Feature and Model Transformation Techniques for Robust Speaker Verification

YIU, Kwok Kwong Michael

A dissertation submitted in partial fulfillment of

the requirements for the degree of

Doctor of Philosophy

Department of Electronic and Information Engineering

The Hong Kong Polytechnic University

September 2004

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

_____(Signed)

YIU KWOK KWONG (Name of student)

Abstract

Feature and Model Transformation Techniques for Robust Speaker Verification

Speaker verification is to verify the identity of a speaker based on his or her own voice. It has potential applications in securing remote access services such as phone-banking and mobile-commerce. While today's speaker verification systems perform reasonably well under controlled conditions, their performance is often compromised under real-world environments. In particular, variations in handset characteristics are known to be the major cause of performance degradation. This dissertation addresses the robustness issue of speaker verification systems in three different angles: speaker modeling, feature transformation, and model transformation.

This dissertation begins with an investigation on the effectiveness of three kernel-based neural networks for speaker modeling. These networks include probabilistic decision-based neural networks (PDBNNs), Gaussian mixture models (GMMs), and elliptical basis function networks (EBFNs). Based on the thresholding mechanism of PDBNNs, the original training algorithm of PDBNNs was modified to make PDBNNs appropriate for speaker verification. Experimental results show that GMM- and PDBNN-based speaker models outperform the EBFN ones in both clean and noisy environments. It was also found that the modified learning algorithm of PDBNNs is able to find decision thresholds that reduce the variation in false acceptance rates, whereas the ad hoc threshold-determination approach used by the EBFNs and GMMs causes

a large variation in the false acceptance rates. This property makes the performance of PDBNN-based systems more predictable.

The effect of handset variation can be suppressed by transforming clean speech models to fit the handset-distorted speech. To this end, this dissertation proposes a model-based transformation technique that combines handset-dependent model transformation and reinforced learning. Specifically, the approach transforms the clean speaker model and clean background model to fit the distorted speech by using maximum-likelihood linear regression (MLLR), which is followed by adapting the transformed models via PDBNN's reinforced learning. It was found that MLLR is able to bring the clean models to a region close to the distorted speech and that reinforced learning is a good means of fine-tuning the transformed models to enhance the distinction between client speakers and impostors.

In addition to model-based approaches, handset variation can also be suppressed by feature-based approaches. Current feature-based approaches typically identify the handset being used as one of the known handsets in a handset database and use the a priori knowledge about the identified handset to modify the features. However, it will be much more practical and cost effective if handset detector-free systems are adopted. To this end, this dissertation proposes a blind compensation algorithm to handle the situation in which no a priori knowledge about the handset is available (i.e., a handset model which is not in the handset database is used). Specifically, a composite statistical model formed by the fusion of a speaker model and a background model is used to represent the characteristics of enrollment speech. Based on the difference

between the claimant's speech and the composite model, a stochastic matching type of approach is proposed to transform the claimant's speech to a region close to the enrollment speech. Therefore, the algorithm can now estimate the transformation online without the necessity of detecting the handset types. Experimental results based on the 2001 NIST Speaker Recognition evaluation set show that the proposed approach achieves significant improvement in both equal error rate and minimum detection cost as compared to cepstral mean subtraction, Znorm, and short-time Gaussianization.

ACKNOWLEDGMENTS

I would like to express sincere gratitude to various bodies in The Hong Kong polytechnic University, where I have the opportunity to study with. My major debt is to my 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 express my appreciation to all the members of the DSP Research Laboratory, past and present, especially for M. C. Cheung, K. K. Kwok, K. Y. Leung, C. H. Sit and C. L. Tsang. 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 three 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

The major contributions of this dissertation are summarised below.

- This dissertation demonstrates that the GMM- and PDBNN- based speaker models outperform the EBFN ones under both clean and noisy environments.

- This dissertation proposes three modifications to the original training algorithm of PDBNNs, which make PDBNNs more appropriate for speaker verification. The modified algorithm is able to find decision thresholds that minimize the variation in false acceptance rate, whereas the ad hoc threshold-determination approach used by the EBFNs and GMMs causes a large variation in the false acceptance error rate. This property makes the performance of PDBNN-based systems more predictable.

- This dissertation proposes a model-based channel compensation algorithm that combines a handset selector with (1) handset-specific transformations, (2) reinforced learning, and (3) stochastic feature transformation to reduce the effect caused by the acoustic distortion. Results show that the proposed algorithm is superior to a number of classical techniques, including CMS, stochastic feature transformation, Hnorm, and speaker model synthesis.

- This dissertation proposes a blind feature-based transformation approach to channel robust speaker verification. The transformation parameters are determined online without any a priori knowledge of channel characteristics. Results show that the proposed transformation approach achieves significant improvement in both equal error rate and minimum detection cost as compared to cepstral mean subtraction, Znorm, and short-time Gaussianization.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

SYMBOLS

$P_{\text{miss}\|\text{target}}$	*Miss rate*, the chance of misclassifying a true speaker as an impostor
$P_{\text{fa}\|\text{nontarget}}$	*False Alarm Rate*, the chance of falsely identifying an impostor as a true speaker
C_{det}	detection cost
C_{miss}	cost of making a false rejection error
C_{fa}	cost of making a false acceptance
P_{target}	the chance of having a true speaker
$P_{\text{nontarget}}$	the chance of having an impostor
C_{miss}	cost of making a false rejection error
C_{fa}	cost of making a false acceptance error
$y(t)$	distorted speech signal
$s(t)$	clean speech signal
$h(t)$	channel's impulse response
$n(t)$	additive noise
$p(\mathbf{x}_t\|\omega_i)$	likelihood function for class ω_i
$\Theta_{r\|i}$	the parameters of the rth mixture component
R	the total number of mixture components
$p(\mathbf{x}_t\|\omega_i, \Theta_{r\|i})$	probabilty density function of the rth component
$P(\Theta_{r\|i}\|\omega_i)$	the prior probability (also called mixture coefficients) of the rth component
$\mathcal{N}(\mu_{r\|i}, \Sigma_{r\|i})$	a Gaussian distribution with mean $\mu_{r\|i}$ and covariance matrix $\Sigma_{r\|i}$
$S(X)$	normalized score
ζ	decision threshold
$y_k(\mathbf{x}_t)$	the k-th network output of EBFNs

w_{kj}	the output weight
$\Phi_j(\mathbf{x}_t)$	the elliptical basic function
M	the number of basic function
T_i	the decision threshold of the ith subnet in a PDBNN
$\phi(\mathbf{x}_t, \mathbf{w}_i)$	the discriminant function of the i subnet in a PDBNN
η_μ, η_σ, and η_t	user-assigned (positive) learning rates
$h_{r\|i}^{(j)}(t)$	the posterior probability
D_2^i	false rejection set
D_3^i	false acceptance set
$l(d)$	penalty function
$S(X_n)$	normalized segmental score
η_r and η_a	reinforced and anti-reinforced learning parameters
FRR	false rejection rate
FAR	false acceptance rate
N_{imp}, N_{spk}	the total number of training segments from impostors and registered speaker
γ_j	spread factor for the jth basic function in an EBFN
z_t	average normalized log-likelihood
$\Lambda^{Znorm}(X)$	Znorm score
$\Lambda^{Hnorm}(X)$	Hnorm score
μ and σ^2	mean and variance for Znorm
$\mu(HS(X))$ and $\sigma(HS(X))$	the handset-dependent normalization parameters
$p(\phi)$	probability density function of parameter vectors ϕ's
$\gamma_i(t)$	the posterior probability of mixture i
η	the metaparameter
$\hat{\mu}_{s,j}$	the extended mean vector of $\mu_{s,j}$

W^k	k-th MLLR adaptation matrix
A^k	the k-th transformation matrix
b^k	the k-th translation vector
$\Re^D$	real number vector space of D dimensions
$e(0), e(1), \ldots, e(K)$	$K+1$ eigenvectors
$w(1), \ldots, w(K)$	K eigenvoice coefficients
C	the DCT matrix
C^{-1}	the inverse DCT matrix

ABBREVIATION

PDBNNs	probabilistic decision-based neural networks
GMMs	gaussian mixture models
EBFNs	elliptical basic function networks
MLLR	maximum-likelihood linear regression
CMS	cepstral mean subtraction
Znorm	zero normalization
FRR	false rejection rate
FAR	false acceptance rate
DET	decision error tradeoff
PMC	parallel model combination
Hnorm	handset normalization
SMS	speaker model synthesis
SFT	stochastic feature transformation
BSFT	blind stochastic feature transformation
EER	equal error rate
LP	linear prediction
VQ	vector quantization
HMMs	hidden markov models
ML	maximum-likelihood
MAP	maximum a posteriori
MCE	minimum classification error
MMI	maximum mutual information
MLP	multiple layer perceptron

RBF	radical basic function
ROC	receiver operating characteristic
RBFNs	radial basic function networks
NTN	neural tree networks
pdf	probability density function
EM	expectation maximization
DBNN	decision-based neural network
LU	locally unsupervised
GS	globally supervised
K-NN	K-nearest neighbors
ACW	adaptive component weighting
RASTA	relative spectral processing
CDCN	codeword-dependent cepstral normalization
SDCN	SNR-dependent cepstral normalization
SNR	signal-to-noise ratio
Tnorm	test normalization
EMAP	extended MAP
MAPLR	maximum a posteriori linear regression
DRT	dimension reduction technique
SD	speaker-dependent
SI	speaker-independent
PCA	principle component analysis
MLED	maximal-likelihood eigenvoice decomposition
DCT	discrete cosine transform
DPMC	data-driven PMC
VTS	vector taylor series

MLE	maximum-likelihood estimation
MFCC	mel-frequency cepstral coefficients
SBR	signal bias removal
UBM	universal background model
DCF	decision cost function
CDF	cumulative density function

Chapter 1

INTRODUCTION TO SPEAKER RECOGNITION

Automatic speaker recognition [29, 42] is to recognize a speaker from his or her voice. Speaker recognition can be divided into *speaker identification* and *speaker verification*. Speaker identification is to determine the identity of an unknown speaker from a group of known speakers. Speaker verification is to verify a speaker's claimed identity based on his or her voice. A speaker claiming an identity is called a *claimant*, and an unregistered speaker posing as a registered speaker is an *imposter*. An ideal speaker verification system should not accept impostors as claimants (*false acceptances*) or reject registered speakers as impostors (*false rejections*).

Speaker recognition can also be divided into text-dependent and text-independent. In text-dependent systems, the same set of keywords is used for enrollment and recognition. In text-independent systems, on the other hand, the phrases or sentences used in verification could be different from those in enrollment. Text-dependent systems require user cooperation and typically use hidden Markov models to represent speakers' speech. Text-dependent systems 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 predefined keywords are not available and users are usually not cooperative or unaware of the recognition task.

1.1 *Components of Speaker Verification Systems*

Typically, a speaker verification system is composed of a front-end feature extractor, a set of client speaker models, a set of background speaker models, and a decision unit. Figure 1.1 illustrates the architecture of a typical speaker verification system.

The feature extractor derives speaker-specific information from speech signals. It is well known from the source-filter theory of speech production that speech spectra implicitly encode the vocal-tract shape information (e.g., length and cross-section area) of a speaker and that pitch harmonics encode the glottal source information. Speaker models are trained from the features extracted from clients' utterances. A set of background models are also trained using the speech of a large number of speakers to represent speaker-independent speech. Basically, the background models are used to normalize the scores of the speaker models in order to minimize non-speaker related variability such as acoustic noise and channel effect. To verify a claimant, speaker scores are normalized by the background scores and the resulting normalized score is compared with a decision threshold. The claimant is accepted (rejected) if the score is larger (smaller) than the threshold.

1.2 *Speaker-Dependent Features*

Although speech signals are nonstationary, their short-term segments can be considered quasi-stationary. Therefore, short-term spectral analysis can be applied to short

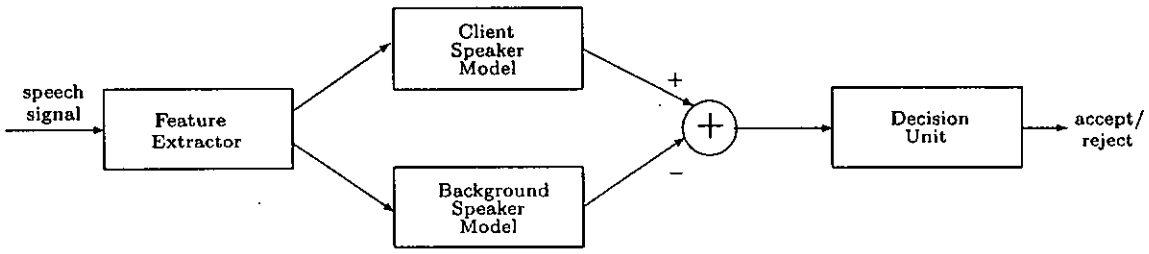


Figure 1.1: The architecture of a typical speaker verification system.

speech segments (typically 20-30ms), which results in a sequence of short-time spectra. In speech and speaker recognition, the short-time spectra are further transformed into features vectors.

In addition to spectral analysis, many speech and speaker recognition systems use linear prediction (LP) analysis [59] to extract feature vectors (known as LP coefficients) from short segments of speech waveforms. One advantage of LP coefficients is that they can be computed efficiently. More importantly, the LP coefficients represent the spectral envelopes of speech signals (information about the formant frequencies and their bandwidth). The spectral envelopes are characterized by the resonance frequencies, length, and spatially varied cross-section areas of the vocal-tract. Because all of these entities are known to be speaker-dependent, the LP coefficients are one of the candidate features for speaker recognition.

Several sets of features (e.g., LP coefficients, impulse responses, autocorrelation coefficients, cross-section areas, and cepstral coefficients) can be derived from LP analysis. Of particular interest is that a simple and unique relationship exists among these features. It has been shown that the cepstral coefficients are the most effective feature for speaker recognition [8].

1.3 Speaker Modeling

Over the years, a variety of speaker modeling techniques have been proposed [79]. This section describes four approaches to speaker modeling.

1.3.1 Template Matching

In template matching, reference templates are used as speaker models [23]. A template is composed of a sequence of feature vectors extracted from a set of fixed sentences uttered by a registered speaker. During recognition, an input utterance is dynamically aligned with the reference templates, and match scores are obtained by measuring the similarity between the aligned utterance and the templates. The use of fixed templates, however, cannot model the wide variability present in the speech signals.

1.3.2 Nearest Neighbors

Vector quantization is a coding technique typically used in transmitting signals at low bit rate. To use VQ for speaker recognition, a personalized codebook is created for each speaker. During recognition, an unknown speaker is identified by selecting the codebook whose code vectors are closest to the input vectors. Since its introduction by Soong [14] in 1985, VQ has been a benchmark method for speaker recognition systems [64], and improvement in the standard VQ approach has also been made [14]. The advantage of VQ is that the problem of segmenting speech into phonetic units can be avoided. Additionally, VQ is more computationally efficient than template matching. The disadvantage of VQ, however, lies in its crude approximation of the features' distribution.

1.3.3 Hidden Markov Models and Gaussian Mixture Models

Hidden Markov models (HMMs) encode both the temporal structure of feature sequences and the statistical variation of the features. Therefore, HMMs can be used as speaker models in text-dependent speaker recognition. The earliest attempt of using HMMs in speaker recognition was reported by Poritz [75]. After Portiz's work, several improved methods such as the mixture autoregressive HMMs [100], sub-word HMMs [34], and semi-continuous HMMs [27] have been proposed. HMMs parameters can be estimated based on maximum-likelihood (ML) or maximum a posteriori (MAP) criteria. Criteria that use discriminative training such as minimum classification error (MCE) [44, 66] or maximum mutual information (MMI) [73, 103] can also be adopted. Multi-state, left-to-right HMMs can be used as speaker- and utterance-specific models.

The HMM approach is similar to the VQ one in that the HMM states are found by a VQ-like procedure. However, unlike VQ, the probabilities of transition between states are encoded, and the order of presentation of the speech data is important. This may cause problems in text-independent speaker recognition where no temporal correlation exists between the training data and the test data. On the other hand, single-state HMMs (also known as Gaussian mixture models (GMMs)) [80, 87] can be applied to text-independent speaker recognition. Like VQ, the feature space of speakers is divided into a number of clusters. However, the likelihood function is continuous rather than discrete and the cluster membership is soft rather than hard. GMMs provide a probabilistic model for each speaker; but unlike HMMs, there is no Markov constraint among the sound classes. Therefore, the order of presentation of

speech data will not affect recognition decisions.

1.3.4 Neural Networks

Neural Networks can be considered as supervised classifiers that learn the complex mappings between data in the input space and the output decision space. This capability is particularly useful when the statistical distributions of data are not known. Neural network-based speaker models can have many forms, including multi-layer perceptrons (MLP) [54, 55], radical basis function (RBF) network [54–56, 74], hybrid MLP-RBF models [6], multi-expert connectionist models [11], and the modified neural tree networks [26].

To apply neural networks for speaker recognition, each speaker is assigned a personalized network that is trained to output a '1' for the voices associated with that speaker and a '0' otherwise. One advantage of using neural networks for speaker recognition is that discriminative information can be easily incorporated into the speaker models by means of supervised learning (See Section 2.1.1). This information can usually improve recognition performance but at the expense of computation resources.

1.4 Performance Evaluations

The Performance of speaker verification systems is usually specified by two types of errors:

1. *Miss rate* ($P_{\text{miss|target}}$)—the chance of misclassifying a true speaker as an impostor.

2. *False Alarm Rate* ($P_{\text{fa|nontarget}}$)—the chance of falsely identifying an impostor as a true speaker.

The miss rate and false alarm rate are also known as the false rejection rate (FRR) and the false acceptance rate (FAR), respectively. In addition to these two error rates, it is also common to report the equal error rate (EER)—the error rate at which $P_{\text{miss|target}} = P_{\text{fa|nontarget}}$.

Because the miss rate and false alarm rate depend on the decision threshold, a $\{P_{\text{miss|target}}, P_{\text{fa|nontarget}}\}$ pair represents one operating point of the system under evaluation. To provide more information about system performance, it is necessary to evaluate the system for a range of thresholds. This results in a *receiver operating characteristic* (ROC) curve, where the miss probability is plotted against the probability of a false alarm, similar to the one used by the face recognition community. However, the speaker verification community has chosen to use a variant of the ROC plots called *detection error tradeoff* (DET) plots [62]. In a DET plot, the axes' scales are normally deviated so that Gaussian distributed scores result in a straight line; the advantage is that systems with almost perfect performance can be compared easily.

In addition to DET curves, speaker verification systems are also compared based on the detection cost:

$$C_{\text{det}} = C_{\text{miss}} \times P_{\text{miss|target}} \times P_{\text{target}} + C_{\text{fa}} \times P_{\text{fa|nontarget}} \times P_{\text{nontarget}},$$

where C_{miss} and C_{fa} are the cost of making a false rejection error and false acceptance error, respectively, and where P_{target} and $P_{\text{nontarget}}$ are, respectively, the chance of having a true speaker and an impostor. Typical values of these figures are $C_{\text{miss}} =$

10, $C_{\text{fa}} = 1$, $P_{\text{target}} = 0.01$, and $P_{\text{nontarget}} = 0.99$ [63]. These values give an expected detection cost of approximately 1.0 for a system without any knowledge of the speakers. The operating point at which the detection cost C_{det} is at a minimum can be plotted on top of the DET curve (See Figure 7.4 for an example).

Because the performance of speaker verification systems depends on the amount of training data, acoustic environment, and the length of test segments, it is very important to report this information in any performance evaluations so that performance of different systems and techniques can be compared. Thus, the NIST established a common set of evaluation data and protocols [38] in 1996. Although only focusing on conversational speech, the NIST speaker recognition evaluations are one of the most important benchmark tests for speaker verification techniques.

1.5 Threshold Determination

Determination of decision thresholds is a very important problem in speaker verification. A large threshold could make the system annoying to users, while a small one could result in a vulnerable system. Conventional threshold determination methods typically compute the distribution of inter- and intra-speaker distances and then choose a threshold to equalize the overlapping area of the distributions [15, 28], i.e., to equalize the false acceptance rate (FAR) and the false rejection rate (FRR). The success of this approach, however, relies on whether the estimated distributions match the speaker- and impostor-class distributions. Another approach derives the threshold of a speaker solely from his or her own voice and speaker model [70]. Session-to-session speaker variability, however, could contribute significant bias to the threshold,

rendering the verification system unusable.

Due to the difficulty in determining a reliable threshold, researchers often report the equal error rate (ERR) of verification systems based on the assumption that an *a posteriori* threshold can be adjusted during verification. Real-world applications, however, are only realistic with *a priori* thresholds that should be determined before verification.

In recent years, research effort has focused on the normalization of speaker scores both to minimize error rates and to determine a reliable threshold. This includes the likelihood-ratio scoring proposed by Higgins et al. [36], where verification decisions are based on the ratio of the likelihood that the observed speech is uttered by the true speaker to the likelihood that it is spoken by an imposter. The a priori threshold is then set to 1, with the claimant being accepted (reject) if the ratio is greater (less) than the threshold. Subsequent work based on likelihood normalization [52, 65], cohort normalized scoring [90], and minimum verification error training [92] also shows that including an impostor model during verification not only improves speaker separability, but also allows decision thresholds to be easily set. Rosenberg and Parthasarathy [91] 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 Reynolds [88], who found that a gender-balanced, randomly selected impostor model performs better, suggesting that more work is required in this area.

Although these previous approaches help select an appropriate threshold, they may cause the system to favor rejecting true speakers, resulting in a high FRR. For

example, Higgins et al. [36] reported that the FRR is more than 10 times larger than the FAR. A recent report [24] based on a similar normalization technique but different threshold setting procedure also found that the average of FAR and FRR is about 3 to 5 times larger than the EER, suggesting that the EER could be an over optimistic estimate of the true system performance.

1.6 Speaker Recognition in Adverse Environments

It is well known that variation in acoustic environments seriously degrades the performance of speaker recognition systems. In particular, the performance of most systems degrades rapidly under adverse conditions such as in the presence of background noise, channel interference, handset variation, intersession variability, and long-term variability of speakers' voice. Speech signals can be affected by many sources of distortion. Among these sources, additive noise and convolutive distortion are the most common.

Additive noise can be classified into different categories according to their properties. For example, stationary noise such as air conditioners or fans has a time-invariant power spectral density, while nonstationary noise such as car passing, keyboard clicks, door slam, and the audio output of radios and TV sets has time-varying properties. Additive noise can also be short-live or continuous depending on their time duration relative to the speech signals.

In addition to additive noise, distortion could also be convolutional. For example, microphones and transmission channels can be considered as digital filters with which the speech signals are convolved. In particular, telephone channels exhibit a bandpass filtering effect on speech signals, where different degrees of attenuation are

exerted on different spectral bands. Reverberation of speech signals is another source of convolutive distortion. This type of distortion results in the addition of a noise component to the speech signals in the log-spectral domain.

Speakers may alter the way they speak under stress or a high level of background noise (known as the Lombard effect). During articulation, speakers may also produce breath noise and lip smacks. All of these distortions can cause severe performance degradation in speaker recognition systems.

This thesis addresses the problems caused by additive noise and convolutive distortion, which can be combined into a composite source of distortion. Specifically, the acquired signal $y(t)$ is expressed as

$$y(t) = s(t) * h(t) + n(t) \tag{1.1}$$

where $s(t)$ is the clean speech signal and $h(t)$, $n(t)$, and '$*$' represent the channel's impulse response, additive noise, and convolution operator, respectively.

1.7 Overview of the Dissertation

1.7.1 Motivations and Research Objectives

Blind deconvolution such as cepstral mean subtraction [28] has been extensively used for overcoming the problem of channel distortion. In this method, the long-term average of cepstral vectors is subtracted from each of the cepstral vectors in an utterance. However, the assumptions that the long-term average is a good estimate of channel characteristics and that the channel is linear are generally invalid. This is because telephone handsets typically exhibit energy-dependent frequency responses

[88], which suggests that linear filtering can only solve part of the problem.

This work aims to develop channel compensation algorithms for telephone-based speaker verification. The main objective is to circumvent the problem of handset variability by estimating and eliminating the nonlinear handset effects in telephone speech. Different channel compensation techniques will be studied in detail.

1.7.2 Organization of the Dissertation

This thesis is organized as follows. Chapter 2 explains and compares three different types of kernel-based probabilistic neural networks. The networks will be used as speaker models in a series of speaker verification experiments. Chapter 3 provides a brief account of the techniques that have been used extensively in addressing the problems of transducer mismatches and robustness in telephone-based speaker recognition. Chapter 4 focuses on model-based compensation and describes the state-of-the-art techniques in detail. Chapter 5 investigates two model adaptation/transformation techniques—reinforced/anti-reinforced learning of probabilistic decision-based neural networks (PDBNNs) and maximum-likelihood linear regression (MLLR)—in the context of telephone-based speaker verification. Chapter 6 focuses on feature-based compensation and highlights the state-of-the-art techniques that are related to this thesis. Chapter 7 proposes a blind compensation algorithm for solving the channel distortion problem. The algorithm is designed to handle the situation in which no a priori knowledge about the channel is available. Finally, a summary of the major findings is provided in Chapter 8.

Chapter 2

KERNEL-BASED NEURAL NETWORKS FOR SPEAKER VERIFICATION

Kernel-based neural networks such as probabilistic decision-based neural networks (PDBNNs), Gaussian mixture models (GMMs), and elliptical basis function networks (EBFNs) have been extensively used in speaker recognition. A common property of these networks is that they capture speaker characteristics via their kernel parameters. This chapter explains how these networks can be applied to speaker verification and compares their performance in terms of verification error rates and robustness against additive noise. It was found that GMM- and PDBNN-based speaker models outperform the EBFN ones in both clean and noisy environments . It was also found that the globally supervised learning of PDBNNs is able to find decision thresholds that minimize the variation in FAR, whereas the ad hoc threshold-determination approach used by the EBFNs and GMMs causes a large variation in FAR. This property makes the performance of PDBNN-based systems more predictable.

2.1 Kernel-Based Probabilistic Neural Networks

2.1.1 Incorporation of Nontarget Information

Early work on text-independent speaker verification used data from target speakers exclusively to train speaker models. One problem of this approach is that information from nontarget speakers (also known as antispeakers or background speakers) is not embedded in the speaker models, which may lead to suboptimal performance. A study in 1991 [36] showed that using nontarget information during verification can greatly enhance speaker verification performance. Nontarget information can be used during model training and recognition. For the former, supervised learning algorithms are used to train a speaker model that discriminates within-class data from out-of-class data. For the latter, likelihood ratio [36] or scoring normalization [90] is applied during recognition.

Neural networks are one of the techniques that can embed nontarget information in the speaker models. For example, the elliptical basis function networks (EBFNs) proposed in Mak and Kung [56] include the cluster centers of antispeakers' speech in their hidden layer. It was shown that EBFNs outperform radial basis function networks (RBFNs) and VQ. The neural tree networks (NTNs) are another type of network that use discriminative training, and research has shown that NTNs are superior to VQ in speaker recognition tasks [25].

Likelihood ratio [36] and scoring normalization that use cohort speakers [90] or the combination of cohort speakers and background speakers [107] have been applied to improve the performance of speaker verification systems. These scoring approaches

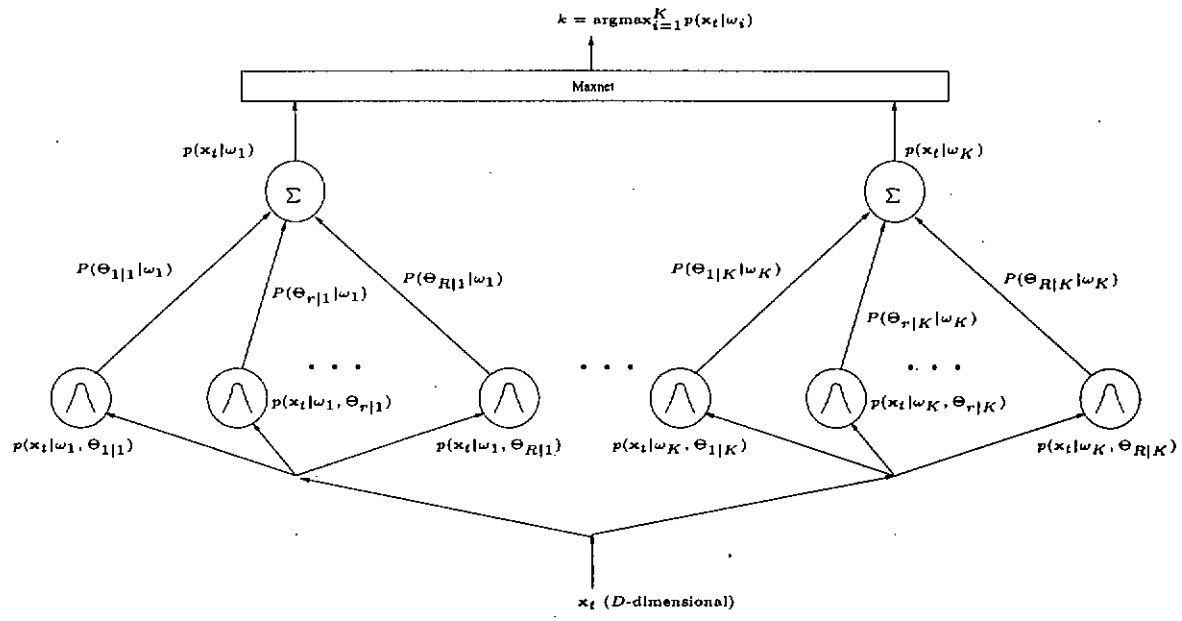


Figure 2.1: Architecture of a GMM-based classifier. Note that the classifier consists of K GMMs for a K-class classification task and that the Maxnet selects the GMM with the largest output as the identified class.

achieve high performance by constructing background models and/or cohort models, which should accurately represent the characteristics of all possible impostors. Typically, the background models and speaker models are trained separately, which means that discriminative information is used during recognition rather than during training.

2.1.2 Gaussian Mixture Models

Figure 2.1 depicts the architecture of a classifier in which each class is represented by a Gaussian mixture model (GMM). GMMs make use of semi-parametric techniques for approximating probability density functions (pdf). As shown in Figure 2.1, the output of a GMM is the weighted sum of R component densities. Given a set of N

independent and identically distributed patterns $X^{(i)} = \{\mathbf{x}_t; t = 1, 2, \ldots, N\}$ associated with class ω_i, we assume that the class likelihood function $p(\mathbf{x}_t|\omega_i)$ for class ω_i is a mixture of Gaussian distributions. More precisely, the likelihood function is expressed as

$$p(\mathbf{x}_t|\omega_i) = \sum_{r=1}^{R} P(\Theta_{r|i}|\omega_i) p(\mathbf{x}_t|\omega_i, \Theta_{r|i}), \tag{2.1}$$

where $\Theta_{r|i}$ represents the parameters of the rth mixture component, R is the total number of mixture components, $p(\mathbf{x}_t|\omega_i, \Theta_{r|i}) \equiv \mathcal{N}(\mu_{r|i}, \Sigma_{r|i})$ 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 $\Sigma_{r|i}$.

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 expectation-maximization (EM) algorithm [19]. More precisely, the parameters of a GMM are estimated iteratively by[1]

$$\mu_{r|i}^{(j+1)} = \frac{\sum_{t=1}^{N} P^{(j)}(\Theta_{r|i}|\mathbf{x}_t)\mathbf{x}_t}{\sum_{t=1}^{N} P^{(j)}(\Theta_{r|i}|\mathbf{x}_t)}, \tag{2.2}$$

$$\Sigma_{r|i}^{(j+1)} = \frac{\sum_{t=1}^{N} P^{(j)}(\Theta_{r|i}|\mathbf{x}_t)\left[\mathbf{x}_t - \mu_{r|i}^{(j+1)}\right]\left[\mathbf{x}_t - \mu_{r|i}^{(j+1)}\right]^T}{\sum_{t=1}^{N} P^{(j)}(\Theta_{r|i}|\mathbf{x}_t)}, \text{ and} \tag{2.3}$$

$$P^{(j+1)}(\Theta_{r|i}) = \frac{\sum_{t=1}^{N} P^{(j)}(\Theta_{r|i}|\mathbf{x}_t)}{N}, \tag{2.4}$$

where, T denotes matrix transpose, j denotes the iteration index, and $P^{(j)}(\Theta_{r|i}|\mathbf{x}_t)$ is the posterior probability of the rth mixture ($r = 1, \ldots, R$). The latter can be

[1]To simplify the notation, ω_i in Eqs. 2.2 though 2.6 has been dropped.

obtained by Bayes' theorem, yielding

$$P^{(j)}(\Theta_{r|i}|\mathbf{x}_t) = \frac{P^{(j)}(\Theta_{r|i})p^{(j)}(\mathbf{x}_t|\Theta_{r|i})}{\sum_{k=1}^{R} P^{(j)}(\Theta_{k|i})p^{(j)}(\mathbf{x}_t|\Theta_{k|i})} \tag{2.5}$$

in which

$$p^{(j)}(\mathbf{x}_t|\Theta_{r|i}) = \frac{1}{(2\pi)^{\frac{D}{2}}|\Sigma_{r|i}^{(j)}|^{\frac{1}{2}}} \exp\left\{-\frac{1}{2}\left[\mathbf{x}_t - \mu_{r|i}^{(j)}\right]^T (\Sigma_{r|i}^{(j)})^{-1}\left[\mathbf{x}_t - \mu_{r|i}^{(j)}\right]\right\}, \tag{2.6}$$

where D is the input dimension.

During recognition, a test vector $\mathbf{x}_t$ of an unknown class is fed to the GMMs, and the class index k is determined by the Maxnet:

$$k = \arg\max_{i=1}^{K} p(\mathbf{x}_t|\omega_i). \tag{2.7}$$

To apply GMMs to speaker verification, each registered speaker in the system is represented by a GMM. To enhance the discrimination between the client speakers and impostors, it is common practice to compute the ratio between the client likelihood and impostor likelihood, where the former is the output of the client's GMM and the latter is the output of a background model [36]. The background model is a GMM trained from the speech of a large number of speakers, who should accurately represent the characteristics of all possible impostors. Alternatively, a set of background models is formed during verification by selecting the GMMs of a small set of client speakers (cohort) whose acoustic characteristics are close to those of the claimant [90].

