150297

TWO-STAGE DECISION MAKING ALGORITHM FOR SPEAKER VERIFICATION

A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of
Dokuz Eylül University
In Partial Fulfillment of the Requirements for
the Degree of Master of Science in Electrical & Electronics Engineering,
Electrical & Electronics Engineering Program

by
Efe Tankut YAPAROĞLU

September, 2004
İZMİR

150297

M.Sc. THESIS EXAMINATION RESULT FORM

We certify that we have read this thesis **"TWO-STAGE DECISION MAKING ALGORITHM FOR SPEAKER VERIFICATION"** completed by **Efe Tankut YAPAROĞLU** under supervision of **Asst. Prof. Dr. Yavuz ŞENOL** and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

[illegible]

Asst. Prof. Dr. Yavuz Şenol

(Supervisor)

[illegible]

Yrd. Doç. Dr. Haldun Sarnel

(Committee Member)

[illegible]

Yrd. Doç. Dr. Hakkı T. Yalazan

(Committee Member)

Approved by the

Graduate School of Natural and Applied Sciences

[illegible]

Prof. Dr. Cahit HELVACI

Director

ACKNOWLEDGMENTS

I would like to express my sincere thanks to Asst. Prof. Dr. Yavuz ŞENOL for his valuable support, encouragement and patience throughout this study.

Another person, I have to give my thanks is my boss Mr. Talat SAĞIROĞLU for his patience and great support. I should also thank to my family and my fiancée Aslı AKGÜN who did not leave me alone in long working hours.

Efe YAPAROĞLU

ABSTRACT

Among biometric approaches, *automatic speaker verification* is becoming more and more popular, for being probably the most natural and economical method of restricting the problems like unauthorized use of computers and communication systems, and multi-level access control. In order to give the right accept/reject decision, it is crucial to generate an efficient model and to implement an accurate decision making algorithm.

In this thesis, first, a baseline system is formed using mel-frequency cepstral coefficients (MFCC) as features, and for speaker modeling a radial basis function (RBF) neural network is used. Then, investigations and experiments have been realized for optimizing the training set, and also, a two-stage decision making algorithm is proposed, which has the aim to eliminate the qualified impostors in the second stage.

Keywords: speaker verification, speaker recognition, radial basis function (RBF) neural networks, mel-scale, training set optimization, cohort model, two-stage decision making algorithm

ÖZET

Biyometrik uygulamalar içersinde *ses ile kimlik doğrulama,* bilgisayar ve haberleşme sistemlerinin yetkisiz kişiler tarafından kullanımı ve çok-seviyeli erişim kontrolü gibi problemlerin önlenmesinde kullanılabilecek en doğal ve ekonomik metodlardan biri olmasından dolayı, giderek popülerlik kazanmaktadır. Bu sistemle doğru kabul/red kararı verebilmek için, verimli bir model oluşturulması ve karar verme algoritmasının kusursuzluğu büyük önem taşımaktadır.

Bu tezde, ilk olarak, konuşmacının ses özelliklerinin temsil edildiği mel-frekanslı sepstral katsayılar ve model olarak da radial taban fonksiyonlu yapay sinir ağları kullanılarak bir iskelet sistem oluşturulmuştur. Daha sonra, yapay sinir ağının eğitim kümesi için optimizasyon incelemesi yapılmış ve ilk kısmı geçen taklitçilerin ikinci kısımda elenmesi amacına dayanan çift seviyeli bir karar verme algoritması ortaya konulmuştur.

Anahtar Sözcükler: ses ile kimlik doğrulama, konuşmacı tanıma, MFCC, radyal taban fonksiyonlu yapay sinir ağları (RBF-YSA), eğitim kümesi optimizasyonu, konuşmacı sesine yakın seslerin modellenmesi, iki aşamalı karar verme algoritması

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE
INTRODUCTION

Speech, being the most natural form of human communication, has been one of the most exciting areas of signal processing. The speech signal carries information related to not only the linguistic message to be conveyed, but also the identity of the speaker, language, emotional status of the speaker, environment and so on. Thinking of a life without the ability of identifying people from their voices, it would be impossible to know whom we are talking on the phone, or who is calling out to us from a distance. Since human brain is still the most advanced and accurate language identification system today, far exceeding the data storage and processing power of a supercomputer, the identification ability seems so simple for us, but computer based implementations are still far from human abilities and any speaker identification system on a computer can not be designed as an optimum solution. However, recent developments in digital signal processors and speech technology have made it possible to design fast, cost effective, high performance speaker recognition systems.

Speech processing is a diverse field with many applications. A few of these areas and how speaker recognition relates to the rest of the field is shown in Figure1.1.

Speaker recognition is concerned with the problems of *identification* and *verification*, each of which may in turn be *text-dependent* or *text-independent*. In speaker identification, the aim is to determine which of the registered speakers a given utterance comes from. The test utterance is scored against all possible speaker models, with the best score determining the speaker identity. In speaker verification, which will be focused on in this work, the aim is to give acceptance or rejection decision for the identity claim of a speaker. The claimant speaks the phrase into a microphone and this signal is analyzed by a verification system that makes the binary decision to accept or reject the user's identity claim or possibly to report insufficient confidence and request additional input before making the decision. Some common examples of speaker verification applications are voice-activated locks, access to restricted computer data, forensic applications, electronic commerce, telephone banking and ATM transactions.

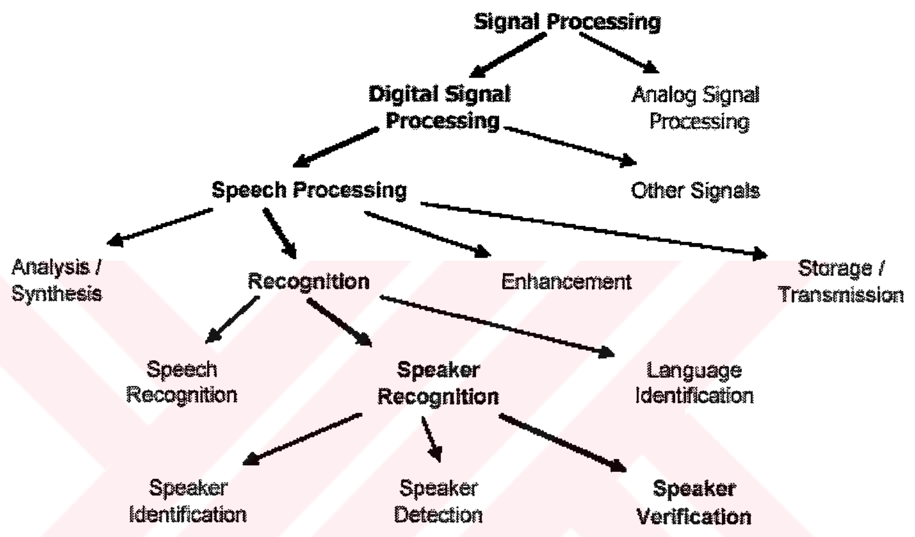

Figure 1.1 Speaker Recognition in Signal Processing

By text-dependent speaker recognition, the same or known text is used for training and test. In contrast, any text is allowed to be uttered in the process of either training or test in the text-independent recognition. By comparison, a text-dependent system is conceptually simple but inflexible, while a text-independent system seems complicated but flexible (Chen, 2002). Moreover, the performance of a text-dependent system is often reasonably better than that of a text-independent system while the text-independent system can perform in a more secure way if the user is allowed to speak any random phrase. No matter what the operating mode is, speaker recognition theoretically belongs to non-verbal speech classification since the information of speaker's characteristics conveyed in speech waves plays a crucial role in this process rather than those verbal contents carried by speech waves.

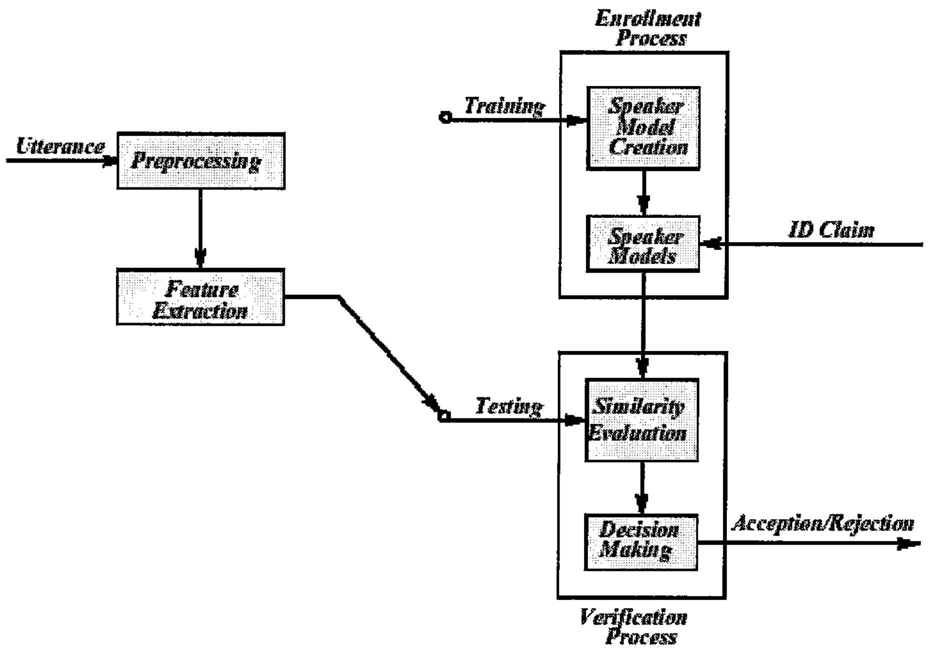

Figure 1.2 Schematic Diagram of a Typical Speaker Verification System

1.1 Motivation

Biometric systems automatically recognize a person by using distinguishing traits. Speaker recognition is a performance biometric, i.e., you perform a task to be recognized. Voice, like other biometrics, cannot be forgotten or misplaced, unlike knowledge-based (e.g. password) or possession-based (e.g. key) access control methods. Speaker recognition systems can be made somewhat robust against noise and channel variations (Mammone et al., 1996), ordinary human changes (e.g. time-of-day voice changes and minor head colds), and mimicry by humans and tape recorders (Higgins et al., 1991). Among biometric approaches, automatic speaker verification (ASV) is probably the most natural and economical method for solving the problems of unauthorized use of computer and communications systems and multilevel access control. Unlike iris and retina recognition applications, voice verification systems are non-intrusive and not considered threatening by users since most people are comfortable with using telephone or microphone (Lucey, 2002), (Altun & Kocer, 2003). With the ubiquitous telephone network and microphones bundled with computers, the cost of a speaker recognition system might only be for software (Rabiner, 1989).

1.2 Problem Definition

Speaker recognition is a difficult task and is still an active research area. ASV (automatic speaker verification) works based on the premise that a person's speech exhibits characteristics that are unique to the speaker. However, this task has been challenged by high variance of input speech signals (Minh, 2000). The source of variance comes mostly from the speakers themselves. Speech signals in training and testing sessions can be greatly different due to many facts such as people voice change with time, health conditions, speaking rates, etc. There are also other factors, beyond speaker variability, that present a challenge to speaker recognition such as acoustical noise and variations in recording environments.

Desired properties of a speaker recognition system can be listed as follows (Wolf, 1972), (Cohen & Zigel, 2003):

- Occur naturally and frequently in normal speech
- Be easily measurable
- Vary as much as possible among speakers
- Operate consistently for each speaker
- Maintain same performance level over time
- Be insensitive to speakers' health or mood
- Be insensitive to background noise and transmission characteristics
- Be insensitive to mimicry and falsification attempts

In this work, a security system will be the case, therefore acceptance of an impostor to the system is completely intolerable. On the other hand frequent rejection of genuine speakers may be disturbing. So the system should be able to clearly differentiate voice characteristics among different people while compensating in-person differences.

1.3 Thesis Goals

The overall goals of this thesis are:

- Perform background search on feature selection, pattern matching and decision making algorithms used in ASV systems,
- Develop a speaker verification system using artificial neural networks, that will serve as a security system,
- Develop new methods to accurately make accept/reject decisions and modify the system to minimize the false acceptance rate (FAR) and keep false rejection rate (FRR) at an agreeable level.

