Processing*, 4(1):56–60, 1996.

[33] F. H. Liu, A. Acero, and R. M. Stern. Effective joint compensation of speech for the effects of additive noise and linear filtering. In *ICASSP-92*, volume 1, pages 257–260, 1992.

[34] F. H. Liu, R. M. Stern, A. Acero, and P. J. Moreno. Environment normalization for robust speech recognition using direct cepstral comparison. In *ICASSP-94*, volume 2, pages 61–64, 1994.

[35] T. F. Lo, K. K. Yiu, and M. W. Mak. A new cepstrum-based channel compensation method for speaker verification. In *Proc. Eurospeech'99*, volume 2, pages 775–778, Sept. 1999.

[36] D. Lowe and A. R. Webb. Exploiting prior knowledge in network optimization: an illustration from medical prognosis. *Network: Computation in Neural Systems*, 1:299–323, 1990.

[37] M. W. Mak, W. G. Allen, and G. G. Sexton. Comparing multi-layer perceptrons and radial basis function networks in speaker recognition. *J. of Microcomputer Applications*, 16:147–159, 1993.

[38] M. W. Mak, W. G. Allen, and G. G. Sexton. Speaker identification using multi-layer perceptrons and radial basis functions networks. *Neurocomputing*, 6:99–118, 1994.

[39] M. W. Mak and C. K. Li. Elliptical basis function networks and radical basis function networks for speaker verification: A comparative study. In *IJCNN'99*, 1999.

[40] M. W. Mak, C. K. Li, and X. Li. Maximum likelihood estimation of elliptical basis function parameters with application to speaker verification. In *Proc. Int. Conf. on Signal Processing*, volume 2, pages 1287–1290, 1998.

[41] M.W. Mak and S.Y. Kung. Estimation of elliptical basis function parameters by the EM algorithms with application to speaker verification. In *IEEE Trans. on Neural Networks*, (in press).

[42] R. J. Mammone, X. Zhang, and R. P. Ramachandran. Robust speaker recognition. *IEEE Signal Processing Magazine*, pages 58–71, September 1996.

[43] D. Mansour and B. H. Juang. A family of distortion measures based upon projection operation for robust speech recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37(11):1659–1671, November 1989.

[44] J. D. Markel, B. T. Oshika, and Jr A. H. Gray. Long-term feature averageing for speaker recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-25:330–337, 1977.

[45] T. Matsui and S. Furui. Likelihood normalization for speaker verification using a phoneme- and speaker-independent model. *Speech Communication*, 17:109–116, 1995.

[46] J. Moody and C. J. Darken. Fast learning in networks of locally tuned processing units. *Neural Computation*, 1(2):281–194, 1989.

[47] P. J. Moreno, B. Raj, E. Gouvea, and R. M. Stern. Multivariate-gaussian-based cepstral normalization for robust speech recognition. In *ICASSP-95*, volume 1, pages 137–140, 1995.

[48] P. J. Moreno, B. Raj, and R. M. Stern. A vector Taylor series approach for environment-independent speech recognition. In *ICASSP-96*, volume 2, pages 733–736, 1996.

[49] P. J. Moreno and R. M. Stern. Sources of degradation of speech recognition in the telephone network. In *ICASSP-94*, volume 1, pages 109–112, 1994.

[50] D. Naik. Pole-filtered cepstral mean subtraction. In *ICASSP95*, volume 1, pages 157–160, 1995.

[51] J. M. Naik, L. P. Netsch, and G. R. Doddington. Speaker verification over long distance telephone lines. In *Proc. ICASSP'89*, volume 1, pages 524–527, 1989.

[52] K. Ng and R. P. Lippmann. Practical charcteristics of neural network and conventional pattern classifiers. *Advances in Neural Inform. Processing Syst.*, 3:970–976, 1991.

[53] J. Oglesby and J. S. Mason. Optimization of neural models for speaker identification. In *Proc. ICASSP'90*, volume 1, pages 261–264, 1990.

[54] E. Parzen. On estimation of a probability density function and mode. *Annals of Mathematical Statistics*, 33:1065–1076, 1962.

[55] T. Poggio and F. Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9):1481–1497, 1990.

[56] A. B. Poritz. Linear predictive hidden Markov models and the speech signal. In *Proc. ICASSP'82*, volume 2, pages 1291–1294, 1982.

[57] M. G. Rahim and B. H. Juang. Signal bias removal by maximum likelihood estimation for robust telephone speech recognition. *IEEE Transactions on Speech and Audio Processing*, 4(1):19–30, Jan 1996.

[58] B. Raj, E. B. Gouvea, P. J. Moreno, and R. M. Stern. Cepstral compensation by polynomial approximation for environment-independent speech recognition. In *ICSLP-96*, volume 4, pages 2340–2343, 1996.

[59] V. Ramamoorthy, N. S. Jayant, R. V. Cox, and M. M. Sondhi. Enhancement of ADPCM speech coding with backward-adaptive algorithms for postfiltering and noise feedback. *IEEE Journal on Selected Areas in Communications*, 6(2):364–382, 1988.

[60] D. A. Reynolds. Speaker identification and verification using gaussian mixture speaker models. In *Speech Communications*, volume 17, pages 91–108, 1995.

[61] D. A. Reynolds. The effects of handset variability on speaker recognition performance: experiments on the switchboard corpus. In *ICASSP96*, volume 1, pages 113–116, 1996.

[62] D. A. Reynolds. HTIMIT and LLHDB: speech corpora for the study of handset transducer effects. In *ICASSP-97*, volume 2, pages 1535 –1538, 1997.