During verification, a sequence of feature vectors X from the claimant is extracted and the following normalized score is computed:

$$S(X) = \log p(X|\omega_s) - \log p(X|\omega_b), \tag{2.8}$$

where $p(X|\omega_s)$ and $p(X|\omega_b)$ are the GMMs' outputs (Eq. 2.1) corresponding to the speaker class ω_s and impostor class ω_b, respectively. The normalized score is then compared with a decision threshold to make a decision:

$$\text{If } S(X) \begin{cases} > \zeta & \text{accept the claimant} \\ \leq \zeta & \text{reject the claimant.} \end{cases} \tag{2.9}$$

That is, to adopt the GMM-based classifier shown in Figure 2.1 to speaker verification, K is set to 2 and Maxnet is changed to compute the log-likelihood difference.

2.1.3 Elliptical Basis Function Networks

Elliptical basis function (EBF) networks [56] are a type of feedforward neural network in which the hidden units evaluate the Mahalanobis distance between the input vectors and a set of vectors called function centers or kernel centers, and the outputs are a linear combination of the hidden nodes' outputs. More specifically, the k-th network output has the form

$$y_k(\mathbf{x}_t) = w_{k0} + \sum_{j=1}^{M} w_{kj}\Phi_j(\mathbf{x}_t), \tag{2.10}$$

where

$$\Phi_j(\mathbf{x}_t) = \exp\left\{-\frac{1}{2\gamma_j}(\mathbf{x}_t - \mu_j)^T \boldsymbol{\Sigma}_j^{-1}(\mathbf{x}_t - \mu_j)\right\}, \tag{2.11}$$

where μ_j and $\boldsymbol{\Sigma}_j$ are the function center (mean vector) and covariance matrix of the j-th basis function, respectively, w_{k0} is a bias term, and γ_j is a smoothing parameter controlling the spread of the j-th basis function.

Figure 2.2 illustrates the architecture of an EBF network with D inputs, M function centers, and K outputs. It clearly shows that EBF networks have a three-layer

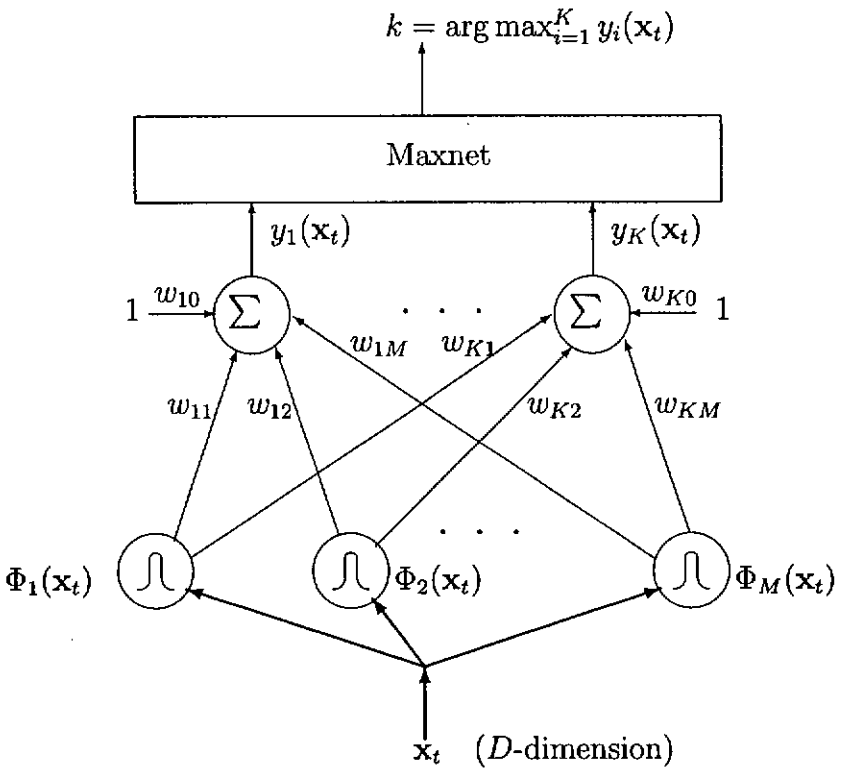


Figure 2.2: Architecture of a K-output EBF network.

architecture: input, hidden, and output layers. The input layer distributes the input patterns $\mathbf{x}_t$ to the hidden layer. Each hidden unit is a Gaussian basis function with shape and location defined by a center μ_j and a covariance matrix $\mathbf{\Sigma}_j$. The number of basis functions (M) is typically much less than that of training vectors.

The three-layer architecture with linear output units leads to a two-stage training procedure. In the first stage, the kernel parameters $\{\mu_j, \mathbf{\Sigma}_j\}_{j=1}^{M}$ in the hidden layer are determined by fast unsupervised learning (such as K-means clustering and sample covariance). In the second stage, the output weights $\{w_{kj}; k = 1, \ldots, K, j = 0, \ldots, M\}$ are determined by least-squares techniques. Because of this two-stage training approach, EBF networks have a shorter training time than the backpropagation net-

works.

The kernel parameters can also be determined in an iterative fashion using the EM algorithm [19]. For each iteration of EM, the mean vectors, covariance matrices, and mixture coefficients are updated according to Eqs. 2.2 through 2.6. It has been shown that using the EM algorithm to estimate the kernel parameters can produce networks that are superior to those using the K-means algorithm and sample covariance [56].

We can observe the similarity between GMM-based and EBFN-based classifiers from their architecture (Figures 2.1 and 2.2). For example, both of them compute the Mahalanobis distance (Eqs. 2.6 and 2.11) between the input vectors and the kernel centers in the hidden layer. However, there are two important differences. First, a GMM computes the likelihood of observing the input vector $\mathbf{x}_t$, whereas an EBF network maps data from the input space to the output space. Second, the kernel parameters of a GMM must be estimated from data derived from its corresponding class. On the other hand, data derived from all known classes (K classes in the case of Figure 2.2) are applied to estimate the kernel parameters of an EBF network. Even if the EBF kernels are divided into K groups and the K sets of kernel parameters are estimated independently using the data derived from the K classes, EBF networks are still different from GMMs in that each of the EBF network's outputs depends on the kernel outputs of the corresponding class as well as those from other classes. The output of a GMM ($p(\mathbf{x}_t|\omega_i)$), on the other hand, depends on the kernel outputs of its own class only. The consequent of this difference is that, for EBF networks, discrimination among all the known classes is considered during the training phrase, but for GMMs, class discrimination is introduced during the recognition phase.

To apply EBFNs to speaker verification, one EBFN is trained for each registered speaker. Specifically, each network is trained to recognize speech patterns from two classes: speaker class and antispeaker class. To achieve this, the hidden nodes are divided into two groups: one corresponds to the speaker class and the other to the antispeaker class. The former is denoted as the speaker kernels and the latter as the antispeaker kernels. The EM algorithm is applied independently to the speaker data and antispeaker data to obtain the speaker kernels and antispeaker kernels, respectively. Then least-squares techniques are applied to determine the output weights. Each network contains two outputs (i.e., $K = 2$ in Figure 2.2), with $y_1(\mathbf{x})$ giving a desired output of 1 and $y_2(\mathbf{x})$ giving a desired output of 0 for speaker's data, and vice versa for antispeakers' data.

During verification, speech patterns $X = \{\mathbf{x}_t; t = 1, \ldots, T\}$ are extracted from the claimant's utterance, and the following score is computed:

$$S(X) = \frac{1}{T} \sum_{t=1}^{T} \frac{\exp\{y_1(\mathbf{x}_t)/2P(\omega_s)\} - \exp\{y_2(\mathbf{x}_t)/2P(\omega_b)\}}{\exp\{y_1(\mathbf{x}_t)/2P(\omega_s)\} + \exp\{y_2(\mathbf{x}_t)/2P(\omega_b)\}}, \tag{2.12}$$

where $P(\omega_s)$ and $P(\omega_b)$ are the prior probabilities of the speaker class and antispeaker class, respectively, and $y_k(\mathbf{x})$ is the k-th output of the network. $P(\omega_s)$ and $P(\omega_b)$ can be easily computed by counting the number of speaker and antispeaker patterns in the training set. Note that dividing the network outputs by the prior probabilities is to rescale the network outputs so that the scaled averages (over the training set X' containing both speaker data and antispeaker data) are approximately equal to 0.5 (i.e., $\frac{1}{T'} \sum_{\mathbf{x} \in X'} y_k(\mathbf{x})/2P(\omega) \approx 0.5$, $\omega \in \{\omega_s, \omega_b\}$). The *softmax* function inside the summation of Eq. 2.12 is intended to prevent any extreme value of $y_k(\mathbf{x})/2P(\omega)$ from

dominating the average outputs. Verification decisions are based on the criterion:

$$\text{If } S(X) \begin{cases} > \zeta & \text{accept the claimant} \\ \leq \zeta & \text{reject the claimant,} \end{cases} \tag{2.13}$$

where $\zeta \in [-1, 1]$ is a speaker-dependent threshold.

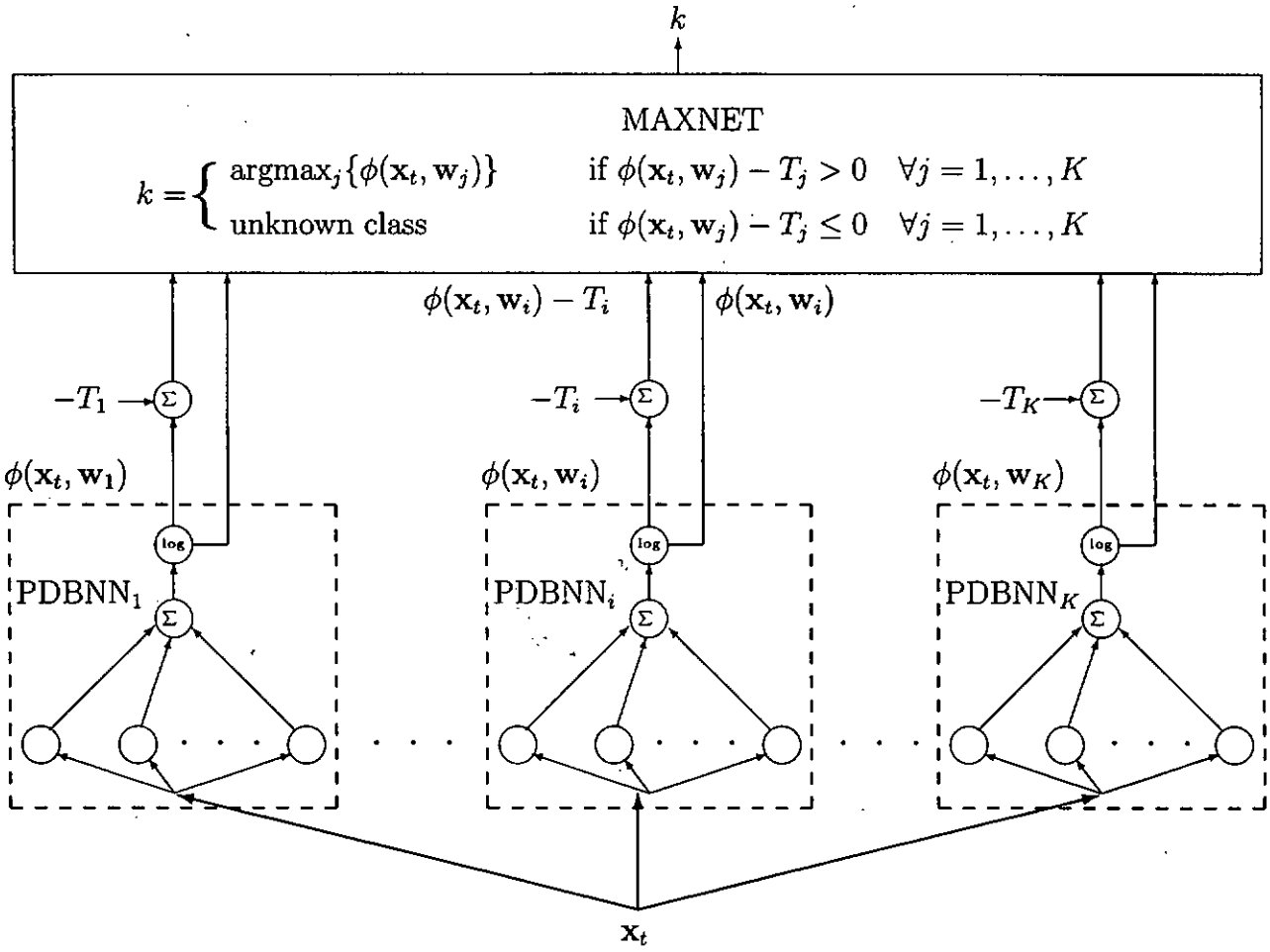


Figure 2.3: Structure of a PDBNN. Each class is modeled by a subnet. The subnet discriminant functions are designed to model the log-likelihood functions given by Eq. 2.14.

2.1.4 *Probabilistic Decision-Based Neural Networks*

Probabilistic decision-based neural networks (PDBNNs) [51] are a probabilistic variant of their predecessor—decision-based neural networks (DBNNs) [46]. Like DBNNs, PDBNNs employ a modular network structure shown in Figure 2.3. 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 form

$$\begin{aligned} \phi(\mathbf{x}_t, \mathbf{w}_i) &= \log p(\mathbf{x}_t|\omega_i) \\ &= \log \left[\sum_{r=1}^{R} P(\Theta_{r|i}|\omega_i) p(\mathbf{x}_t|\omega_i, \Theta_{r|i}) \right], \end{aligned} \tag{2.14}$$

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 i-th subnet.

Learning in PDBNNs is divided into two phases: locally unsupervised (LU) and globally supervised (GS). In the LU learning phase, PDBNNs adopt the EM algorithm to maximize the likelihood function

$$\begin{aligned} l(\mathbf{w}_i; X^{(i)}) &= \sum_{t=1}^{N} \log p(\mathbf{x}_t|\omega_i) \\ &= \sum_{t=1}^{N} \log \left[\sum_{r=1}^{R} P(\Theta_{r|i}|\omega_i) p(\mathbf{x}_t|\omega_i, \Theta_{r|i}) \right] \end{aligned} \tag{2.15}$$

with respect to the parameters $\mu_{r|i}$, $\Sigma_{r|i}$, and $P(\Theta_{r|i}|\omega_i)$, where $X^{(i)} = \{\mathbf{x}_t; t = 1, 2, \ldots, N\}$ denotes the set of N independent and identically distributed training patterns from class i. The EM algorithm leads to an iterative update procedure identical to Eqs. 2.2 through 2.6.

After the maximum-likelihood estimation, the PDBNN has one more learning phase—the GS learning, which minimizes the classification error. Specifically, when

misclassification occurs, reinforced learning and anti-reinforced learning will be performed on the target class and unduly class.

Assume that we have a set of training patterns $X = \{\mathbf{x}_t; t = 1, 2, \ldots, M\}$. Now further divide the data set X into (1) "the positive training set" $X^+ = \{\mathbf{x}_t; \mathbf{x}_t \in \omega_i, t = 1, 2, \ldots, N\}$ and (2) "the negative training set" $X^- = \{\mathbf{x}_t; \mathbf{x}_t \notin \omega_i, t = N + 1, N + 2, \ldots, M\}$. In PDBNN, an energy function is defined as

$$E = \sum_{t=1}^{M} l(d(t)), \tag{2.16}$$

where

$$d(t) = \begin{cases} T_i - \phi(\mathbf{x}_t, \mathbf{w}_i) & \text{if } \mathbf{x}_t \in \omega_i \\ \phi(\mathbf{x}_t, \mathbf{w}_i) - T_i & \text{if } \mathbf{x}_t \notin \omega_i, \end{cases} \tag{2.17}$$

where T_i is the threshold value and its initial value is set to zero. The penalty function l can be either a piecewise linear function

$$l(d) = \begin{cases} d & \text{if } d \geq 0 \\ 0 & \text{if } d < 0 \end{cases} \tag{2.18}$$

or a sigmoidal function

$$l(d) = \frac{1}{1 + e^{-d}}, \tag{2.19}$$

The use of sigmoidal function ensures that the energy function is not dominated by a single error.

In the globally supervised (GS) training phase, target values are used to fine-tune the decision boundaries. Specifically, for any patterns $\mathbf{x}_t$ not belong to the i-th class but are misclassified to the i-th class, or for any patterns belong to the i-th class but are misclassified to another class, reinforced and/or anti-reinforced learning are

applied to update the mean vectors and covariance matrices of subnet i. Thus, we have

$$\begin{aligned}
\mu_{r|i}^{(j+1)} &= \mu_{r|i}^{(j)} + \eta_\mu \sum_{t,\mathbf{x}_t \in \mathcal{D}_2^i} h_{r|i}^{(j)}(t) \Sigma_{r|i}^{-1(j)} \left[\mathbf{x}_t - \mu_{r|i}^{(j)}\right] \\
&\quad -\eta_\mu \sum_{t,\mathbf{x}_t \in \mathcal{D}_3^i} h_{r|i}^{(j)}(t) \Sigma_{r|i}^{-1(j)} \left[\mathbf{x}_t - \mu_{r|i}^{(j)}\right] \\
\Sigma_{r|i}^{(j+1)} &= \Sigma_{r|i}^{(j)} + \frac{1}{2}\eta_\sigma \sum_{t,\mathbf{x}_t \in \mathcal{D}_2^i} h_{r|i}^{(j)}(t) \left(H_{r|i}^{(j)}(t) - \Sigma_{r|i}^{-1(j)}\right) \\
&\quad -\frac{1}{2}\eta_\sigma \sum_{t,\mathbf{x}_t \in \mathcal{D}_3^i} h_{r|i}^{(j)}(t) \left(H_{r|i}^{(j)}(t) - \Sigma_{r|i}^{-1(j)}\right), \qquad (2.20)
\end{aligned}$$

where $H_{r|i}^{(j)}(t) = \Sigma_{r|i}^{-1(j)} \left[\mathbf{x}_t - \mu_{r|i}^{(j)}\right] \left[\mathbf{x}_t - \mu_{r|i}^{(j)}\right]^T \Sigma_{r|i}^{-1(j)}$, $h_{r|i}^{(j)}(t)$ is the posterior probability identical to Eq. 2.5, 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:

- $D_2^i = \{\mathbf{x}_t; \mathbf{x}_t \in \omega_i, \quad \mathbf{x}_t \text{ is misclassified to another class } \omega_j \}$
- $D_3^i = \{\mathbf{x}_t; \mathbf{x}_t \notin \omega_i, \quad \mathbf{x}_t \text{ is classified to } \omega_i\}$

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

$$T_i^{(j+1)} = \begin{cases} T_i^{(j)} - \eta_t l'(T_i^{(j)} - \phi(\mathbf{x}_t, \mathbf{w}_i)) & \text{if } \mathbf{x}_t \in \omega_i \quad \text{(reinforced learning)} \\ T_i^{(j)} + \eta_t l'(\phi(\mathbf{x}_t, \mathbf{w}_i) - T_i^{(j)}) & \text{if } \mathbf{x}_t \notin \omega_i \quad \text{(anti-reinforced learning)}, \end{cases} \qquad (2.21)$$

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. The PDBNN algorithm tries to minimize the classification error at each epoch and is not guaranteed to converge.

In this work, three modifications were made to the PDBNN's training algorithm to make PDBNNs appropriate for speaker verification. First, the original PDBNNs used

one threshold per network. However, in this work, one network was used to model the speaker class and another one to model the antispeaker class (i.e., $i = 1$ or 2 in Eq. 2.14 and $L = 2$ in Figure 2.3). To make PDBNNs applicable to speaker verification, the likelihood computation was modified such that only one threshold is required. Specifically, instead of comparing the network's log-likelihood against its corresponding threshold, as in the original PDBNNs, a normalized score was compared against a single decision threshold, as in Eqs. 2.8 and 2.9.

The second modification changes the frequency of updating the decision threshold. The original PDBNN adopts the batch-mode supervised learning (see Eq. 2.20). In this work, a sequential update mode was adopted. Specifically, the GS training was modified as follows: Let X_n be the n-th segment extracted from the speaker's speech patterns $X^{(s)}$ or from antispeakers' speech patterns $X^{(b)}$, the normalized segmental score is computed by evaluating

$$\begin{aligned} S(X_n) &= S_s(X_n) - S_b(X_n) \\ &= \frac{1}{T} \sum_{\mathbf{x} \in X_n} \{\phi_s(\mathbf{x}) - \phi_b(\mathbf{x})\}, \end{aligned} \tag{2.22}$$

where $\phi_s(\mathbf{x})$ and $\phi_b(\mathbf{x})$ are the log-likelihood function (Eq. 2.14) of the speaker class and antispeaker (background) class, respectively. For the n-th segment, the following criteria were used to determine whether to update the decision threshold $\zeta_n^{(j)}$:

$$\text{If } S(X_n) \begin{cases} > \zeta_{n-1}^{(j)} \text{ and } X_n \in X^{(s)} & X_n \text{ is correctly classified, no need to update} \\ \leq \zeta_{n-1}^{(j)} \text{ and } X_n \in X^{(b)} & X_n \text{ is correctly classified, no need to update} \\ > \zeta_{n-1}^{(j)} \text{ and } X_n \in X^{(b)} & \text{false acceptance, need to update} \\ \leq \zeta_{n-1}^{(j)} \text{ and } X_n \in X^{(s)} & \text{false rejection, need to update} \end{cases} \tag{2.23}$$

where $\zeta_{n-1}^{(j)}$ is the decision threshold of the PDBNN speaker model after learning from segment X_{n-1} at epoch j. Therefore, whenever misclassification occurs, the threshold $\zeta_{n-1}^{(j)}$ is updated according to

$$\zeta_n^{(j)} = \begin{cases} \zeta_{n-1}^{(j)} - \eta_r l'(\zeta_{n-1}^{(j)} - S(X_n)) & \text{if } X_n \in X^{(s)} \text{ and } S(X_n) < \zeta_{n-1}^{(j)} \\ \zeta_{n-1}^{(j)} + \eta_a l'(S(X_n) - \zeta_{n-1}^{(j)}) & \text{if } X_n \in X^{(b)} \text{ and } S(X_n) \geq \zeta_{n-1}^{(j)} \end{cases} \tag{2.24}$$

where η_r and η_a are reinforced and anti-reinforced learning parameters, respectively (more on this in the next paragraph), $l(d) = \frac{1}{1+e^{-d}}$ is a penalty function, and $l'(d)$ is the derivative of $l(\cdot)$.

In the third modification, a new method was introduced to compute learning rates. In the original PDBNNs, the learning rates for optimizing the thresholds are identical for both reinforced and antireinforced learning. However, in some situations, there may be many false acceptances and only a few false rejections (or vice versa), which means anti-reinforced learning will occur more frequently than reinforced learning (or vice versa). To reduce the imbalance in the learning frequency, the reinforced (anti-reinforced) learning rate η_r (η_a) is made proportional to the rate of false rejections (acceptance) weighted by the total number of impostor (speaker) segments:

$$\eta_r = \frac{\mathrm{FRR}^{(j-1)}}{\mathrm{FAR}^{(j-1)} + \mathrm{FRR}^{(j-1)}} \frac{N_{\mathrm{imp}}}{N_{\mathrm{imp}} + N_{\mathrm{spk}}} \eta \tag{2.25}$$

$$\eta_a = \frac{\mathrm{FAR}^{(j-1)}}{\mathrm{FAR}^{(j-1)} + \mathrm{FRR}^{(j-1)}} \frac{N_{\mathrm{spk}}}{N_{\mathrm{imp}} + N_{\mathrm{spk}}} \eta, \tag{2.26}$$

where $\mathrm{FRR}^{(j-1)}$ and $\mathrm{FAR}^{(j-1)}$ represent the false rejection rate and false acceptance rate at epoch $j-1$, respectively; and N_{imp} and N_{spk} represent the total number of training segments from the impostors and the registered speaker, respectively; and η is a positive learning parameter. The first term of Eqs. 2.25 and 2.26 increases the

learning rate if the corresponding error rate is large, which has the effect of rapidly reducing the corresponding error rate. The second term weights the learning rate according to the proportion of training segments in the opposite class, which has the effect of reducing the learning rate of the frequent learner and increasing the learning rate of the nonfrequent learner. This arrangement can prevent reinforced learning or anti-reinforced learning from dominating the learning process and aims to increase the convergence speed of the decision threshold.

2.2 Applications to Speaker Verification

2.2.1 Speech Corpus and Feature Extraction

The YOHO corpus [41], collected by ITT Defense Communication Division, was used in this work. Yoho is a large-scale, scientifically controlled speech corpus for testing speaker verification systems at high confidence level. The corpus features "combination lock" phrases, 138 speakers (108 male, 30 female), intersession variability, and high-quality telephone speech sampled at 8kHz with 16 bits per sample. The recording system of YOHO was set up in the corner of a large office. Low level noise could be heard from adjoining offices. A handset containing an omnidirectional electret microphone without noise-canceling features was used for recordings. There are four enrollment sessions for each speakers. Each of these sessions contains 24 utterances. Likewise, there are ten verification sessions for each speaker, with each session containing four utterances. Each utterance is composed of three 2-digit numbers (e.g. 34-52-67). The combination-lock phrases together with intersession variability make

YOHO ideal for speaker verification research.

In this work, all of the 138 speakers in the YOHO corpus have been used for experimental evaluations. Gaussian white noise with different noise power was also added to the clean YOHO corpus. Both the clean and noisy YOHO corpora were used in the evaluations.

The feature extraction procedure is as follows. For each utterance, the silent regions were removed by a silent detection algorithm based on the energy and zero crossing rate of the signals. The remaining signals were preemphasized by a filter with transfer function $1 - 0.95z^{-1}$. Twelfth-order LP-derived cepstral coefficients were computed using a 28ms Hamming window at a frame rate of 71Hz.

2.2.2 Enrollment Procedures

In the verification experiments, each registered speaker was represented by three different speaker models (GMM, EBFN, and PDBNN). A GMM-based speaker model consists of two GMMs, one representing the individual speaker and the other representing all other speakers (called antispeakers hereafter). An EBFN- or PDBNN-based speaker model consists of a single EBFN or PDBNN representing the corresponding registered speaker as well as all antispeakers. For each registered speaker, all utterances in the four enrollment sessions corresponding to the speaker and a predefined set of antispeakers (each speaker has his/her own set of antispeakers) were used to train a speaker model. The speaker model was trained to recognize the speech derived from two classes: speaker class and antispeaker class. To this end, two groups of kernel functions (one group representing the client speaker and the other representing

the speakers in the antispeaker class) were assigned to each speaker model. Hereafter, we denote the group corresponding to the speaker class as the speaker kernels and the one corresponding to the antispeaker class as the antispeaker kernels. For each registered speaker, a unique antispeaker set containing 16 antispeakers was created. The 16 antispeakers were randomly selected from the speaker set excluding the true speaker. Speech features derived from this set were subsequently used to estimate the antispeaker kernels by using the EM algorithm. The antispeaker kernels enable us to integrate scoring normalization [53] into the speaker models, which enhances the models' capability in discriminating the true speakers from the impostors.

Each of the GMMs and PDBNNs is composed of 12 inputs (12th-order cepstral coefficients were used as features), a predefined number of kernels, and one output. On the other hand, the EBFNs contain 12 inputs, a predefined number of kernels, and 2 outputs with each output representing one class (speaker class and antispeaker class).

We applied the K-means algorithm to initialize the initial positions of the speaker kernels. Then, the kernels' covariance matrices were initialized by the K-nearest neighbors algorithm ($K = 2$). 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 fine-tune the mean vectors, covariance matrices, and mixture coefficients (see Eqs. 2.2 through 2.6). All of the covariance matrices are diagonal. The same procedure was also applied to determine the mean vectors and covariance matrices of the antispeaker kernels, using the speech data derived from the

antispeaker set. It was found that initializing the mean vectors by K-means and the covariance matrices by K-NN reduces the number of EM iterations required to determine the maximum-likelihood solution. Because the K-means and K-NN algorithms run much faster than the EM algorithm, this approach can reduce the training time considerably.

The process for constructing a PDBNN-based speaker model involves two phases: locally unsupervised (LU) training and globally supervised (GS) training. The LU training phase is identical to the GMM training described in Section 2.1.2. In the GS training phase, the speaker's enrollment utterances and the utterances from all enrollment sessions of the antispeakers were used to determine a decision threshold (see Section 2.2.4).

For EBFN-based speaker models, the speaker kernels and antispeaker kernels obtained from GMM training were combined to form a hidden layer. In this work, γ_j in Eq. 2.11 was determined heuristically by

$$\gamma_j = \frac{9}{5} \sum_{k=1}^{5} \|\mu_k - \mu_j\| \tag{2.27}$$

where μ_k denotes the k-th nearest neighbor of μ_j in the Euclidean sense. We have empirically found that using five nearest centers and multiplying the resulting average distance by 9 give reasonably good results. However, no attempts have been made to optimize these values. Finally, singular value decomposition was applied to determine the output weights. Details of the enrollment procedure for EBFNs can be found in [56].

2.2.3 *Verification Procedures*

Verification was performed using each speaker in the YOHO corpus as a claimant, with 64 impostors being randomly selected from the remaining speakers (excluding the antispeakers and the claimant) and rotating through all speakers. For each claimant, the feature vectors of the claimant's utterances from his or her 10 verification sessions in YOHO were concatenated to form a claimant sequence. Likewise, the feature vectors of the impostor's utterances were concatenated to form an impostor sequence.

Verification Procedures for PDBNNs and GMMs

For PDBNNs and GMMs, the following steps were performed during verification. The feature vectors from the claimant's speech $\mathcal{T}^c = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{T_c}\}$ was divided into a number of overlapping segments containing $T(< T_c)$ consecutive vectors as shown below[1]

$$\overbrace{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_6, \cdots\cdots, \mathbf{x}_T}^{\text{1st segment, } \tau_1}, \mathbf{x}_{T+1}, \mathbf{x}_{T+2}, \cdots, \mathbf{x}_{T_c}$$

$$\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5, \overbrace{\mathbf{x}_6, \cdots\cdots, \mathbf{x}_{T+5}}^{\text{2nd segment, } \tau_2}, \mathbf{x}_{T+6}, \cdots, \mathbf{x}_{T_c}$$

For the t-th segment ($\mathcal{T}_t \subset \mathcal{T}^c$), the average normalized log-likelihood

$$z_t = \frac{1}{T} \sum_{\mathbf{x} \in \mathcal{T}_t} \{\phi_S(\mathbf{x}) - \phi_A(\mathbf{x})\} \tag{2.28}$$

of the PDBNN-based and GMM-based speaker models was computed, where $\phi_S(\mathbf{x})$ and $\phi_A(\mathbf{x})$ represent the log-likelihood function (Eq. 2.15) of the speaker model and

[1]The claimant can be the true speaker (in which case $\mathcal{T}^c$ represents a claimant sequence) or he/she can be an impostor (in which case $\mathcal{T}^c$ represents an impostor sequence).

antispeaker model, respectively. Verification decisions were based on the criterion:

$$\text{If } z_t \begin{cases} > \zeta & \text{accept the claimant} \\ \leq \zeta & \text{reject the claimant} \end{cases} \tag{2.29}$$

where ζ is a speaker-dependent decision threshold (see Section 2.2.4 for the procedure of determining ζ). A verification decision was made for each segment, with the error rate (either false acceptance or false rejection) being the proportion of incorrect verification decisions to the total number of decisions. In this work, T in Eq. 2.28 was set to 500 (i.e., 7 seconds of speech), and each segment was separated by five vector positions. More specifically, the t-th segment contains the vectors

$$\mathcal{T}_t = \{\mathbf{x}_{5(t-1)+1}, \mathbf{x}_{5(t-1)+2}, \cdots, \mathbf{x}_{5(t-1)+500}\}, \tag{2.30}$$

where $5(t-1)+500 < T_c$. Note that dividing the vector sequence into a number of segments has also been successfully used in [56, 87] for increasing the number of decisions.

Verification Procedures for EBFNs

For the EBF-based speaker models, verification decisions were based on the difference between the scaled network outputs [56]. Because the ratio of training vectors between the speaker class and antispeaker class is about 1 to 16, the networks favor the antispeaker class during verification by always giving an output close to 1 for the antispeaker class and close to 0 for the speaker class. This problem can be solved by scaling the outputs during verification so that the new average outputs are approximately equal to 0.5 for both classes. This can be achieved by dividing the output

$y_k(\mathbf{x})$ by $2P(\omega_k)$, where $P(\omega_k)$ is the prior probability of class ω_k. Specifically, we compute the scaled output $\tilde{y}_k(\mathbf{x}) = \frac{y_k(\mathbf{x})}{2P(\omega_k)}$ with $k = 1, 2$ so that $\frac{1}{N}\sum_{\mathbf{x}\in X}\tilde{y}_k(\mathbf{x}) \approx 0.5$, where N is the number of training vectors in the training set X. A simple way to estimate the prior probability $P(\omega_k)$ is to divide the number of patterns in class ω_k by the total number of patterns in the training set.

Similar to the PDBNN-based and GMM-based speaker models, we divided the claimant's utterance $\mathcal{T}^c$ into a number of overlapping segments. For each segment $\mathcal{T}_t$ (with segment length T), the scores

$$z_{t,k} = \frac{1}{T}\sum_{\mathbf{x}\in\mathcal{T}_t}\frac{\exp\{\tilde{y}_k(\mathbf{x})\}}{\sum_{r=1}^{2}\exp\{\tilde{y}_r(\mathbf{x})\}} \quad k = 1, 2 \tag{2.31}$$

corresponding to the speaker and antispeaker classes were computed. Note that we have made use of the softmax function inside the summation of Eq. 2.31. Its purpose is to ensure that $z_{t,k}$ is in the range $[0, 1]$ and that $\sum_k z_{t,k} = 1$, thereby preventing any extreme value of $\tilde{y}_k(\mathbf{x})$ from dominating the average outputs.

Verification decisions were based on the criterion:

$$\text{If } z_{t,1} - z_{t,2} \begin{cases} > \zeta & \text{accept the claimant} \\ \leq \zeta & \text{reject the claimant} \end{cases} \tag{2.32}$$

where $\zeta \in [-1, 1]$ is a speaker-dependent threshold (see Section 2.2.4) controlling the false rejection rate (FRR) and the false acceptance rate (FAR). Again, a verification decision was made for each segment (as defined in Eq. 2.30) at a rate of one decision per five feature vectors. Computing the difference between the two outputs is equivalent to normalizing the score in GMMs. Thus, scoring normalization was intergrated into the network architecture.