1.4 Thesis Organization

In Chapter 2, the fundamentals of voice recognition by humans are given, describing in details the dynamics of speech production and speech perception systems that human beings possess. Also, the history and development of voice recognition technology is mentioned.

In Chapter 3, the steps of speaker verification systems are described in details, including acquisition of speech as a digital signal, extraction of the features efficiently, types of features used in speaker verification systems, modeling of speakers, scoring, and the decision making concept.

In Chapter 4, a baseline speaker verification system, utilising mel-frequency cepstral coefficients (MFCC) as features and radial basis function (RBF) network as the verification model, is proposed and described in details. Also, investigations are made in the same chapter to optimize the speaker ratios in the training set, and the resulting ratios formed the baseline of the system proposed in Chapter 5.

A new two-stage decision making algorithm is proposed in Chapter 5 and described in details. The results for this proposed algorithm are reported.

The conclusions of the thesis are reported in Chapter 6.

CHAPTER TWO
VOICE RECOGNITION TECHNOLOGY

Speech processing by computer is a major field of endeavor. It is multidisciplinary, encompassing electrical engineering, computer science, linguistics, speech communication, telecommunications, among others (Rodman, 1998). Speaker recognition is a complementary research area to speech recognition. Both techniques use similar methods of speech signal processing up to a point, but speech recognition, if it is to be speaker independent, must purposefully ignore any unique speech characteristics of the speaker, and focus on those aspects of the speech signal richest in linguistic information. Conversely, speaker recognition must amplify those unique speech characteristics that individuate a person.

This chapter first gives a brief explanation about voice production and perception theory of human beings. Then, history of voice recognition technology follows.

2.1. What Makes Each Human Voice Distinct

2.1.1 Speech Production, Human Articulary System

Every person has vocal characteristics which are sufficiently unique so that one can recognize an individual by their voice alone. This leads one to believe that there is a significant amount of phonetic information in the acoustic speech signal which is independent from the message to be conveyed. These characteristics are directly related to the physiology of the speaker. Figure 2.1 displays a schematic of the human vocal apparatus (Xafopoulos, 2001). The physical features shown are crucial components in the formation of distinct sounds, or phonemes, which are used to construct words. The process of generating speech begins in the lungs. Muscle concentration forces air out of the lungs, into the *bronchi* and through the *trachea.* The air is then forced through the *larynx*, which contains the *vocal cords* and the *glottis*. The glottis is the opening between the vocal cords through which air can be forced. Above the larynx is the *epiglottis*, which permits food to enter the *esophagus*, while covering and protecting the respiratory path.

By definition, the vocal tract begins at the vocal cords and ends at the lips, while the *nasal tract* begins at the *velum* and ends at the nostrils. The velum regulates the opening to the nasal cavity, and when lowered, acoustically couples the nasal tract to the vocal tract. The final, and the often most critical components of the vocal tract are the lips, teeth, and tongue, contained in the oral cavity (Morgan & Scofield, 1991).

To understand how speakers are distinguished by their voices, it is important to first understand how the acoustic signal is produced. A widely used model of speech production system, the *source-filter model*, is shown in Figure 2.2.

The source-filter model patterns the vocal tract as a (usually linear) time varying filter. The source energy for this filter is the excitation signal. The different ways of coding this excitation signal are generally what separates these source-filter speech coders from one another. The source-filter model results from considering the excitation and vocal tract as separable components in the production of speech. The excitation is produced at some point in the vocal tract, and then the excitation is spectrally shaped (or filtered) by the rest of the vocal tract.

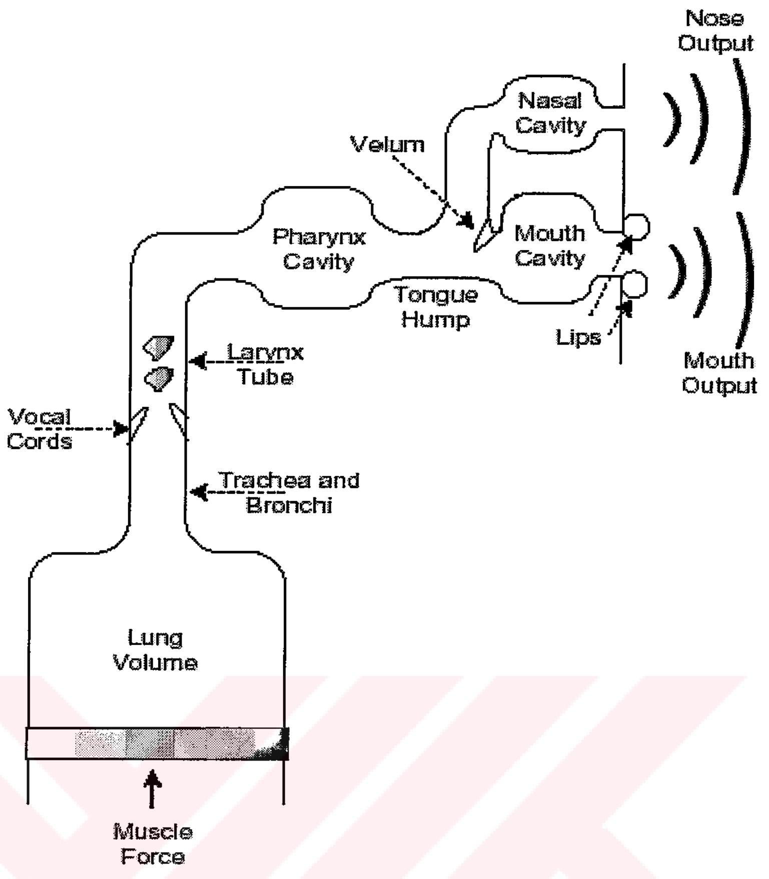

Figure 2.1 Schematic Diagram of Human Vocal System

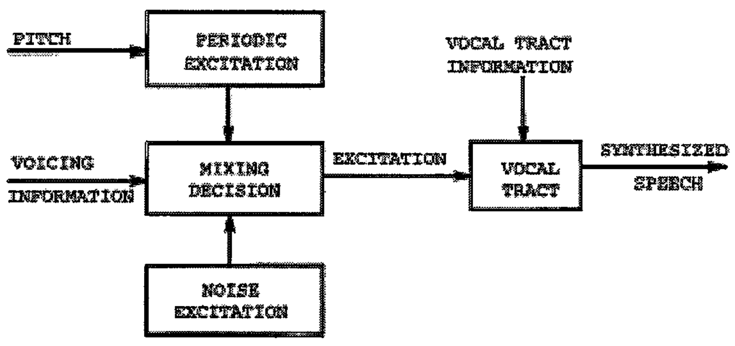

Figure 2.2 General Source-Filter Model

The Vocal Tract: The throat, nose, tongue, and mouth form a resonating air-filled cavity that predominantly dictates the sound produced by the human vocal system. The resonant frequencies of this tube are called *formant frequencies*. Different configurations of the vocal tract result in different formant frequencies. The formant frequencies are one of the two major factors that dictate which phoneme will be produced by the vocal tract. The other major factor is the excitation of the vocal tract.

Excitation: For voiced speech, a periodic waveform provides the excitation to the vocal tract. The periodic waveform results from the glottal pulses created by the rapid opening and closing of the vocal cords. A simple and widely used model for unvoiced speech is shaped white noise. White noise is random and has a flat spectral shape where all frequencies have equal power. The white noise is assumed to be generated when air passes through a constriction. Some sounds such as /z/ are produced by both exciting the vocal tract with a periodic excitation and by forcing air through a constriction in the vocal tract. This is called *mixed excitation.* One of the challenges in speech coding is to be able to accurately represent sounds that are voiced, unvoiced, or mixed.

2.1.2 Speech Perception, Human Auditory System

The basilar membrane is a key component of the inner ear. Oversimplified, sound vibrations cause movement of the basilar membrane by transduction through the middle ear. Movement of the basilar membrane stimulates hair cells, which in turn produce impulses in the auditory nerve fibers.

Ohm and Von Helmholtz (Helmholtz, 1954) were the first to present the notion that the basilar membrane acts as a spectrum analyzer. Von Békésy expounded upon this theory and demonstrated that the basilar membrane vibrates locally, and the point of vibration is related monotonically to the frequency of the acoustic stimulus (Békésy, 1953; 1960). Von Békésy proved that the basilar membrane was a spectrum analyzer, not an array of tuned resonators, but a nonuniform (almost logarithmically scaled) transmission line with limited but distinct spectral resolution. Further experimentation showed that this limited spectral resolution was characterized by *critical bands* (Beranek, 1986), which can be thought of as a frequency span, or frequency “bin,” into which sounds are lumped perceptually. It is also adequate to say that the critical band is a frequency range, defined by its band edges (specific frequencies), outside of which subjective responses change abruptly.

Figure 2.3 shows the results of these experiments for single ear listening. As can be seen from this figure, at center frequencies greater than 500 Hz, critical bandwidth increases approximately linearly as center frequency increases logarithmically.

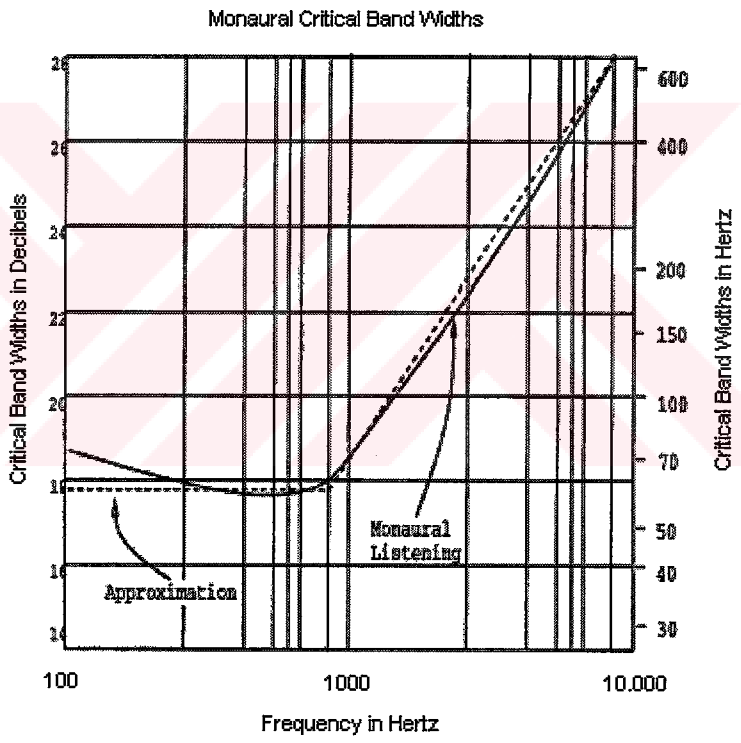

Figure 2.3 Frequency width of critical bands as a function of band center frequency

Figure 2.3 is the basis for the Bark domain and the Mel domain, which are used for representing human ear’s response to sounds. Both the Bark and the Mel domains were created to have a constant number of each unit (Barks or Mels) in each critical band. The Bark domain was normalized to have 1 Bark per critical band. Barks and Mels are perceptually based frequency units that increase, almost logarithmically, with frequency.

Table 2.1 (Beranek, 1986) illustrates the relationship between the frequency units of Barks, Mels, and Hertz. The table shows that each critical band contains a logarithmically increasing frequency bandwidth in the linear scale of Hertz. Approximately 150-200 Mels span each critical band. By definition, there is 1 Bark per critical band.

Table 2.1 The relationship between the frequency units: Barks, Hertz, and Mels.

Frequency (Hz)	Critical Band No. (Barks)	Mels
20-100	1	0-150
100-200	2	150-300
200-300	3	300-400
300-400	4	400-500
400-510	5	500-600
510-630	6	600-700
630-770	7	700-800
770-920	8	800-950
920-1080	9	950-1050
1080-1270	10	1050-1150
1270-1480	11	1150-1300
1480-1720	12	1300-1400
1720-2000	13	1400-1550
2000-2320	14	1550-1700
2320-2700	15	1700-1850
2700-3150	16	1850-2000
3150-3700	17	2000-2150
3700-4400	18	2150-2300
4400-5300	19	2300-2500
5300-6400	20	2500-2700
6400-7200	21	2700-2850
7200-9500	22	2850-3050

Figure 2.4 shows a graph of Barks versus Mels. Although the line is somewhat linear, it is not exactly linear. This is because all of the information known about critical bands is a result of experimental tests, which are far from exact. The fact that both units are so close to being linearly related even though they are formed on the basis of separate experimental tests, supports the validity of these frequency scalings.