[63] D. A. Reynolds and R. C. Rose. Robust text-independent speaker identification using Gaussian mixture speaker models. *IEEE Trans. on Speech and Audio Processing*, 3(1):72–83, 1995.

[64] D. A. Reynolds, M. A. Zissman, T. F. Quatieri, G. C. O'Leary, and B. A. Carlson. The effects of telephone transmission degradations on speaker recognition performance. In *ICASSP95*, volume 1, pages 329–332, 1995.

[65] M. D. Richard and R. P. Lippmann. Neural network classifiers estimate Bayesian a posteriori probabilities. *Neural Computation*, 3:461–483, 1991.

[66] S. Roberts and L. Tarassenko. A probabilistic resource allocating network for novelty detection. *Neural Computation*, 6:270–284, 1994.

[67] R. C. Rose, E. M. Hofstetter, and D. A. Reynolds. Integrated models of signal and background with application to speaker identification in noise. *IEEE Trans. on Speech and Audio Processing*, 2(2):245–257, 1994.

[68] A. E. Rosenberg, J. Delong, C. H. Lee, B. H. Juang, and F. K. Soong. The use of cohort normalized scores for speaker verification. In *Proc. ICSLP 92*, pages 599–602, 1992.

[69] A. E. Rosenberg and S. Parthasarathy. Speaker background models for connected digits password speaker verification. In *Proc. ICASSP'96*, volume 1, pages 81–84, 1996.

[70] A. E. Rosenberg, O. Siohan, and S. Parthasarathy. Speaker verification using minimum verification error training. In *Proc. ICASSP'98*, pages 105–108, 1998.

[71] D. W. Ruck, S. K. Rogers, M. Kabrisky, M. E. Oxley, and B. W. Suter. The multilayer perceptron as an approximation to a Bayes optimal discriminant function. *IEEE Trans. on Neural Networks*, 1(4):296–298, December 1990.

[72] A. Sankar and C. H. Lee. A maximum-likelihood approach to stochastic matching for robust speech recognition. In *IEEE Trans. on Speech and Audio Processing*, volume 4, pages 190–202, 1996.

[73] F. K. Soong and A. E. Rosenberg. On the use of instantaneous and transitional spectral information in speaker recognition. *IEEE Trans. on ASSP*, 36(6):871–879, June 1988.

[74] F. K. Soong, A. E. Rosenberg, L. R. Rabiner, and B. H. Juang. A vector quantization approach to speaker recognition. In *Proc. ICASSP 85*, volume 1, pages 387–390, 1985.

[75] D. F. Specht. Probabilistic neural networks. *Neural Networks*, 3:109–118, 1990.

[76] Y. Tohkura. A weighted cepstral distance measure for speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-35(10):1414–1422, October 1987.

[77] H. G. C. Tråvén. A neural network approach to statistical pattern classification by semiparametric estimation of probability density functions. *IEEE Trans. on Neural Networks*, 2(3):366–377, May 1991.

[78] A. R. Webb. Functional approximation by feedforward networks: a least-squares approach to generalization. *IEEE Transactions on Neural Networks*, 5(3):363–371, 1994.

[79] K. K. Yiu, M. W. Mak, and C. K. Li. Probabilistic decision-based neural networks for speech pattern classification. *ICSP'98*, 2:1378–1381, 1998.

[80] K. K. Yiu, M. W. Mak, and C. K. Li. Gaussian mixture models and probabilistic decision-based neural networks for pattern classification: A comparative study. *Neural Computing & Applications*, 8:235–245, 1999.

[81] W. D. Zhang, K.K. Yiu, M. W. Mak, C. K. Li, and M. X. He. A priori threshold determination for phrase-prompted speaker verification. In *Eurospeech'99*, volume 2, pages 1023–1026, 1999.

[82] Y. Zhao. An EM algorithm for linear distortion channel estimation based on observations from a mixture of gaussian sources. In *IEEE Trans. on Speech and Audio Processing*, volume 7, pages 400–413, 1999.

[83] M. S. Zilovic, R. P. Ramachandran, and R. J. Mammone. Speaker identification based on the use of robust cepstral features obtained from pole-zero transfer functions. *IEEE Trans. on Speech and Audio Processing*, 6(3):260–267, 1998.

AUTHOR'S PUBLICATIONS

International Journal Papers

1. K. K. Yiu, M. W. Mak and C. K. Li. Gaussian Mixture Models and Probabilistic Decision-Based Neural Networks for Pattern Classification: A Comparative Study, *Neural Computing and Applications*, Vol. 8, pp. 235–245, 1999.

International Conference Papers

2. K. K. Yiu, M. W. Mak, T. F. Lo and C. K. Li. Probabilistic Decision-Based Neural Networks for Speaker Verification with A Priori Decision Thresholds. In *1999 International Symposium on Signal Processing and Intelligent Systems*, pages 612–717, 1999.

3. K. K. Yiu, M. W. Mak and C. K. Li. Probabilistic Decision-Based Neural Networks for Speech Pattern Classification. In *International Conference on Signal Processing (ICSP'98)*, pages. 1378–1381, 1998.

4. W. D. Zhang, K. K. Yiu, M. W. Mak, C. K. Li and M. X. He. A Priori Threshold Determination for Phrase-Prompted Speaker Verification.

   In *Eurospeech'99*, Vol. 2, pages. 1023–1026, Sept. 1999, Hungary.

5. T. F. Lo, M. W. Mak and K. K. Yiu. A New Cepstrum-Based Channel Compensation Method for Speaker Verification.

   In *Eurospeech'99*, Vol. 2, pages. 775–778, Sept. 1999, Hungary.