In this work, equal error rate (EER)—false acceptance rate being equal to false rejection rate—was used as a performance index to compare the verification performance among different speaker models. Because the speaker models remain fixed once they have been trained, EER can be used to compare the models' ability in discriminating speaker features from impostor features.

2.2.4 Determination of Decision Thresholds

As mentioned before, it is necessary to determine a decision threshold for each speaker model during enrollment. These thresholds will be used during verification.

The procedures for determining the decision thresholds of PDBNNs, GMMs, and EBFNs are different. For the GMM and EBFN speaker models, the utterances from all enrollment sessions of 16 randomly selected antispeakers were used for threshold determination. Specifically, these utterances were concatenated and the verification procedures were applied. The thresholds ζ's in Eqs. 2.29 and 2.32 were adjusted until the corresponding FAR fell below a predefined level; in this work, the level was set to 0.5%. Antispeakers' utterances, rather than a speaker's utterances, were used because it is easier to collect the speech of a large number of antispeakers. Hence, the thresholds obtained are more reliable than those that would have been obtained from the speaker's speech. In addition, using a predefined FAR to determine the decision thresholds makes it easier to predict the robustness of the verification system against impostor attacks [108].

The modified threshold determination procedure described in Section 2.1.4 was used to determine PDBNNs' decision thresholds. To keep the mean vectors and

covariance matrices the same as the maximum-likelihood estimates, only the decision thresholds were adjusted during globally supervised training, with the mean vectors and covariance matrices remain unchanged.

2.2.5 Pilot Experiments

The architecture of GMMs, EBFNs, and PDBNNs depends on several free parameters, including the number of speaker kernels, the number of antispeaker kernels, and the number of antispeakers for finding the antispeaker kernels. To determine these parameters, a series of pilot experiments involving 30 speakers from the YOHO corpus were performed. Equal error rates (EERs) were used as the performance indicator. To determine an appropriate number of speaker kernels, speaker models with different numbers of speaker kernels were constructed, and the numbers of antispeaker kernels and antispeakers were fixed to 160 and 16, respectively. Table 2.1(a) shows the average EERs obtained by the GMM-based speaker models. Evidently, the EER decreases as the number of kernels increases. The decrease in EER becomes less significant after the number of speaker kernels reaches 40.

To determine an appropriate number of antispeakers for determining the antispeaker kernels, we varied the number of antispeakers while fixing the number of speaker kernels and antispeaker kernels to 40 and 160, respectively. Table 2.1(b) shows the average EER obtained by the GMM-based speaker models. Optimal performance is obtained when the number of antispeakers is 32. In order to reduce processing time, we used 16 antispeakers in the rest of the experiments.

We have also varied the number of antispeaker kernels while fixing the number

Number of Speaker's Kernels	EER (%)
10	2.78
20	1.51
40	0.77
80	0.57
160	0.48

(a)

Number of Antispeakers	EER (%)
4	2.02
8	1.30
16	0.77
32	0.48
64	0.81

(b)

Number of Antispeaker Kernels	EER (%)
40	0.83
80	0.83
160	0.77
320	0.75
640	0.79

(c)

Table 2.1: Average equal error rates based on 30 GMM-based speaker models with different numbers of (a) speaker kernels, where the number of antispeakers and the number of antispeaker kernels were set to 16 and 160, respectively; (b) antispeakers, where the number of speaker kernels and antispeaker kernels were set to 40 and 160, respectively; and (c) antispeaker kernels, where the number of speaker kernels and antispeakers were set to 40 and 16, respectively.

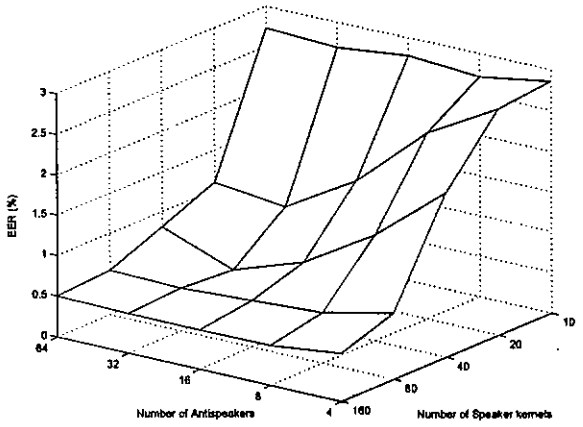


Figure 2.4: EER surfaces plot.

of speaker kernels and the number of antispeakers to 40 and 16, respectively. Table 2.1(c) shows the average EER obtained by the GMM-based speaker models. The results show that no significant reduction in error rate can be achieved when the number of antispeaker kernels reaches 160. Therefore, 160 antispeaker kernels were used in subsequent experiments.

Table 2.1(c) shows that the error rate is fairly insensitive to the number of antispeaker kernels provided that the number is sufficiently large. To determine the sensitivity with respect to the number of speakers's kernels and the number of antispeakers, we plot the error surface in Figure 2.4. This Figure suggests that the larger the number of speaker's kernels and antispeakers, the better the performance.

Information criterion such as Bayesian Information Criterion (BIC) [94] can also be used to determine the model order. To use BIC, a regularization term is added to the log-likelihood function to penalize the complexity of the model.

Because the EBFNs, GMMs, and PDBNNs use the same set of kernels, it is not necessary to repeat the above experiments for EBFNs and PDBNNs.

2.2.6 Large-Scale Experiments

Based on the results in Section 2.2.5, we set the number of speaker kernels and the number of antispeaker kernels to 40 and 160, respectively; and for each speaker model, we used 16 antispeakers for determining the parameters in the antispeaker kernels. Note that we have selected a suboptimal number of antispeakers to reduce the computation time in creating the speaker models.

All speakers (108 male, 30 female) in the YOHO corpus were used to evaluate the performance of EBFNs, GMMs, and PDBNNs in close-set text-independent speaker verification. To demonstrate the robustness of these classifiers, speech from the enrollment sessions of the YOHO corpus was used for training while speech from the verification sessions was used for testing.

Table 2.2 summarizes the average FAR, FRR, and EER obtained by the PDBNN-, GMM- and EBFN-based speaker models. All results are based on the average of 138 speakers in the YOHO corpus. The results, in particular the EERs, demonstrate the superiority of the GMMs and PDBNNs over the EBFNs. The EER of GMMs and PDBNNs are the same because their kernel parameters are identical. Table 2.2 shows that the EER obtained by the EBFNs is greater than that of the GMMs and PDBNNs.

In terms of FAR and FRR, Table 2.2 demonstrates the superiority of the threshold determination procedure of PDBNNs. In particular, Table 2.2 clearly shows that the globally supervised learning of PDBNNs can maintain the average FAR at a very low level during verification, whereas the ad hoc approach used by the EBFNs and GMMs produces a much larger average FAR. Recall from our previous discussion that the

Speaker Model	FAR (%)	FRR (%)	EER (%)
GMMs	8.01	0.08	0.33
EBFs	15.24	0.50	0.48
PDBNNs	1.10	1.87	0.33

Table 2.2: Average error rates achieved by the GMMs, EBFNs, and PDBNNs based on 138 speakers in the YOHO corpus. The decision thresholds for GMMs and PDBNNs were determined by setting the predefined FAR to 0.5%; whereas, the thresholds for PDBNNs were determined by reinforced learning (see Section 2.2.4).

predefined FAR for determining the decision thresholds of EBFNs and GMMs was set to 0.5%. The average FAR of EBFNs and GMMs are, however, very different from this value. This suggests that it may be difficult to predict the performance of the EBFNs and GMMs in detecting impostor attacks.

Figure 2.5 depicts the FAR and FRR of individual speakers in the GMM-, EBFN- and PDBNN-based speaker verification systems. Evidently, most of the speakers in the PDBNN-based system exhibit a low FAR. On the other hand, the GMMs and EBFNs exhibit a much larger variation in FAR. We conjecture that the globally supervised learning in PDBNNs is able to find decision thresholds that minimize the variation in FAR.

Figure 2.6 shows the DET curves [62] corresponding to Speaker 164 for different types of speaker models. In the DET plots, we use a nonlinear scale for both axes so that systems producing Gaussian distributed scores will be represented by straight lines. This property helps spread out the receiver operating characteristics (ROCs), making comparison of well-performed systems much easier. Note that the DET curves

for the GMM and PDBNN are identical in this experiment because the globally supervised training updates the decision thresholds only. It is evident from Figure 2.6 that the GMM- and PDBNN-based speaker models outperform the EBFN one.

2.2.7 Compared with Related Work

There are several speaker verification evaluations based on the YOHO corpus in the literature. For examples, Reynolds et al. [81] obtained a 0.51% EER and Higgins et al. [36] achieved a 1.7% EER. Both systems are based on GMMs. Note that the performance of our system (0.33% EER) is better that of [81] and [36]. However, these error rates can only be loosely compared with each other because the evaluations were not performed under identical conditions (different training/testing paradigms and background speaker sets).

To compare with a more classical approach, we have repeated the same experiment using Vector Quantization (VQ) speaker models [96]. Using 64-center VQ speaker codebooks,[2] we obtained an EER of 1.29%. When the number of code vectors per VQ codebook reduces to 32, the EER increases to 1.61%. These EERs are much higher than that of the PDBNNs and GMMs.

It is also important to point out that EER is not the only criterion for judging speaker verification performance. In practical situations, we also need to consider the tradeoff between false acceptance and false rejection. The key finding of this work is that the PDBNNs can effectively control this tradeoff through their threshold determination mechanism.

[2]The number of centers in VQ codebooks must be a power of 2.

	PDBNN/GMM		EBFN	
	Train	Test	Train	Test
EER(%)	4.12	24.61	6.86	27.17

Table 2.3: Performance of the PDBNN, GMM, and EBFN in the 2-D speaker verification problem.

2.3 Comparison of Decision Boundaries

To further illustrate the difference among the three kernel-based probabilistic neural networks, let us compare the decision boundaries in a speaker verification problem using 2-D speech features. Details of the speaker verification experiments can be found in Section 2.2. We extracted the first and second cepstral coefficients of Speaker 162 and those of his antispeakers and impostors from the YOHO corpus to create a set of two-dimensional (2-D) speech data. Similar to the enrollment procedure in the speaker verification experiments, a PDBNN, a GMM, and an EBFN (all with 2 inputs and 6 centers) were trained to classify the patterns into two classes. Therefore, except for the reduction in feature dimension, the training methods, learning rate, and verification methods are identical to the speaker verification experiments described in Section 2.2.

Table 2.3 compares the performance of the three speaker models, and Figure 2.7 shows the test data, decision boundaries, function centers, and contours of basis function outputs formed by these models. The decision boundaries are based on the equal error thresholds obtained from the data set. It is evident from Figure 2.7(a) that the decision boundaries formed by the EBFN enclose two regions, which belong to the speaker class, with a large amount of test data, whereas, the complement

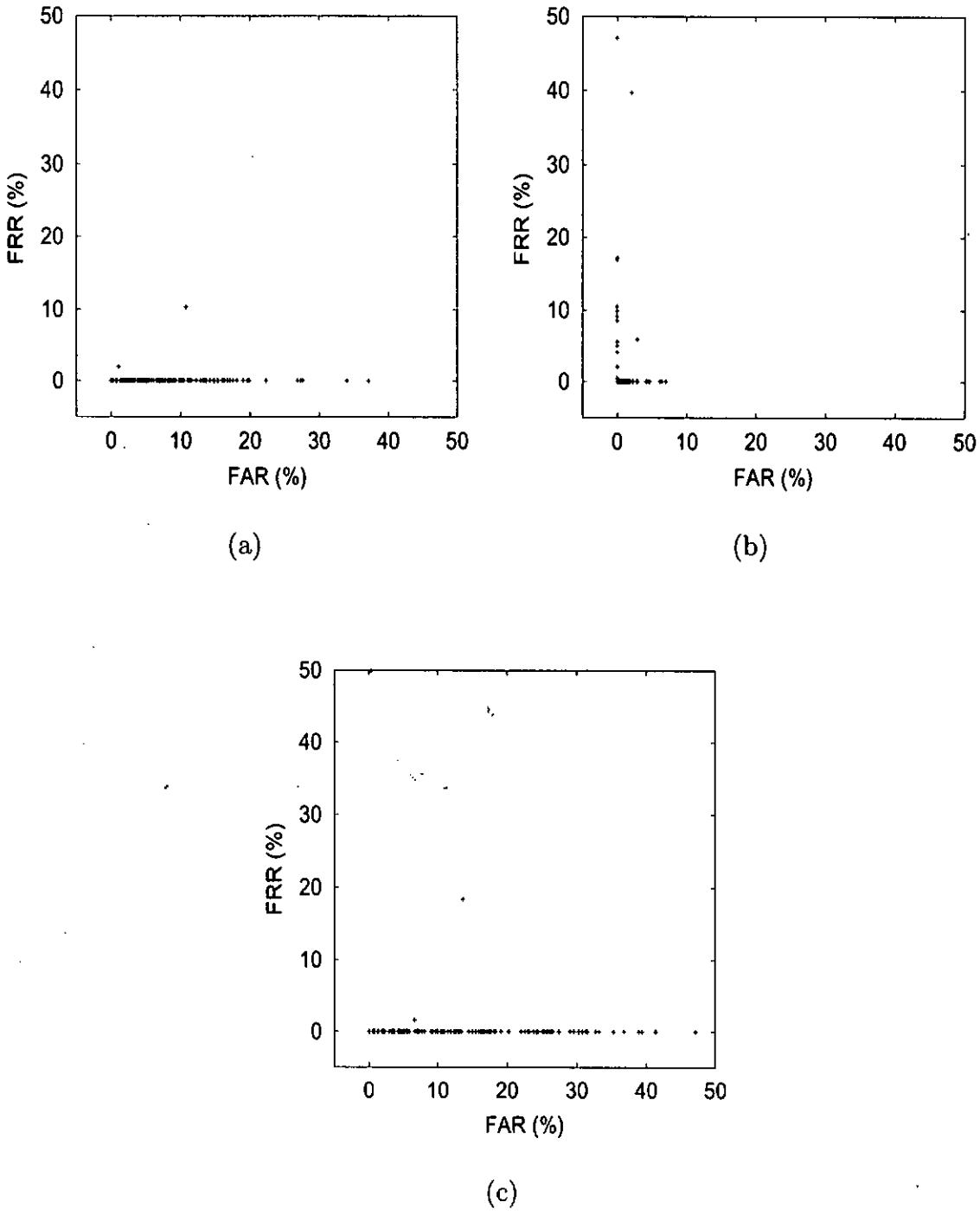


Figure 2.5: FRRs versus FARs (during verification) of 138 speakers using (a) GMMs, (b) PDBNNs, and (c) EBFNs as speaker models.

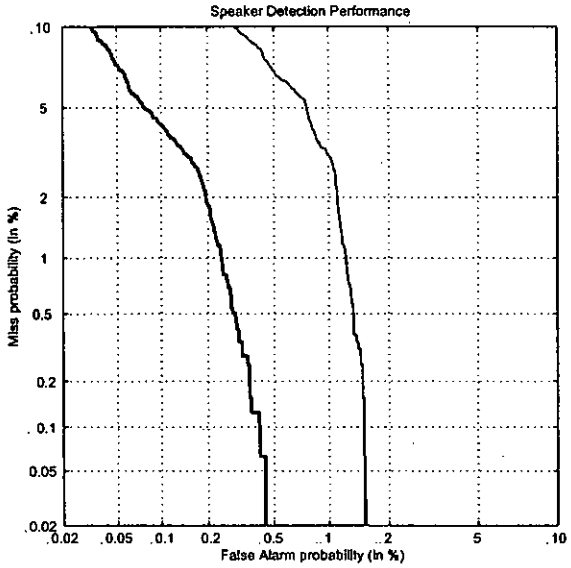


Figure 2.6: DET curves corresponding to Speaker 164. Thick curve: EBFN-based speaker model. Thin curve: GMM-based and PDBNN-based speaker models.

region, which belongs to the impostor class, extends to infinity. On the other hands, the decision boundaries created by the GMM and PDBNN extend to infinity in the feature space for both speaker class and impostor class. Both the decision boundaries (Figure 2.7) and the EER (Table 2.3), suggest that the GMM and PDBNN provide better generalization than the EBFN. These results also agree with what we have found in Table 2.2. The poor performance in EBFNs may be caused by the least squares approach to finding the output weights. Because the EBFNs formulate the classification problem as a function interpolation problem (mapping from the feature space to 0 or 1), overfitting will easily occur if there are too many hidden nodes but too few training samples.

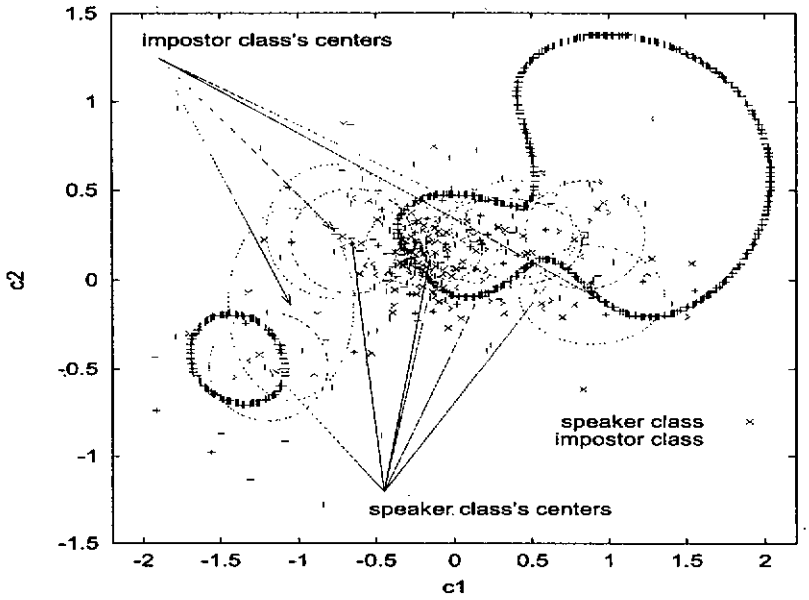
The aim of this Section is to illustrate the decision boundaries on a 2-dimensional space. We have used 6 speaker kernels and 6 anti-speaker kernels for the speaker mod-

els. No attempts have been made to optimize these parameters. Hence, a sub-optimal performance for elliptical basic function networks and Gaussian mixture models is possible. The inferior performance of the RBFNs may be caused by their inherent characteristics—attempt to minimize the desired outputs and actual outputs. This can easily lead to overfitting. To illustrate this situation, we have trained an RBFN with 30 centers and 2 GMMs, each which 15 centers, to solve a binary classification problem. As shown in Figure 2.8, the decision boundaries created by RBFN attempt to fit most of the data, whereas those created by the GMMs exhibit better generalization.

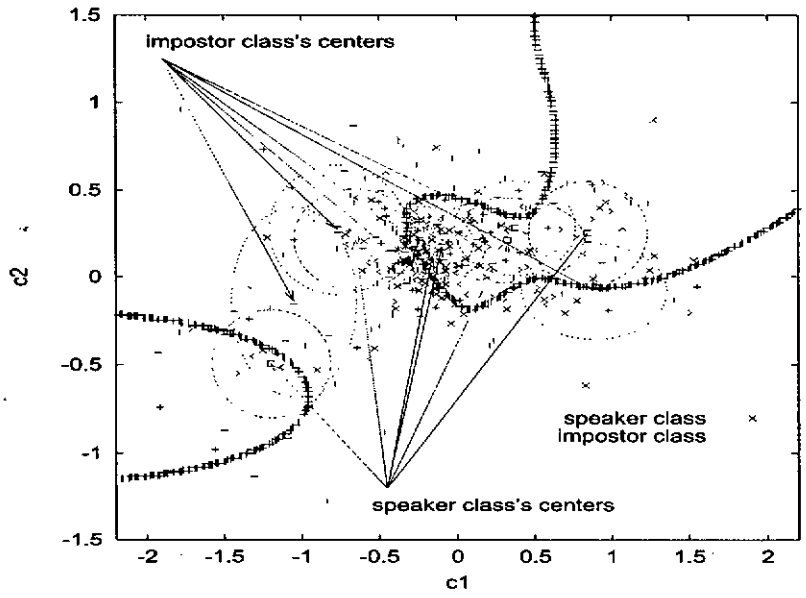
To achieve the best generalization, it is necessary to optimize the complexity of the model. The concept of bias-variance trade-off [12] can provide considerable insight into this phenomenon. In bias-variance decomposition, the generalization error is decomposed into bias and variance components. A too simple, or too inflexible model will have a large bias. On the other hand, a too flexible model (too flexible in relation to a particular training data) will have a large variance. The best generalization is obtained when optimum balance between the bias and variance can be established.

2.4 Robustness Against Noise

One of the main challenges in speaker recognition is to recognize speakers in adverse conditions. Noise is commonly considered as an additive component to the speech signals. Speaker models trained by using clean speech signals are usually subject to performance degradation in noisy environments. This section compares the speaker verification performance of the kernel-based speaker models described in Section 2.1



(a)



(b)

Figure 2.7: Speaker verification problem using 2-D speech features. The figures depict the decision boundaries, function centers, and contours of constant basis function outputs (thin ellipses) produced by (a) EBFNs and (b) GMMs and PDBNNs. The $\times$ and $+$ signs represent the speaker's data and impostors' data, respectively.

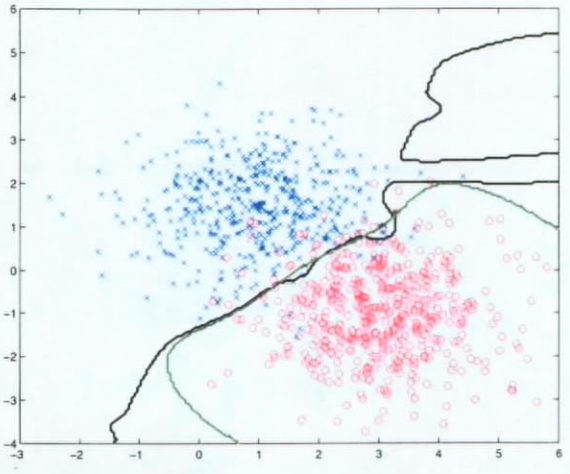


Figure 2.8: Decision boundary created by RBFNs (black) and GMMs (green).

under clean and noisy environments.

To test the robustness of different speaker models against additive noise, zero-mean Gaussian noise was added to the YOHO speech so that the resulting corrupted speech has an SNR of 10dB, 6dB, and 3dB. Segmentation files[3] derived from the clean YOHO corpus were used to specify the silence/speech regions of the corrupted YOHO speech. In practice, the same speech detector that we applied to YOHO speech should also be used to segment the corrupted YOHO speech. Our objective, however, is to compare the robustness of different speaker models against additive noise. Therefore, using the same segmentation files to define the speech regions of both clean and corrupted speech prevents the error introduced by the speech detector from interfering our comparison.

Tables 2.4 shows the average FAR, FRR, and EER obtained by the GMM-,

[3]These files only specify the silence and speech regions of the utterances. A speech detection algorithm based on zero-crossing rate and average amplitude was used to determine the speech regions.

PDBNN- and EBFN-based speaker models under different SNRs. The results show that the error rates of all models increase as the noise power increases. Such performance degradation is mainly caused by the mismatches in training and testing environments. Additive white noise contaminates the speech signals and therefore changes their acoustic characteristics. The speaker models, which were trained with clean speech, produced a reasonably low error rate for speech with a high SNR. However, their performance degraded rapidly when they were applied to noisy speech. Evidently, the EERs of PDBNNs and GMMs are smaller than that of the EBFNs under all SNRs. Although PDBNNs and GMMs provide better generalization, the performance of PDBNNs and GMMs are still unacceptable at low SNRs. In addition to additive noise, telephone speech may also be distorted by handsets and telephone channels. These issues are addressed in Chapters 5 and 7.

In order to investigate the effect of intermittent noise, machine-gun noise from the noisex-92 corpus was added to the YOHO corpus to produce corrupted speech with signal-to-noise ratio (SNR) of 10dB, 7dB, 4dB, and 1dB. The SNR is defined by

$$\text{SNR} = 10\log_{10}\frac{\sigma_s^2}{\sigma_n^2}, \tag{2.33}$$

where σ_s^2 and σ_n^2 represent the average energy of signal and noise, respectively.

In the experiment, GMMs were used as speaker and background models. The EERs corresponding to the machine-gun noise experiments (see Table 2.5) are much lower than those corresponding to the white-noise experiments. Because machine-gun noise is a kind of intermittent noise, not the whole utterance was corrupted by the noise. On the other hand, the white noise corrupted the whole utterance and caused

SNR	FAR (%)	FRR (%)	EER (%)
3 dB	43.52	54.91	27.30
6 dB	42.51	53.59	20.32
10 dB	41.20	50.70	12.79
clean	8.01	0.08	0.33

(a)

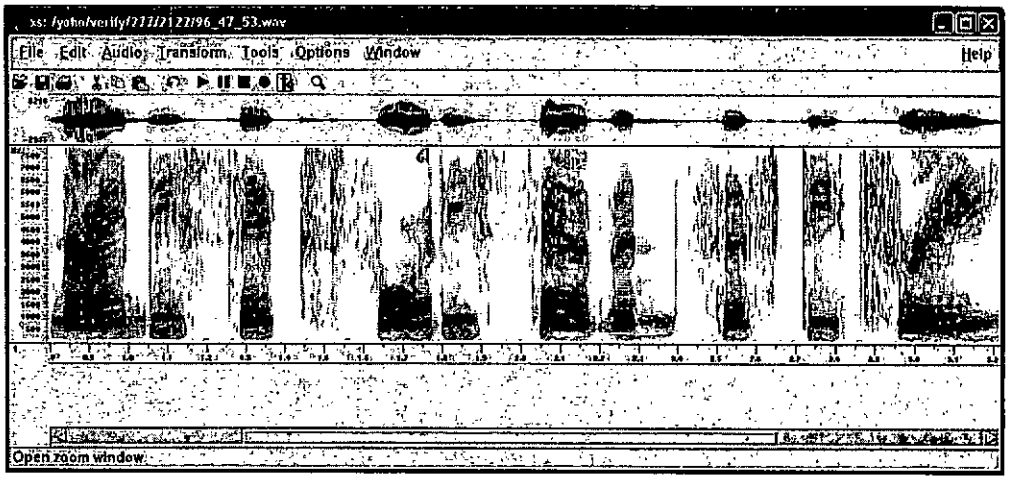
SNR	FAR (%)	FRR (%)	EER (%)
3 dB	19.52	77.53	27.30
6 dB	17.03	77.53	20.32
10 dB	13.67	76.38	12.79
clean	1.10	1.87	0.33

(b)

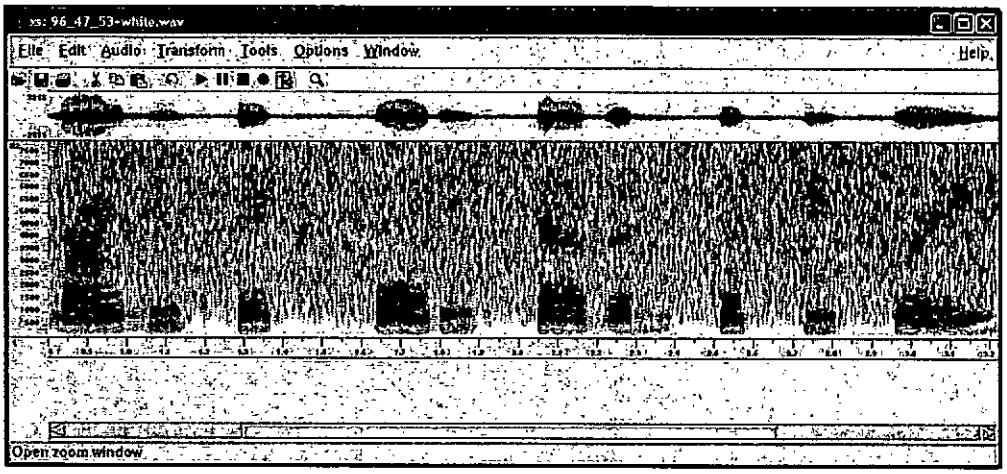
SNR	FAR (%)	FRR (%)	EER (%)
3 dB	30.48	65.91	30.32
6 dB	29.97	65.16	22.45
10 dB	29.22	61.06	14.58
clean	15.24	0.50	0.48

(c)

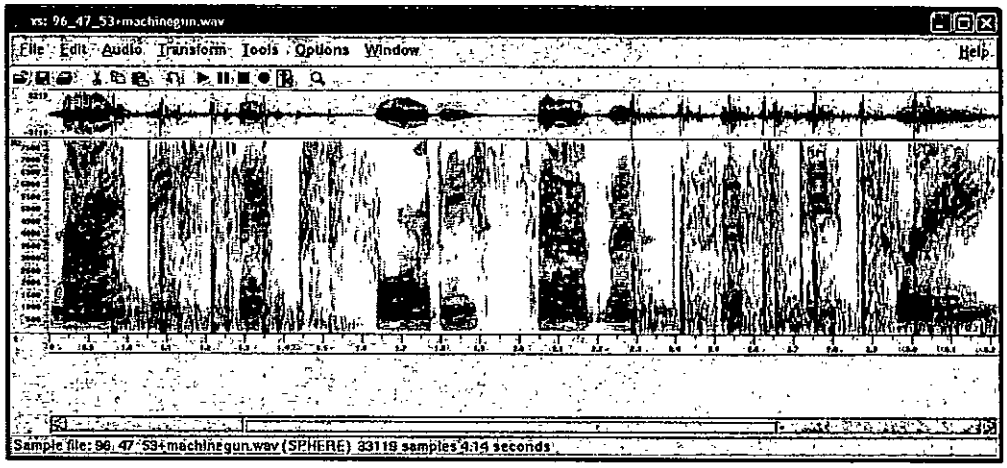
Table 2.4: Average error rates obtained by (a) GMMs, (b) PDBNNs, and (c) EBFNs at different signal-to-noise ratios in the white-noise experiments.



(a)



(b)



(c)

Figure 2.9: (a) Original clean speech signals. (b) Speech signals corrupted by white noise. (c) Speech signals corrupted by machine-gun noise.

SNR	FAR (%)	FRR (%)	EER (%)
1 dB	2.49	28.65	2.00
4 dB	2.39	22.17	1.89
7 dB	2.36	16.19	1.77
10 dB	2.43	11.41	1.63
clean	15.24	0.50	0.48

Table 2.5: Average error rates obtained by GMMs at different signal-to-noise ratios in the machine-gun noise experiments.

a more severe distortion (cf. Figure 2.9(b) and Figure 2.9(c)).

2.5 Conclusions

This chapter addresses the problem of building a speaker verification system using kernel-based probabilistic neural networks. The modeling capability and robustness of these pattern classifiers are compared. Experimental results based on 138 speakers and the visualization of decision boundaries suggest that GMM- and PDBNN-based speaker models outperform the EBFN ones. Results also show that the proposed modifications to the PDBNN's supervised learning algorithm not only make PDBNNs amenable to speaker verification tasks but also make their performance more predictable. This work also found that PDBNNs and GMMs are more robust than EBFNs in recognizing speakers in noisy environments.

Chapter 3

TECHNIQUES FOR ROBUST SPEAKER VERIFICATION

In recent years, a great deal of effort has been spent on the problems of transducer mismatches and robustness in telephone-based speaker recognition. These efforts have resulted in five types of techniques: channel equalization, feature transformation, model transformation, background noise compensation, and score normalization. Channel equalization estimates undistorted features or features that are insensitive to channel variation from channel-distorted speech signals. Feature-based compensation transforms channel-distorted speech features to fit clean speaker models, whereas model-based compensation adapts or transforms the parameters of clean models to fit new acoustic environments. Background noise compensation aims to reduce the background noise in noisy speech or to minimize the effect of additive noise on the feature vectors. Score-based compensation aims to minimize environment-dependent bias by normalizing the distribution of speaker scores. This chapter provides a brief account of these techniques. Further details of individual techniques will be elaborated in subsequent chapters.

3.1 Channel Equalization

Channel equalization aims to derive noise-resistant feature parameters rather than to remove the noise. Typically, weak or no assumptions are made on the noise and the clean models can be used directly without retraining. However, it is impossible to use the characteristics specific to a particular noise type because noise statistics are not explicitly estimated.

There are two main schools of thought in channel equalization: intraframe processing and interframe processing.

3.1.1 Intraframe Processing

This school of thought looks at the local spectral characteristics of a given frame of speech. This has led to some influential methods such as cepstral weighting [101], bandpass liftering [43], and more recently adaptive component weighting (ACW) [7] and pole-zero postfiltering [110].

Cepstral weighting emphasizes the reliable components in feature vectors and suppresses the components that are susceptible to noise and channel variation. ACW 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 the former assumes that all speech frames are subject to the same distortion, whereas the latter does not require this assumption. Pole-zero postfiltering is based on the postfiltering technique commonly used in speech enhancement [78]. Although it can be considered as a special kind of cepstral weighting, the weighting procedure is defined in terms of a transfer function.

It has been shown that ACW performs significantly better than cepstral weighting in a telephone-based speaker identification task [7] and that the performance of ACW and pole-zero postfiltering are comparable [110]. However, these experiments have limitations, as the speech corpora (KING and TIMIT with channel simulators) they used do not allow for proper examination of handset variability. Effort must therefore be made to investigate whether these approaches are robust to handset variation.

3.1.2 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 (CMS) [8], pole-filtered cepstral mean subtraction [69], delta cepstrum [28], and relative spectral (RASTA) processing [35].

In cepstral mean subtraction, the stationary channel effects are compensated for by subtracting the long-term average of the cepstral vectors from the channel-distorted cepstrum. Pole-filtered cepstral mean subtraction extends this idea one step further. Instead of subtracting the cepstral mean, this method subtracts 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 has been shown that the average of these modified LP filters is a better estimate of the channel as compared to the long-term average of the cepstral vectors [69].

Although the mean-subtraction approaches can significantly improve performance

in "mismatched" conditions, considerable loss of recognition accuracy is experienced when they are used in "matched" conditions. This is due to the implicit assumption of the mean-subtraction approaches: the long-term cepstral mean of clean speech is zero. The requirement of long-term averages 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 [28]. 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 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 passband. Therefore, delta processing introduces more modification to the linguistic components of speech than RASTA processing. While cepstral mean subtraction is nothing more than removing the DC component of a vector sequence, RASTA processing recursively removes the temporal average, making the current output depends on its past.