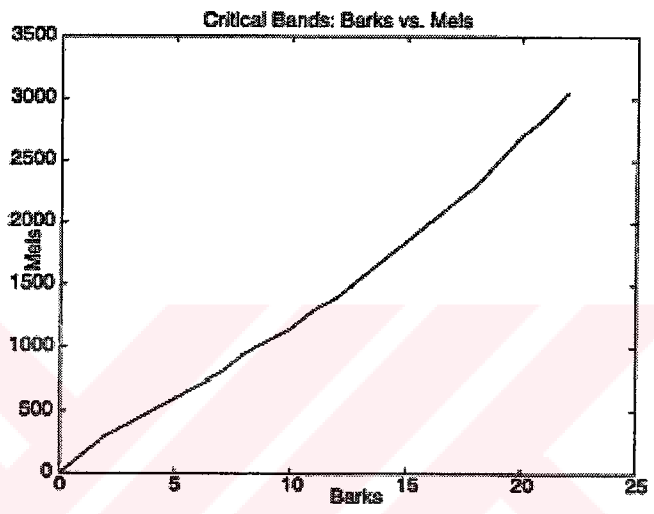

Figure 2.4 Comparing the experimentally derived frequency scales of Barks versus Mels

2.2. History

Research on speaker recognition began in the 1960's when scientists attempted to use the speech spectrogram as a tool for speaker recognition (Bolt et al., 1969), (Kersta, 1962), (Stevens et al., 1968), (Tosi et al., 1972). Even with human experts interpreting the spectrograms, the results were limited. At the time computer technology was not sufficiently advanced to aid the process.

In late 1960's, Atal and Itakura independently developed a spectral analysis method, now known as linear prediction (Chen, 1998). While motivations were different, they made an identical assumption, that the speech signal at time *t* could be approximately predicted by a linear combination of its past values. The linear prediction was used to model the frequency response of the vocal tract.

Advances in computer technology in the post-1960s triggered a series of research projects on speaker recognition. Although progress was made in the area of text-dependent speaker recognition, text-independent systems that could deal with channel and speaker variability were not as successful.

It took a number of years for the research to achieve commercialization. The earliest of those commercial systems applied speaker verification to door-access control (Markowitz, 2002). Most of those systems were designed to accept text-dependent input via microphone and they generally used dynamic time warping with filter banks. Later systems incorporated normalization and adaptation techniques. The most successful of the early algorithms was developed by Texas Instruments (Doddington, 1976). The "TI algorithm" is still used in some commercial products.

The mel-cepstrum was introduced in 1980, which was a result of a study of human auditory perception. In 1983, statistical language models began to be used. The spectral dynamics were to be included as additional features in late 1980's.

The 1990's and 2000's have witnessed the flowering of commercial speaker recognition. Algorithms diversified to include Hidden Markov Models (HMM), Vector Quantization (VQ), Gaussian Mixture Models (GMM), various types of Neural Networks, and performance enhancement techniques, such as channel compensation, background noise cancellation and anti-speaker modeling. Recent projects have reported very low error rates, and that, speaker verification is ready to compete with other more mature biometrics in real world deployments.

CHAPTER THREE

THEORY OF SPEAKER VERIFICATION

The general approach to ASV consists of five steps: digital speech data acquisition, feature extraction, pattern matching, making an accept/reject decision, and enrollment to generate speaker reference models. A block diagram of this procedure is shown in Figure 3.1. These steps of speaker verification will be detailed in this chapter.

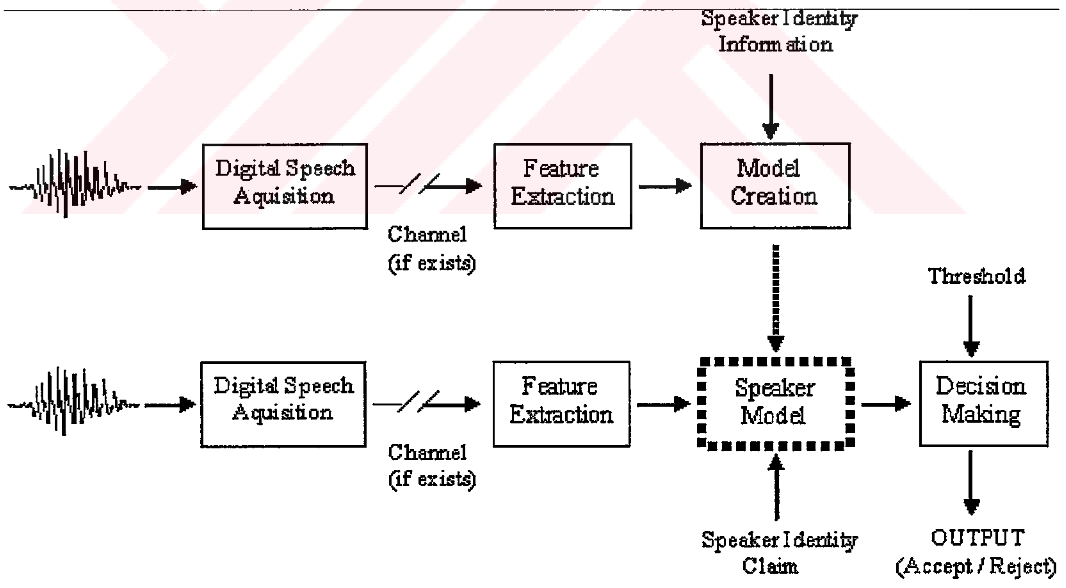

Figure 3.1 Block Diagram of Generic Speaker Verification System

The upper part of the diagram represents the enrollment phase while the lower part represents the verification (or test) phase.

3.1 Digital Speech Acquisition

Initially, the acoustic sound pressure wave is transformed into a digital signal suitable for voice processing. A microphone or telephone handset can be used to convert the acoustic wave into an analog signal. This analog signal is conditioned with antialiasing filtering (and possibly additional filtering to compensate for any channel impairments). The antialiasing filter limits the bandwidth of the signal to approximately the Nyquist rate (half the sampling rate) before sampling. The conditioned analog signal is then sampled to form a digital signal by an analog-to-digital (A/D) converter. Today's A/D converters for speech applications typically sample with 12–16 bits of resolution at 8000–20000 samples per second. Higher sampling rate is commonly used to allow a simpler analog antialiasing filter and to control the fidelity of the sampled signal precisely (e.g., sigma–delta converters).

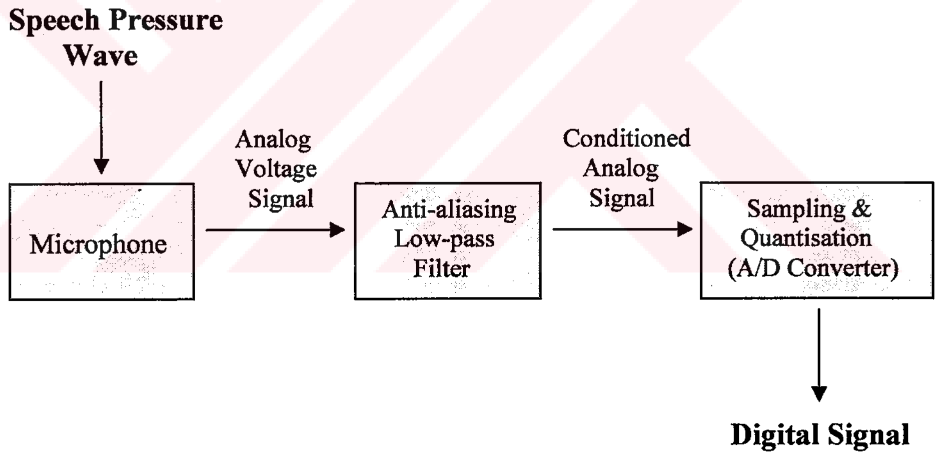
Speech Pressure Wave
Microphone
Analog Voltage Signal
Anti-aliasing Low-pass Filter
Conditioned Analog Signal
Sampling & Quantisation (A/D Converter)
Digital Signal

Figure 3.2 Acquisition of Digital Speech

In local speaker-verification applications, the analog channel is simply the microphone, its cable, and analog signal conditioning. Thus, the resulting digital signal can be very high quality, lacking distortions produced by transmission of analog signals over long-distance telephone lines (Rabiner, 1989).

3.2 Feature Extraction

Feature extraction, by definition, is the estimation of variables called feature vector from another set of variables, at a considerably lower information rate. Feature selection is the transformation of these observation vectors to feature vectors. Thus, in the process of ASV, the goal of feature selection is to find a transformation that yields relatively a lower dimensional feature space, that would preserve information pertinent to the application and would enable meaningful comparisons to be performed between the feature vectors and the speaker models using simple means of similarity.

As more features are used, the feature dimensions increase, which imposes severe requirements on computation and storage in both training and testing. The demand for a large amount of training data to represent a speaker's voice characteristics grows exponentially with the dimension of the feature space (Figure 3.3). This severely restricts the usefulness of nonparametric procedures and higher order transforms (Rabiner, 1989), (Cohen, 2003), (Premakanthan & Mikhael, 2001). The principal component analysis (PCA) and the factor analysis (FA) have been used as statistical methods to reduce the dimensionality. The PCA is used to find a lower dimensional representation that accounts for the variance of the speakers while the Factorial analysis (FA) seeks a lower dimensional representation that accounts for correlation among the features. To make the ASV problem more mathematically tractable, techniques that use analysis of the variance methods have been used. These methods involve the calculation of Fisher's F ratio tests. The discrimination of speakers increases if the statistical distributions of different speakers are concentrated at widely different locations in the parameter space (Orman, 2000). In a one dimensional parameter space, the ratio of interspeaker to intraspeaker variance, which is called as F-ratio, gives a good measure of the discriminative performance for the evaluated feature. A high value of F ratio tests is desirable for speaker verification. The F ratio is given by,

$$F_{ratio} = \frac{Variance\ of\ the\ Speaker\ Means}{Average\ of\ the\ Intraspeaker\ Variances} \tag{3.1}$$

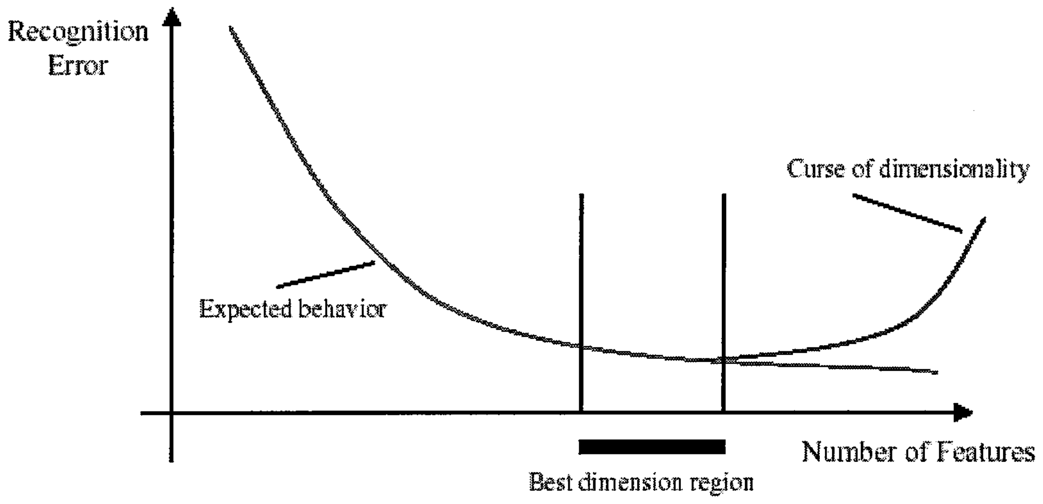

Figure 3.3 "Curse of Dimensionality" in Feature Selection

3.2.1 Pre-processing

Pre-processing is an essential part of feature extraction and as the name implies, preprocessing involves the conditioning of digital speech signal prior to extracting the speaker-specific features from the speech signal. Figure 3.4 displays the steps of preprocessing.