3.1.3 Limitations of Interframe and Intraframe Processing

Although the preceding methods have been successfully applied to reduce channel mismatches, they have their own limitations. For example, all of them assume that the channel can be approximated by a linear filter. Most telephone handsets, however, exhibit energy-dependent frequency responses [88] for which linear filtering may be a poor approximation. Reynolds [82] also found that current techniques can only compensate for part of the mismatches. Therefore, a more complex representation of handset characteristics is required.

3.2 Feature Transformation

This approach aims to make the classifiers more robust by compensating the distortion in the feature domain during the classification stage. This is typically achieved by applying an affine transformation to bring the test data closer to the training data [60]. 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. This approach has the advantage that both convolutional distortion and additive noise can be compensated simultaneously. Interestingly, the cepstral mean subtraction, pole-filtered cepstral subtraction, and cepstral weighting are special cases of affine transformation [60].

Codeword-dependent cepstral normalization (CDCN) [1] is another feature-based approach that can handle both channel distortion and background noise. In CDCN,

additive noise and convolutive distortion are modeled as codeword-dependent cepstral biases. The CDCN, however, only works well when the background noise level is low.

When stereo corpora are available, channel distortion can be estimated directly by comparing the clean features against their corresponding distorted features. For example, in SNR-dependent cepstral normalization (SDCN) [1], cepstral biases for different signal-to-noise ratios are estimated in a maximum-likelihood framework. In probabilistic optimum filtering [71], transformation is performed by combining the outputs of a set of multi-dimensional least-square filters. These methods, however, rely on the availability of stereo corpora, which could be difficult to obtain.

The requirement of stereo corpora can be avoided by making use of the information embedded in clean speech models. For example, in stochastic matching [93], the cepstral biases and affine transformation matrices are determined by maximizing the likelihood function of the clean models given the transformed data. The linear transformation in [93] can be replaced by a neural network to compensate for non-linear distortion [98]. However, closed form solutions can no longer be obtained and the generalized EM algorithm is required.

3.3 Model Adaptation and Transformation

Model-based approaches attempt to modify the clean speech models such that the density functions of the resulting models fit the distorted data better. The advantage of this approach is that no assumption about the speech signals is required and that modifying the testing data is not necessary. Influential model-based approaches include (1) stochastic matching [93] and stochastic additive transforms [89], where

the models' means and variances are adjusted by stochastic biases; (2) maximum-likelihood linear regression (MLLR) [49], where the mean vectors of clean speech models are linearly transformed; and (3) the constrained reestimation of Gaussian mixtures [21], where both mean vectors and covariance matrices are transformed. Recently, MLLR has been extended to maximum-likelihood linear transformation [32], in which the transformation matrices for the variances can be different from those for the mean vectors. Meanwhile, the constrained transformation in [21] has been extended to piecewise-linear stochastic transformation [20], where a collection of linear transformations are shared by all the Gaussians in each mixture. The random bias in [93] has also been replaced by a neural network to compensate for nonlinear distortion [98]. All of these extensions show improvement in recognition accuracy.

Because the preceding methods "indirectly" adjust the model parameters via a small number of transformations, they may not be able to capture the fine structure of the distortion. While this limitation can be overcome by the Bayesian techniques [39, 48], where model parameters are adjusted "directly", the Bayesian approach requires a large among of adaptation data to be effective. Because both direct and indirect adaptations have their own strengths and weaknesses, a natural extension is to combine them so that the two approaches can complement each other [67, 95].

3.4 Background Noise Compensation

It has been shown that about 40% of telephone conversations contain competing speech, music, or traffic noise [22]. This figure suggests the importance of background noise compensation in telephone-based speaker recognition. Early approaches include

spectral subtraction [13] and projection-based distortion measure [61]. More recently, statistical-based methods such as noise integration model [89] and signal bias removal [77] 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 because clean speech is usually not available.

3.5 Joint Additive and Convolutional Noise Compensation

There have been several proposals aimed at addressing the problem of convolutional distortion and additive noise simultaneously. In addition to the affine transformation and CDCN mentioned before, these proposals include stochastic pattern matching [93], parallel model combination [33], state-based compensation for continuous-density hidden-Markov models [4], and maximum-likelihood estimation of channels' autocorrelation functions and noise [109]. Although these techniques have been successful in improving speech recognition performance, caution must be taken when they are applied to speaker recognition. This is because adapting a speaker model to new environments will affect its capability in recognizing speakers [10].

3.6 Score Normalization

In speaker verification, the distributions of client scores and impostor scores could have large variation. Score normalization techniques have been proposed to reduce score variation. The basis idea is to normalize the verification scores by using mean and variance that are estimated from impostor-score distribution. The three common ways to compute the impostor-score distribution are Znorm, Hnorm, and Tnorm.

Znorm

In Znorm [50], each speaker model is tested against a set of impostors utterances, resulting in an impostor-score distribution. The speaker-dependent normalization parameters—mean μ and variance σ^2—are estimated from the impostor-score distribution. During recognition, these parameters are then applied to the log-likelihood ratio score $\Lambda(X)$ as follows:

$$\Lambda^{Znorm}(X) = \frac{\Lambda(X) - \mu}{\sigma}. \tag{3.1}$$

Hnorm

Handset-score normalization (Hnorm) [83] estimates the handset-dependent normalization parameters (biases and scales) by testing a claimant model against handset-dependent utterances produced by impostors. This can avoid collecting a large amount of speaker-specific utterances because test utterances from impostors can be easily obtained. The resulting normalization parameters are speaker- and handset-specific. During recognition, the handset type $HS(X)$ of the test utterance X is detected by a handset detector. Hnorm is then applied to the log-likelihood ratio score $\Lambda(X)$ as follows:

$$\Lambda^{Hnorm}(X) = \frac{\Lambda(X) - \mu(HS(X))}{\sigma(HS(X))}, \tag{3.2}$$

where $\mu(HS(X))$ and $\sigma(HS(X))$ represents the handset-dependent normalization parameters. To avoid bimodal distributions due to different genders, the parameters were derived using utterances of impostors, which are of the same gender as the claimant.

Tnorm

Another score normalization technique, namely, test normalization (Tnorm) [9], uses impostor models instead of impostors' utterances to calculate the normalization parameters during verification. For a given test utterances, impostors models are used to calculate impostors' log-likelihood scores and the normalization parameters are estimated. Unlike Hnorm, the same utterance is used for computing speaker scores and for estimating normalization parameters; as a result, acoustic mismatches between the speaker scores and normalization parameters can be avoided.

Chapter 4

MODEL ADAPTATION AND TRANSFORMATION

Environmental robustness is an important issue in telephone-based speaker verification because users of speaker verification systems tend to use different handsets in different situations. Different handsets may introduce different degrees of distortion to the speech signals. It has been noticed that recognition accuracy degrades dramatically when users use different handsets for enrollment and verification. This lack of robustness with respect to handset variability makes speaker verification over telephone networks a challenging task.

This chapter focuses on model-based compensation and describes the state-of-the-art techniques in detail. Techniques that are related to this thesis will also be emphasized.

4.1 Maximum a Posteriori Adaptation

With some modifications, standard speaker adaptation methods, such as maximum a posteriori (MAP) [48], can be used to adapt clean speaker models to suit a new acoustic environment. Specifically, given observation data $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_T\}$, the MAP estimate is expressed as follows:

$$\hat{\phi} = \arg\max_{\phi} p(\phi|\mathbf{X}) = \arg\max_{\phi} p(\mathbf{X}|\phi)p(\phi), \tag{4.1}$$

where $p(\phi)$ is a probability density function of parameter vectors ϕ's.

In most practical situations, only a limited amount of adaptation data is available for adaptation. The MAP approach can deal with this data-sparsity problem effectively by making use of the prior distribution of clean models. The parameters of the clean models are adjusted to fit the adaptation data subject to the constraint imposed by the prior density of the model's parameters. The prior information prevents inappropriate parameters modification unless the adaptation data provide strong evidence.

The MAP re-estimation formula for the means of a GMM $\Lambda = \{\omega_i, \mu_i, \Sigma_i\}_{i=1}^{M}$ is given by

$$\hat{\mu}_i = \frac{\eta\mu_i + \sum_{t=1}^{T} \gamma_i(t)\mathbf{x}_t}{\eta + \sum_{t=1}^{T} \gamma_i(t)} \qquad i = 1, \ldots, M, \tag{4.2}$$

where $\gamma_i(t)$ is the posterior probability of mixture i and η is a metaparameter controlling the contribution of the prior mean μ_i and the adaptation data $\mathbf{x}_t$ on the adapted mean $\hat{\mu}_i$. When the adaptation data is sparse, the prior mean has a strong influence on the adapted mean. On the other hand, when the amount of adaptation data increases, the MAP estimate converges to the maximum-likelihood estimate, a phenomenon known as asymptotic convergence.

The disadvantage of MAP adaptation is that its rate of convergence is slow, especially when the clean models have a large number of free parameters. The effectiveness of MAP adaptation depends on whether adaptation data are available for most (if not all) model parameters. This is because MAP is an unconstrained adaptation method, i.e., adaptation is performed only on those model parameters who have "seen" the

adaptation data. If there is no adaptation data for mixture i, the sum $\sum_{t=1}^{T} \gamma_i(t)$ in Eq. 4.2 will become zero. As a result, MAP will not adapt the parameters of the mixture.

The problem of unobserved mixture components can be addressed by exploiting the correlation between model parameters. For example, the linear regression relationship among different mixture components can be used to adapt the unobserved mixture components [17]. Other examples include the extended MAP (EMAP) method [97] and the quasi-Bayes technique with correlated mean vectors [40]. These techniques use the correlation between the parameters of different speech units to estimate the mean vectors of the unobserved speech units.

Another problem of MAP adaptation is that estimating the prior distribution $p(\phi)$ is often difficult. Typically, the prior can be obtained by clustering the training data based on similarity measures. One model can be derived for each cluster, and each model is regarded as an observation drawn from the prior distribution $p(\phi)$. Then, the hyperparameters of the prior are estimated based on sampled moments [95].

4.2 Maximum-Likelihood Linear Regression

Another popular model-based technique is maximum-likelihood linear regression (MLLR) [49]. In MLLR, a set of transformation matrices is used to transform the Gaussian parameters (i.e., the means and possibly covariances) of the clean models (or speaker-independent models) to maximize the likelihood of the adaptation data. MLLR is very effective for rapid adaptation because the transformations can be estimated from a small amount of adaptation data. Specifically, denote $\mu_{s,j}$ as the j-th

mean vector of the clean speaker model, the adapted mean vector $\mu_{ad\_s,j}$ is given by

$$\mu_{ad\_s,j} = W^k \hat{\mu}_{s,j} = A^k \mu_{s,j} + \mathbf{b}^k,$$

where $\hat{\mu}_{s,j} = [{\mu_{s,j}}^T, 1]^T$ is the extended mean vector of $\mu_{s,j}$. The k-th adaptation matrix $W^k = [A^k, \mathbf{b}^k]$ is composed of a translation vector $\mathbf{b}^k \in \Re^D$ and a transformation matrix $A^k \in \Re^D \times \Re^D$, where D is the dimensionality of the feature vectors.

It is possible to estimate a separate transformation matrix for each mixture component. This will probably provide sufficient flexibility to compensate for any nonlinear distortion. However, this will also increase the number of parameters to be estimated, which in turn requires more adaptation data to estimate the parameters robustly. In another extreme, a single global transformation can be used for all components. This global tying enables MLLR to adapt parameters that have not been observed in the adaptation data. A better approach is to tie mixture components with similar properties together to from a regression class [31]. The members of the same regression class use the same transformation. The more components we tie together, the fewer parameters we need to estimate; however, the transformation will also be coarser. A regression class tree can be used to decide which components should be tied together. The regression classes can be generated dynamically according to a binary decision tree to deal with the tradeoff between specificity and robustness. Specifically, at every node in the tree, a transformation matrix is estimated and shared for all Gaussian components beneath the node. New regression classes can be incrementally created when sufficient data become available.

When the number of regression classes is equal to the number of Gaussian compo-

nents, the convergence characteristic of MLLR is similar to that of MAP adaptation because in that case each Gaussian component is adapted independently just like MAP adaptation. The number of parameters to be estimated in MLLR is usually larger than that of MAP adaptation. To shorten computation time, a small number of regression classes should be used; however, the performance will quickly saturate when the amount of adaptation data increases. To address this problem, researchers proposed to combine MLLR with MAP [5, 95]. This approach guarantees rapid MLLR adaptation for small amount of data and asymptotic convergence with large amount of adaptation data.

4.3 Maximum a Posteriori Linear Regression

There are several methods to combine MLLR and MAP in order to take advantages of both methods. For example, we can firstly use MLLR to transform the model parameters and then use MAP to adapt the transformed models. It is also possible to incorporate the MAP concept into MLLR. One well known approach of this type is called maximum a posteriori linear regression (MAPLR).

When compensating environmental variation in speaker recognition, the transformation parameters, η, are assumed to be fixed but unknown. These parameters are usually determined by using maximum-likelihood formulations. More specifically, the maximum-likelihood estimates of the transformation parameters are given by

$$\hat{\eta}_{ML} = \arg\max_{\eta} p(Y|\Lambda_X, \eta) \tag{4.3}$$

and the transformed model is

$$\Lambda_Y^{MLLR} = f_{\hat{\eta}_{ML}}(\Lambda_X),$$

where Y is a distorted speech sequence, Λ_X is a GMM representing the clean speech, Λ_Y^{MLLR} is a GMM that fits the distorted speech Y in a maximum-likelihood sense, $p(Y|\Lambda_X, \eta)$ is the likelihood function of the transformed GMM $f_\eta(\Lambda_X)$ given the distorted speech sequence Y, and $f_{\hat{\eta}_{ML}}(\cdot)$ is a transformation function with parameters $\hat{\eta}_{ML}$. Maximum likelihood is used because it is computationally attractive and analytical solutions for the affine transformation family can be obtained easily. However, maximum-likelihood estimation only uses the information in the adaptation data and does not constrain the transformation parameters or incorporate into them any prior knowledge. Constraining the transformation parameters is beneficial when the amount of adaptation data is limited or prior information about the density of the transformation parameters is known.

By assuming the transformation parameters η to be a set of random variables, the prior information can be characterized by a probability density function (pdf), $p(\eta)$. This function constrains the values that the parameters η can take. The prior density $p(\eta)$ can be combined with the likelihood function $p(Y|\Lambda_X, \eta)$ by using the MAP formulation:

$$\begin{aligned} \hat{\eta}_{MAP} &= \arg\max_{\eta} p(\eta|Y, \Lambda_X) \\ &= \arg\max_{\eta} p(Y|\Lambda_X, \eta)p(\eta) \end{aligned} \tag{4.4}$$

and

$$\Lambda_Y^{MAP} = f_{\hat{\eta}_{MAP}}(\Lambda_X),$$

where $\hat{\eta}_{MAP}$ is the MAP estimate of the transformation parameters, Λ_Y^{MAP} is the MAP-transformed speech model, and $f_{\hat{\eta}_{MAP}}(\cdot)$ is a transformation function. This technique is referred to as MAPLR in the literature [16]. If the prior distribution reflects the the actual plausibility of the transformation parameters, then the MAP-estimated parameters should be more robust because incorrect transformation can be avoided.

4.4 Eigenvoice

One drawback of MAP and MLLR is that they require a large amount of adaptation data to estimate the free parameters robustly. In recent years, model-based algorithms that allow rapid adaptation have been proposed. For example, the eigenvoice approach [45] constrains the adapted models to be a linear combination of a small set of basis vectors, which greatly reduces the number of free parameters to be estimated. As a result, rapid adaptation can be achieved with a small amount of adaptation data. The basis vectors are derived offline from a set of reference speakers. These basis vectors form an orthogonal basis and guarantee to represent most of the interspeaker variations among the reference speakers.

In the eigenvoice approach, T speaker-dependent models are trained using the speech of T speakers in the reference set. For each speaker model, the mean vectors are stacked to from a supervector of dimension MD, where M is the number of Gaussian components in the speaker model and D is the dimensionality of the feature vectors. For a system with T reference speakers, T suppervectors will be formed. The order of the parameters in the supervectors is unimportant as long as

the order is identical across all supervectors. Dimension reduction techniques such as principle component analysis (PCA) is applied to the T supervectors to yield T eigenvectors, each of dimension MD. The first K eigenvectors ($K < T \ll MD$) capture most of the variations in the speaker models. These vectors are referred to as *eigenvoices*. These K eigenvoices span a space called K-space, and any adapted models are constrained to be located in this space. For each client speaker, K eigenvoice coefficients $(w(1), \ldots, w(K))$ are estimated using an EM-based algorithm called maximal-likelihood eigenvoice decomposition (MLED) [45]. Given the eigenvoice coefficients for a client speaker, his or her model is represented by a point P in the K-space, i.e.,

$$P = e(0) + w(1)e(1) + \cdots + w(K)e(K). \tag{4.5}$$

Figure 4.1 illustrates the eigenvoice adaptation process.

PCA creates a set of orthogonal basis and guarantees that truncating the expansion after the K-th eigenvector gives minimum mean-squared error. The eigenvoice approach uses these K eigenvectors to capture speaker variations. The approach constrains the adapted models to be a linear combination of these K eigenvectors. As K is usually small, the degree of freedom for the speaker models is also small.

Because the adapted models are highly constrained, their performance quickly saturates when large amount of adaptation data become available. To improve performance, the eigenvoice approach can be combined with MAP or MLLR [72]. Specifically, an eigenvoice model is trained and used as an initial model for MLLR or MAP adaptation. This approach allows the adapted models to characterize new speakers

not belonging to any reference speakers.

4.5 *Parallel Model Combination*

Parallel model combination (PMC) [30] is a model-based compensation technique that approximates the distribution of noisy speech by combining clean speech models with noise models in the log-spectral domain. Unlike conventional model retraining, where the clean speech data must be available online, PMC uses clean speech models to represent the statistics of clean speech data so that putting clean speech data online is unnecessary. Another advantage of PMC is its low computational complexity because it is not necessary to retrain the speaker models when background noise changes.

Figure 4.2 illustrates the PMC process. The process starts with a set of clean speech models and a noise model. For additive noise, it is simpler to model the noise effects in the linear- or log-spectral domain. Because feature vectors are often composed of cepstrum, delta cepstrum, and delta-delta cepstrum, they must be transformed to the linear- or log-spectral domain. This can be achieved by applying inverse discrete cosine transform (DCT) to the cepstral vectors. For noise vectors with mean μ_n^c and covariance matrix Σ_n^c in the cepstral domain, the mapping to the log-spectral domain is given by

$$\mu_n^l = C^{-1}\mu_n^c \tag{4.6}$$

$$\Sigma_n^l = C^{-1}\Sigma_n^c(C^{-1})^T, \tag{4.7}$$

where C is the DCT matrix and μ_n^l and Σ_n^l represent the mean and covariance of the noise in the log-spectral domain, respectively. Similarly, the clean speech parameters,

μ_x^c and Σ_x^c, can also be mapped to μ_x^l and Σ_x^l using Eqs. 4.6 and 4.7.

In the linear-spectral domain, the distribution of noise is log-normal, with mean μ_N and covariance matrix Σ_N given by

$$\begin{aligned} \mu_N[i] &= \exp\{\mu_n^l[i] + \Sigma_n^l[i,i]/2\} \\ \Sigma_N[i,j] &= \mu_N[i]\mu_N[j](\exp\{\Sigma_n^l[i,j]\} - 1), \end{aligned}$$

where i and j are the indexes to the corresponding vectors and matrices. Assuming that the noise power and speech power are independent and additive, the combined mean and covariance in the linear-spectral domain are given by

$$\begin{aligned} \mu_Y &= g\mu_X + \mu_N \\ \Sigma_Y &= g^2\Sigma_X + \Sigma_N, \end{aligned}$$

where g is a gain matching term introduced to compensate for (1) the level difference between the observed speech and noise and (2) the energy mismatch between training and testing speech.

In the popular log-normal approximation, it is assumed that the sum of two log-normal distributions is also log-normal. Using inverse formulae, the mean and covariance of the corrupt speech in the log-spectral domain can be written as:

$$\begin{aligned} \mu_y^l[i] &= \ln \mu_Y[i] - \frac{1}{2}\ln\left\{\frac{\Sigma_Y[i,j]}{\mu_Y[i]\mu_Y[j]} + 1\right\} \\ \Sigma_y^l[i,j] &= \ln\left\{\frac{\Sigma_Y[i,j]}{\mu_Y[i]\mu_Y[j]} + 1\right\}. \end{aligned} \tag{4.8}$$

Finally, the parameters are transformed back to the cepstral domain using the follow-

ing equations:

$$\mu_y^c = C\mu_y^l$$
$$\Sigma_y^c = C\Sigma_y^l C^T.$$

There are three main variants of PMC: log-normal, data-driven, and numerical integration. The log-normal approximation is popular but it cannot be used directly for delta and delta-delta cepstra. A more accurate variant is the data-driven PMC (DPMC) [30], which uses Monte Carlo simulations to compute the means and covariance matrices. Numerical integration [30] is the most accurate PMC method, but it is also the most computationally expensive.

4.6 Vector Taylor Series

The vector Taylor series (VTS) approach assumes that the clean speech signal $x[m]$ is corrupted by additive noise $n[m]$ and channel distortion. Specifically, the distorted speech $y[m]$ is expressed as

$$y[m] = x[m] * h[m] + n[m], \tag{4.9}$$

where $*$ denotes convolution and $h[m]$ is the channel's impulse response. In the cepstral domain, Eq. 4.9 becomes

$$\mathbf{y} = \mathbf{x} + \mathbf{h} + \mathbf{g}(\mathbf{n} - \mathbf{x} - \mathbf{h}), \tag{4.10}$$

where $\mathbf{y}$, $\mathbf{x}$, $\mathbf{n}$ and $\mathbf{h}$ are the cepstrum of noisy speech, clean speech, noise, and the channel, respectively, and $\mathbf{g}(\mathbf{z})$ is a nonlinear function given by

$$\mathbf{g}(\mathbf{z}) = C\ln(1 + \exp^{C^{-1}\mathbf{z}}), \tag{4.11}$$

where C is the DCT matrix. By substituting Eq. 4.11 into Eq. 4.10, we can obtain the cepstrum of noisy speech if the cepstrum of noise and the cepstrum of channel's impulse response are known.

It is reasonable to assume that the distribution of $\mathbf{x}$ is a mixture of Gaussians and that of $\mathbf{n}$ is a single Gaussian. Even if these assumptions are valid, the cepstrum of the corrupted speech $\mathbf{y}$ in Eq. 4.10 is not Gaussian because of the nonlinearity in Eq. 4.11. However, we can assume that a nonlinear function of Gaussian is also Gaussian; therefore the same decoder that is used for decoding $\mathbf{x}$ can also be used for decoding $\mathbf{y}$.

Moreno et al. [68] suggested to approximate the nonlinearity in Eq. 4.11 by a first-order Taylor series. Acero et al. [2] extended the work of Moreno et al. to compute the mean and covariance matrix of $\mathbf{y}$ in the cepstral domain. In [2], $\mathbf{x}$, $\mathbf{h}$, and $\mathbf{n}$ are assumed to be independent Gaussian vectors with means $\mu_{\mathbf{x}}$, $\mu_{\mathbf{h}}$, and $\mu_{\mathbf{n}}$ and covariance matrices $\Sigma_{\mathbf{x}}$, $\Sigma_{\mathbf{h}}$, and $\Sigma_{\mathbf{n}}$, respectively. With these assumptions, Eq. 4.10 can be approximated by a first-order Taylor series expansion around $(\mu_{\mathbf{n}}, \mu_{\mathbf{x}}, \mu_{\mathbf{h}})$, that is,

$$\begin{aligned} \mathbf{y} &= \mu_{\mathbf{x}} + \mu_{\mathbf{h}} + \mathbf{g}(\mu_{\mathbf{n}} - \mu_{\mathbf{x}} - \mu_{\mathbf{h}}) + A(\mathbf{x} - \mu_{\mathbf{x}}) + A(\mathbf{h} - \mu_{\mathbf{h}}) \\ &\quad + (I - A)(\mathbf{n} - \mu_{\mathbf{n}}), \end{aligned}$$

where the matrix A is given by

$$A = CFC^{-1} \tag{4.12}$$

and F is a diagonal matrix whose elements are given by vector $\mathbf{f}(\mu)$

$$\mathbf{f}(\mu) = \frac{1}{1 + \exp^{C^{-1}\mu}}. \tag{4.13}$$

The mean of $\mathbf{y}$, $\mu_{\mathbf{y}}$, is given by

$$\mu_{\mathbf{y}} = \mu_{\mathbf{x}} + \mu_{\mathbf{h}} + \mathbf{g}(\mu_{\mathbf{n}} - \mu_{\mathbf{x}} - \mu_{\mathbf{h}}) \tag{4.14}$$

and its covariance matrix $\Sigma_{\mathbf{y}}$ is given by

$$\Sigma_{\mathbf{y}} \approx A\Sigma_{\mathbf{x}}A^T + A\Sigma_{\mathbf{h}}A^T + (I - A)\Sigma_{\mathbf{n}}(I - A)^T. \tag{4.15}$$

It was found that the performance of VTS is close to that of a well-matched system and the Taylor series approximation appears to be more accurate than the log-normal approximation in PMC [2].

4.7 Probabilistic Decision-Based Neural Networks

Probabilistic decision-based neural networks (PDBNNs) were proposed by Lin, Kung, and Lin for face detection and recognition [51], and recently PDBNNs have shown promise in text-independent speaker verification [105]. One unique feature of PDBNNs is their two-phase learning rule: locally unsupervised (LU) and globally supervised (GS). In the LU phase, PDBNNs adopt the maximum-likelihood principle to estimate the network parameters. In the globally supervised (GS) phase, discriminative training based on gradient descent and reinforced learning is used to fine-tune the network parameters.

The following training strategy can be adopted to make PDBNNs appropriate for environment adaptation [105]. The strategy begins with the training of a clean speaker model and a clean background model using the LU training of PDBNNs. This step aims to maximize the likelihood of the training data. The clean models are then adapted to a channel-dependent speaker model and a channel-dependent background

model using GS training. While clean speech data are used in the LU training, distorted training data derived from the target channel are used in the GS training. The GS training uses gradient descent and reinforced learning to update the models' parameters so that the classification error of the adapted models on the distorted data is minimized. Hence, the resulting models will be speaker- and channel-specific.

By using the distorted data derived from H channels, the above training strategy will produce H channel-dependent speaker models for each client speaker. Likewise, H channel-dependent background models will also be created, and they are shared among all of the client speakers. Figure 5.1 (left portion) illustrates the idea of PDBNN-based adaptation for robust speaker verification.

4.8 Speaker Model Synthesis

Recently, Teunen et al. [99] proposed a new model-based compensation technique called speaker model synthesis (SMS) to address the channel mismatch problem. SMS begins with using maximum a posteriori (MAP) adaptation to adapt a channel- and gender-independent background model (also called root model) to obtain a number of channel-dependent background models, one for each known channel. The transformations among these channel-dependent background models are then computed offline. Specifically, the transformation $T_{ab}(\cdot)$ from channel a to channel b are computed as

follows:

$$T_{ab}(\omega_i) = \omega_i \left(\frac{\omega_{b,i}}{\omega_{a,i}}\right) \tag{4.16}$$

$$T_{ab}(\mu_i) = \mu_i + (\mu_{b,i} - \mu_{a,i}) \tag{4.17}$$

$$T_{ab}(\sigma_i^2) = \sigma_i^2 \left(\frac{\sigma_{b,i}^2}{\sigma_{a,i}^2}\right), \tag{4.18}$$

where $\{\omega_{a,i}, \mu_{a,i}, \sigma_{a,i}^2\}$ and $\{\omega_{b,i}, \mu_{b,i}, \sigma_{b,i}^2\}$ are the mixing coefficients, means and variances of the i-th mixture component in the background models corresponding to channels a and b, respectively. The transformation is valid because the models for both channel a and channel b are adapted from the same root model. Note that the transformation ensures that $T_{ab}(\lambda_a) = \lambda_b$.

During enrollment, the channel used by the client is detected; the corresponding background model is then adapted to create a channel-dependent speaker model. During verification, the handset used by the claimant is detected. If the detected handset matches the channel of the speaker model, then the corresponding channel-dependent speaker and background models are used for verification. However, if the detected handset does not correspond to the speaker model (i.e., a mismatch occurs), then a new speaker model corresponding to the detected channel is synthesized by using the transformation that maps the enrollment channel to the detected channel (see Figure 4.3). The synthesized speaker model and the background model corresponding to the detected channel are then used for computing the log-likelihood ratio for decision making. Because all speaker and background models are adapted from the root model, a one-to-one correspondence between individual Gaussians in these models can be preserved. Figure 4.3 illustrates the idea of SMS.

Research [99] has shown that SMS achieves performance that matches the best result reported in the 1998 NIST evaluation set; in particular, its performance is similar to that of Hnorm but only requires a fraction of Hnorm's training time.

The idea of mapping speaker models from one channel to another in SMS has been extended to feature-domain channel compensation. In [85], Reynolds suggests a technique called feature mapping (see also Section 6.5) in which feature vectors from different channels are transformed to a channel-independent feature space by a mapping function whose parameters are learned from a set of channel-dependent models.

4.9 Summary

Table 4.1 summarizes the characteristics of the model-based compensation techniques discussed in this chapter, and Table 4.2 compares these techniques in terms of training methods, requirement on the amount of adaptation data, and computational complexity.

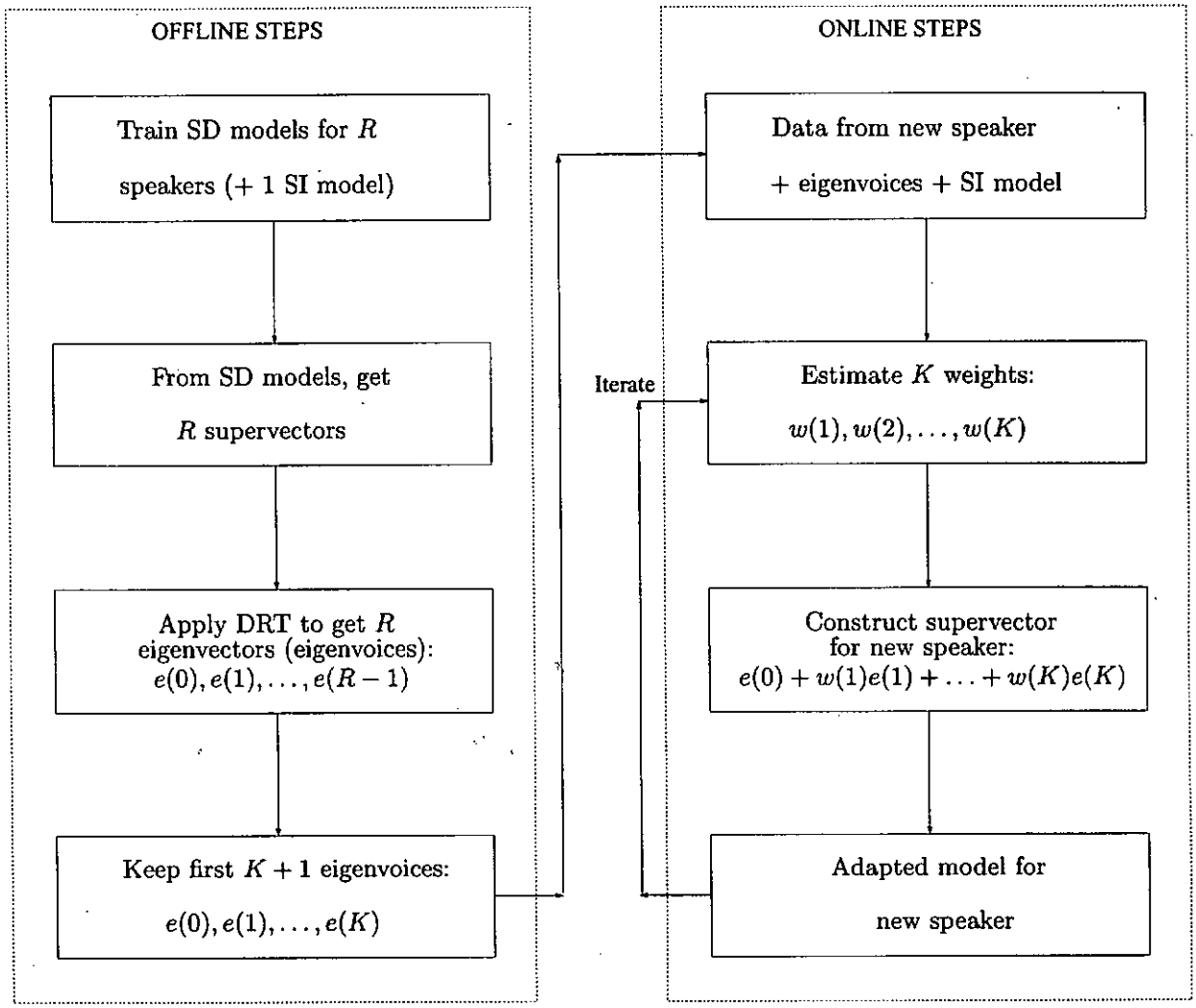


Figure 4.1: The process of eigenvoice speaker adaptation. (Adapted from: R. Kuhn, et al. Rapid Speaker Adaptation in Eigenvoice Space, *IEEE Trans. on Speech and Audio Processing*, vol. 8, no. 6, pp. 695-707, 2000.)

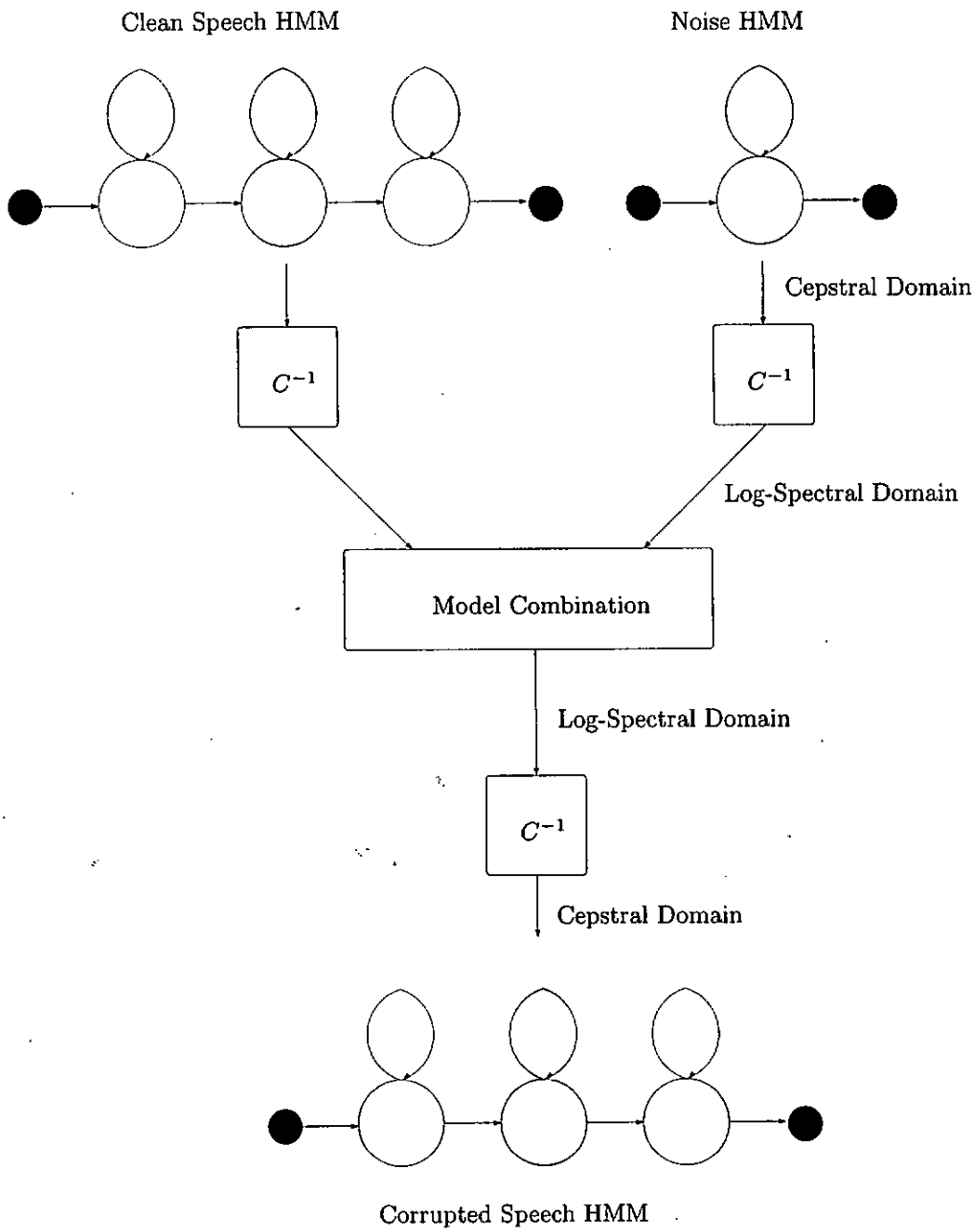


Figure 4.2: The PMC process. (Adapted from: G. J. Gales, *Model Based Techniques for Noise Robust Speaker Recognition*, Engineering Department, PhD Thesis, Cambridge University, 1995.)

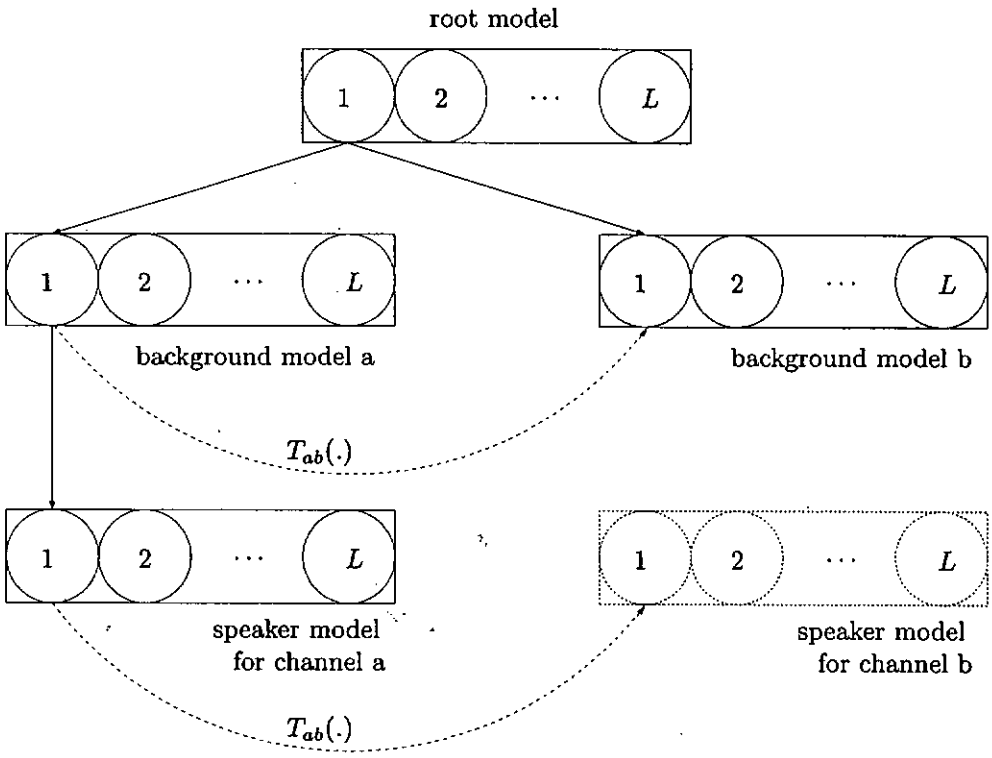


Figure 4.3: The idea of speaker model synthesis. (Adapted from: R. Teunen, B. Shahshahani and, L. Heck, A Model-Based Transformational Approach to Robust Speaker Recognition, in Proc. *ICSLP 2000*, vol. 2, pp. 495-498, 2000.)

Compensation Method	Characteristics
MAP	- Constraint and prior information can be incorporated into adaptation via prior density functions. - Reduce to the maximum-likelihood approach when the amount of adaptation data increases (i.e., asymptotic convergence). - Slow convergence for large models. - Adaptation is unconstrained in that unobserved mixture components will not be adapted. - The parameters of prior distributions are difficult to estimate.
MLLR	- Use a set of transformation matrices to transform speaker-independent models to speaker-dependent models. - Able to achieve rapid adaptation using a small amount of adaptation data. - Use a regression tree to tie mixtures into groups. - Performance can be saturated quickly when the amount of adaptation data increases.
MAPLR	- Incorporate the Bayesian concept into MLLR. - Transformation parameters are treated as random variables with prior distributions. - More robust than MLLR because incorrect transformations can be avoided.
Eigenvoice	- Constrain the adapted models to be a linear combination of eigenvectors. - Can perform rapid adaptation using a small amount of adaptation data. - The eigenvectors form orthogonal bases and guarantee to represent most of the interspeaker variations. - Performance can be saturated quickly the large amount of adaptation data increases.
PMC	- Approximate noisy speech models by combining clean speech models with noise models in the log-spectral domain. - Three variants of PMC: log-normal, data-driven, and numerical integration. - Log-normal PMC cannot be used directly for delta and delta-delta cepstra. - Accuracy: numerical integration > data-driven > log-normal
VTS	- Approximate the nonlinearity between the noisy speech models, clean speech models, and the channel models using a first-order Taylor series expansion. - Performance is close to that of well-matched system. - Appears to be more accurate than the log-normal PMC.
PDBNNs	- Apply discriminative training to perform adaptation. - Channel-dependent speech data are required. - The adapted models will be speaker- and channel-specific.
SMS	- Based on MAP adaptation. - Make use of the one-to-one correspondence between individual Gaussians. to compute transformation between different channels.

Table 4.1: Summary of the state-of-the-art model-based compensation techniques discussed in this chapter.

Model-based Compensation	Training Methods	Requirement on the Amount of Adaptation Data	Computation Complexity
MAP	Bayesian (EM-based)	Large	Low
MLLR	Maximum-Likelihood (EM-based)	Moderate	High
MAPLR	Bayesian (EM-based)	Small	High
Eigenvoice	Maximum-Likelihood (EM-based)	Very small	Low
PMC	Models are combined using mismatch functions	Large	High
VTS	EM-based	Large	High
PDBNN	Reinforced Learning (Discriminative)	Small	High
SMS	Bayesian (EM-based)	Small	Low

Table 4.2: Comparison of model-based compensation techniques in terms of training methods, requirement on the amount of adaptation data, and computation complexity.

Chapter 5

MODEL ADAPTATION AND TRANSFORMATION FOR CHANNEL COMPENSATION

As mentioned in Chapter 4, with some modifications, MLLR and the learning algorithm of PDBNNs can be applied to model adaptation. One of the positive properties of PDBNNs is their supervised training procedure. However, the procedure will change the model parameters only if the adaptation data are sufficiently close to the model centers. MLLR, on the other hand, applies a transformation matrix to a group of acoustic centers so that all centers will be transformed even if the adaptation data are far away from the model centers. As a result, MLLR provides a quick improvement, but its performance quickly saturates as the amount of adaptation data increases. This chapter proposes a model adaptation/transformation technique that combines the advantage of PDBNNs' supervised training procedure and MLLR transformation. Experimental results show that the proposed techniques outperforms several classical ones, including CMS, Hnorm, Tnorm, and speaker model synthesis.

5.1 Cascading MLLR Transformation and PDBNN Adaptation

Although PDBNN's reinforced learning uses handset- and speaker-dependent patterns to adapt the model parameters, a previous study [106] showed that its performance

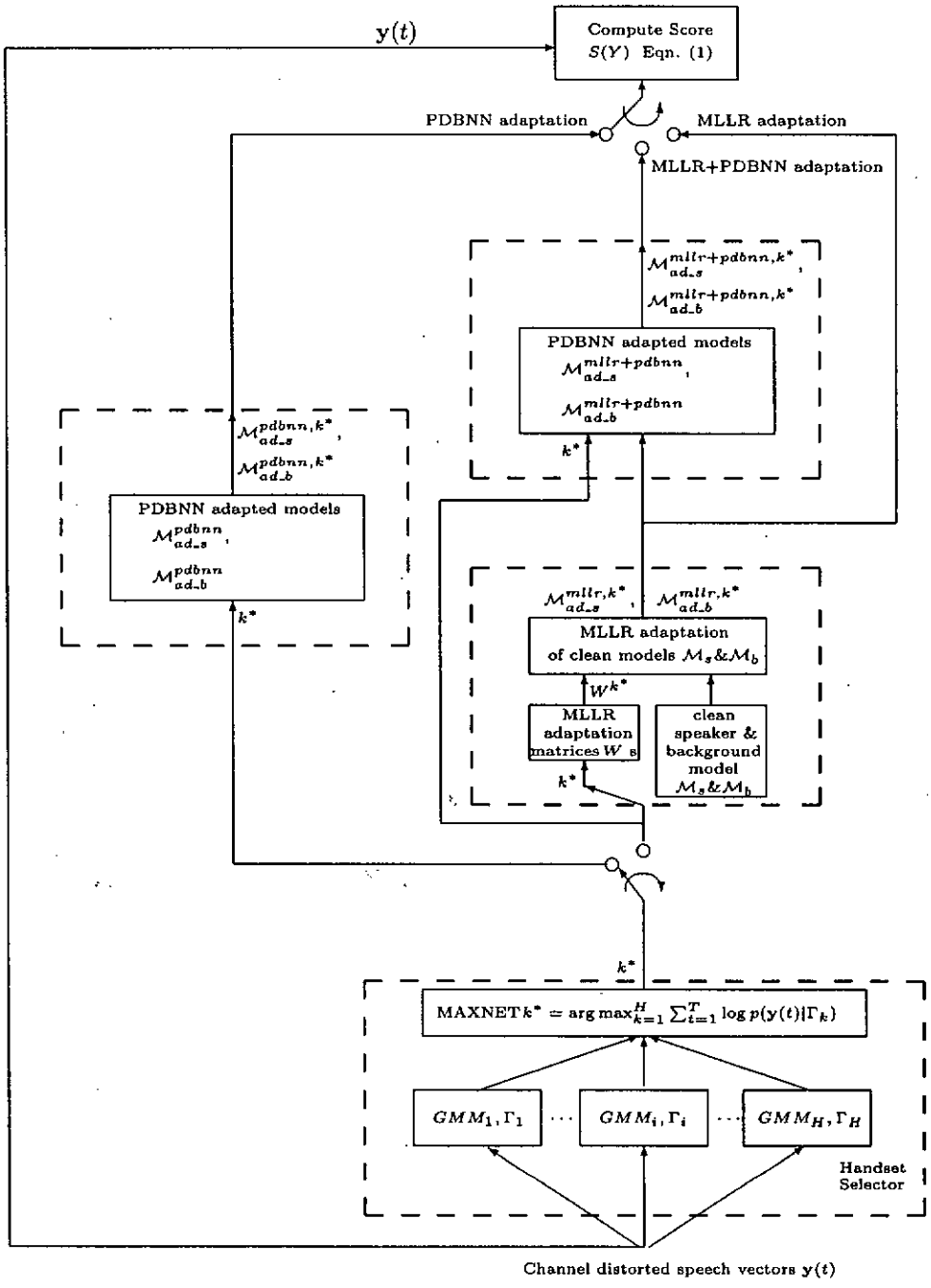


Figure 5.1: The combination of handset identification and model adaptation for robust speaker verification. *Note:* adaptation is applied to both the speaker and background models.

is inferior to that of MLLR adaptation. A possible reason is that PDBNN's discriminative training is sensitive to the centers' locations of the "initial model". Because the adaptation data may be far away from the centers of the clean speaker models, PDBNN's reinforced learning can only adapt some of the model centers in the clean speaker models. This phenomenon is very similar to that of MAP adaptation in which adaptation is performed only on those model parameters who have "seen" the adaptation data. If a mixture component is far away from all of the adaptation data, MAP will not adapt the parameters of that mixture. To overcome this obstacle, we propose using MLLR to transform the clean speaker models to a region close to the handset-distorted speech. These MLLR-transformed models are then used as the initial models for PDBNN's reinforced learning. Figure 5.1 (right portion without the bypass) shows how the cascade of MLLR transformation and PDBNN adaptation can be integrated into a composite speaker verification system.

5.2 *Cascading MLLR Transformation and PDBNN Adaptation Using Transformed Features*

Although cascading MLLR transformation and PDBNN adaptation should be better than using MLLR transformation or PDBNN adaptation alone, this approach is not very practical. This is because PDBNN's reinforced learning requires client-dependent speech data to be collected from each of the environments that the client may encounter during verification. A possible solution is to use stochastic feature transformation (SFT) [57] to transform the clean speech data to environment-specific speech data. Figure 5.2 illustrates the idea of the proposed method.

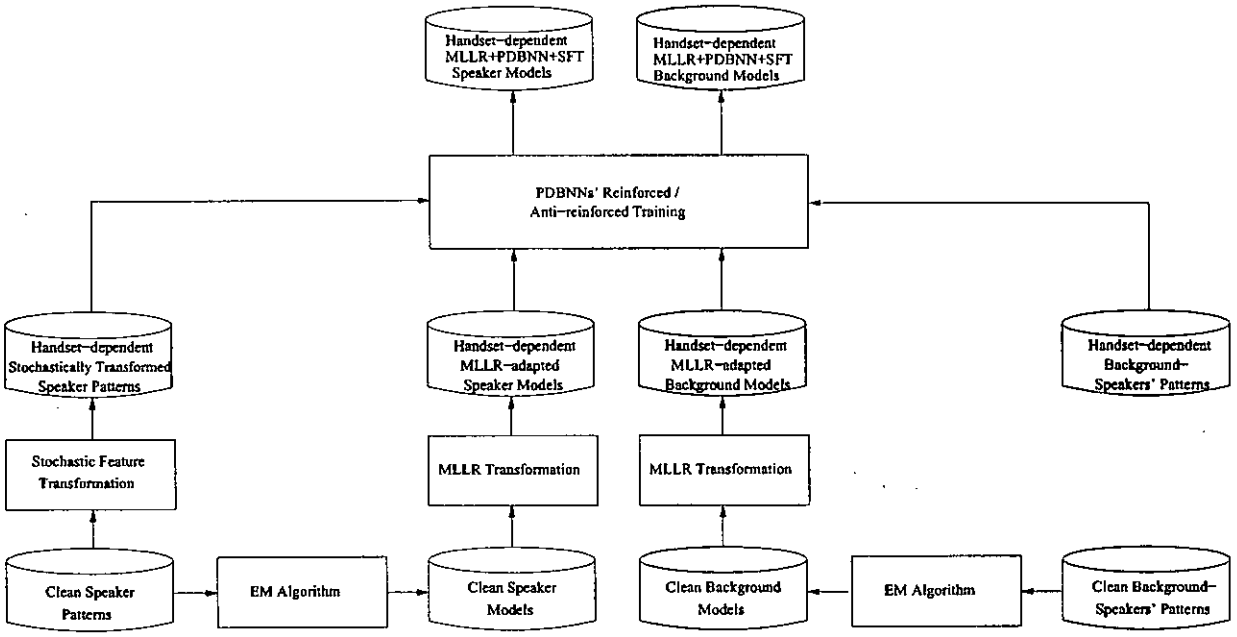


Figure 5.2: The process of fine-tunning MLLR-adapted models using transformed features. For each known handset, a precomputed MLLR adaptation matrix is used to transform the clean models to handset-dependent MLLR-adapted models. Then, PDBNN adaptation is performed on the MLLR-adapted models using handset-dependent, stochastically-transformed patterns to obtain the final adapted models.

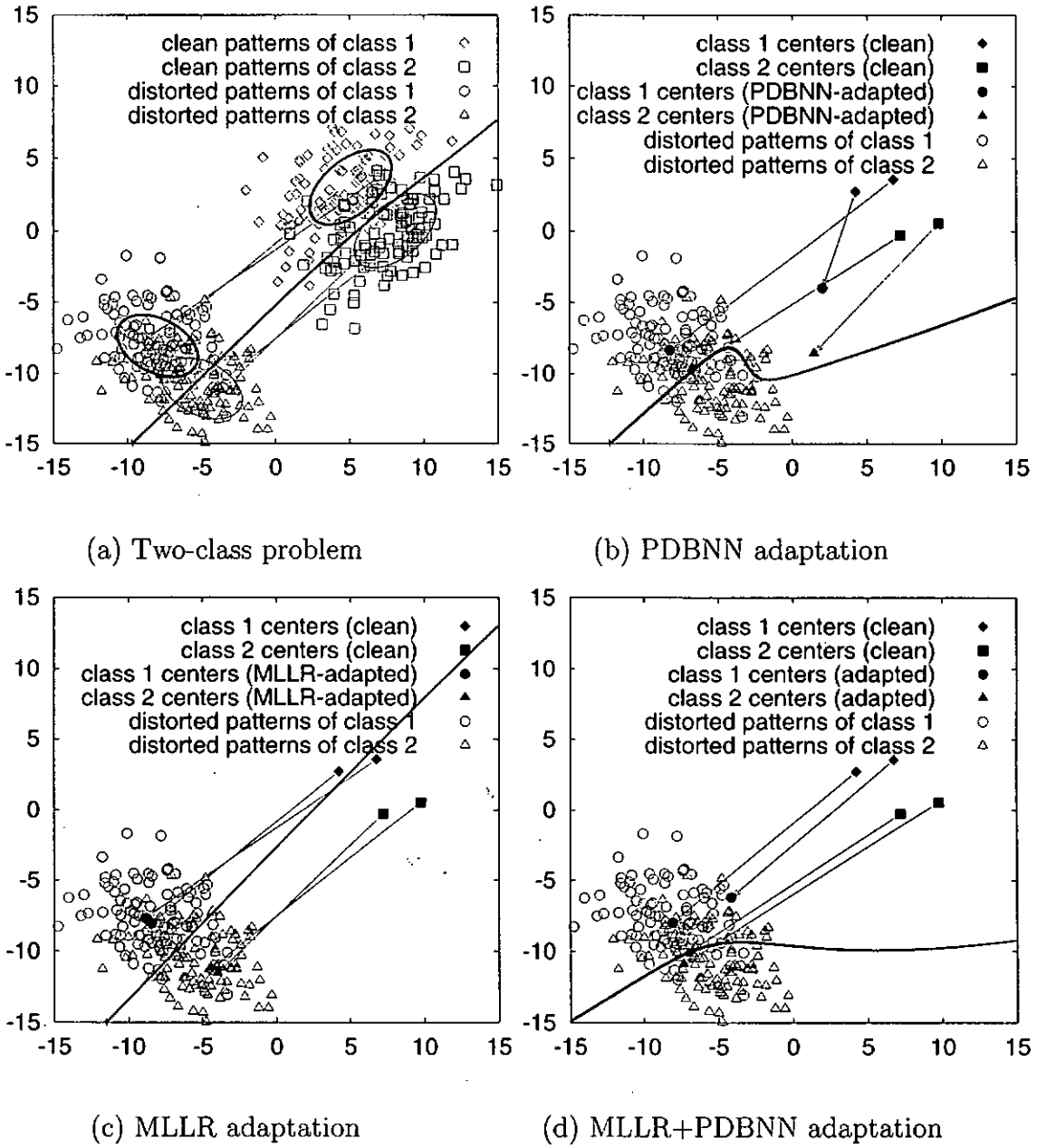
For each known handset, a precomputed MLLR matrix is used to transform the clean models to handset-dependent MLLR-adapted models. Then, PDBNN adaptation is performed on the MLLR-adapted models using handset-dependent, stochastically-transformed patterns to obtain the final adapted models. Because handset-specific, client-dependent speech patterns are difficult to obtain, stochastic feature transformation is applied to transform the clean client patterns, as illustrated in Figure 5.2, to handset-dependent client patterns. The key idea is that SFT-transformed client patterns are artificially generated to provide the PDBNN adaptation with the required data in the handset-distorted space for fine-tuning the

MLLR-adapted models. For the background speakers' data, transforming the features is not necessary because plenty of environment-dependent data are available.

5.3 A Two-Dimensional Example

Figure 5.3 illustrates the idea of environment adaptation in a two-class problem. Figure 5.3(a) plots the clean and distorted patterns of Class 1 and Class 2. The upper right (respectively, lower left) clusters represent the clean (respectively, distorted) patterns. The ellipses show the corresponding equal density contours. The decision boundary in Figure 5.3(a) was derived from the clean GMMs that were trained using the clean patterns. As can be observed from the lower left portion of Figure 5.3(a), this decision boundary is not appropriate for separating the distorted patterns.

The clean models were adapted using the methods explained in Sections 4.2, 4.7, and 5.1 and the results are shown in Figures 5.3(b) through 5.3(d). In Figures 5.3(b), 5.3(c), and 5.3(d), markers ♦ and ■ represent the centers of the clean models and • and ▲ represent the centers of the adapted models. The arrows indicate the adaptation of model centers and the thick curves show the equal-error decision boundaries derived from the adapted models. For PDBNN-adaptation (Figure 5.3(b)), two clean GMMs were trained independently using the clean patterns of each class. The distorted patterns of both classes were used to adapt the clean models using PDBNN's reinforced learning. For MLLR-adaptation (Figure 5.3(c)), a GMM was trained using the clean patterns of both classes, which was followed by the estimation of MLLR parameters using the clean GMM and the distorted patterns of both classes. The two clean GMMs corresponding to the two classes were then transformed using the



(a) Two-class problem

(b) PDBNN adaptation

(c) MLLR adaptation

(d) MLLR+PDBNN adaptation

Figure 5.3: (a) Scatter plots of the clean and distorted patterns in a 2-class problem. The thick and thin ellipses represent the equal density contour of Class 1 and Class 2, respectively. The upper right (respectively lower left) clusters contain the clean (respectively distorted) patterns. The decision boundary (thick curve) was derived from the clean GMMs which were trained using the clean patterns. (b) Centers of the clean models and the PDBNN-adapted models. The thick curve is the decision boundary created by the PDBNN-adapted models. (c) Centers of the clean models and MLLR-adapted models. The thick curve is the decision boundary created by the MLLR-adapted models. (d) Centers of the clean models and MLLR+PDBNN adapted models. (The clean models were firstly transformed by MLLR; then the transformed models were further adapted by PDBNN adaptation to obtain the final model.) In (b), (c), and (d), only the distorted patterns are plotted for clarity. The arrows indicate the displacement of the original centers and the adapted centers.

estimated MLLR parameters. For the cascaded MLLR-PDBNN adaptation (Figure 5.3(d)), the clean models were firstly transformed by MLLR and then PDBNN adaptation was performed on the adapted models to obtain the final model.

A comparison between Figure 5.3(b) and Figure 5.3(c) reveals that while PDBNN-adaptation can only transform some of the model centers to the region of the distorted clusters, MLLR-based adaptation can transform all of the centers of the clean GMMs to a region around the distorted clusters. By cascading MLLR transformation and PDBNN adaptation, all of the centers of the clean GMMs can be transformed to a region within the distorted clusters before the decision boundary is fine-tuned. The adaptation capability of cascading MLLR and PDBNN is also demonstrated in the speaker verification evaluation described in Section 5.5.

5.4 Handset Selector

Unlike speaker adaptation in speech recognition where the adapted system will be used by the same speaker in subsequent sessions, in speaker verification, the claimant in each verification session may be a different person. Therefore, one cannot use the claimant's speech for adaptation, because by doing so the client's speaker model will be transformed to fit the claimant's speech regardless of the genuineness of the claimant. This will result in high false acceptance error if the claimant turns out to be an impostor. Therefore, instead of using the claimant speech for determining the transformation parameters or adapting the client model directly, it can be used indirectly as follows. Before verification takes place, one set of transformation parameters (or adapted speaker models) is obtained for each of the handsets (or handset

types) that the clients are likely to use. Then, during verification, the most likely handset (or handset type) that is used by the claimant is identified and the best set of transformation parameters (or the best adapted model) is selected accordingly.

A recently proposed handset selector [57, 58, 102] can be adopted to identify the most likely handset given an utterance. Specifically, H GMMs, $\{\Gamma_k\}_{k=1}^{H}$, as shown in Figure 5.1, are independently trained using the distorted speech recorded from the corresponding telephone handsets. During recognition, the claimant's features $\mathbf{y}(t), t = 1, \ldots, T$, are fed to all GMMs. The most likely handset k^* is selected by the Maxnet as illustrated in Figure 5.1.

For PDBNN adaptation, the precomputed PDBNN-adapted speaker model ($\mathcal{M}_{ad\_s}^{pdbnn,k^*}$) and background model ($\mathcal{M}_{ad\_b}^{pdbnn,k^*}$) corresponding to the k^*-th handset are used for verification. For MLLR adaptation, the precomputed MLLR adaptation matrix (W^{k^*}) for the k^*-th handset is used to transform the clean speaker model ($\mathcal{M}_s$) to the MLLR-adapted speaker model ($\mathcal{M}_{ad\_s}^{mllr,k^*}$). The same matrix is also used to transform the clean background model ($\mathcal{M}_b$) to the MLLR-adapted background model ($\mathcal{M}_{ad\_b}^{mllr,k^*}$). For MLLR+PDBNN adaptation, the precomputed MLLR adaptation matrix (W^{k^*}) for the k^*-th handset is firstly used to transform the clean models ($\mathcal{M}_s$, $\mathcal{M}_b$) to the MLLR-adapted model ($\mathcal{M}_{ad\_s}^{mllr,k^*}$, $\mathcal{M}_{ad\_b}^{mllr,k^*}$). Then, PDBNN adaptation is performed on the MLLR-adapted models to obtain the MLLR+PDBNN adapted models ($\mathcal{M}_{ad\_s}^{mllr+pdbnn,k^*}$, $\mathcal{M}_{ad\_b}^{mllr+pdbnn,k^*}$). These models will be used for verifying the claimant.

5.5 Experiments

5.5.1 Speech Corpus

The HTIMIT corpus [84], which contains the speech of 384 speakers, was used to evaluate the adaptation approaches. HTIMIT was collected with the objective of minimizing confounding factors so that the recorded speech differs predominately in handset transducer effects. This was achieved by playing a gender-balanced subset of the TIMIT corpus through nine telephone handsets and a Sennheizer head-mounted microphone. Therefore, unlike other telephone speech database where no handset labels are given, every utterance in HTIMIT is labeled with a handset name (cb1–cb4, el1–el4, pt1, or senh). The SA sentences (i.e., the rainbow sentences) spoken by all speakers also provide the best case scenario for discriminative training techniques such as PDBNN adaptation. Because of these special characteristics, HTIMIT has been used in several speaker recognition studies, including Quartieri et al. [76] and Mak and Kung [57].

Speakers in the corpus were divided into a speaker set (consisting of 100 speakers) and an impostor set (consisting of 50 speakers). Each speaker in the corpus spoke two dialectal sentences (the SA sentence set), five phonetically-compact sentences (the SX sentence set) and three phonetically-diverse sentences (the SI sentence set). Each speaker spoke the same set of sentences in the SA sentence set. In the SX sentence set, some speakers spoke the same sentences. However, all sentences in the SI sentence set are different. Therefore, the HTIMIT corpus allows us to perform text-independent speaker verification by using SA and SX sentence sets as the training set and the SI

sentence set as the test set.

5.5.2 Training the Handset Selector

A handset selector consisting of 10 handset-dependent GMMs was trained using the speech of all client speakers. Specifically, for each handset, the utterances from the SA and SX sentence sets of 100 clients speakers were used to train the corresponding handset-dependent GMM.

5.5.3 Enrollment Procedures

Utterances from the head-mounted microphone (senh) were considered to be clean and were used for creating the speaker models and background models. A Pilot experiment was performed to compare the performance of using maximum-likelihood estimation (MLE) and MAP adaptation [86] to create speaker models. Different combination of features (MFCCs, delta MFCCs, and delta-delta MFCCs) and interframe processing techniques (CMS [8] and RASTA [35]) were also investigated. For MAP adaptation, the number of speaker centers was equal to the number of background centers because the speaker models were adapted from the background model. For MLE, we used fewer speaker centers for the speaker models that use both static and dynamic features (delta MFCCs and delta-delta MFCCs) as inputs, because when the feature dimension increases, more parameters are required to be estimated. We reduced the number of speaker centers so that the parameters can be estimated robustly.

Table 5.1 shows the equal error rates (EERs) of different features and model creation methods. The EERs were based on the average of 100 target speakers and

50 impostors without any channel compensation; hence, they represent the inherent channel-robustness of the features. Evidently, all feature combinations and training approaches achieve a similar level of performance. However, MFCCs with MLE training achieves the lowest EER. RASTA processing, which is equivalent to bandpass filtering the MFCC sequences, will smooth out the sequences and make the speakers less distinguishable; as a result, its performance is poorer than that of CMS.

We chose to use MFCCs as features and MLE as the training method in the rest of the experiments because the time to estimate the MLLR transformation matrices depends on the feature dimension. By keeping the dimension small, the time required to estimate the transformation matrices can be greatly reduced. To demonstrate this phenomenon, we performed another experiment to compare the time used for computing the MLLR parameters with feature dimensions set to 12 and 36. The number of training patterns is the same in both cases. The time to compute the MLLR parameters for 12-dimensional features is 1,310 seconds while that for 36-dimensional features is 94,530 seconds. This suggests that the feature dimension is a predominant factor in determining the computation time.

Feature	**Interframe Processing Method**	**MAP**	**MLE**	
		Equal Error Rate (%)	Equal Error Rate (%)	No. of Speaker Centers
MFCC	CMS	11.86	11.23	32
MFCC+ΔMFCC	CMS	12.01	14.36	16
MFCC+ΔMFCC+$\Delta\Delta$MFCC	CMS	11.39	15.85	8
MFCC	RASTA	12.61	12.91	32

Table 5.1: EER (in %) of different features combination and training approaches. The number of centers in the background models was set to 64 in all cases.

Based on the results in the pilot experiment, we assigned a 32-center GMM ($\mathcal{M}_s$) for each client speaker to characterize his or her voice. For each speaker model, the training vectors were derived from the SA and SX utterances of the corresponding speaker. A 64-center GMM universal background model ($\mathcal{M}_b$), which was trained using all of the SA and SX utterances from all speakers in the speaker set, was used to normalized the speaker scores (see Eq. 5.1).

The feature vectors were 12-th order mel-frequency cepstral coefficients (MFCCs) computed every 14ms using a Hamming window of 28ms. To remove silence frames, the phonetic transcription files (.phn) in the corpus were used. All silence regions, as specified in the transcription files, were excluded from the feature extraction process. Therefore, all frames are based on the speech regions of the utterances.

5.5.4 Model Adaptation Procedures

For PDBNN-based adaptation, the clean speaker model ($\mathcal{M}_s$) and the clean background model ($\mathcal{M}_b$) described before were used as the initial models for globally supervised training. The SA and SX utterances of the target speaker and background speakers from a telephone handset were used as positive and negative training patterns, respectively. Hence, for each target speaker, a handset-dependent speaker model and a handset-dependent background model were created for each handset (including the head-mounted microphone used for enrollment). As shown in Figure 5.1, the PDBNN-adapted models are denoted as $\mathcal{M}_{ad\_s}^{pdbnn,k}$ and $\mathcal{M}_{ad\_b}^{pdbnn,k}$.

For MLLR-based adaptation, we used a single, full transformation matrix to compensate for the mismatch between two environments. Specifically, a clean background

model ($\mathcal{M}_b$) was trained using the clean speech of all speakers in the speaker set. Then, the speech data from another handset were used to estimate a transformation matrix W^k corresponding to that handset using MLLR. This procedure was repeated for all handsets in the HTIMIT corpus. As illustrated in Figure 5.1, the MLLR-adapted models are denoted as $\mathcal{M}_{ad\_s}^{mllr,k}$ and $\mathcal{M}_{ad\_b}^{mllr,k}$.

For MLLR+PDBNN adaptation, both MLLR transformation and PDBNN globally supervised training were applied to compensate for the acoustic mismatch. First, MLLR was used to transform the clean speaker model ($\mathcal{M}_s$) and the clean background model ($\mathcal{M}_b$) to handset-dependent models. PDBNN globally supervised training were then applied to the handset-dependent models. The handset-dependent SA and SX utterances of the target speaker and background speakers were used as the training patterns in the PDBNN supervised training. The resulting models are denoted as $\mathcal{M}_{ad\_s}^{mllr+pdbnn,k}$ and $\mathcal{M}_{ad\_b}^{mllr+pdbnn,k}$, as illustrated in Figure 5.1.