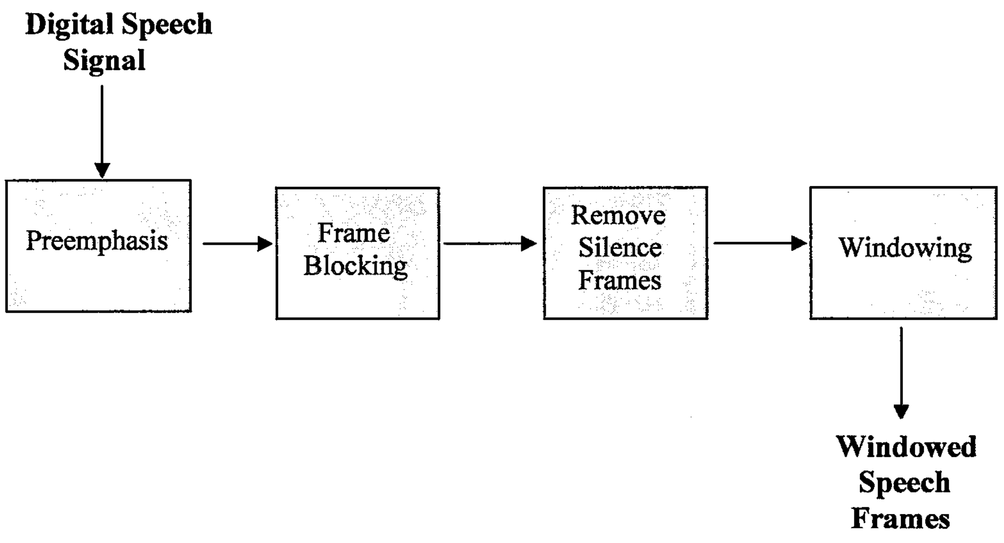

Figure 3.4 Block Diagram of Pre-processing

3.2.1.1 Pre-emphasis

The reasons for employing a preemphasis filter are twofold. First, it has been argued (Ehab et al., 2000) that minimum phase component of the glottal signal can be modeled by a simple two-real-pole filter whose poles are near $z=1$. Further, the lip radiation characteristic, with its zero near $z=1$, tends to cancel the spectral effects of one of the glottal poles. By introducing a second zero near $z=1$, the spectral contributions of the larynx and lips have been effectively eliminated and the analysis can be asserted to be seeking parameters corresponding to the vocal tract only. It is clear that the pre-emphasis will give the higher formants in the vocal tract a better chance to influence the outcome. Since the pre-emphasis suppresses the low frequencies, it is useful to eliminate the 50 Hz power supply noise eventually generated by the sound card.

$$y(n) = x(n) - a\,x(n-1), \qquad 1 \le n < M \tag{3.2}$$

The FIR filter can be implemented as in Equation (3.2), where M is the number of samples in the speech signal x(n), and y(n) is the pre-emphasized signal. The constant "*a*" is generally selected between 0.95 and 0.98.

3.2.1.2 Frame Blocking

The speech signal can be considered a quasi-stationary signal. An example of speech signal is shown in Figure 3.5. When examined over a sufficiently short period of time (between 5 and 100 msec), its characteristics are fairly stationary. However, over long periods of time (on the order of 1/5 seconds or more) the signal characteristic change to reflect the different speech sounds being spoken. Therefore, short-time spectral analysis is the most common way to characterize the speech signal. Most of the state-of-the-art systems today use a frame duration between 10 and 30 miliseconds.

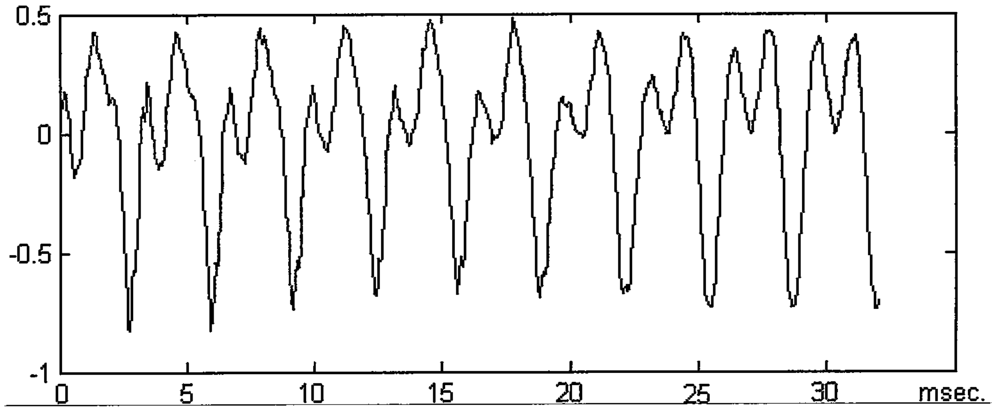

Figure 3.5 Example Speech Frame of 32 msec.

3.2.1.3 Removing Silence Frames

Next step in preprocessing is to remove the silence frames which can occur before, during, or after the utterance. Since silence frames contain no speaker-specific information, these frames should be eliminated not to degrade the recognition performance. There are several methods for removing silence frames, one of them is the Rabiner and Sambur method (1975), which first finds a threshold value from the minimum and maximum frame energies, then compares each individual frame with this threshold and eliminates the frames below threshold.

$$\zeta = \min(100\,e_{\min}, 0.75\,e_{\max} + 0.25\,e_{\min}) \qquad (3.3)$$

Where "ζ" is the silence removing threshold, e_{min} is the energy of the minimum-energy frame, e_{max} is the energy of the maximum-energy frame.

3.2.1.4 Windowing

The last step in pre-processing is to window each individual frame so as to minimize the signal discontinuities at the beginning and end of each frame. The concept here is to minimize the spectral distortion by using the window to taper the signal to zero at the beginning and end of each frame. If we define the window as *w(n)*, $0 \le n \le N-1$, where *N* is the number of samples in each frame, then the result of windowing the signal is given as:

$$y_1(n) = x_1(n) * w(n), \qquad 0 \le n \le N-1 \tag{3.4}$$

Different window shapes are realized by applying a weighting function. Most typical is the Hamming window with $\alpha_w = 0.54$,

$$w(n) = \frac{\alpha_w - (1-\alpha_w)\cos(2\pi n/(N_s - 1))}{\beta_w} \tag{3.5}$$

The β_w parameter is chosen for normalization so that the energy of the signal will be unchanged through the operation. The Hamming window's shape provides spectral analysis with a flatter pass band and significantly less stop band ripple, both properties are important for obtaining smoothly varying parametric estimates.

3.2.2 Types of Features Used in ASR

Most frequently used features in ASR systems can be listed as below:

Vocal Tract model features

- Autocorrelation Coefficients (COR)
- Linear Prediction Coefficients (LPC)
- Partial Correlation Coefficients (PARCOR)
- Log Area Ratio Coefficients (LAR)
- Perceptional Linear Prediction (PLP)

Spectral and Cepstral features

- Line Spectrum Pairs (LSP)
- Linear Prediction Cepstral Coefficients (LPCC)
- Bank of filters (linear)
- Bank of filters (Mel)
- Mel-Frequency Cepstral Coefficients (MFCC)

Linear prediction cepstral coefficients (LPC) and mel-frequency cepstral coefficients (MFCC) are the most popular type of features used in ASR systems, and therefore they will be described in the following section.

3.2.3 Linear Prediction Cepstral Coefficients (LPCC)

As the name implies the linear predictor predicts the current sample of the speech signal from a linear combination of past samples. LPC is a very important spectral estimation technique because it provides an estimate of the poles (hence the formants) of the vocal tract transfer function (Woo, 2000). The LPC algorithm is an *n*th order predictor which attempts to predict the value of any point in a time-varying linear system based on the values of previous *n* samples. The representation of the vocal tract transfer function, *H(z)*, can be given by the following equation:

$$H(z) = \frac{G}{1 - \sum_{i=1}^{P} a(i) z^{-i}} \tag{3.6}$$

The values *a(i)* are called the *prediction coefficients*, while *G* represents the amplitude, or gain, associated with the vocal tract excitation. The notation z^{-1} in the domain of z-transform represents a system function and corresponds to a unit delay in the time domain. For discrete-time signals, the z-transforms can be considered a generalization of the Fourier transform. The poles of the transfer function in Equation (3.6) are determined from the roots of the polynomial in the denominator. The LPC can only derive the resonant frequencies, or the formants, but not the zeros. The LPC does not adequately estimate signals that have no poles, such as some unvoiced speech noise. The non-linear signal components adversely affect the LPC estimates.

For the speech signal *s(n)*, the predicted speech sample *ŝ(n)* is a function of *a(i)* and prior speech samples according to:

$$\tilde{s}(n) = \sum_{i=1}^{P} a(i)s(n-i) \tag{3.7}$$

LPC analysis involves solving for the *a(i)* terms according to least squared error criteria. If the error is defined as:

$$\begin{aligned} e(n) &= s(n) - \tilde{s}(n) \\ &= s(n) - \sum_{i=1}^{P} a(i)s(n-i) \end{aligned} \tag{3.8}$$

Then taking the derivative of the squared error with respect to the coefficients *a(j)*, and setting it equal to zero gives:

$$\begin{aligned} &\frac{\partial}{\partial_{a(j)}}[s(n) - \sum_{i=1}^{P} a(i)s(n-i)]^2 = 0 \\ &[s(n) - \sum_{i=1}^{P} a(i)s(n-i)]\, s(n-j) = 0 \quad \text{for } 1 \le j \le P \end{aligned} \tag{3.9}$$

Thus,

$$s(n)s(n-j) = \sum_{i=1}^{P} a(i)s(n-i)s(n-j) \quad \text{for } 1 \le j \le P \tag{3.10}$$

A possible method for solving the matrix is called the autocorrelation method, which assumes that the signal is stationary within the analysis windows. The autocorrelation solution (3.5) can be expressed as:

$$R(j) = \sum_{i=1}^{P} a(i)R(|i-j|) \quad \text{for } 1 \le j \le P \tag{3.11}$$

Where *R(j)* is an even function (*R(j)* = *R(-j)*) and is computed from:

$$R(j) = \sum_{m=0}^{N-1-j} s(m)s(m+j) \quad \text{for } 1 \le j \le P \tag{3.12}$$

Once the autocorrelation terms *R(j)* have been calculated, a recursive algorithm, called Durbin's recursion, is used to determine the values of *a(i).* The initial state of the recursion begins with an energy term, which contains the summed, squared energy in the windowed signal,

$$E^0 = R(0) \tag{3.13}$$

At each step in the recursion the following calculations are performed:

$$\begin{aligned} &k(i) = (R(i) - \sum_{j=1}^{i-1} a^{i-1}(j)R(i-j))/E^{i-1} \quad \text{for } 1 \le j \le P \\ &a^i(i) = k(i) \\ &a^i(j) = a^{i-1}(j) - k(i)a^{i-1}(i-j) \quad \text{for } 1 \le j \le i-1 \\ &E^i = (1-k(i)^2)E^{i-1} \end{aligned} \tag{3.14}$$

The final solution for *a(j)* is given by $a^P(j)$ for $1 \le j \le P$. Given that the vocal tract does not produce a "purely" linear speech signal, the solution for *a(j)* is optimal, but not exact. The most difficult predictable part of the speech signal is the glottal pulse, since it contains a large amount of energy which "instantaneously" appears in the signal.

One can calculate the cepstrum in two ways, one is by using simple recursion and the other is with the Fourier transform.