The MLLR+PDBNN+SFT adaptation is identical to MLLR+PDBNN adaptation except that the former uses stochastically transformed speaker features in the PDBNN globally supervised training. Specifically, the clean speaker patterns from the senh handset were transformed using SFT to obtain a set of artificial, handset-dependent patterns for PDBNN adaptation. Because handset-dependent patterns for the background speakers can be easily obtained, it is not necessary to perform SFT on background speakers' speech. Although HTIMIT provides training data for all speakers and handsets, not all of these data were used in this part of the experiments. This is because in practical implementation of speaker verification systems, each client speaker typically provides a few utterances from a single handset only. The experi-

ments reported here aim to simulate a more realistic environment than HTIMIT can provide by artificially generating environment-dependent data for PDBNN adaptation. Therefore, the implementation constraints of MLLR+PDBNN adaptation—the necessity of handset-dependent speaker patterns—can be relaxed.

The SFT parameters were obtained as follows. The clean utterances (SA and SX utterances from handset senh) of 20 speakers were used to create a 2-center GMM clean model Λ_X. Using this clean model and the handset-dependent utterances (SA and SX) of these 20 speakers, a set of feature transformation parameters was computed for each handset. Further details on SFT can be found in Section 6.4.

A preliminary evaluation was performed to compare the performance of MLLR transformation using 5, 20 and 100 speakers to estimate the adaptation matrices W^ks. While the performance improves with the number of speakers, the computation time also increases with the total number of training patterns. However, because this is a one-time (offline) computation, the training cost is not as critical as that of the enrollment computation (see Section 5.6.2 for more discussions). Therefore, 20 and 100 speakers were used to estimate the MLLR transformation matrices in the experiments.

We have also conducted experiments using Hnorm [83], Tnorm [9], stochastic feature transformation (SFT) [57], and speaker model synthesis (SMS) [99] for comparison. Hnorm aims to make the scores derived from different handset types more comparable, which is an important step for ensuring consistency across handsets. Tnorm does not make explicit use of handset types; its main purpose is to reduce the miss rate at low false-alarm rate instead of reducing the EERs. SMS is a model-

based compensation approach that combines handset selector and handset-dependent transformations to overcome the handset mismatch problem.

For Hnorm, the speech patterns derived from handset-dependent utterances of 49 same-gender (same as the client speaker), nontarget speakers were used to compute the handset-dependent score means and standard deviations. As a result, each client speaker model is associated with 10 handset-dependent score means and standard deviations. These parameters were used during verification to normalize the claimant's scores. For Tnorm [9], verification utterances were fed to all of the 99 nontarget speaker models to calculate the score mean and standard deviation. These parameters were then used to normalize the claimant's scores. The SFT parameters were estimated using the same 20 and 100 speakers, as in PDBNN model adaptation and MLLR model transformation. For SMS, the transformations between different handsets were derived from the parameters of the corresponding handset-dependent background models, which were trained using all of the SA and SX utterances of the corresponding handsets from all speakers in the speaker set.

5.5.5 Verification Procedures

During verification, a pattern sequence Y derived from each of the SI sentences of the claimant was fed to the GMM-based handset selector $\{\Gamma_i\}_{i=1}^{10}$. Handset-dependent speaker and background models or adaptation matrix were selected according to the handset selector's output (see Figure 5.1). The test utterance was then fed to an adapted speaker model $\mathcal{M}_{ad\_s}^{\Psi,k^*}$ and an adapted background model $\mathcal{M}_{ad\_b}^{\Psi,k^*}$ to obtain

the score

$$S(Y) = \log p(Y|\mathcal{M}_{ad\_s}^{\Psi,k^*}) - \log p(Y|\mathcal{M}_{ad\_b}^{\Psi,k^*}), \tag{5.1}$$

where $\Psi \in$ {PDBNN, MLLR, PDBNN+MLLR, PDBNN+MLLR+SFT} represents the type of adaptation used for obtaining the adapted models. $S(Y)$ was compared with a global, speaker-independent threshold to make a verification decision. In this work, the threshold was adjusted to determine the equal error rates (EERs).

For each handset and adaptation method, a set of client scores and a set of impostor scores were collected as follows. Each of the 100 adapted speaker models was tested by the SI utterances of the corresponding speaker and 50 impostors to produce a set of client scores and impostor scores corresponding to that speaker.[1] For each handset in an experimental setting, there were 300 client speaker trials (100 client speakers $\times$ 3 sentences per speaker) and 15,000 impostor trials (50 impostors per client speaker $\times$ 100 client speakers $\times$ 3 sentences per impostor). The client scores and impostor scores of 100 speaker models were lumped together to form a set of client-speaker scores and a set of impostor scores. By adjusting a decision threshold, these two sets of scores produce a detection error tradeoff (DET) curve [62] and a speaker-independent equal error rate (EER). Note that the DET curve and EER are handset-dependent because they are derived from handset-dependent scores. To obtain handset-independent DET curves and EERs, the client and impostor scores of 10 handsets were concatenated to form a set of handset-independent client scores and impostor scores.

[1] Client speakers will not be used as impostors.

5.6 Results and Discussions

5.6.1 Comparison in Terms of Error Rates

Table 5.2 show the results of different environment adaptation approaches, including cepstral mean subtraction (CMS), test normalization (Tnorm) [9], handset normalization (Hnorm) [83], speaker model synthesis (SMS) [99], PDBNN adaptation, stochastic feature transformation (SFT) [57], MLLR, MLLR+Hnorm, MLLR+PDBNN and MLLR+PDBNN+SFT adaptation. All error rates were based on the average of 100 target speakers and 50 impostors. The column with label "Average" shows the average of 10 handset-dependent EERs corresponding to the 10 handsets, and the column with label "Global" shows the handset-independent EERs obtained by merging the scores of 10 handsets. Therefore, the decision thresholds used for obtaining the average EERs are speaker-independent but handset-dependent, and the thresholds used for obtaining the global EERs are speaker- and handset-independent.

Figure 5.4 shows the DET curves corresponding to handset el2; there were 300(100×3) speaker trails and 15000(100×50×3) impostor trials in the experiment. Figure 5.5 shows the composite DET curves obtained by merging the scores of all handsets; there were 3000(300×10) speaker trails and 150000(15000×10) impostor trials in the experiment. The following discussions are based on the handset-independent EERs.

Evidently, all cases of environment adaptation (except Tnorm) show significant reduction in error rates when compared to CMS. In particular, MLLR+PDBNN and MLLR+PDBNN+SFT adaptation achieve the largest error reduction. Table 5.2 also

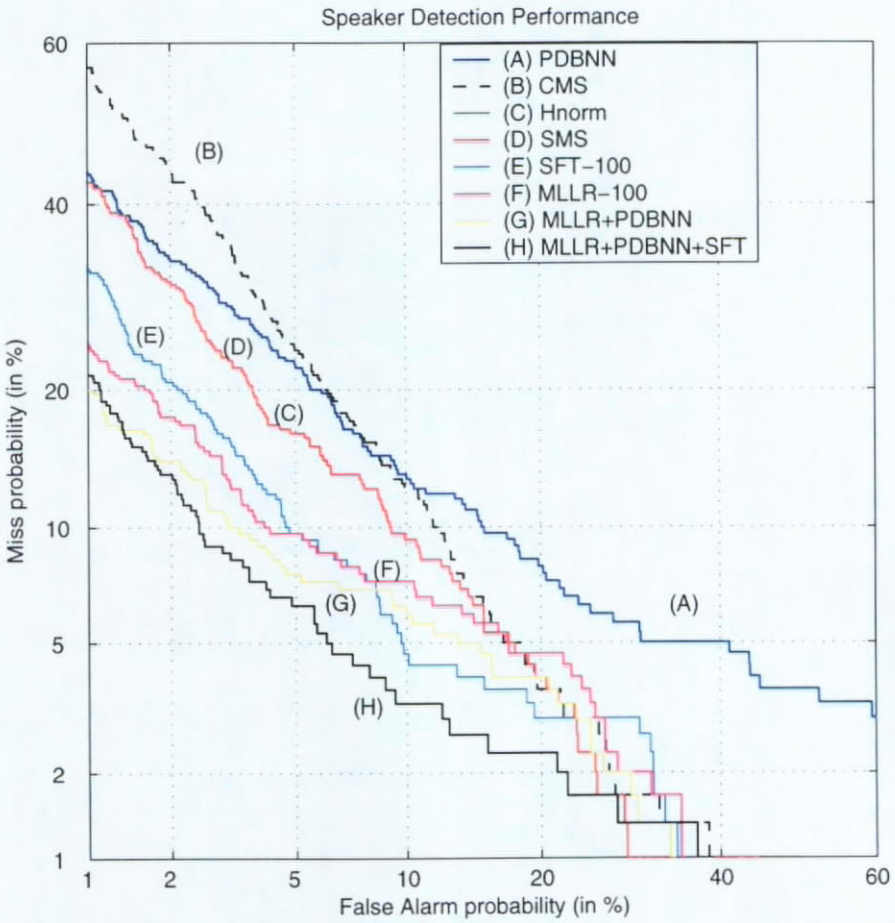


Figure 5.4: DET curves corresponding to handset el2 based on different environment adaptation approaches: PDBNN, cepstral mean subtraction (CMS), Handset normalization (Hnorm), speaker model synthesis (SMS), stochastic feature transformation (SFT), MLLR, MLLR+PDBNN, and MLLR+PDBNN+SFT. For ease of comparison, methods labeled from (A) to (H) in the legend are arranged in descending order of EERs.

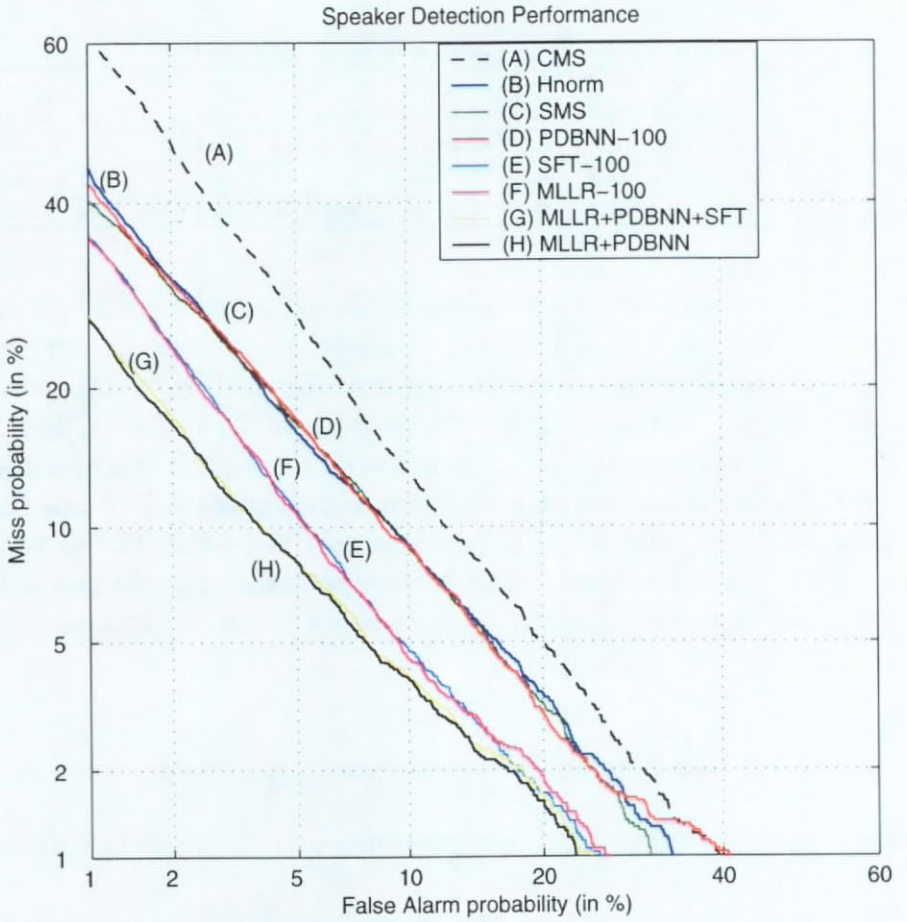


Figure 5.5: Composite DET curves based on different environment adaptation approaches: cepstral mean subtraction (CMS), Handset normalization (Hnorm), speaker model synthesis (SMS), PDBNN, stochastic feature transformation (SFT), MLLR, MLLR+PDBNN+SFT, and MLLR+PDBNN. The composite DET curve is created by merging the scores of all speakers and handsets. For ease of comparison, methods labeled from (A) to (H) in the legend are arranged in descending order of EERs.

Adaptation Method	Equal Error Rate (%)											
	cb1	cb2	cb3	cb4	el1	el2	el3	el4	pt1	senh	Average	Global
CMS	8.21	8.50	21.20	15.40	8.15	11.20	11.49	8.85	10.56	6.79	**11.03**	**11.29**
Tnorm	8.88	8.94	22.58	14.94	9.30	9.78	10.40	8.64	8.51	5.54	**10.74**	**11.39**
Hnorm	7.30	6.98	13.81	10.42	7.42	9.40	10.32	7.62	9.34	7.06	**8.96**	**9.51**
SMS	6.61	5.78	14.93	10.99	6.68	7.94	8.98	6.88	7.86	5.55	**8.22**	**9.51**
PDBNN-100	7.72	8.48	10.02	9.66	6.72	11.59	8.64	9.59	8.99	3.01	**8.44**	**9.34**
SFT-20	4.18	3.61	17.64	11.81	4.93	7.24	7.82	3.85	6.64	3.60	**7.13**	**7.42**
SFT-100	4.28	3.61	17.55	11.06	4.81	7.60	7.34	3.87	6.12	3.65	**6.98**	**7.15**
MLLR-20	4.69	3.34	17.23	10.21	5.52	7.35	9.66	4.76	8.84	3.54	**7.51**	**8.33**
MLLR-100	4.52	3.14	15.17	9.58	4.79	7.60	6.46	4.84	6.94	3.69	**6.67**	**7.12**
MLLR-100+Hnorm	5.01	4.12	15.22	9.34	4.95	7.01	5.96	4.29	6.65	3.69	**6.62**	**7.77**
SFT-20+Hnorm	4.18	3.61	17.64	11.81	4.93	7.24	7.87	3.85	6.64	3.60	**7.13**	**7.42**
MLLR+PDBNN-100	4.30	3.12	10.16	6.89	5.01	6.84	6.36	4.87	6.53	3.43	**5.75**	**6.40**
MLLR+PDBNN+SFT-100	4.21	3.01	12.86	7.65	4.97	5.65	6.05	4.26	5.46	3.43	**5.75**	**6.53**

Table 5.2: Equal error rates (in %) achieved by cepstral mean subtraction (CMS), Tnorm, Hnorm, speaker model synthesis (SMS), PDBNN adaptation, stochastic feature transformation (SFT), SFT+Hnorm, MLLR, MLLR+Hnorm, MLLR+PDBNN, and MLLR+PDBNN+SFT adaptation. Note that CMS and Tnorm do not require the handset selector. All results were based on 100 target speakers and 50 impostors. The MLLR and SFT transformation matrices were estimated using 20 and 100 speakers (denoted as '-20' and '-100', respectively) in the speaker set. The label "Average" refers to the average of handset-dependent EERs, and the label "Global" refers to the handset-independent EERs obtained by pooling the scores of all handsets.

demonstrates that model-based adaptation and feature-based transformation (SFT) are comparable in terms of error rate reduction. The results are also consistent with those of Figure 5.5, which shows that a large portion of the DET curve of SFT-100 overlaps with that of MLLR-100.

Although PDBNN adaptation uses discriminative training to adapt the model parameters to fit a new environment, the results show that using PDBNN adaptation alone is not desirable because its performance is inferior to that of MLLR adaptation. This may be due to the intrinsic design of PDBNNs, i.e., the supervised reinforced/antireinforced learning of PDBNNs are designed for fine-tuning the decision

boundaries by slightly adjusting the model centers responsible for the misclassifications. As a result, some of the centers will not be adapted at all. Because only misclassified data are used for adaptation and their amount is usually small after LU training, moving all the centers from the clean space to the environmentally distorted space may be difficult. On the other hand, MLLR adaptation finds a transformation matrix to maximize the likelihood of the adaptation data. As a result, moving all of the model centers to the environmentally distorted space is much easier.

PDBNN adaptation requires speaker-specific training data from all possible handsets that the users may use. MLLR adaptation, on the other hand, only requires some environment-specific utterances to estimate the global transformation matrices, which requires much less training data; better still, the utterances do not necessarily need to be produced by the same client speaker. PDBNN adaptation also requires additional training for the inclusion of new speakers because the new speaker models and background models should be adapted using gradient descent to environment-dependent models. On the other hand, for MLLR adaptation, transformation can be applied to create new speaker models and background models once the MLLR transformation matrices have been estimated.

The results also demonstrate that SFT and MLLR achieve a comparable amount of error reduction. A comparison between SFT-20 and MLLR-20 (where the training utterances of 20 speakers were used to estimate the transformation parameters) reveals that SFT performs slightly better when the amount of training data is small. This is because the number of free parameters in feature transformation is much less than that of MLLR. However, the performance of SFT quickly saturates when the number

of training patterns increases, as indicated in SFT-100 and MLLR-100. While MLLR requires much more data to estimate the global transformation matrices robustly, its performance is better than that of SFT when sufficient training data are available.

As shown in Table 5.2, the EERs achieved by Tnorm and CMS are very close. Note that the compensation in Tnorm and CMS are "blind" in that no handset information is used. The results show that Hnorm outperforms the classical CMS; however, the improvement is not as significant as SFT and MLLR. Recall that both Hnorm and Tnorm work in the likelihood-ratio score space using two normalization parameters only (bias and scale). SFT and MLLR, on the other hand, compensate for the mismatch at the feature space and model space, respectively. Both methods can translate and scale the components of feature vectors or models' centers. Because the number of free parameters in SFT and MLLR is much larger than that of Hnorm and Tnorm, SFT and MLLR are more effective provided that their free parameters can be estimated correctly.

Table 5.2 and Figure 5.5 also show that SMS outperforms the classical CMS and achieves a performance comparable to Hnorm, which agrees with the results in [99].

Because SFT and Hnorm work in different spaces (SFT transforms speech features in the feature space and Hnorm performs score normalization in the score space), we have also combined these two techniques to see whether further improvement can be made. The results ("SFT-20+Hnorm" in Table 5.2) show that no improvement was obtained when compared with SFT-20. We have also combined MLLR and Hnorm to see the effect of combining model transformation with Hnorm. The results ("MLLR-100+Hnorm" in Table 5.2) show that MLLR+Hnorm performs slightly worse than

that of MLLR alone in terms of EER. A possible reason is that the SFT (MLLR) has already transformed the features (clean models) to a region close to the clean features (distorted features); this action effectively achieves what Hnorm is supposed to achieve. Therefore, performing Hnorm on SFT scores or MLLR scores has little effect. Another possible reason is that all of these methods (SFT, MLLR, and Hnorm) use the same handset selector for choosing their parameters. Because the handset information is identitical for each verification session, no additional benefit can be obtained by combining SFT and Hnorm or MLLR and Hnorm.

Although it may not be desirable to use PDBNN's reinforced learning alone, it is amenable to the fine-tuning of MLLR-transformed centers. This is evident from the row MLLR+PDBNN in Table 5.2, where error reductions of 10% with respect to MLLR-100 and 31% with respect to PDBNN-100 were achieved. The main reason behind these error reductions is that MLLR can transform the clean model centers to the region of distorted clusters. These initial centers give a better "seed" for the reinforced learning.

For MLLR+PDBNN+SFT adaptation, an EER of 6.53% is achieved (the bottom row of Table 5.2), which is slightly higher than that of MLLR+PDBNN adaptation (6.40%). The DET curves of MLLR+PDBNN+SFT and MLLR+PDBNN in Figure 5.5 are highly overlapped, which suggests that the performance of MLLR+PDBNN with and without using stochastically-transformed data are almost identical for a wide range of decision thresholds. Bear in mind that MLLR+PDBNN adaptation requires speaker- and handset-dependent training data, whereas MLLR+PDBNN+SFT adaptation requires handset-dependent training data only. Therefore, in terms of

practicality, MLLR+PDBNN+SFT adaptation has advantage over MLLR+PDBNN adaptation.

5.6.2 Comparison in Terms of Computational Complexity

In this section, we examine the computational complexity of MLLR and PDBNN adaptation. Suppose the speaker model is a GMM with M_1 components and the background model is a GMM with M_2 components. Let D be the dimension of feature vectors, and T be the number of frames in the adaptation data.

PDBNN

In PDBNN adaptation, for each epoch, Eq. 2.22 is used to compute a segmental score for each training segment. Let S_1 and S_2 be the numbers of speaker's segments and background segments, respectively and N be the size of each segment. The time complexity is $O((S_1 + S_2)ND(M_1 + M_2))$. If that segment is misclassified, reinforced or anti-reinforced learning will be applied, which will require $O(ND(M_1 + M_2))$ operations. Therefore, PDBNN has an overall complexity of $O((S_1 + S_2)ND(M_1 + M_2))$.

MLLR

Suppose a global transformation matrix is used to compensate the channel mismatch effect. Based on the notation of [95], computing the posterior probability requires $O(TDM_2)$ operations where T, D, and M_2 represents the number of adaptation vectors, the dimension of feature vectors, and the number of Gaussians in the GMM, respectively. To compute the intermediate variables h_{ij} and z_{ij}, MLLR requires

$O(TD^2M_2)$ operations. To compute W, we need to solve a system of $D\times(D+1)$ linear equations, which requires $O(D^4)$ operations. Updating the means requires multiplying the mean vector by the adaptation matrix W, which needs $O(M_2D^2)$ operations. Therefore, MLLR has an overall complexity of $O(D^4 + M_2D^2 + TD^2M_2)$.

To compare the computation time of different adaptation approaches, we have measured their training and verification time. The simulations were performed on a Pentium IV 2.66GHz CPU. The measurements were based on 30 speakers and 50 impostors in the HTIMIT corpus and the total CPU time running the whole task was recorded.

The training time is composed of two parts: systemwise computation time and enrollment computation time. The former represents the time to carry out the tasks that only need to be performed once. These tasks include the computation of MLLR transformation matrices and background model training. The enrollment computation time represents the time to enroll a new speaker and to adapt his or her models. It was determined by taking the average enrollment time of 30 client speakers using seven utterances per speaker. The verification time is the time taken to verify a claimant based on a single utterance. It was determined by taking the average verification time of 80 claimants (30 client speakers and 50 impostors) using three utterances per claimant.

Table 5.3 shows the training and verification time for Hnorm, Tnorm, PDBNN-100, MLLR-100, MLLR+PDBNN-100 and MLLR+PDBNN+SFT-100. Evidently, PDBNN adaptation requires an extensive amount of computation resources during enrollment because a handset-dependent speaker model and a handset-dependent

background model were created for each speaker and acoustic environment. It is of interest to compare the enrollment time of the techniques that involve PDBNN adaptation. Table 5.3 shows that the enrollment time for PDBNN-100 is longer than that of MLLR+PDBNN-100 and MLLR+PDBNN+SFT-100. Because MLLR moves the model centers to a feature space close to the adapted data, PDBNN adaptation converges rapidly. As a result, the enrollment time for MLLR+PDBNN-100 and MLLR+PDBNN+SFT-100 is less than that for PDBNN-100.

The long systemwise computation time of the methods that involve MLLR suggests that computing the MLLR transformation matrices requires an extensive amount of computation resources. This is because to use MLLR adaptation, it is required to estimate 10 transformation matrices for 10 different environments offline, and for each environment, patterns from all background speakers are involved in the estimation of the transformation matrix. However, once the matrices have been computed, MLLR adaptation requires considerably less computation resources than PDBNN adaptation to enroll a new speaker (compare the enrollment computation time of PDBNN-100 and MLLR-100 in Table 5.3). This is because to enroll a speaker, PDBNN adaptation requires to create 10 speaker models and 10 background models for 10 different acoustic environments, whereas the MLLR transformation matrices have been precomputed offline. Fortunately, creating a speaker model by PDBNN adaptation involves the patterns from the corresponding speaker and the background speakers only, and adaptation takes place only for those patterns that cause misclassification.

Table 5.3 shows that Hnorm requires considerably more computation resources

Adaptation Method	Training Time		Verification Time
	Systemwise Computation Time (seconds)	Enrollment Computation Time (seconds)	(seconds)
Hnorm	324.30	240.78	2.48
Tnorm		0.96	37.54
PDBNN-100		8,489.54	1.20
MLLR-100	130,713.30	0.96	0.38
MLLR+PDBNN-100		2,561.74	1.59
MLLR+PDBNN+SFT-100		5,157.77	1.27

Table 5.3: Training and verification time used by different adaptation approaches. The training time is composed of two parts: systemwise computation time and enrollment computation time. The former represents the time to carry out the tasks that only need to be performed once. These tasks include the computation of the MLLR transformation matrices and background model training. The enrollment computation time represents the time to enroll a new speaker and to adapt his/her models.

than Tnorm during enrollment. However, Tnorm takes a longer time to verify a speaker than Hnorm. Although both Hnorm and Tnorm work in the likelihood ratio score space, they have different complexity when it comes to training and testing. The main computation of Hnorm lies in the training phase and is proportional to the total number of nontarget utterances because these utterances are fed to the speaker and background models to calculate the nontarget scores, which in turn are used to calculate the normalization parameters. Once the Hnorm parameters have been calculated, rapid verification can be achieved. For Tnorm, on the other hand, verification is computationally intensive and the amount of computation is proportional to the total number of speaker models used for deriving the normalization parameters. For a better estimation of Tnorm parameters, a large number of scores are required, which in turn require a large number of speaker models. This can greatly lengthen the verification time. Therefore, among all adaptation methods, only Tnorm cannot

achieve near realtime performance during verification.

5.6.3 Comparison in Terms of Storage Requirements

Because storage requirements are also important in practical implementation, it is of interest to compare the disk space usage of different adaptation methods. Table 5.4 shows that PDBNNs adaptation requires the largest amount of disk space because it requires the storage of one speaker model and one background model for each handset and target speaker. On the other hand, Hnorm, MLLR, and SFT require a small amount of disk space per handset. The disk space requirement of PDBNN is proportional to the number of acoustic environments, E, because it is necessary to adapt a handset-specific speaker model and a handset-specific background model for each environment. Note that Tnorm does not require any disk storage because its normalization parameters are computed during verification.

5.7 Conclusions

We have presented model adaptation and transformation approaches to addressing the problem of environmental mismatch in telephone-based speaker verification systems. A handset selector is combined with (1) handset-dependent model transformation, (2) reinforced learning, and (3) stochastic feature transformation to reduce the effect caused by the acoustic distortion. Experimental results based on 150 speakers of the HTIMIT corpus show that the model adaptation based on the combination of MLLR and PDBNN outperforms a number of classical techniques, including CMS, stochastic feature transformation, Hnorm, and speaker model synthesis.

Adaptation Method	Number of Parameters	Disk Usage in Bytes
CMS (No adaptation)	$PN_c + N_b$	326,400
Hnorm	$PN_c + N_b + 2DE$	327,360
SMS	$PN_b + EN_b$	704,000
MLLR	$PN_c + N_b + D(D+1)E$	332,640
SFT	$PN_c + N_b + 2DE$	327,360
PDBNN	$(PN_c + N_b)E$	3,264,000

P = Population size (100)

D = Feature dimension (12)

N_c = Number of parameters per speaker model ($= [1 + D + D]32 = 800$)

N_b = Number of parameters per background model ($= [1 + D + D]64 = 1600$)

E = number of acoustic environments (10)

Table 5.4: Disk space requirements of different adaptation methods. The numbers inside brackets are the values of the corresponding parameters used in the experiments. *Note:* the figures in the last column are calculated by assuming that each floating-point number occupies 4 bytes.

Chapter 6

FEATURE TRANSFORMATION

One of the key problems in phone-based speaker verification is the acoustic mismatch between speech gathered from different handsets. One possible approach to resolving the mismatch problem is *feature transformation.* This chapter focuses on feature-based compensation and highlights the state-of-the-art techniques that are related to this thesis.

6.1 Cepstral Mean Subtraction

When signal $x[n]$ passes through a linear filter with impulse response $h[n]$, we obtain a signal $y[n]$

$$y[n] = x[n] * h[n], \tag{6.1}$$

where $*$ denotes convolution. Based on Eq. 6.1, we can express the cepstrum of $y[n]$ in terms of the cepstrum of $x[n]$ and $h[n]$ as follows:

$$\mathbf{y}_t = \mathbf{x}_t + \mathbf{h}, \tag{6.2}$$

where t denotes the frame index. For an utterance with T frames, the mean of $\mathbf{y}_t$ is given by

$$\bar{\mathbf{y}} = \frac{1}{T}\sum_{t=0}^{T-1}\mathbf{y}_t = \frac{1}{T}\sum_{t=0}^{T-1}(\mathbf{x}_t + \mathbf{h}) = \bar{\mathbf{x}} + \mathbf{h}. \tag{6.3}$$

Cepstral mean subtraction (CMS) [8, 28] consists of subtracting the sample mean $\bar{\mathbf{y}}$ from the cepstral vector $\mathbf{y}_t$ to form the normalized cepstral vector $\hat{\mathbf{x}}_t$, i.e.,

$$\hat{\mathbf{x}}_t = \mathbf{y}_t - \bar{\mathbf{y}}. \tag{6.4}$$

Substituting Eqs. 6.2 and 6.3 into Eq. 6.4, we obtain

$$\begin{aligned} \hat{\mathbf{x}}_t &= \mathbf{x}_t + \mathbf{h} - (\bar{\mathbf{x}} + \mathbf{h}) \\ &= \mathbf{x}_t - \bar{\mathbf{x}}. \end{aligned} \tag{6.5}$$

Note that the normalized cepstrum $\hat{\mathbf{x}}_t$ will become close to the clean cepstrum $\mathbf{x}_t$ when $\bar{\mathbf{x}} \approx 0$. In other words, CMS can remove the channel cepstrum $\mathbf{h}$ from channel distorted speech $\mathbf{y}_t$ when the expectation of clean speech cepstrum is zero.

It has been shown that cepstral mean subtraction can minimize the filtering effect of linear channels [8]. The method provides significant robustness against channel variations as long as the utterances have at least 2 seconds of speech [8]. To use the method effectively, mean subtraction should be performed on every utterance for both training and recognition.

However, considerable loss of recognition accuracy is experienced when CMS is used for speaker recognition in which both training and recognition are performed on the same channel. This is due to the implicit assumption of CMS: the long-term cepstral mean of clean speech is zero. This assumption is correct only if the utterances are phonetically balanced; in other words, the speech in each utterance should contain approximately the same amount of voiced, unvoiced, and plosive sound.

6.2 Signal Bias Removal

Rahim and Juang [77] proposed an iterative technique called signal bias removal (SBR) to minimize the effects of channel mismatch. SBR can be considered as a blind equalization method that reduces the acoustic differences between the training and testing environments. The method can deal with both additive cepstral bias and additive spectral bias.

In SBR, the channel is represented by an additive bias term $\mathbf{b}$. This bias term is estimated from the distorted cepstral vectors using the following maximum-likelihood formulation:

$$p(\mathbf{Y}|\bar{\mathbf{b}}, \Lambda) = \max_{\mathbf{b},i} \left[\prod_{t=1}^{T} p(\mathbf{y}_t - \mathbf{b}|\lambda_i) \right], \tag{6.6}$$

where $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_t, \ldots, \mathbf{y}_T\}$ is an observation sequence of T frames and $\Lambda = \{\lambda_i; i = 1, 2, \ldots, M\}$ contains the hidden Markov models of M different speech units. For a given Λ, the maximum-likelihood estimator, $\bar{\mathbf{b}}$, is the one that satisfies Eq. 6.6.

SBR uses a two-step iterative procedure to remove channel bias. In the first step, given a set of centroids μ_i of distorted speech, the channel bias $\bar{\mathbf{b}}$ is estimated from the test utterance, i.e.,

$$\bar{\mathbf{b}} = \frac{1}{T} \sum_{t=1}^{T} (\mathbf{y}_t - \mu_{i^*}), \tag{6.7}$$

where T is the number of frames in the utterance and μ_{i^*} is the nearest neighbor to $\mathbf{y}_t$ according to a distance criterion that is consistent with the probability distribution of the distorted signal $\mathbf{y}_t$:

$$i^* = \arg\min_i \|\mathbf{y}_t - \mu_i\|. \tag{6.8}$$

In the second step, the estimated bias, $\bar{\mathbf{b}}$, is subtracted from the distorted signal to

recover the undistorted cepstrum, i.e.,

$$\hat{\mathbf{x}}_t = \mathbf{y}_t - \bar{\mathbf{b}} \qquad \forall t. \tag{6.9}$$

These two-step procedure results in a maximization of Eq. 6.6. The procedure is iterated until a local optimal solution for $\mathbf{b}$ is reached.

Instead of using the cepstral mean as the channel bias as in CMS, SBR estimates the maximum likelihood of the bias. Experimental results suggest that SBR outperforms than CMS [77].

6.3 Codeword-Dependent Cepstral Normalization

In codeword-dependent cepstral normalization (CDCN) [1, 3], the additive noise and channel equalization vectors are estimated simultaneously using the EM algorithm. Specifically, the distorted cepstrum is expressed as

$$\mathbf{y} = \mathbf{x} + \mathbf{h} + \mathbf{r}(\mathbf{x}, \mathbf{n}, \mathbf{h}), \tag{6.10}$$

where $\mathbf{x}$, $\mathbf{n}$, and $\mathbf{h}$ represent the cepstrum of clean speech, additive noise, and channel's impluse response, respectively, and $\mathbf{r}(\mathbf{x}, \mathbf{n}, \mathbf{h})$ is a nonlinear function of $\mathbf{x}$, $\mathbf{n}$, and $\mathbf{h}$. The goal of CDCN is to estimate the uncorrupted cepstral vectors $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_T\}$ from a distorted test utterance $\mathbf{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_T\}$.

Acero etal. [1, 3] showed that CDCN provides dramatic improvement in performance when a speech recognizer is trained on speech recorded from one microphone and tested on speech recorded from another microphone. It was also shown that CDCN does not degraded performance when both training and testing use the same microphone.