<u>Using the Fourier Method:</u> Speech wave *x(n)* can be expressed as the convolution of glottal pulses *g(n)* and vocal tract impulse response *v(n)*. In other words,

$$x(n) = g(n) * v(n) \tag{3.15}$$

Letting the logarithmic operation for the discrete Fourier transformation be D,

$$D\{x(n)\} = D\{g(n) * v(n)\} = D\{g(n)\} + D\{v(n)\} \tag{3.16}$$

The inverse discrete Fourier transform for *D{x(n)}* is called a cepstrum. In other words,

$$\begin{aligned} c(n) &= \frac{1}{2\pi}\int_0^{2\pi} \log|X(\omega)|\, e^{jn\omega} d\omega, \\ X(\omega) &= |X(z)|_{z=e^{j\omega t}} \end{aligned} \tag{3.17}$$

The cepstrum for *x(n)* turns out to be the sum of the cepstrum for *g(n)* and the cepstrum for *v(n)*. The independent variable of the cepstrum has a time dimension (frequency). In the case of a voiced sound, *D{g(n)}* appears as a component in the neighborhood of $1/F_0$ (F_0 is the fundamental frequency) on the time axis, and *D{v(n)}* as a component of the short time domain. Thus, a window is opened in the cepstrum and the short time range components are extracted (this is accomplished by removing *g(n)*), and if a discrete Fourier transformation is performed in this, the spectral envelope is obtained.

Using the Linear Prediction Coefficients: The LPC-derived cepstral coefficients are defined as follows, where c_i is the i^{th} cepstral coefficient and a_k are the prediction coefficients:

$$\begin{aligned} c_1 &= a_1 \\ c_i &= a_i + \sum_{k=1}^{i-1}((1-(k/i))a_k c_{i-k}), \quad 1 < i \leq N \end{aligned} \tag{3.18}$$

Unlike LPC coefficients, cepstral coefficients are independent and the distance between cepstral coefficients vectors can be calculated with a Euclidean-type distance measure.

3.2.4 Mel-Frequency Cepstral Coefficients (MFCC)

The "Mel" is a unit of measure of perceived pitch or frequency of a tone (Kamm et al., 1998). This approach uses a bank of highly overlapped band-pass filters that roughly approximates the frequency response of basilar membrane in human ear to cover the frequency range of interest in a speech signal. The measurement from the outputs of those band-pass filters can be essentially treated as a short-time spectral envelope.

A mel-scale filter bank (Xin et al., 2001) consists of a sequence of overlapping triangular filters with center frequencies and bandwidths determined by the Mel frequency scale (see Figure 3.6). The mel frequency scale is based on results from psychophysical studies on humans. Each interval on the mel-scale corresponds to the perceived relative pitch of a reference tone. First, a reference tone of 1000 Hz is defined to be 1000 mels, then the reference tone is changed until the subject indicates that the pitch has doubled, this frequency corresponds to 2000 mels. This process is repeated to define the entire mel scale. Experiments by Stevens and Volkmann showed that subjective pitch in mels increases less and less rapidly as a test frequency is increased linearly.

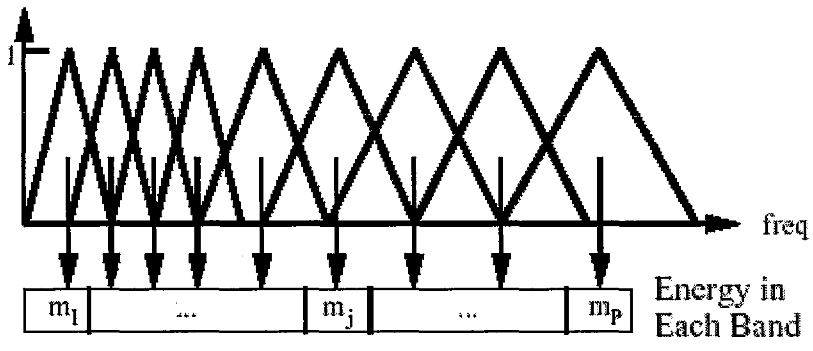

Figure 3.6 Mel-Scale Filter Bank

The relation between the Mel scale and the standard frequency scale is formulated as follows:

$$\mathrm{mel}(f) = 2595 \log_{10} (1 + \frac{f}{700}) \tag{3.19}$$

First step in calculation of MFCC features is segmentation of voice active regions of utterances to the overlapping frames and windowing with a function that is generally a Hamming or a Hanning type. In the second step, the magnitude spectrum of each frame is calculated via discrete Fourier transform. At the third step, the logarithm of magnitude spectrum is calculated and passed through a mel scale filter bank. The last step is to calculate the cosine transform of coefficients, which gives us the mel-frequency cepstral coefficients (Equation 3.20).

$$\mathrm{MFCC}_i = \sqrt{\frac{2}{N}} \sum_{j=1}^{N} m_j \cos\left(\frac{\pi i}{N}(j - 0.5)\right) \tag{3.20}$$

Where N is the number of bandpass filters, m_j is the log bandpass filter output amplitudes.

3.3 Creation of Speaker Models

Creating a speaker model from extracted features is another crucial step in speaker verification systems. The type of speaker recognition system, i.e. text dependence, directly effects modeling technique. There are different approaches proposed and tested in the literature.

For text independent systems, vector quantization (VQ) based methods, Hidden Markov Model (HMM) based methods, phoneme based methods and artificial neural network (ANN) based methods can be addressed as the main categories for speaker verification systems.

In VQ, codebooks consisting of a small number of representative feature vectors are used for characterizing speaker specific features. In this method, a speaker-specific codebook is created from training data. In test stage, the likelihood score is calculated to be used in decision-making stage as illustrated in Figure 3.7. Multi-state HMM based methods are also especially important in text dependent and vocabulary dependent systems. In text independent systems they are utilized, as well.

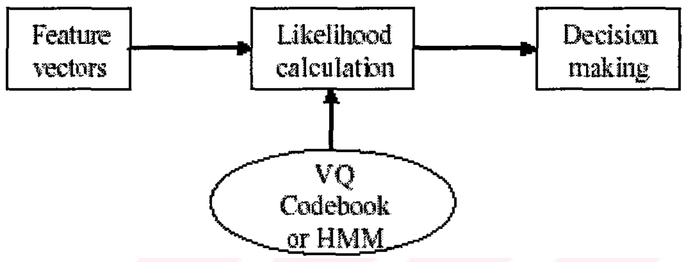

Figure 3.7 Representation of VQ and HMM based systems

It was shown (Matsui & Furui, 1992) that, although speaker recognition performance is strongly correlated with the number of mixtures, number of states of the model is not very effective. This suggests that information of state transitions is not significant for speaker recognition purposes. Therefore, using single state continuous or semi-continuous ergodic HMM is the most appropriate choice for building such systems. A single state HMM corresponds to the Gaussian mixture model (GMM) representation.

In VQ or HMM, phoneme information in speech frames is handled implicitly. On the other hand, in phoneme-based systems, phonemes or phoneme classes are explicitly recognized and modeled separately. Then, in authentication session, each speech frame is assigned to a phoneme model and likelihood score is extracted taking its phoneme class into account. Also, large vocabulary speech recognition systems are used for more exact phoneme models. But such systems are reported not to outperform GMM based systems. Difficulties in exact phoneme segmentation and request for increasing amount of enrollment and test data are possible causes.

GMM, as a simple but efficient method, is still the most popular modeling method for today's speaker verification and identification systems. In GMM, each speaker is modeled as a weighted sum of K multidimensional (L-dimensional) Gaussian probability density functions. If λ_i is the model for i^{th} speaker,

$$\lambda_i = \{p_k^i, \mu_k^i, S_k^i\} \quad 1 \leq k \leq K \tag{3.21}$$

where,

$$\mu_k^i = [\mu_{k,1}^i \mu_{k,2}^i \cdots \mu_{k,L}^i] \tag{3.22}$$

$$S_k^i = \begin{bmatrix} \sigma_{k,1}^i & & & & \\ & \sigma_{k,2}^i & 0 & 0 & \\ & & \ddots & & \\ & 0 & & \sigma_{k,L-1}^i & \\ & & & & \sigma_{k,L}^i \end{bmatrix} \tag{3.23}$$

In these equations, $P_k^i, \mu_k^i, and\ S_k^i$ are mixture coefficient, mean vector and covariance matrix for the k_{th} Gaussian component of the model. In speech recognition applications that use MFCC as the feature vector, covariance matrix is assumed to be diagonal. This is because elements of MFCC are more or less uncorrelated.

Setting of K mean vectors is a crucial step in this process. This can be achieved by a vector quantization algorithm or Expectation Maximization (EM) algorithm. There are lots of methods proposed in the literature for this purpose.

3.4 Scoring and Decision Making

In decision making phase, the aim is to create a likelihood score using the speaker model and the features extracted from the incoming authentication utterance (Naci, 2003). In speaker identification systems, the speaker model which gives the highest similarity score can be referred to as the owner of the spoken utterance. But if a verification system is the case, decision making phase is more painful. Setting an acceptance-rejection threshold is a complex issue since likelihood score range changes from one speaker's model to another's. Having computed a match score between the input speech-feature vector and a model of the claimed speaker's voice, a verification decision is made whether to accept or reject the speaker or to request another utterance (Bhattacharyya et al., 2001).

Two kinds of errors can occur in this decision-making:

α error : Rejecting a true speaker also called False Rejection Rate (FR),
β error : Accepting an imposter also called False Acceptance Rate (FA).

The speaker verification system performance and robustness are measured in terms of False Acceptance (FA) and False Rejection (FR) rates.

The FR rate for i_{th} speaker is measured as:

$$\alpha_i = \frac{\textit{Number of Rejects}}{\textit{Number of Utterances}} \tag{3.24}$$

The average FR rate for set of speakers is measured as:

$$\overline{\alpha} = \frac{1}{m}\sum_{i=1}^{m}\alpha_i \tag{3.25}$$

The FA rate for ith speaker is measured as:

$$\beta_i = \frac{\textit{Number of Impostor Accepts}}{\textit{Number of Utterances}} \qquad (3.26)$$

The average FA rate for set of speakers is measured as:

$$\bar{\beta} = \frac{1}{m}\sum_{i=1}^{m}\beta_i \qquad (3.27)$$

The FA and FR rates in the above formulas are measured for a specific threshold and change as thresholds are changed. As threshold increases, FR increases at the expense of FA. Conversely, if threshold decreases, FR decreases while FA increases. Hence a single threshold dependent FA-FR figure gives a very limited description of the system. The goal of dynamic evaluation is to provide a description of the system performance, which is as independent as possible of the threshold values. Receiver Operating Characteristic (ROC) is such a method for representing the dynamic characteristic (Figure 3.8). It is a plot of FR vs. FA for a varying threshold.

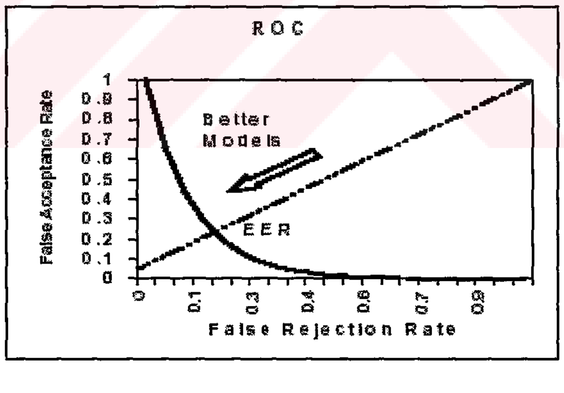

Figure 3.8 Receiver Operating Characteristics (ROC) Curve

Keeping the whole ROC curve lacks conciseness, and it is classically felt desirable to condense system performance into a single figure. Traditionally Equal Error Rate (EER), which is the point on the curve where $\alpha = \beta$, is chosen for this purpose. The EER in the above figure is 0.2. As ROC curves move towards the origin the EER decreases and the system performance improves (figure 3. 9).

The ROC curves can be speaker dependent or independent. If α and β are used to plot the ROC we get a speaker independent ROC. Whereas if we plot α_i and β_i we get the ROC for speaker *i*. There is no simple way of calculating an average ROC from individual speaker ROCs. Current practice consists in characterizing each individual ROC curve by its EER and summarizing the performance of the system by the average EER.