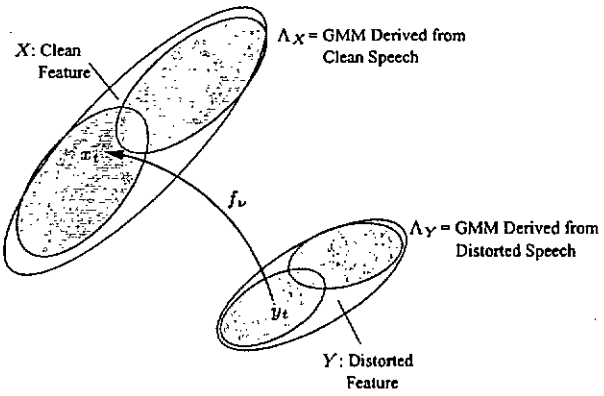
6.4 *Stochastic Feature Transformation*

Stochastic feature transformation (SFT) [57, 58] is inspired by the stochastic matching method of Sankar and Lee [93]. Stochastic matching was originally proposed for speaker adaptation and channel compensation. Its main goal is to transform the distorted data to fit the clean speech models or to transform the clean speech models to better fit the distorted data. In the case of feature transformation, the channel is represented by either a single cepstral bias ($\mathbf{b} = [b_1, b_2, \ldots, b_D]^T$) or a bias together with an affine transformation matrix ($A = \text{diag}\ \{a_1, a_2, \ldots, a_D\}$). In the latter case, the componentwise form of the transformed vectors is given by

$$\hat{x}_{t,i} = f_\nu(\mathbf{y}_t)_i = a_i y_{t,i} + b_i, \tag{6.11}$$

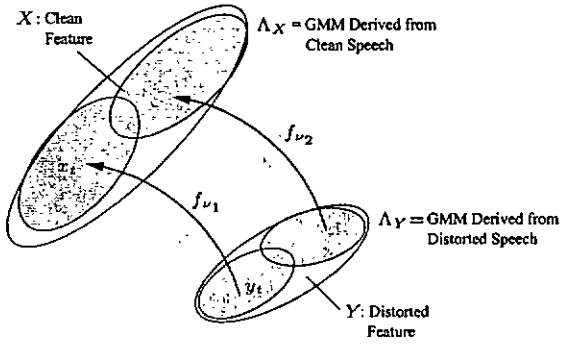
where $\mathbf{y}_t$ is a D-dimensional distorted vector, $\nu = \{a_i, b_i\}_{i=1}^D$ is the set of transformation parameters, and $f_\nu(\cdot)$ denotes the transformation function. Intuitively, the bias $\mathbf{b}$ compensates the convolutive distortion and the matrix A compensates the effects of noise. Figure 6.1(a) illustrates the concept of stochastic feature transformation with a single set of linear transformation parameters ν per handset.

The first-order transformation in Eq. 6.11, however, has two limitations. First, it assumes that all speech signals are subject to the same degree of distortion, which may be incorrect for nonlinear channels where signals with higher amplitude are subject to a higher degree of distortion, because of the saturation effect in transducers. Second, the use of a single transformation matrix is inadequate for an acoustic environment with varying noise levels. Mak and Kung [47, 57] proposed to overcome these limitations by nonlinear transformation.



$$\nu' = \arg\max_{\nu} p(f_\nu(Y)|\Lambda_X, \nu)$$
$$\hat{x}_{t,i} = f_\nu(\mathbf{y}_t)_i = a_i y_{t,i} + b_i$$

(a)



$$\nu = (\nu_1, \nu_2)$$
$$\nu' = \arg\max_{\nu} p(f_\nu(Y)|\Lambda_X, \nu)$$
$$\hat{x}_{t,i} = \sum_{k=1}^{2} g_k(\mathbf{y}_t)(c_{ki} y_{t,i}^2 + a_{ki} y_{t,i} + b_{ki})$$

(b)

Figure 6.1: The idea of stochastic feature transformation is illustrated here. (a) Linear transformation with a single set of transformation parameters per handset and (b) nonlinear transformation with two sets of transformation parameters (ν_1 and ν_2) per handset. (Source: S. Y. Kung, M. W. Mak and S. H. Lin, *Biometric Authentication: A Machine Learning Approach*, Prentice Hall, 2004.)

6.4.1 Nonlinear Feature Transformation

The proposal in [47, 57] is based on the notion that different transformation matrices and bias vectors can be applied to transform the vectors in different regions of the feature space. This can be achieved by extending Eq. 6.11 to

$$\hat{x}_{t,i} = f_\nu(\mathbf{y}_t)_i = \sum_{k=1}^{K} g_k(\mathbf{y}_t)(c_{ki}y_{t,i}^2 + a_{ki}y_{t,i} + b_{ki}), \tag{6.12}$$

where $\nu = \{a_{ki}, b_{ki}, c_{ki}; k = 1, \ldots, K; i = 1, \ldots, D\}$ is the set of transformation parameters and

$$g_k(\mathbf{y}_t) = P(k|\mathbf{y}_t, \Lambda_Y) = \frac{\omega_k^Y p(\mathbf{y}_t|\mu_k^Y, \Sigma_k^Y)}{\sum_{l=1}^{K} \omega_l^Y p(\mathbf{y}_t|\mu_l^Y, \Sigma_l^Y)} \tag{6.13}$$

is the posterior probability of selecting the k-th transformation given the distorted speech $\mathbf{y}_t$. Note that the selection of transformation is probabilistic and data-driven. In Eq. 6.13, $\Lambda_Y = \{\omega_k^Y, \mu_k^Y, \Sigma_k^Y\}_{k=1}^K$ is the speech model that characterizes the distorted speech and

$$p(\mathbf{y}_t|\mu_k^Y, \Sigma_k^Y) = (2\pi)^{-\frac{D}{2}}|\Sigma_k^Y|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(\mathbf{y}_t - \mu_k^Y)^T(\Sigma_k^Y)^{-1}(\mathbf{y}_t - \mu_k^Y)\right\} \tag{6.14}$$

is the density of the k-th distorted cluster. Figure 6.1(b) illustrates the idea of nonlinear feature transformation with different transformations for different regions of the feature space. Note that when $K = 1$ and $c_{ki} = 0$, Eq. 6.12 is reduced to Eq. 6.11 (i.e., the standard stochastic matching is a special case of the proposed approach).

Given a clean speech model $\Lambda_X = \{\omega_j^X, \mu_j^X, \Sigma_j^X\}_{j=1}^M$ derived from the clean speech of several speakers, the maximum-likelihood estimates of ν can be obtained by max-

imizing an auxiliary function

$$Q(\nu'|\nu) = \sum_{t=1}^{T}\sum_{j=1}^{M}\sum_{k=1}^{K} h_j(f_\nu(\mathbf{y}_t))g_k(\mathbf{y}_t)\log\left\{\omega_j^X\omega_k^Y p(\mathbf{y}_t|\mu_j^X,\Sigma_j^X,\nu_k')\right\}$$
$$= \sum_{t=1}^{T}\sum_{j=1}^{M}\sum_{k=1}^{K} h_j(f_\nu(\mathbf{y}_t))g_k(\mathbf{y}_t)\log\left\{\omega_j^X\omega_k^Y p(f_{\nu_k'}(\mathbf{y}_t)|\mu_j^X,\Sigma_j^X)|J_{\nu_k'}(\mathbf{y}_t)|\right\} \tag{6.15}$$

with respect to ν'. In Eq. 6.15, ν' and ν represent the new and current estimates of the transformation parameters, respectively. T is the number of distorted vectors; $\nu_k' = \{a_{ki}', b_{ki}', c_{ki}'\}_{i=1}^{D}$ denotes the k-th transformation; $|J_{\nu_k'}(\mathbf{y}_t)|$ is the determinant of the Jacobian matrix, the (r,s)-th entry of which is given by $J_{\nu_k'}(\mathbf{y}_t)_{rs} = \partial f_{\nu_k'}(\mathbf{y}_t)_s/\partial y_{t,r}$; and $h_j(f_\nu(\mathbf{y}_t))$ is the posterior probability given by

$$h_j(f_\nu(\mathbf{y}_t)) = P(j|\mathbf{y}_t,\Lambda_X,\nu) = \frac{\omega_j^X p(f_\nu(\mathbf{y}_t)|\mu_j^X,\Sigma_j^X)}{\sum_{l=1}^{M}\omega_l^X p(f_\nu(\mathbf{y}_t)|\mu_l^X,\Sigma_l^X)}, \tag{6.16}$$

where

$$p(f_\nu(\mathbf{y}_t)|\mu_j^X,\Sigma_j^X) = (2\pi)^{-\frac{D}{2}}|\Sigma_j^X|^{-\frac{1}{2}} \cdot$$
$$\exp\left\{-\frac{1}{2}(f_\nu(\mathbf{y}_t)-\mu_j^X)^T(\Sigma_j^X)^{-1}(f_\nu(\mathbf{y}_t)-\mu_j^X)\right\}. \tag{6.17}$$

Ignoring the terms independent of ν' and assuming diagonal covariance (i.e., Σ_j^X $= \text{diag}\left\{(\sigma_{j1}^X)^2,\ldots,(\sigma_{jD}^X)^2\right\}$, and likewise for Σ_k^Y), Eq. 6.15 can be written as

$$Q(\nu'|\nu) = \sum_{t=1}^{T}\sum_{j=1}^{M}\sum_{k=1}^{K} h_j(f_\nu(\mathbf{y}_t))g_k(\mathbf{y}_t) \cdot$$
$$\left\{-\frac{1}{2}\sum_{i=1}^{D}\frac{(c_{ki}'y_{t,i}^2 + a_{ki}'y_{t,i} + b_{ki}' - \mu_{ji}^X)^2}{(\sigma_{ji}^X)^2} + \sum_{i=1}^{D}\log(2c_{ki}'y_{t,i} + a_{ki}')\right\} \tag{6.18}$$

The generalized EM algorithm can be applied to find the maximum-likelihood estimates of ν. Specifically, in the E-step, Eqs. 6.16 and 6.17 are used to compute

$h_j(f_\nu(\mathbf{y}_t))$ and Eqs. 6.13 and 6.14 are used to compute $g_k(\mathbf{y}_t)$; then in the M-step, ν' is updated according to

$$\nu' \leftarrow \nu' + \eta \frac{\partial Q(\nu'|\nu)}{\partial \nu'}, \tag{6.19}$$

where η is a positive learning factor and $\eta = 0.001$ in this work. $\partial Q(\nu'|\nu)/\partial \nu'$ is the derivative of $Q(\nu'|\nu)$ with respect to a'_{ki}, b'_{ki} and c'_{ki}, that is,

$$\frac{\partial Q(\nu'|\nu)}{\partial a'_{ki}} = \sum_{t=1}^{T}\sum_{j=1}^{M} h_j(f_\nu(\mathbf{y}_t))g_k(\mathbf{y}_t) \cdot \left\{ -\frac{y_{t,i}(c'_{ki}y_{t,i}^2 + a'_{ki}y_{t,i} + b'_{ki} - \mu_{ji}^X)}{(\sigma_{ji}^X)^2} + \frac{1}{2c'_{ki}y_{t,i} + a'_{ki}} \right\} \tag{6.20}$$

$$\frac{\partial Q(\nu'|\nu)}{\partial b'_{ki}} = \sum_{t=1}^{T}\sum_{j=1}^{M} h_j(f_\nu(\mathbf{y}_t))g_k(\mathbf{y}_t) \cdot \left\{ -\frac{(c'_{ki}y_{t,i}^2 + a'_{ki}y_{t,i} + b'_{ki} - \mu_{ji}^X)}{(\sigma_{ji}^X)^2} \right\} \tag{6.21}$$

$$\frac{\partial Q(\nu'|\nu)}{\partial c'_{ki}} = \sum_{t=1}^{T}\sum_{j=1}^{M} h_j(f_\nu(\mathbf{y}_t))g_k(\mathbf{y}_t) \cdot \left\{ -\frac{y_{t,i}^2(c'_{ki}y_{t,i}^2 + a'_{ki}y_{t,i} + b'_{ki} - \mu_{ji}^X)}{(\sigma_{ji}^X)^2} + \frac{2y_{t,i}}{2c'_{ki}y_{t,i} + a'_{ki}} \right\}. \tag{6.22}$$

These E- and M-steps are repeated until $Q(\nu'|\nu)$ ceases to increase. In [47, 57], Eq. 6.19 was repeated 20 times in each M-step. The exact number of iterations in the M-step is not important as long as $Q(\nu'|\nu)$ is increasing in each iteration. The most significant improvement occurs during the first 5 iterations. It was found that using 20 iterations in the M-step gives better performance and faster convergence.

The posterior probabilities $g_k(\mathbf{y}_t)$ and $h_j(f_\nu(\mathbf{y}_t))$ suggest that there are K regions in the distorted feature space and M regions in the clean feature space. As a result, there are KM possible transformations; however, this number can be reduced to K

by arranging the indexes j and k such that the symmetric divergence

$$D(\Lambda_{X,j}||\Lambda_{Y,k}) = \frac{1}{2}tr\left\{(\Sigma_j^X)^{-1}\Sigma_k^Y + (\Sigma_k^Y)^{-1}\Sigma_j^X - 2I\right\} + \frac{1}{2}(\mu_j^X - \mu_k^Y)^T\left[(\Sigma_k^Y)^{-1} + (\Sigma_j^X)^{-1}\right](\mu_j^X - \mu_k^Y)$$

between the j-th mixture of Λ_X and the k-th mixture of Λ_Y is minimal.

6.4.2 Piecewise-Linear Feature Transformation

When $c_{ji} = 0$, Eq. 6.12 becomes a piecewise linear version of the standard stochastic matching in Eq. 6.11. The maximum-likelihood estimate of ν can be obtained by the EM algorithm. Specifically, in the M-step, the derivative of Eq. 6.18 with respect to ν' is set to 0, which results in

$$s_{ki}{a'_{ki}}^2 + (u_{ki} - q_{ki}b'_{ki})a'_{ki} - v_k = 0 \quad \text{and} \quad q_{ki}a'_{ki} + r_{ki}b'_{ki} - p_{ki} = 0, \tag{6.23}$$

where

$$\begin{aligned}
p_{ki} &= \sum_{t=1}^{T}\sum_{j=1}^{M} h_{tj}g_{tk}\mu_{ji}^X/(\sigma_{ji}^X)^2, \\
q_{ki} &= \sum_{t=1}^{T}\sum_{j=1}^{M} h_{tj}g_{tk}y_{t,i}/(\sigma_{ji}^X)^2, \\
r_{ki} &= \sum_{t=1}^{T}\sum_{j=1}^{M} h_{tj}g_{tk}/(\sigma_{ji}^X)^2, \\
s_{ki} &= \sum_{t=1}^{T}\sum_{j=1}^{M} h_{tj}g_{tk}y_{t,i}^2/(\sigma_{ji}^X)^2, \\
u_{ki} &= \sum_{t=1}^{T}\sum_{j=1}^{M} h_{tj}g_{tk}\mu_{ji}^X y_{t,i}/(\sigma_{ji}^X)^2, \text{ and} \\
v_k &= \sum_{t=1}^{T}\sum_{j=1}^{M} h_{tj}g_{tk},
\end{aligned}$$

where $h_{tj} \equiv h_j(f_\nu(\mathbf{y}_t))$ and $g_{tk} \equiv g_k(\mathbf{y}_t)$ are estimated during the E-step.

6.5 *Feature Mapping*

Feature mapping [85] is a general feature-domain channel-compensation technique that extends the mapping idea of speaker model synthesis (SMS) (see Section 4.8). Similar to SMS, a channel-independent root GMM is trained using all data from many channels; then channel-dependent GMMs are created by adapting the root GMM using channel-dependent data. Let GMM (CD1), $\Lambda^{CD1} = \{\omega_i^{CD1}, \mu_i^{CD1}, \Sigma_i^{CD1}\}$, be the channel-dependent GMM for channel CD1 and GMM (CI), $\Lambda^{CI} = \{\omega_i^{CI}, \mu_i^{CI}, \Sigma_i^{CI}\}$, be the channel-independent root model. The mapping of a feature $\mathbf{x}$ in the space modeled by Λ^{CD1} to the channel-independent feature, $\mathbf{y}$, is given by

$$\mathbf{y} = (\mathbf{x} - \mu_i^{CD1})\frac{\sigma_i^{CI}}{\sigma_i^{CD1}} + \mu_i^{CI},$$

where $i = \arg\max_{1 \le j \le M} \omega_j^{CD1} p_j^{CD1}(x)$ and M is the number of mixtures. This mapping transforms $\mathbf{x} \sim \mathcal{N}(\mu_i^{CD1}, \sigma_i^{CD1})$ into $\mathbf{y} \sim \mathcal{N}(\mu_i^{CI}, \sigma_i^{CI})$.

During verification, the handset type of the testing utterance is detected by a handset selector and then each of the feature vectors in the utterance is mapped to the channel-independent space based on the closest Gaussian in the channel-dependent GMM. The mapped features are then input to the speaker and root GMMs to compute a likelihood-ratio score.

Feature mapping has two advantages. First, the technique can be easily incorporated into any recognition systems without model retraining. Second, feature mapping allows the aggregation of information obtained from several channel types into a single channel-independent feature space. This capability is useful for model building or updating.

Feature mapping and SMS are related in that they both learn the transformations or mappings between channel-dependent models derived from a channel-independent model via MAP adaptation. However, there are also important differences. For example, while SMS focuses on synthesizing speaker models for different channels, feature mapping focuses on mapping features from different channels into a common channel-independent feature space. In this respect, both SMS and feature mapping are related to stochastic matching [93].

Experimental results based on NIST landline and cellular telephone speech corpora suggest that feature mapping performs significantly better than CMS, and its performance is similar to that of Hnorm and SMS.

6.6 Summary

Table 6.1 summarizes the characteristics of the feature-based compensation techniques discussed in this chapter, and Table 6.2 compares these techniques in terms of training methods, requirements on the amount of adaptation data, and computational complexity.

Feature-based Compensation	Characteristics
CMS	- Minimize linear channel effects by subtracting the mean cepstrum from individual cepstral vectors. - Assume that the expectation of clean speech cepstrum is zero. - Should be applied to both training and recognition utterances. - Performance degrades under matched conditions.
SBR	- Estimate channel bias based on maximum-likelihood methods. - Perform better than CMS.
CDCN	- Estimate additive background noise and channel distortion using the EM algorithm. - Improve performance under mismatch conditions and does not degrade performance under matched conditions. - Computationally demanding.
SFT	- Transform distorted data to better fit clean speech models. - Can be easily extended to nonlinear, multiple transformations to overcome nonlinear distortion. - Transformation parameters are estimated using the generalized EM algorithm. - Low computational complexity.
Feature Mapping	- Extension of SMS to the feature domain. - Based on MAP adaptation. - Make use of the one-to-one correspondence between individual Gaussians to compute the transformation between different channels.

Table 6.1: Summary of the state-of-the-art feature-based compensation techniques discussed in this chapter.

Feature-based Compensation	Training Methods	Requirement on the Amount of Adaptation Data	Computation Complexity
CMS	Not required	Small	Very low
SBR	Maximum-Likelihood (EM-based)	Moderate	Moderate
CDCN	Maximum-Likelihood (EM-based)	Moderate	High
SFT	Maximum-Likelihood (EM-based)	Moderate	Low
Feature Mapping	Bayesian (EM-based)	Moderate	Low

Table 6.2: Comparison of feature-based compensation techniques in terms of training methods, requirement on the amount of adaptation data, and computation complexity.

Chapter 7

FEATURE TRANSFORMATION FOR CHANNEL COMPENSATION

One popular approach to compensating for handset distortion is to divide handsets into several broad categories according to the type of transducer (e.g., carbon button and electret). During operations, a handset selector is used to identify the most likely handset type from speech signals and handset distortion is compensated for based on some a priori information about the identified type in the database. Although this method works well in landline phones, it may encounter difficulty in mobile handsets because they have a large number of categories, new handset models are frequently released, and models can become obsolete in a short time. Maintaining a handset database for storing the information of all possible handset models is a great challenge and updating the compensation algorithm whenever a new handset is released is also difficult. Therefore, it is imperative to develop a channel compensation method that does not necessarily require a priori knowledge of handsets. This chapter proposes a blind compensation algorithm to solve this problem. The algorithm is designed to handle the situation in which no a priori knowledge about the channel is available (i.e., a handset model not in the handset database is being used). Because the algorithm does not require a handset selector, it is suitable for a broader scale of deployment

than the conventional approaches.

7.1 *Blind Stochastic Feature Transformation*

Channel compensation can be divided into supervised or unsupervised. Supervised compensation assumes that the channel or handset characteristics are known a priori. Therefore, channel-specific compensation can be derived before recognition takes place. If handset labels are available during recognition, the corresponding channel-specific compensation can be applied to reduce the mismatch effect. Alternatively, one can detect the handset label from the speech signal during verification [57]. However, this approach may not be practical because users may use a new handset, which is not well represented in the training set, during verification. While this problem can be partially resolved by using a handset classifier with out-of-handset rejection capability [58, 102], it is difficult to find a threshold for detecting unseen handsets. On the other hand, unsupervised (blind) compensation does not assume any knowledge of the channel characteristics. In particular, it adapts speaker models or transforms speaker features to accommodate channel variations based on verification utterances only. Therefore, handset detectors are no longer required.

In speaker verification, it is important to ensure that channel variations are suppressed so that the interspeaker distinction can be enhanced. In particular, given a claimant's utterance recorded in an environment different from that during enrollment, one aims to transform the features of the utterance so that they become compatible with the enrollment environment. Therefore, it is not appropriate to transform the claimant's utterance either to fit the speaker model only or to fit the

background model only because the former will result in an unacceptably high FAR (false acceptance rate) and the latter an excessive FRR (false rejection rate). This chapter proposes a feature-based *blind* transformation approach to solving this problem. Specifically, a feature-based transformation is estimated based on the statistical difference between a test utterance and a composite acoustic model formed by combining the speaker and background models. The transformation is then used to transform the test utterance before verification. The transformation is blind in that it compensates the handset distortion without a priori information about the channel's characteristics. Hereafter, this transformation approach is referred to as blind stochastic feature transformation (BSFT).

7.1.1 Estimation of Transformation Parameters

Figure 7.1 illustrates a speaker verification system with BSFT, whose operations are divided into two separate phases: enrollment and verification.

1. *Enrollment Phase.* The speech of all client speakers are used to create a compact universal background model (UBM) Λ_b^M with M components. Then, for each client speaker, a compact speaker model Λ_s^M is created by adapting the UBM Λ_b^M using maximum a posteriori (MAP) adaptation [86] (see also section 4.1). Because verification decisions are based on the likelihood of the speaker model and background model, both models must be considered when the transformation parameters are computed. This can be achieved by fusing Λ_b^M and Λ_s^M to form a $2M$-component composite GMM Λ_c^{2M}. During the fusion process, the means and covariances remain unchanged but the value of each mixing coeffi-

cient is divided by 2. This step ensures that the output of the composite GMM represents a probability density function. Figure 7.2 illustrates the model fusion process.

2. *Verification Phase.* Distorted features $Y = \{\mathbf{y}_1, \ldots, \mathbf{y}_T\}$ extracted from a verification utterance are used to compute the transformation parameters $\nu = \{A, \mathbf{b}\}$. This is achieved by maximizing the likelihood of the composite GMM Λ_c^{2M} given the transformed features $\hat{X} = \{\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_T\}$:

$$\hat{\mathbf{x}}_t = f_\nu(\mathbf{y}_t) = A\mathbf{y}_t + \mathbf{b}, \quad t = 1, \ldots, T, \tag{7.1}$$

where A is a $D \times D$ identity matrix for zeroth-order transformation and $A = \text{diag}\{a_1, a_2, \ldots, a_D\}$ for first-order transformation, and $\mathbf{b}$ is a bias vector. The transformed vectors $\hat{X}$ are then fed to a full size speaker model Λ_s^N and a full size UBM Λ_b^N for computing verification scores in terms of likelihood ratio[1]:

$$s(\hat{X}) = \log p(\hat{X}|\Lambda_s^N) - \log p(\hat{X}|\Lambda_b^N).$$

The transformation parameters $\nu = \{A, \mathbf{b}\}$ can be estimated by the EM algorithm. More specifically, given the current estimate $\nu' = \{A', \mathbf{b}'\}$, we compute

$$\begin{aligned} \nu &= \arg\max_{\nu} Q(\nu|\nu') \\ &= \arg\max_{\nu} \sum_{t=1}^{T} \sum_{j=1}^{2M} h_j(f_{\nu'}(\mathbf{y}_t)) \log \left\{ \omega_{c,j} p(f_\nu(\mathbf{y}_t)|\mu_{c,j}, \Sigma_{c,j}, \nu) \left| J_\nu(\mathbf{y}_t) \right| \right\}, \end{aligned}$$

where $\{\omega_{c,j}, \mu_{c,j}, \Sigma_{c,j}\}_{j=1}^{2M}$ are the parameters of Λ_c^{2M}, $h_j(f_{\nu'}(\mathbf{y}_t))$ is the posterior

[1] Λ_s^N, Λ_b^N, Λ_s^M, and Λ_b^M are GMM. However, the number of Gaussians in Λ_s^N and Λ_b^N are significantly larger than that of Λ_s^M and Λ_b^M, i.e., $N \gg M$.

probability

$$h_j(f_{\nu'}(\mathbf{y}_t)) = P(j|\mathbf{y}_t, \Lambda_c^{2M}, \nu') = \frac{\omega_{c,j} p(f_{\nu'}(\mathbf{y}_t)|\mu_{c,j}, \Sigma_{c,j})}{\sum_{l=1}^{2M} \omega_{c,l} p(f_{\nu'}(\mathbf{y}_t)|\mu_{c,l}, \Sigma_{c,l})},$$

and $|J_\nu(\mathbf{y}_t)|$ is the determinant of a Jacobian matrix with (r, s)-th entry given by $J_\nu(\mathbf{y}_t)_{rs} = \partial f_\nu(\mathbf{y}_t)_s / \partial y_{t,r}$.

The main idea of BSFT is to transform the distorted features to fit the composite GMM Λ_c^{2M}, which ensures that the transformation compensates the acoustic distortion.

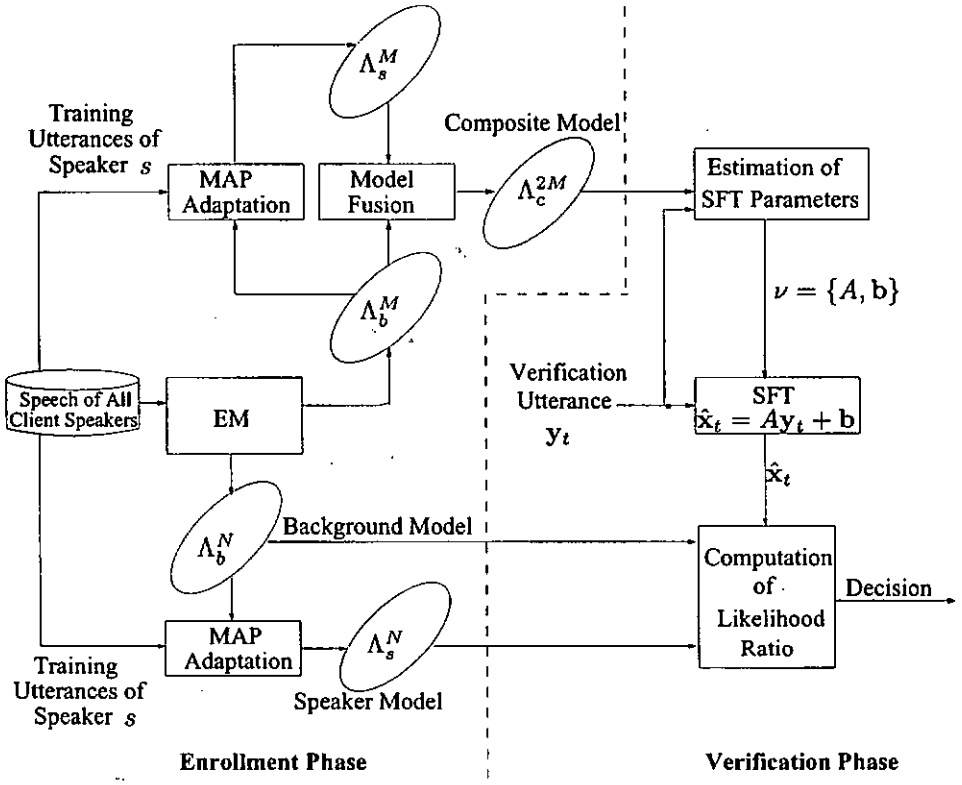


Figure 7.1: Estimation of BSFT parameters. The background model Λ_b^N, speaker model Λ_s^N, and composite model Λ_c^{2M}, produced during the enrollment phase, are subsequently used for verification purposes.

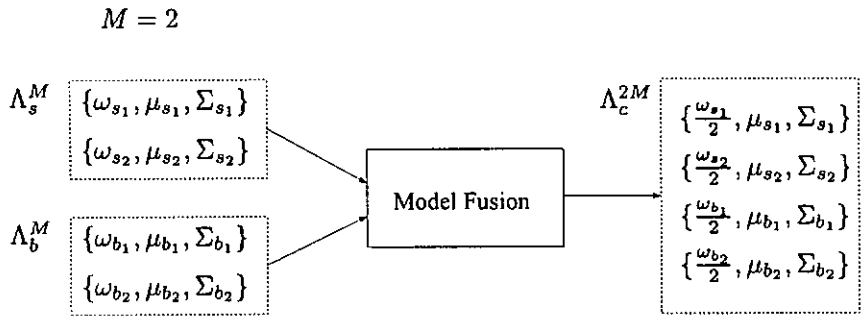


Figure 7.2: Model Fusion. The compact background model Λ_b^M and compact speaker model Λ_s^M are combined to form the compact composite model Λ_c^{2M}.

Because the computation complexity of estimating SFT parameters grows with the amount of adaptation data and the total number of mixture components in the GMMs, BSFT will become computationally intensive when the number of components is large. To perform rapid adaptation, we propose adopting a light-weight approach to computing transformation parameters. One of the positive properties of SFT is that the transformation can be estimated using GMMs with only a few components. In the light-weight approach, we synthesize a compact composite GMM (Λ_c^{2M}) by fusing a compact speaker GMM (Λ_s^M) and a compact background GMM (Λ_b^M), both with M components where $M \ll N$. It was found that a good trade-off between performance and computation complexity can be maintained by using a suitable value of M.

7.1.2 A Two-Dimensional Example

Figure 7.3 illustrates the idea of BSFT in a classification problem with two-dimensional input patterns. Figure 7.3(a) plots the clean and distorted patterns of Class 1 and Class 2. The upper right (respectively, lower left) clusters represent the clean (respectively, distorted) patterns. The ellipses show the corresponding equal

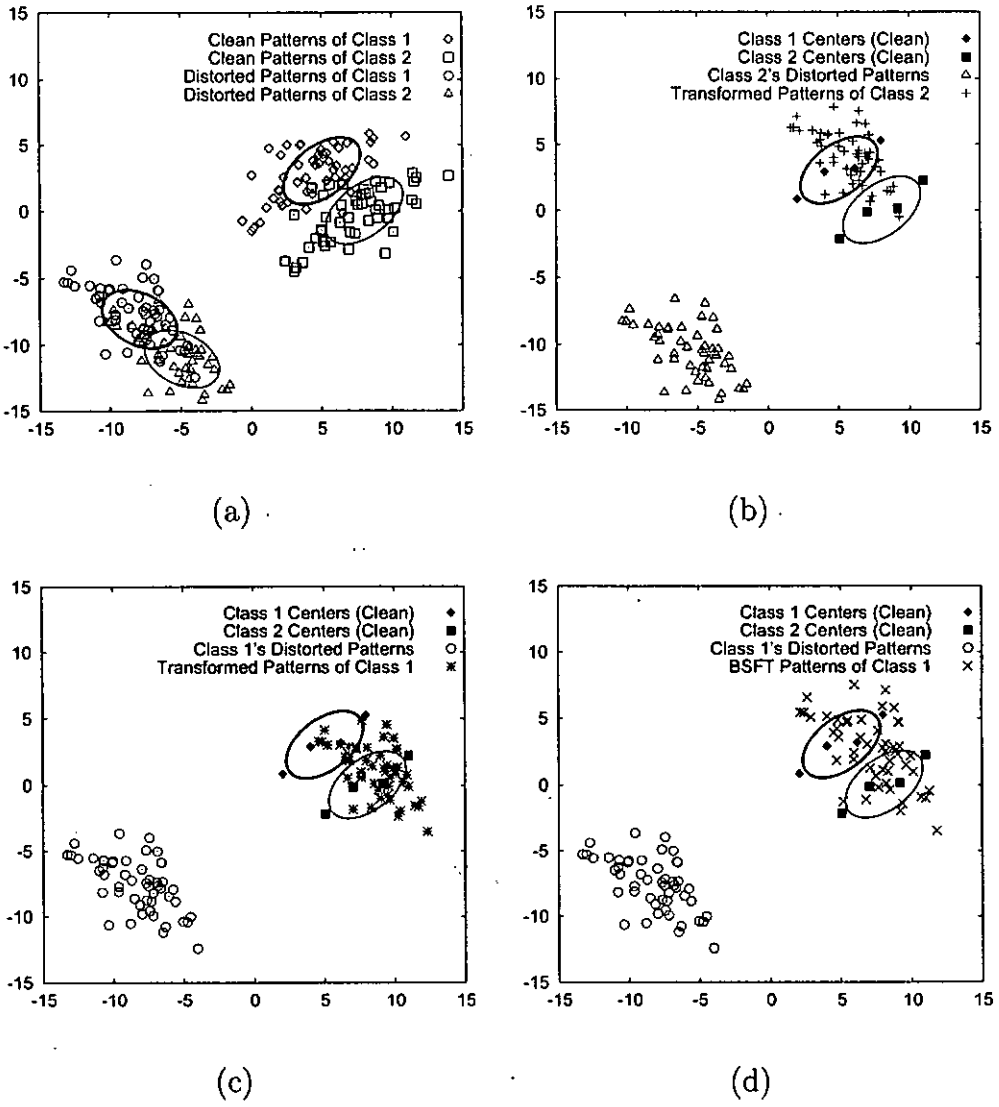


Figure 7.3: A Two-class problem illustrating the idea of BSFT. (a) Scatter plots of the clean and distorted patterns corresponding to Class 1 and Class 2. The thick and thin ellipses represent the equal density contours of Class 1 and Class 2, respectively. The upper right (respectively, lower left) clusters contain the clean (respectively, distorted) patterns. (b) Distorted patterns of Class 2 were transformed to fit Class 1's clean model. (c) Reversely, distorted patterns of Class 1 were transformed to fit Class 2's clean model. (d) Distorted data of Class 1 were transformed to fit the clean models of both Class 1 and Class 2 using first-order BSFT. For clarity, only the distorted patterns before and after transformation were plotted in (b) through (d).

density contours. Symbols ♦ and ■ represent the centers of the clean models. Figure 7.3(b) illustrates a transformation matching the distorted data of Class 2 and the GMM of Class 1 (GMM1). Because the transformation only takes GMM1 into account, while ignoring GMM2 completely, it results in a high error rate. Similarly, the transformation in Figure 7.3(c) also has a high error rate. The transformation in Figure 7.3(d) was estimated from the distorted data of Class 1 and a composite GMM formed by fusing GMM1 and GMM2. In this case, the transformation adapts the data to a region close to both GMM1 and GMM2, because it takes both GMMs into account. Therefore, instead of transforming the distorted data to a region around GMM1 or GMM2 as in Figures 7.3(b) and 7.3(c), the transformation in Figure 7.3(d) attempts to compensate the distortion. The capability of BSFT is also demonstrated in a speaker verification task to be described next.

7.2 Experimental Evaluations

7.2.1 Enrollment and Verification

Per discussion earlier, the experiments were divided into two phases: enrollment and verification.

1. *Enrollment Phase.* A 1,024-component UBM Λ_b^{1024} (i.e., N = 1,024 in Figure 7.1) was trained using the training utterances of all target speakers. The same set of data was also used to train an M-component UBM (Λ_b^M in Figure 7.1). For each target speaker, a 1,024-component speaker-dependent GMM Λ_s^{1024} was created by adapting Λ_b^{1024} using MAP adaptation [86]. Similarly, Λ_s^M was created

by adapting Λ_b^M, and the two models were fused to form a composite GMM Λ_c^{2M}. The value of M was varied from 2 to 64 in the experiments.

2. *Verification Phase.* For each verification session, a feature sequence Y was extracted from the utterance of a claimant. The sequence was used to determine the BSFT parameters (A and $\mathbf{b}$ in Eq. 7.1) to obtain a sequence of transformed vectors $\hat{X}$. The transformed vectors were then fed to Λ_s^{1024} and Λ_b^{1024} to obtain verification scores for decision making.

7.2.2 Speech Data and Features

The 2001 NIST speaker recognition evaluation set [37], which contains cellular phone speech of 74 male and 100 female target speakers extracted from the SwitchBoard-II Phase IV Corpus, was used in the evaluation. Each target speaker has 2 minutes of speech for training (i.e., enrollment); a total of 850 male and 1,188 female utterances are available for testing (i.e., verification). Each verification utterance has a length of between 15 and 45 seconds and is evaluated against 11 hypothesized speakers of the same sex as the speaker of the verification utterance. Out of these 11 hypothesized speakers, one is the target speaker who produced the verification utterance. Therefore, there are one target and 10 impostor trials for each verification utterance, which amounts to a total of 2,038 target trials and 20,380 impostor attempts for 2,038 verification utterances.

Mel-frequency cepstral coefficients (MFCCs) [18] and their first-order derivatives were computed every 14ms using a Hamming window of 28ms. Cepstral mean subtraction (CMS) [28] was applied to the MFCCs to remove linear channel effects. The

MFCCs and delta MFCCs were concatenated to form 24-dimensional feature vectors.

7.2.3 Performance Measures

Detection error trade-off (DET) curves and equal error rates (EERs) were used as performance measures. They were obtained by pooling all scores of both sex from the speaker and impostor trials. In addition to DET curves and EERs, decision cost function (DCF) was also used as performance measure. The DCF is defined as

$$\begin{aligned} \text{DCF} &= C_{Miss} \times P_{Miss|Target} \times P_{Target} \\ &+ C_{FalseAlarm} \times P_{FalseAlarm|Nontarget} \times P_{Nontarget}, \end{aligned}$$

where P_{Target} and $P_{Nontarget}$ are the prior probability of target and nontarget speakers, respectively, and where C_{Miss} and $C_{FalseAlarm}$ are the costs of miss and false alarm errors, respectively. Following NIST's recommendation [63], these parameters were set as follows: $P_{Target} = 0.01$, $P_{Nontarget} = 0.99$, $C_{Miss} = 10$, and $C_{FalseAlarm} = 1$.

7.3 Results and Discussions

7.3.1 Verification Performance

Figure 7.4 and Table 7.1 show the results of the baseline (CMS only), Znorm [83], and BSFT with different order and number of components M.[2] In Figure 7.4, there are 2038 speaker trails and 20380 impostor trials. Evidently, all cases of BSFT show significant reduction in error rates when compared to the baseline. In particular, Table 7.1 shows that first-order BSFT with Znorm achieves the largest error reduction. The

[2]Theoretically, the larger the value of M, the better the results. However, setting M larger than 64 will result in unacceptably long verification time.

DET curves also show that BSFT with Znorm performs better than the baseline and Znorm alone for all operating points.

Because the evaluation trials in NIST01 are gender-matched, gender-dependent background models can also be used for enrollment and estimation of BSFT parameters. In another experiment, speaker models were adapted from gender-dependent background models using MAP adaptation. A compact gender-dependent background model (with 64 components) was used to estimate the BSFT parameters. As shown in Table 7.1 and Figure 7.5, using gender-dependent background model helps to reduce the EERs and minimum DCF further for all cases of BSFT. However, Znorm and BSFT with Znorm seem to perform better when the background model is gender-independent. This may be attribute to the fact that less data are available for determining the Znorm parameters (score mean and variance) for each speaker when gender-dependent background models were used, which results in less reliable Znorm scores for verification. For the gender-independent case, the training utterances of 60 speakers from the "devtest" section of NIST2001 were used for estimating the Znorm parameters. For the gender-dependent case, however, the Znorm parameters of each speaker were estimated from the respective gender of these 60 speakers. Among these 60 speakers, 38 are male and 22 are female, and each of them has one training utterance. As a results, the Znorm parameters of the female speakers were determined by 22 utterances only.

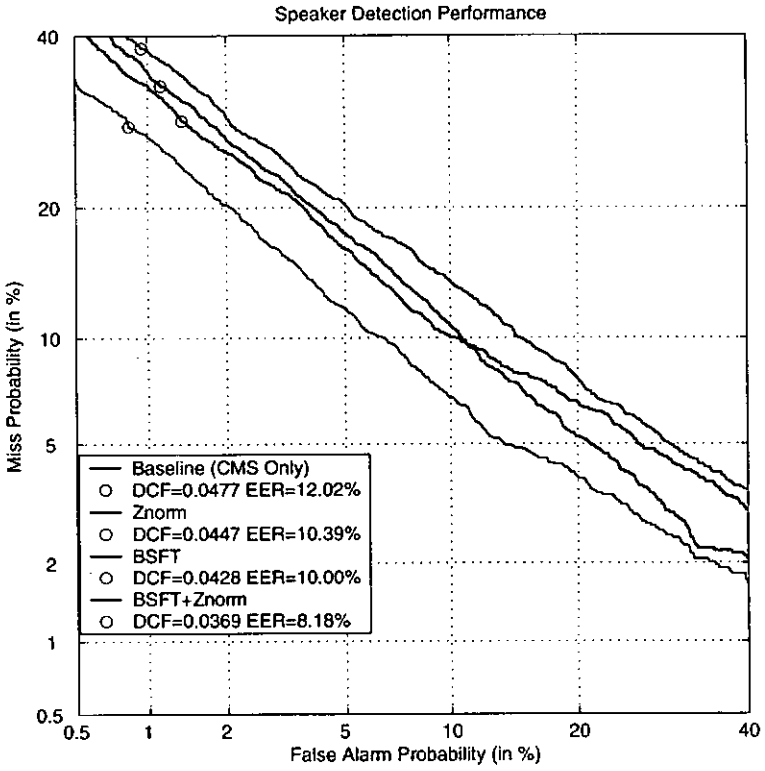


Figure 7.4: *DET curves comparing speaker verification performance using CMS (black), Znorm (blue), first-order BSFT (red), and first-order BSFT with Znorm (green). For BSFT, the number of components M in the compact GMMs was set to 64. The circles represent the errors at which minimum decision costs occur. A gender-independent background model was used in all cases.*

7.3.2 Comparison with Other Models

It is of interest to compare BSFT with the short-time Gaussianization approach proposed in Xiang et al. [104] because both methods transform distorted features in the feature space and their transformation parameters are estimated by the EM algorithm [19]. The short-time Gaussianization achieves an EER of 10.84% in the NIST 2001 evaluation set [104], whereas BSFT achieves an EER of 9.26%, which represent an

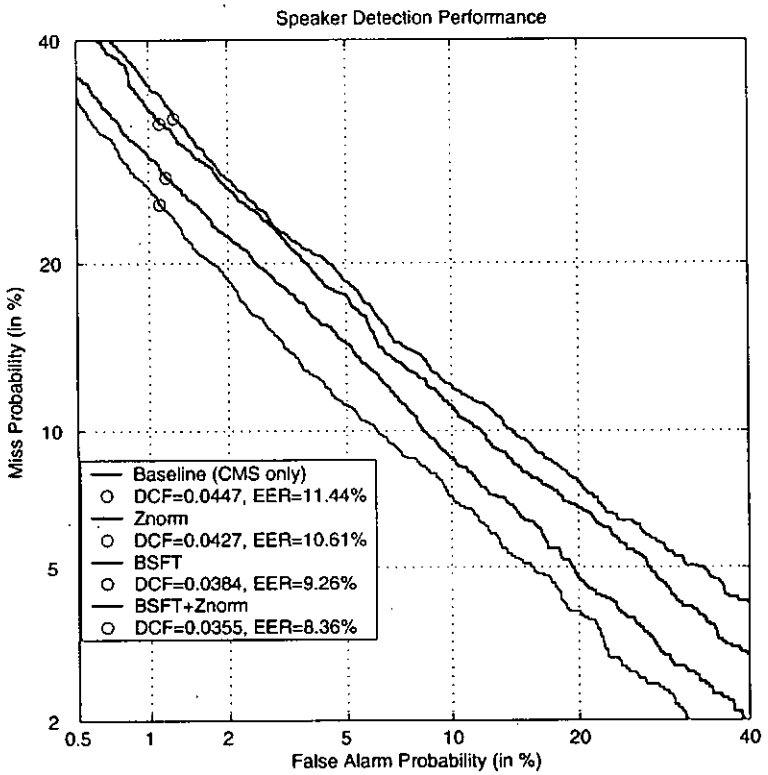


Figure 7.5: *DET curves comparing speaker verification performance using CMS (black), Znorm (blue), first-order BSFT (red), and first-order BSFT with Znorm (green). For BSFT, the number of components M in the compact GMMs was set to 64. The circles represent the errors at which minimum decision costs occur. Gender-dependent background models were used in all cases.*

error reduction of 14.58%.[3] The minimum decision cost of BSFT is also lower than that of short-time Gaussianization (0.0384 versus 0.0440).

7.3.3 Computation Consideration

In BSFT, a set of transformation parameters ν is computed by the EM algorithm in which the likelihood function of a composite GMM given the transformed test data is maximized. In short-time Gaussianization, a linear, global transformation matrix,

[3]Because Xiang et al. did not use Znorm in [104], their results should be compared with the one without Znorm here.

which aims to decorrelate the distorted features, is estimated by the EM algorithm using the training data of all background speakers. The distorted features are then transformed and mapped to fit a normal distribution. The linearly transformed features are divided into a number of overlapping segments, with each segment containing a number of consecutive transformed vectors. The consecutive vectors in a segment are then sorted in ascending order. The rank of the central frame is used to find a warped feature so that its cumulative density function (CDF) matches the CDF of a standard normal distribution.

It can be argued that the inferior performance of Gaussianization is due to the nonadaptive nature of its transformation parameters. However, the adaptive nature of BSFT comes with a computational price: different transformation parameters have to be computed for each speaker. Therefore, it is vital to have a cost effective computation approach for BSFT. Note that the computation complexity of estimating BSFT parameters grows with the amount of adaptation data (i.e., the value of T in Eq. 7.1) and the number of mixture components in the compact GMMs (i.e., the value of M). To reduce computation time, M should be significantly smaller than N, the number of components in the full size speaker and background models. This is particularly important for the computation of BSFT parameters during the verification phase because the computation time of this phase is a significant part of the overall verification time. The evaluations suggest that a good tradeoff between performance and computation complexity can be achieved by using a suitable value of M.

7.3.4 Hnorm Vs. Znorm

In chapter 5, we used HTIMIT as the evaluation set. One nice property of HTIMIT is that it provides explicit handset labels for all utterances in the corpus. Because of this property, HTIMIT is particularly suitable for evaluating the Hnorm approach to channel compensation where handset-dependent normalization parameters are estimated by testing each speaker model against handset-dependent speech produced by impostors.

In this chapter, we used NIST 2001 as the evaluation set. The evaluation set also provides algorithmically-determined handset labels. However, most of the training and testing segments were obtained from handsets with an electret microphone and only some were obtained from handsets with a carbon-button microphone. Hence, we chose to use Znorm for score normalization because Znorm does not require handset-specific data to compute the normalization parameters.

7.4 Conclusions

This chapter has presented a new approach, namely blind stochastic feature transformation, to channel robust speaker verification and provided experimental results on the 2001 NIST evaluation set. The algorithm computes feature transformation parameters based on the statistical difference between a test utterance and a composite GMM formed by combining the speaker and background models. The transformation is then used to transform the test utterance to fit the clean speaker model and background model before verification. Experimental results show that the proposed

algorithms achieves significant improvement in both equal error rate and minimum detection cost when compared to cepstral mean subtraction, Znorm, and short-time Gaussianization.

Compensation Method	SFT Order	M	Background Model (Λ_b^N)			
			Gender-Independent		Gender-Dependent	
			Equal Error Rate (%)	Minimum Decision Cost	Equal Error Rate (%)	Minimum Decision Cost
Baseline	NA	NA	12.02	0.0477	11.44	0.0477
BSFT	Zeroth	2	11.90	0.0473	11.49	0.0440
BSFT	Zeroth	4	11.82	0.0458	11.16	0.0427
BSFT	Zeroth	8	11.39	0.0449	10.89	0.0428
BSFT	Zeroth	16	11.24	0.0450	10.79	0.0420
BSFT	Zeroth	32	11.22	0.0450	10.80	0.0422
BSFT	Zeroth	64	11.16	0.0443	10.61	0.0414
BSFT	First	2	12.00	0.0506	11.29	0.0445
BSFT	First	4	11.55	0.0471	10.27	0.0425
BSFT	First	8	10.70	0.0464	9.77	0.0409
BSFT	First	16	10.47	0.0454	9.48	0.0394
BSFT	First	32	10.43	0.0446	9.38	0.0395
BSFT	First	64	10.00	0.0428	9.26	0.0384
Znorm	NA	NA	10.39	0.0447	10.61	0.0427
BSFT+Znorm	First	64	8.18	0.0369	8.36	0.0355

Table 7.1: Equal error rates and minimum decision cost achieved by the baseline (CMS only), Znorm, and zeroth- and first-order BSFT with different order and number of components M in the compact GMMs. The number of components in the full size speaker and background models is 1,024. The columns "Gender-Independent" and "Gender-Dependent" represents the types of background models being used for obtaining the results.

Chapter 8

CONCLUSIONS

This work has addressed two important issues in speaker verification: speaker modeling and channel mismatch compensation.

For the former, techniques that minimize the error rate of telephone-based speaker verification systems have been investigated. For example, to make PDBNNs more appropriate for speaker verification, the original training algorithm of PDBNNs have been modified. The structural properties and learning rules of probabilistic decision-based neural networks (PDBNNs), Gaussian mixture models (GMMs), and elliptical basis function networks (EBFNs) have been compared using three problem sets. Experiments using these three kernel-based probabilistic neural networks as speaker models have been carried out to evaluate their suitability for speaker verification.

For the channel mismatch problem, we used two channel compensation approaches to addressing the problem of environmental mismatch in telephone-based speaker verification systems. These approaches are model transformation/adaptation and feature transformation. For the model transformation/adaptation approach, we have proposed an algorithm combining MLLR, PDBNN, and stochastic feature transformation and compared its performance with that of the state-of-the-art techniques. For the feature transformation approach, we have proposed a blind feature trans-

formation algorithm in which no a priori information of channel characteristics are required.

Here, we summarize our major findings of this work:

- The use of GMMs and PDBNNs as speaker models is promising. Experimental results, based on 138 speakers and visualization of decision boundaries, suggest that GMM- and PDBNN-based speaker models outperform the EBFN ones. It was also found that PDBNNs and GMMs are more robust than EBFNs in recognizing speakers in noisy environments.

- The globally supervised training of PDBNNs is able to find decision thresholds that not only reduce the variation of false acceptance errors among all speakers in the system but also maintain the errors at a low level. Results also show that the proposed modifications on the PDBNN's supervised learning not only make PDBNNs amenable to speaker verification but also lead to more predictable performance.

- Comparisons with conventional model-based channel compensation techniques suggest that combining MLLR and PDBNN outperforms a number of classical techniques, including CMS, stochastic feature transformation, Hnorm, and speaker model synthesis.

- Unsupervised (blind) compensation does not assume any knowledge of the channel characteristics. It adapts speaker models or transforms speaker features to accommodate channel variations based on verification utterances only. Exper-

imental results show that the proposed blind feature transformation algorithm achieves significant improvement in both equal error rate and minimum detection cost when compared to cepstral mean subtraction, Znorm, and short-time Gaussianization.

Chapter 9

FUTURE WORK

The blind compensation method discussed in Chapter 7 is designed to handle handsets that are unknown to the speaker verification system. However, estimating SFT parameters online requires additional computation during the verification phase. One possible solution to reducing computation without sacrificing the advantages of blind compensation is to precompute the transformation parameters of some commonly used handsets and store them in a handset database; during verification, the most appropriate transformation parameters are selected from the database if the handset being used is detected to be one of the a priori known models. In this case, the supervised compensation approach discussed in [58] can be adopted. This paves the ground for an integrated approach combining blind and supervised compensation techniques. Figure 9.1 illustrates how the blind and supervised compensation can be integrated into a speaker verification system. This integrated approach can enjoy the best of the two worlds: (1) whenever a known handset is detected, precomputed transformation parameters can be used and (2) should the handset being used is unknown to the system, the system can compute the transformation parameters online using blind or unsupervised techniques.

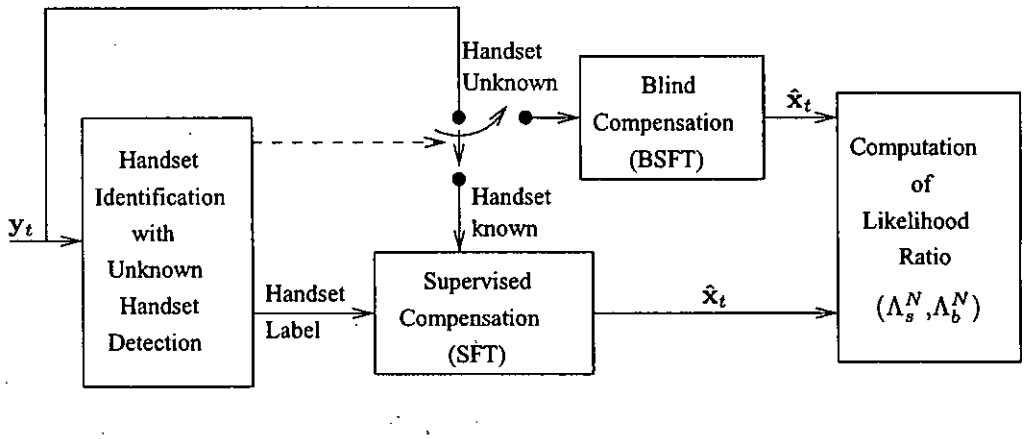


Figure 9.1: Integration of Blind and Supervised Compensation.

BIBLIOGRAPHY

[1] A. Acero. *Acoustical and Environmental Robustness in Automatic Speech Recognition.* Kluwer Academic Publishers, Dordrecht, 1992.

[2] A. Acero, L. Deng, T. Kristjansson, and J. Zhang. HMM adaptation using vector Taylor series for noisy speech recognition. In *ICSLP 2000*, volume 3, pages 869–872, 2000.

[3] 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.

[4] M. Afify, Y. Gong, and J. P. 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):524–537, 1998.

[5] S. Ahn, S. Kang, and H. Ko. Effective speaker adaptations for speaker verification. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 5–9, 2000.

[6] E. Ambikairajah, M. Keane, A. Kelly, L. Kilmartin, and G. Tattersall. Predictive models for speaker verification. *Speech Communication*, 13:417–425, 1993.

[7] 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.

[8] 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.

[9] R. Auckenthaler, M. Carey, and H. Lloyd-Thomas. Score normalization for text-independent speaker verification systems. *Digital Signal Processing*, 10:42–54, 2000.

[10] F. Beaufays and M. Weintraub. Model transformation for robust speaker recognition from telephone data. In *ICASSP'97*, volume 2, pages 1063–1066, 1997.

[11] Y. Bennani. Multi-expert and hybrid connectionist approach for pattern recognition: Speaker identification task. *Int. J. of Neural Systems*, 5(3):207–216, 1994.

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

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

[14] I. Booth, M. Barlow, and B. Watson. Enhancements to DTW and VQ decision algorithms for speaker recognition. *Speech Communication*, 13:427–433, 1993.

[15] D. K. Burton. Text-dependent speaker verification using vector quantization source coding. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, ASSP-35(2):133–143, 1987.

[16] C. Chesta, O. Siohan, and C. H. Lee. Maximum a posteriori linear regression for hidden markov model adaptation. In *Eurospeech'99*, volume 1, pages 211–214, 1999.

[17] S. Cox. Predictive speaker adaptation in speech recognition. *Computer Speech and Language*, 9:1–17, 1995.

[18] S. B. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Trans. on ASSP*, 28(4):357–366, August 1980.

[19] 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.

[20] V. D. Diakoloukas and V. V. Digalakis. Maximum-likelihood stochastic-transformation adaptation of hidden markov models. *IEEE Trans. on Speech and Audio Proc.*, 7(2):177–187, March 1999.

[21] V. Digalakis, D. Rtichev, and L. Neumeyer. Speaker adaptation using constrained reestimation of Gaussian mixtures. *IEEE Trans. On Speech and Audio Proc.*, pages 357–366, 1995.

[22] 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.

[23] G.R. Doddington. *A Computer Method of Speaker Verification.* PhD thesis, Dept. of Electrical Engineering, University of Wisonsin, Madison, 1970.

[24] J. B. Pierrot. et al. A comparison of a priori threshold setting procedures for speaker verification in the CAVE project. In *Proc. ICASSP'98*, pages 125–128, 1998.

[25] K. Farrell, S. Kosonocky, and R. Mammone. Neural tree network/vector quantization probability estimators for speaker recognition. In *Proc. Workshop on Neural Networks for Signal Processing*, pages 279–288, 1994.

[26] K. R. Farrell, R. J. Mammone, and K. T. Assaleh. Speaker recognition using neural networks and conventional classifiers. 2(1):194–205, 1994.

[27] M. E. Forsyth, A. M. Sutherland, and J. A. Jack. HMM speaker verification with sparse training data on telephone quality speech. *Speech Communication*, 13:411–416, 1993.

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

[29] S. Furui. Recent advances in speaker recognition. *Pattern Recognition Letters*, 18:859–872, 1997.

[30] G. J. Gales. *Model Based Techniques for Noise Robust Speaker Recognition.* PhD thesis, Engineering Department, Cambridge University, 1995.

[31] M. J. F. Gales. The generation and use of regression class trees for MLLR adaptation. Technical report, Technical Report CUED/F-INFENG/TR263, Cambridge University, 1996.

[32] M. J. F. Gales. Maximum likelihood linear transformations for HMM-based speech recognition. *Computer Speech and Language*, 12:75–98, 1998.

[33] 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.

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

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

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

[37] *http://www.nist.gov/speech/tests/spk/2001/doc.* The NIST year 2001 speaker recognition evaluation plan.

[38] *http://www.nist.gov/speech/tests/spk/index.htm.*

[39] Q. Huo and C. H. Lee. On-line adaptive learning of the continuous density hidden Markov model based on approximate recursive Bayes estimate. *IEEE Trans. on Speech and Audio Processing*, 5(2):161–172, March 1997.

[40] Q. Huo and C. H. Lee. On-line adaptive learning of the correlated continuous density hidden Markov models for speech recognition. *IEEE Trans. on Speech and Audio Processing*, 6(4):386–397, 1998.

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

[42] Jr. J. P. Campbell. Speaker recognition: A tutorial. *Proc. of the IEEE*, 85(9):1437–1462, 1997.

[43] 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.

[44] S. Katagiri, B. H. Juang, and C. H. Lee. Pattern recognition using a family of design algorithm based upon the generalized probabilistic descent method. *Proc. IEEE*, 86(11):2345–2373, 1998.

[45] R. Kuhn, J. C. Junqua, P. Nguyen, and N. Niedzielski. Rapid speaker adaptation in eigenvoice space. *IEEE Trans. on Speech and Audio Processing*, 8(6):695–707, 2000.

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

[47] S. Y. Kung, M. W. Mak, and S. H. Lin. *Biometric Authentication: A Machine Learning Approach.* Prentice Hall, 2004.

[48] C. H. Lee, C. H. Lin, and B. H. Juang. A study on speaker adaptation of the parameters of continuous density hidden Markov models. *IEEE Trans. on ASSP-39*, 39(4):806–814, April 1991.

[49] C. J. Leggetter and P. C. Woodland. Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models. *Computer Speech and Language*, 9(4):806–814, 1995.

[50] K. P. Li and J. E. Porter. Normalizations and selection of speech segments for speaker recognition scoring. In *ICASSP-88*, volume 1, pages 595–598, 1988.

[51] 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.

[52] 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.

[53] 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.

[54] 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.

[55] 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.

[56] M. W. Mak and S. Y. Kung. Estimation of elliptical basis function parameters by the EM algorithms with application to speaker verification. *IEEE Trans. on Neural Networks*, 11(4):961–969, 2000.

[57] M. W. Mak and S. Y. Kung. Combining stochastic feautre transformation and handset identification for telephone-based speaker verification. In *Proc. ICASSP'2002*, pages I701–I704, 2002.

[58] M. W. Mak, C. L. Tsang, and S. Y. Kung. Stochastic feature transformation with divergence-based out-of-handset rejection for robust speaker verification. *EURASIP J. on Applied Signal Processing*, 4:452–465, 2004.

[59] J. Makhoul. Linear prediction: A tutorial review. *Proceedings of the IEEE*, 63(4):561–580, April 1975.

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

[61] 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.

[62] A. Martin, G. Doddington, T. Kamm, M. Ordowski, and M. Przybocki. The DET Curve in assessment of detection task performance. In *Eurospeech'97*, pages 1895–1898, 1997.

[63] M. Przybocki A. Martin. NIST's assessment of text independent speaker recognition performance 2002. In *The Advent of Biometircs on the Internet, A COST 275 Workshop*, Rome, Italy, Nov. 2002.

[64] T. Matsui and S. Furui. Comparison of test-independent speaker recognition methods using VQ-distortion and discrete/continuous HMM's. *IEEE Trans. on Speech and Audio Processing*, 2(3):456–459, 1994.

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

[66] E. McDermott. *Discriminative Training for Speech Recognition.* Ph.D. Thesis, Waseda University, Japan, 1997.

[67] C. Mokbel. Online adaptation of HMMs to real-life conditions: A unified framework. *IEEE Trans. on Speech and Audio Processing*, 9(4):342–357, May 2001.

[68] P. J. Moreno. *Speech Recognition in Noisy Environments.* PhD thesis, Department of Electrical and Computer Engineering, Carnegie Mellon University, 1996.

[69] D. Naik. Pole-filtered cepstral mean subtraction. In *ICASSP'95*, volume 1, pages 157–160, 1995.

[70] 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.

[71] L. Neumeyer and M. Weintraub. Probabilistics optimal filtering for robust speech recognition. *ICASSP'94*, pages 417–420, 1994.

[72] P. Nguyen, C. Wellekens, and J. C. Junqua. Maximum likelihood eigenspace and MLLR for speech recognition in noisy environments. In *Eurospeech'99*, volume 6, pages 2519–2522, 1999.

[73] Y. Normandin. *Hidden Markov Models, Maximum Mutual Information Estimation, and the Speech Recognition Problem.* Ph.D. thesis, Dept. Elect. Eng., McGill University, Canada, 1991.

[74] J. Oglesby and J. S. Mason. Radial basis function networks for speaker recognition. In *ICASSP'91*, pages 393–396, 1991.

[75] A. B. Portiz. Linear predictive hidden markov models and the speech signal. *ICASSP'82*, 2:1291–1294, 1982.

[76] T. F. Quartieri, D. A. Reynolds, and G. C. O'Leary. Estimation of handset nonlinearity with application to speaker recognition. *IEEE Trans. on Speech and Audio Processing*, 8(5):567–584, 2000.

[77] 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.

[78] 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.

[79] D. A. Reynolds. An overview of automatic speaker recognition technology.

[80] D. A. Reynolds. Experimental evaluation of features for robust speaker identification. *IEEE Trans. on Speech and Audio Processing*, 2(4):639–643, 1994.

[81] D. A. Reynolds. Speaker identification and verification using Gaussian mixture speaker models. *Speech Communications*, 17:91–108, 1995.

[82] D. A. Reynolds. The effects of handset variability on speaker recognition performance: Experiments on the switchboard corpus. In *ICASSP'96*, volume 1, pages 113–116, 1996.

[83] D. A. Reynolds. Comparison of background normalization methods for text-independent speaker verification. In *Eurospeech'97*, pages 963–966, 1997.

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

[85] D. A. Reynolds. Channel robust speaker verification via feature mapping. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 6–10, 2003.

[86] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn. Speaker verification using adapted Gaussian mixture models. *Digital Signal Processing*, 10:19–41, 2000.

[87] 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.

[88] 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 *ICASSP'95*, volume 1, pages 329–332, 1995.

[89] 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.

[90] 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.

[91] 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.

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

[93] A. Sankar and C. H. Lee. A maximum-likelihood approach to stochastic matching for robust speech recognition. *IEEE Trans. on Speech and Audio Processing*, 4(3):190–202, 1996.

[94] G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6:461–464, 1978.

[95] O. Siohan, C. Chesta, and C. H. Lee. Joint maximum a posteriori adaptation of transformation and HMM parameters. *IEEE Trans. on Speech and Audio Processing*, 9(4):417–428, May 2001.

[96] 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.

[97] R. Stern and M. Lasry. Dynamic speaker adaptation for feature-based isolated word recognition. *IEEE Trans. on coustics, Speech, and Signal Processing*, 35(6):751–763, 1987.

[98] A. C. Surendran, C. H. Lee, and M. Rahim. Nonlinear compensation for stochastic matching. *IEEE Trans. on Speech and Audio Processing*, 7(6):643–655, 1999.

[99] R. Teunen, B. Shahshahani, and L. Heck. A model-based transformational approach to robust speaker recognition. In *ICSLP*, volume 2, pages 495–498, 2000.

[100] N. Z. Tishby. On the application of mixture AR hidden markov models to text independent speaker recognition. *IEEE Transactions on Signal Processing*, 39(3):563–570, March 1991.

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

[102] C. L. Tsang, M. W. Mak, and S. Y. Kung. Divergence-based out-of-class rejection for telephone handset identification. In *Proc. Int. Conf. on Spoken Language Processing*, pages 2329–2332, 2002.

[103] V. Valtchev. *Discriminative Methods in HMM-based Speech recognition.* Ph. D. thesis, University of Cambridge, UK, 1995.

[104] B. Xiang, U. Chaudhari, J. Navratil, G. Ramaswamy, and R. Gopinath. Short-time Gaussianization for robust speaker verification. In *Proc. IEEE ICASSP'02*, volume 1, pages 681–684, 2002.

[105] K. K. Yiu, M. W. Mak, and S. Y. Kung. A comparative study on kernel-based probabilistic neural networks for speaker verification. *International Journal of Neural Systems*, 12(5):381–391, 2002.

[106] K. K. Yiu, M. W. Mak, and S. Y. Kung. Environment adaptation for robust speaker verification. In *Eurospeech'03*, 2003.

[107] W. D. Zhang, M. W. Mak, and M. X. He. A two-stage scoring method combining world and cohort model for speaker verification. In *Proc. ICASSP'00*, pages 1193–1196, June 2000.

[108] 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.

[109] Y. Zhao. An EM algorithm for linear distortion channel estimation based on observations from a mixture of Gaussian sources. *IEEE Trans. on Speech and Audio Processing*, 7(4):400–413, 1999.

[110] 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 S. Y. Kung. A Comparative Study on Kernel-Based Probabilistic Neural Networks for Speaker Verification, *International Journal of Neural Systems*, Vol. 12, No. 5, 381-391, 2002.

2. K. K. Yiu, M. W. Mak and S. Y. Kung. Blind Stochastic Feature Transformation for Channel Robust Speaker Verification, accepted by *J. of VLSI Signal Processing*.

3. K. K. Yiu, M. W. Mak and S. Y. Kung. Environment Adaptation for Robust Speaker Verification by Cascading Maximum Likelihood Linear Regression and Reinforced Learning, resubmitted to *Computer Speech and Language*.

International Conference Papers

4. K. K. Yiu, M. W. Mak and S. Y. Kung. Environment Adaptation for Robust Speaker Verification, in *Eurospeech'03*, Geneva, Sept. 2003, pp. 2973-2976.

5. K. K. Yiu, M. W. Mak and S. Y. Kung. Speaker Verification with A Priori Threshold Determination Using Kernel-Based Probabilistic Neural Networks, in *Int. Conf. on Neural Information Processing 2002*, (ICONIP'02), Singapore, Nov. 2002, pp. 2386-2390.

6. K. K. Yiu, M. W. Mak and S. Y. Kung. Kernel-Based Probabilistic Neural Networks with Integrated Scoring Normalization for Speaker Verification, in *Pacific-Rim Conference on Multimedia 2002* (PCM'2002), 2002, pp. 623-630.

7. K. K. Yiu, M. W. Mak, M. C. Cheung, and S. Y. Kung. A New Approach to Channel Robust Speaker Verification via Constrained Stochastic Feature Transformation, accepted by *Int. Conf. on Spoken Language Processing, 2004.*

8. M. C. Cheung, K. K. Yiu, M. W. Mak and S. Y. Kung. Multi-Sample Fusion with Constrained Feature Transformation for Robust Speaker Verification, accepted by *Int. Conf. on Spoken Language Processing, 2004.*

9. K. K. Yiu, M. W. Mak, M. C. Cheung, and S. Y. Kung. Blind Stochastic Feature Transformation for Speaker Verification over Cellular Networks, accepted by *ISIMP, 2004.*