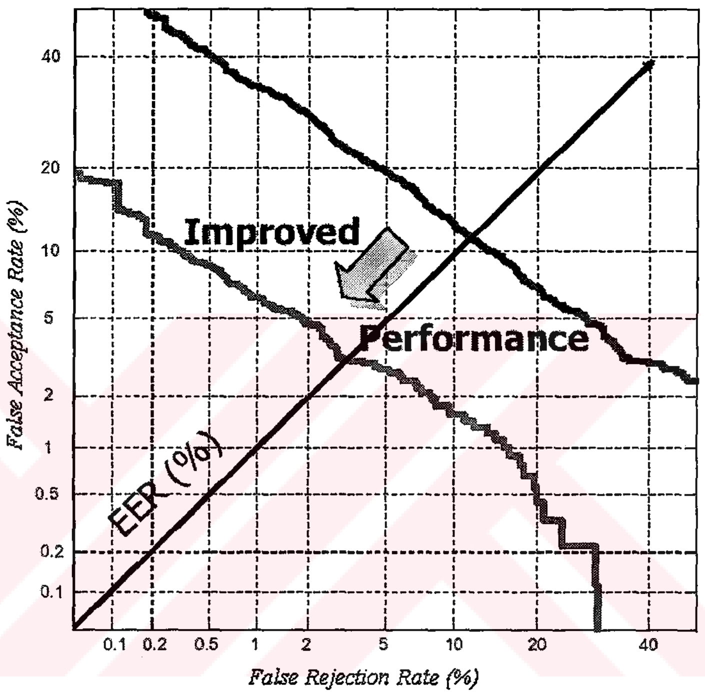

Figure 3.9 ROC Curve – Improved Performance

Proper setting of the thresholds is very critical to performance of speaker verification systems. Thresholds can be of 2 types namely, speaker dependent thresholds where every speaker has his individual threshold calculated from his individual FA-FR characteristics or speaker independent thresholds where a common threshold is used for all speakers based on the average FA-FR characteristics of the system.

The exact choice of the threshold value is determined by the operating constraints, such as, a required FR rate α_0 or a required FA rate β_0 or by evaluating the costs and benefits of marginal improvements in FA and FR and choosing a threshold that optimizes the benefits (see figure 3.10). With cautious constraints, the threshold value can be made speaker specific, speaker adaptive, and/or risk adaptive (e.g., break-ins may be more likely at night) (Rabiner, 1989).

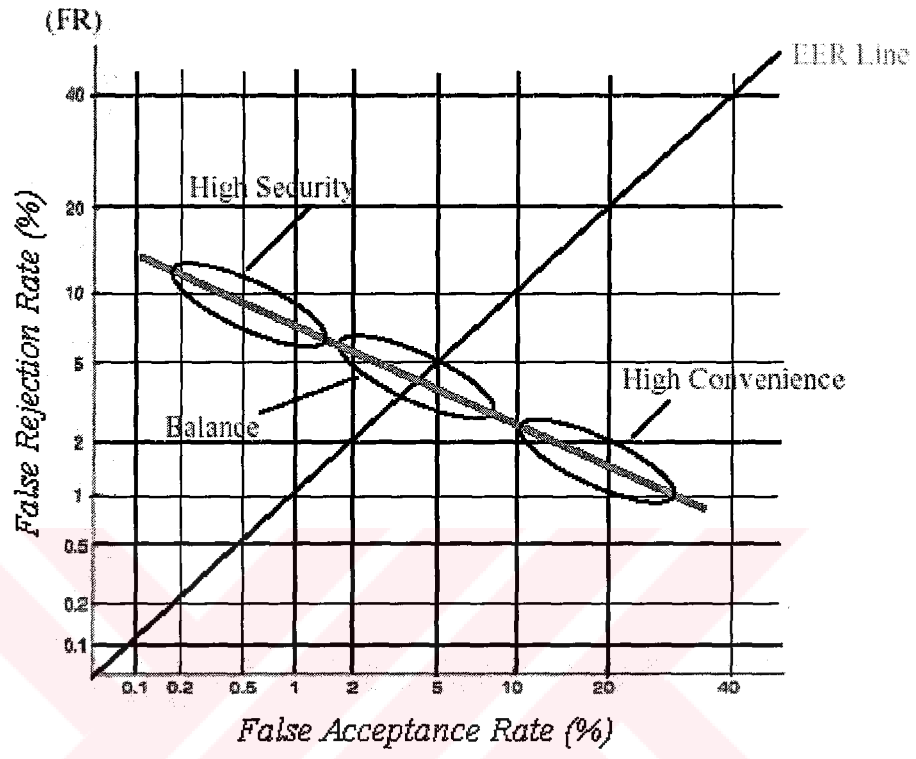

Figure 3.10 Detection Error Tradeoff (DET) Curve

The thresholds can be set a priori or a posteriori. In a priori threshold setting the threshold is estimated from tuning data, which can be the training data or a new set of unseen data. There is no intersection of the tuning data and test data. In a posteriori threshold setting the thresholds are set using the test data. Performance measures using a priori thresholds are more realistic than a posteriori thresholds.

CHAPTER FOUR
SYSTEM DESIGN AND IMPLEMENTATION

The aim of this work is to form a high performance speaker verification system that can be used especially for security purposes. For this, a baseline verification system is proposed and, several experiments and investigations are made to improve the verification performance of the proposed system. For the experiments, we used the IViE corpus (WEB_1, 2002) which is formed by 116 speakers (equal percentage of male and female speakers), each uttering 20 sentences in various lengths, which equals to 2320 utterances in total. The recordings were made in sampling rate of 16kHz.

Tests are performed using MATLAB software. First, the experiments are done for the baseline speaker verification system which is described below together with detailed results. In the second part, balancing of the training data is investigated for obtaining optimum performance. Then, by generating a cohort model, a new scoring technique is proposed, and results are reported as well. Finally, feature extraction algorithm is tuned to further decrease the verification error to a minimum value.

4.1 Implementation of the Baseline Speaker Verification System

The implemented baseline system consists of; pre-processing, MFCC features extraction, radial basis function (RBF) neural network implementation, and performance evaluation. Each individual has his/her own model, created in the training phase. The experiments are done using these models and verification scores are obtained.

4.1.1 Pre-processing

In this part, the speech signal sampled in 16kHz in digital format is taken and passed from a pre-emphasis filter of the form *"$y(n) = x(n) - 0.95\, x(n-1)$"* to suppress low frequency components. Then, the pre-emphasized speech utterance is divided into 256 sample frames, overlapping by 128 samples (16 msec. frame size, 8 msec. overlapping). Next step is to remove the silence frames according to Rabiner and Sambur method (see part 3.2.1.3), since the silence frames contain no speaker-specific information. Then, the non-silence frames are windowed by a Hamming window to minimize the signal discontinuities at the beginning and end of each frame.

4.1.2 MFCC Features Extraction

Next process is to convert the pre-emphasized, framed and windowed non silence speech frames into a spectral-domain representation. For verification purpose, we use MFCC (Mel Frequency Cepstral Coefficients) as the feature vector. These features are based on the known variation of the human ear's critical bandwidth. Frequency filters spaced linearly at low frequencies and logarithmically at high frequencies have been used to capture the phonetically important characteristics of speech. This is expressed in the mel-frequency scales, which are the linear frequency spacing below 1000 Hz and a logarithmic spacing above 1000 Hz (Rabiner & Huang, 1993). Feature extraction based on Mel Frequency Cepstral Coefficients (MFCC) utilizes the filter bank of which center frequency and bandwidth are scaled by subjective measure, Mel. As a reference point, the pitch of a 1kHz tone, 40 dB above the perceptual hearing threshold, is defined as 1000 mels. It is common knowledge that perceptually motivated feature sets (MFCC is an example of these type of feature sets) give better speaker recognition performance when compared with other methods, such as liner prediction cepstrum coefficients (LPCC) (Song & Rosenberg, 1988) and log-cepstrum coefficients (logCC) (Rabiner & Schafer, 1978).

The procedure of MFCC feature extraction is as shown in Figure 4.1.

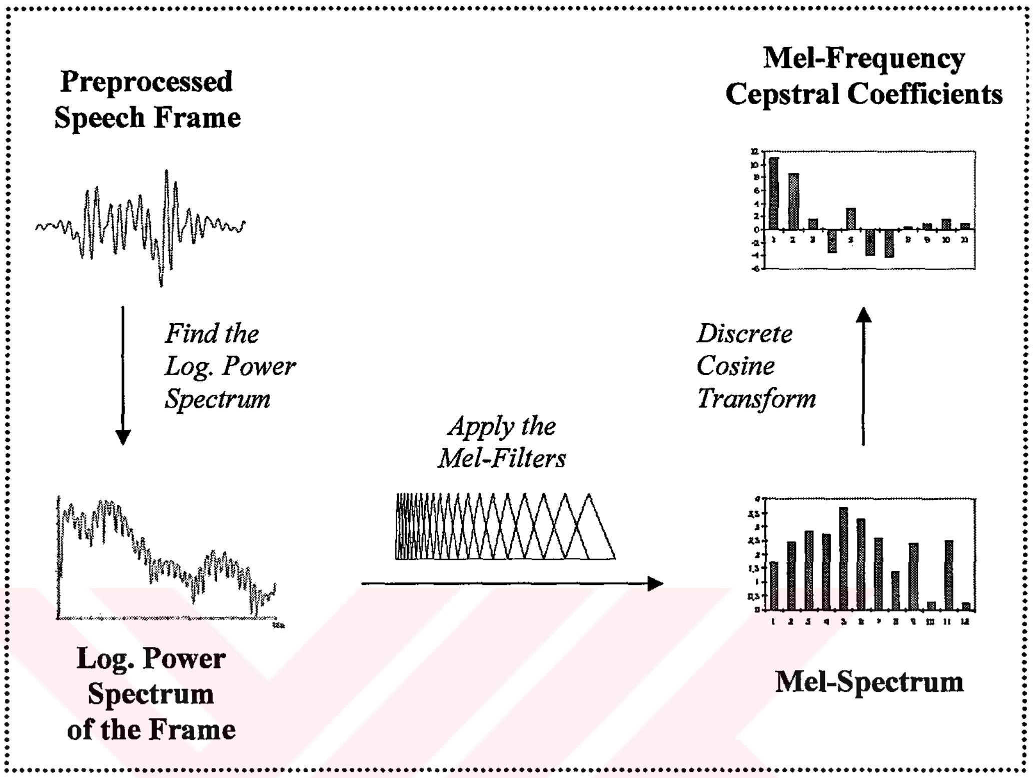
Preprocessed Speech Frame
Mel-Frequency Cepstral Coefficients
Find the Log. Power Spectrum
Discrete Cosine Transform
Apply the Mel-Filters
Log. Power Spectrum of the Frame
Mel-Spectrum

Figure 4.1 Schematic Representation of MFCC Feature Extraction

In this stage, pre-emphasized and windowed non-silence speech frames are taken and, the MFCC features are extracted. In our baseline system, 12 mel-filters are used and 11 mel-frequency cepstral coefficients are extracted as feature vector for each frame. The first coefficient component, C_0, is excluded since it carries little speaker specific information.

4.1.3 Radial Basis Function Network for Speaker Modelling

As the speaker model, we used radial basis function (RBF) network since Gaussian approximation is reported to give good results with spectral features such as MFCC's. To give a brief information, an RBF is a multidimensional function that depends on the distance between the input vector and a center vector. Figure 4.2 shows the basic structure of the RBF network. The input layer has neurons with a linear function that simply feeds the input signals to the hidden layer. Moreover, the connection between the input layer and the hidden layer are not weighted, that is, each hidden neuron receives each corresponding input value unaltered. The hidden neurons are processing units that perform the radial basis function. In contrast to the MLP network, the RBF networks usually has only one hidden layer. Examples for the transfer function of the hidden neurons in RBF network are: Gaussian, Multiquadratic, Inverse multiquadratic, Thin-plate-spline, Piece-wise linear, and Cubic approximation function.

In our system, we will use the Gaussian function as the transfer function of the hidden neurons, since it is common knowledge that Gaussian approximation provides good results in modeling the natural signals such as speech signals. The Gaussian function has the form:

$$e^{-\frac{r^2}{\sigma^2}} \tag{4.1}$$

Where, σ is a real parameter (called a scaling parameter) and r is the distance between the input vector and the center vector. The distance is measured by the Euclidean norm.

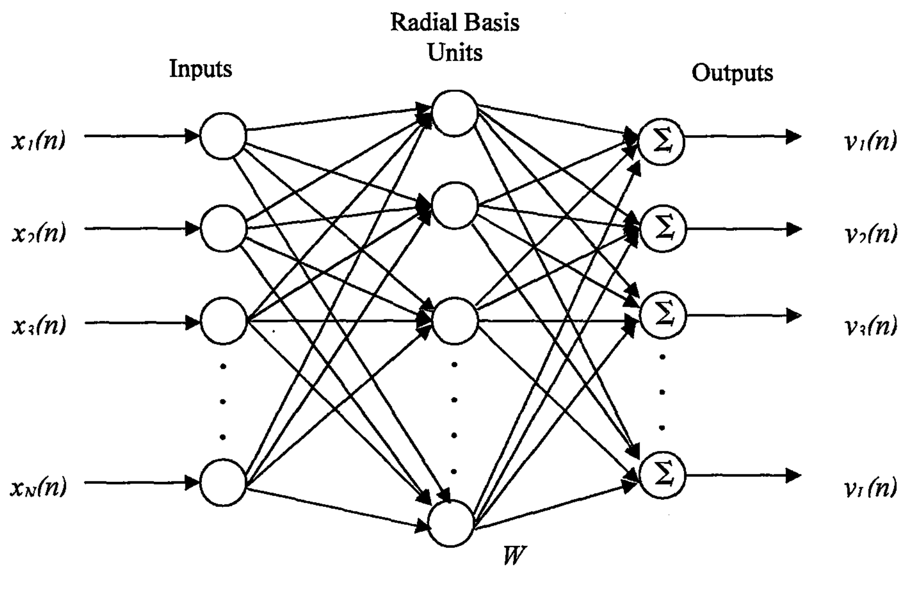

Figure 4.2 Schematic Representation of an RBF Network

The output of each neuron in hidden layer is:

$$h_i(n) = f_i(\|\boldsymbol{X}(n) - \boldsymbol{C}_i(n)\|), \quad \mathrm{i} = 1,2,\ldots,\mathrm{H} \tag{4.2}$$

Where, $\boldsymbol{X}(n)$ is the input vector at time n, $\boldsymbol{C}_i$ is the center vector of the i^{th} hidden neuron, H is the number of neurons in the hidden layer and, f_i is the transfer function of the i^{th} hidden neuron.

The connection between the hidden layer and the output layer are weighted. Each neuron of the output layer has a linear input-output relationship so that they perform simple summations; that is, the output of the i^{th} neuron in the output layer at time n is:

$$y_i(n) = \sum_{j=1}^{\mathrm{H}} W_{ij} h_j(n) = \sum_{j=1}^{\mathrm{H}} W_{ij} f_j(\|\boldsymbol{X}(n) - \boldsymbol{C}_j\|), \tag{4.3}$$

Where, $(i = 1,2,\ldots,L)$, L is the neuron number of the output layer and W_{ij} is the connection weight between the j^{th} neuron in the hidden layer and i^{th} neuron in the output layer.

It has been shown experimentally that if a sufficient number of hidden neurons is used and the center vectors are suitably distributed in the input domain, then the RBF network is able to approximate a wide class of nonlinear multidimensional functions. The approximation performance of an RBF network critically depends on the choice of the centers (Luo & Unbehaugen, 1997), (Chen et al., 1991).

From Figure 4.2 and Equations (4.2) and (4.3), we know that RBF network is specified by two set of parameters: the connection weights and the center vectors. These parameters can be determined from the available sample vectors (training data) by solving the optimization problem:

$$E = \sum_{n=1}^{M} ||\boldsymbol{Y}(n) - \hat{\boldsymbol{Y}}(n)||^2 \tag{4.4}$$

Where, M is the number of available sample vectors, $\boldsymbol{Y}(n)$ is the computed output and $\hat{\boldsymbol{Y}}(n)$ is the desired output from the network.

For the center vectors the simplest technique involves choosing these vectors randomly from a subset of the available sample vectors. However, in such a case the number of hidden neurons needs to be relatively large to cover the entire input domain. An improved approach is to apply the so-called *k*-means clustering algorithm (Moody & Darken, 1989). This algorithm finds a set of cluster centers and partitions the training samples into subsets. Each cluster center is associated with one of the H hidden neurons in the RBF network. The data are partitioned in such a way that the training points are assigned to the cluster with the nearest center (Luo & Unbehaugen, 1997).

For the connection weights determination, the following recursive procedure can be used:

1) Initialize randomly all connection weights.
2) Compute the output vector $\boldsymbol{Y}(n)$ by the Equation (4.3).
3) Compute the error term $e_i(n)$ of each output neuron

$$e_i(\mathrm{n}) = y_i(\mathrm{n}) - \hat{y}_i(\mathrm{n}) \tag{4.5}$$

4) Adjust the connection weights according to

$$W_{ij}(\mathrm{n}+1) = W_{ij}(\mathrm{n}) + \gamma \mathrm{e}_i(\mathrm{n}) f_i(||\boldsymbol{X}(n) - \boldsymbol{C}_i||) \tag{4.6}$$

Where γ is the learning rate parameter.

5) Compute the total error

$$e(n) = \|\boldsymbol{Y}(n) - \hat{\boldsymbol{Y}}(n)\|^2 \qquad (4.7)$$

and iterate the computation *y* returning to Step(2) until this error is less than the specified one.

4.1.4 Performance of the Baseline System

In our baseline system described above, each speaker model is trained using RBF network, with a world set of 27 non-speakers. The output layer of the RBF network had 2 neurons, where the first output neuron represents the probability of being a true speaker feature, the second output neuron represents the probability of being a non-speaker feature (Figure 4.3).

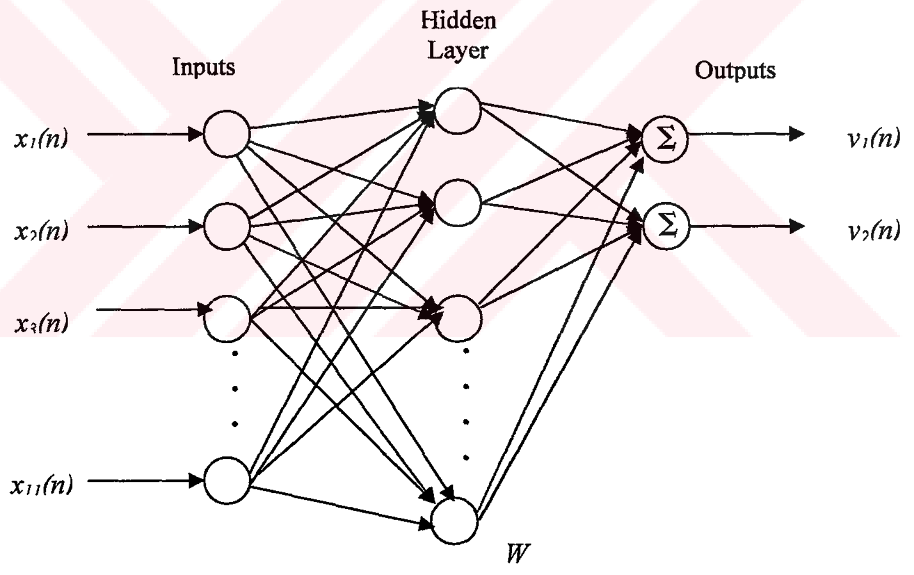

Figure 4.3 The Proposed RBF Network

The RBF network was trained to give the outputs as in the table 4.1.

Table 4.1 The Proposed RBF Network Outputs

X(n)	y1(n)	y2(n)
True Speaker Feature	1	0
Non-Speaker Feature	0	1

The score of each test utterance was evaluated as:

$$S = \frac{1}{M} \sum_{n=1}^{M} y_1(n) - y_2(n) \tag{4.8}$$

Where, *M* is the number of feature vectors in the test utterance.

Training phase takes for about 20 minutes by a Pentium-4, 1.7GHz PC with 256MB RAM, including MFCC features extraction. Tests are done with 288 non-speaker utterances and 26 true speaker utterances. In this stage, the performance was evaluated in terms of equal error rate (EER), since the speaker acceptance/rejection threshold greatly affects the false acceptance (FA) and false rejection (FR) error values (see Figure 4.4 and 4.5).

Models are created for 4 male speakers and 4 female speakers (The abbreviations of the speaker names are given in Table 4.2). The amount of true speaker features was chosen to be equal to the amount of non-speaker features in the training set. Also, the amount of male non-speaker features were equal to the amount of female non-speaker features in the world set. The test scores for the baseline system are as stated below:

Table 4.2 Test scores in terms of EER for the baseline system

	Female Speakers					Male Speakers				
Speaker	**wsc**	**wkt**	**mmm**	**cmf**	**Average**	**wlh**	**wer**	**mpm**	**cmc**	**Average**
EER (%)	9.7	9.3	6.4	6.9	**8.1**	3.2	5.5	4.9	6.7	**5.1**

As seen in Table 4.2, the average equal error rate of these eight speaker test models for the baseline system is 6.6 %. The next investigation will be made in order to decrease the equal error rate (in part 4.2) by optimizing the training set in terms of:

1) The ratio of same gender speakers/opposite gender speakers in non-speaker set,
2) The ratio of speaker features/non-speaker features in the training set.

Figures 4.4 and 4.5 are the FAR-FRR graphs for two different speakers which are trained and tested according to the baseline model. The equal error rate, as a performance value, is the point where FAR equals to FRR.

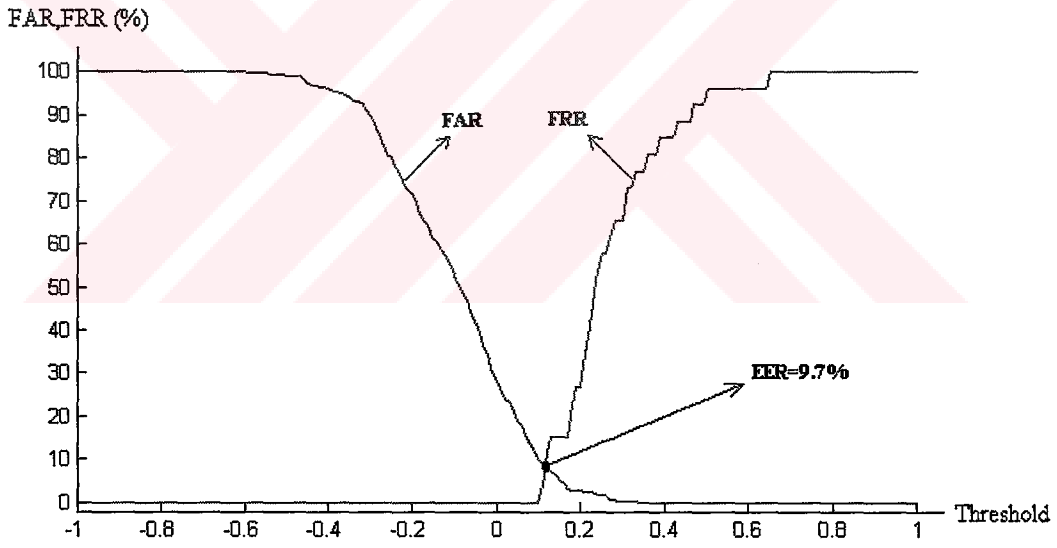

Figure 4.4 Performance of Speaker "wsc" for Varying Thresholds

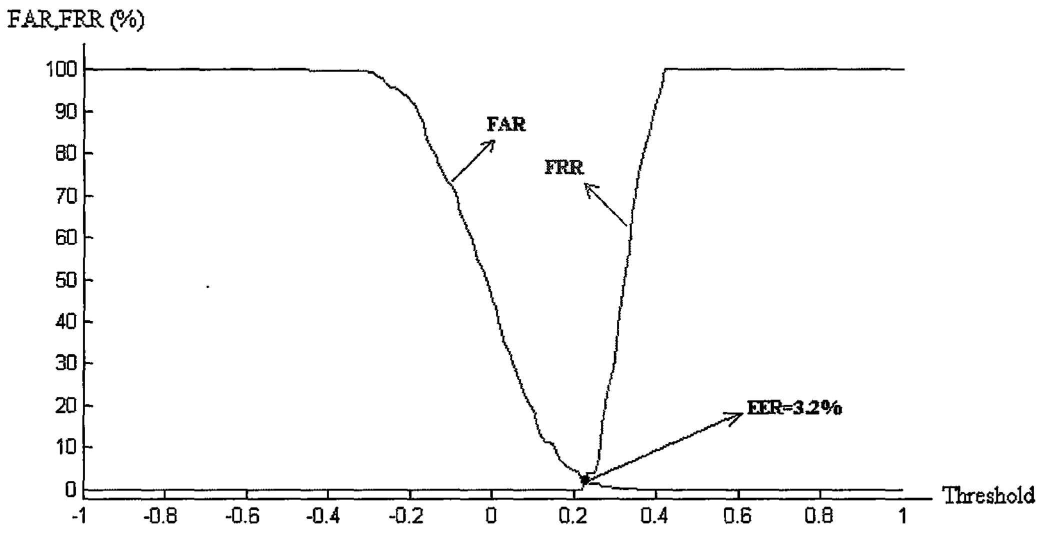

Figure 4.5 Performance of Speaker "wlh" for Varying Thresholds

4.2 Optimizing the Training Set

In the baseline system, the feature vectors in the training set were composed of two types of features:

1) True speaker features (also denoted as "speaker features" only).
Amount: 12 different utterances, total 540 feature vectors .

2) Non-speaker features (also denoted as "world set").
Amount: 27 non-speakers, 20 feature vectors from each, total 540 feature vectors.

Parts 4.2.1 and 4.2.2 investigate the optimization of the training set.

4.2.1 Ratio of Same-Gender/Opposite-Gender Features in the Training Set

The concept in the title of this part can be explained as:

If the verification model belongs to a male speaker, the male non-speakers are the same-gender non-speakers, the female non-speakers are the opposite-gender non-speakers.

We made experiments to determine the optimum ratio of speaker genders in the non-speaker set. The results are shown in Table 4.3.

Table 4.3 EER Performances for Different Percentages of Same-Gender Non-Speakers in Training Set

EER(%)		Speaker Name								
		wsc	wkt	mmm	cmf	wlh	wer	mpm	cmc	Average EER
Percentage Of Same Gender Non-Speakers	**33%**	10.1	11.6	7.6	8.9	4.5	7.2	6.3	10.3	**8.3**
	50%	9.7	9.3	6.4	6.9	3.2	5.5	4.9	6.7	**6.6**
	67%	7.1	6.9	6.0	5.9	1.8	4.1	4.3	6.4	**5.4**
	80%	6.6	6.2	4.9	5.7	1.3	4.5	4.3	5.6	**4.9**
	90%	**6.9**	**6.1**	**4.6**	**4.5**	**1.2**	**3.3**	**3.2**	**4.6**	**4.3**
	100%	7.7	8.2	6.9	7.4	2.4	6.2	5.8	7.4	**4.5**

It can be seen from Table 4.3 that, lower equal error rate (EER) value is achieved by increasing the percentage of same-gender non-speaker features in the training set, up to 90 percent. Then, the EER slightly increases when the network is trained with 100 percent of same-gender non-speakers. This means that, we will continue our tests with the percentage of 90%. (e.g. if the verification model belongs to a male speaker, then the network will be trained with 90% male-non-speakers and 10% female-non-speakers. If the verification model belongs to a female speaker, this is the direct contrary).

Figures 4.6 and 4.7 are the FAR-FRR graphs for the situation when there is 90% same-gender non-speakers and 10% opposite-gender non-speakers. When Figures 4.6 and 4.7 are compared with Figures 4.4 and 4.5, the improvement in performance can easily be observed.

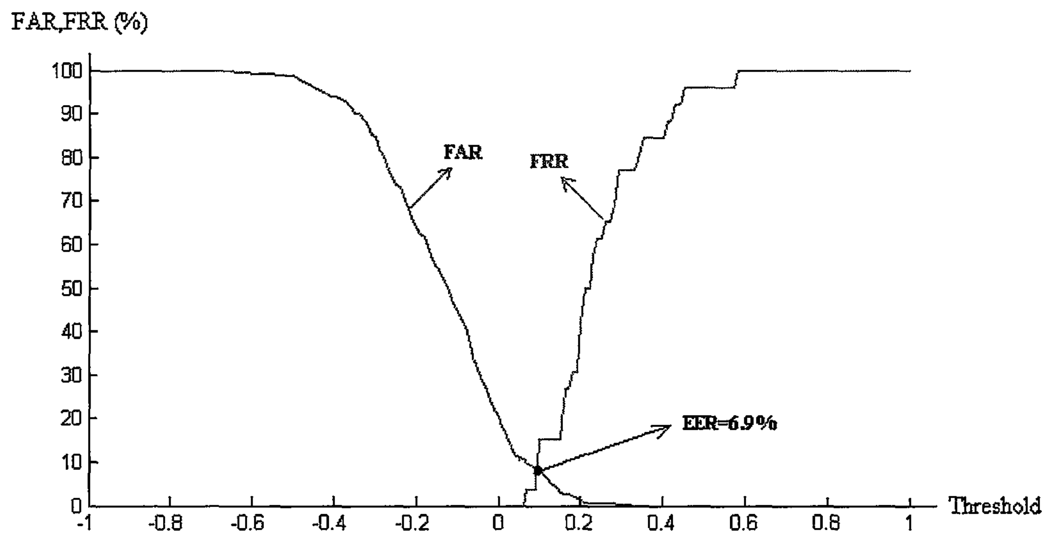

Figure 4.6 Performance of Speaker "wsc" when trained with 90% same-gender and 10% opposite-gender non-speakers

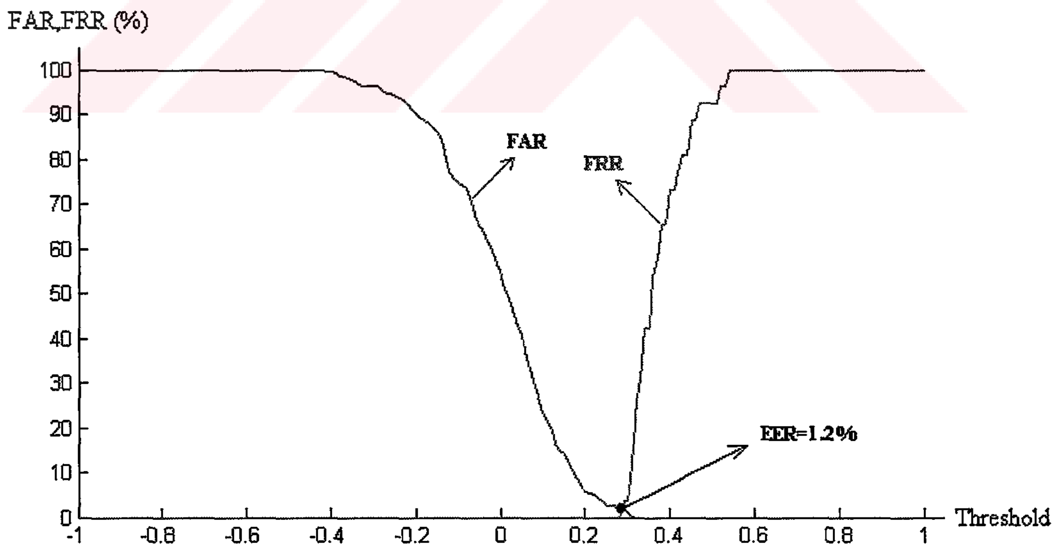

Figure 4.7 Performance of Speaker "wlh" when trained with 90% same-gender and 10% opposite-gender non-speakers

4.2.2 Ratio of True Speaker / Non-Speaker Features in the Training Set

As we know, the RBF Network should be trained with both true-speaker and non-speaker features in order to discriminate the speaker's specific vocal properties from the other speakers. The ratio of these true speaker features to non-speaker features is also an important point that affects the verification performance.

The results from experiments for different true-speaker percentages in the training set is as shown in Table 4.4 (non-speaker features set is composed of 90% same-gender non-speakers, and 10% opposite-gender non-speakers).

Table 4.4 EER Performances for Different Percentages of True Speaker Features in Training Set

EER(%)		Speaker Name								
		wsc	wkt	mmm	cmf	wlh	wer	mpm	cmc	Average EER
Percentage Of True Speaker Features	**33%**	8.1	7.2	5.7	6.2	2.6	4.3	4.0	5.4	**5.4**
	40%	7.3	7.1	5.5	5.4	1.5	3.8	3.8	5.1	**4.9**
	50%	**6.9**	**6.1**	**4.6**	**4.5**	**1.2**	**3.3**	**3.2**	**4.6**	**4.3**
	60%	7.9	6.9	5.4	5.6	2.0	4.1	4.2	5.3	**5.2**
	67%	13.5	11.8	8.6	9.4	6.1	6.9	7.2	9.5	**9.1**

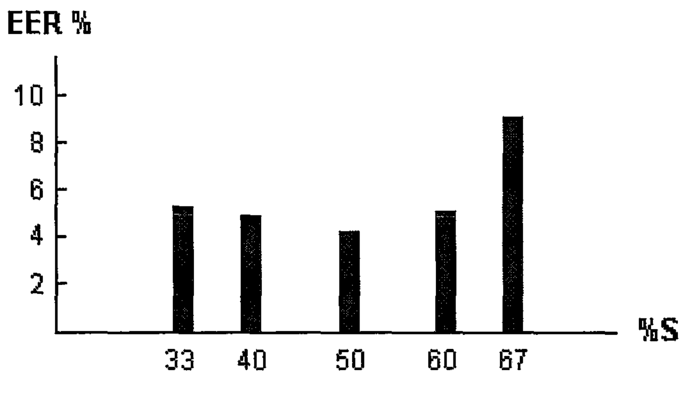

Figure 4.8 Effect of True Speaker Features Percentage in the Training Set
(%S = Speaker features / Total features in training set)

As seen from Table 4.4 and Figure 4.8, minimum equal error rate value (EER) is achieved when the number of speaker features in the training set becomes equal to the number of non-speaker features (when %S = 50%). Also, it is obvious from Table 4.4 that when the percentage of speaker features in the training set is increased, the error rate is considerably increased.

Evaluating the test results, in Part 4.2.1 and Part 4.2.2 for optimizing the training set, we are going into Chapter 5 with the knowledge such that, we will train our main RBF network with equal amount of speaker and non-speaker features, where the non-speaker features are composed of 90% same-gender non-speakers and 10% opposite-gender non-speakers.

CHAPTER FIVE

A NEW TWO-STAGE DECISION MAKING ALGORITHM

In a speaker verification system, a decision should be made such that; either the claimant is accepted or the claimant is rejected. Also, in some systems there may be a doubtful region and within this, the claimant may be asked to repeat his/her utterance one more time. The acceptance, rejection or unsure decisions are made according to the predefined threshold(s) of the speaker model. So, determination of threshold(s) is an important process in forming a speaker verification model.

In this chapter, we will propose a two-stage decision making algorithm. First stage is to test the claimant utterance with the "world model" which is an RBF network formed by a training set according to the results obtained in Section 4.2. In the first stage, the claimant is directly rejected if his/her score is below the rejection threshold "ζ_{WL}", or directly accepted if his/her score is greater than the acceptance threshold "ζ_{WH}". Therefore, we say that the claimant with a score below the threshold ζ_{WL} or above the threshold ζ_{WH} does not pass into the second stage. If the claimant's score is between these thresholds, the utterance is fed into the "cohort model" which is the second stage of decision making. Cohort means the group of speakers who are acoustically close to the genuine speaker. The reason behind cohort selection is that a speaker verification system trained with cohort speakers can deal with impostor attacks more precisely. However, the disadvantage of a cohort model is that it may not discriminate well the genuine speaker features from the world set of non-speakers.

The procedure of world model creation, selection of cohort features and training the cohort model is shown in Figure 5.1.

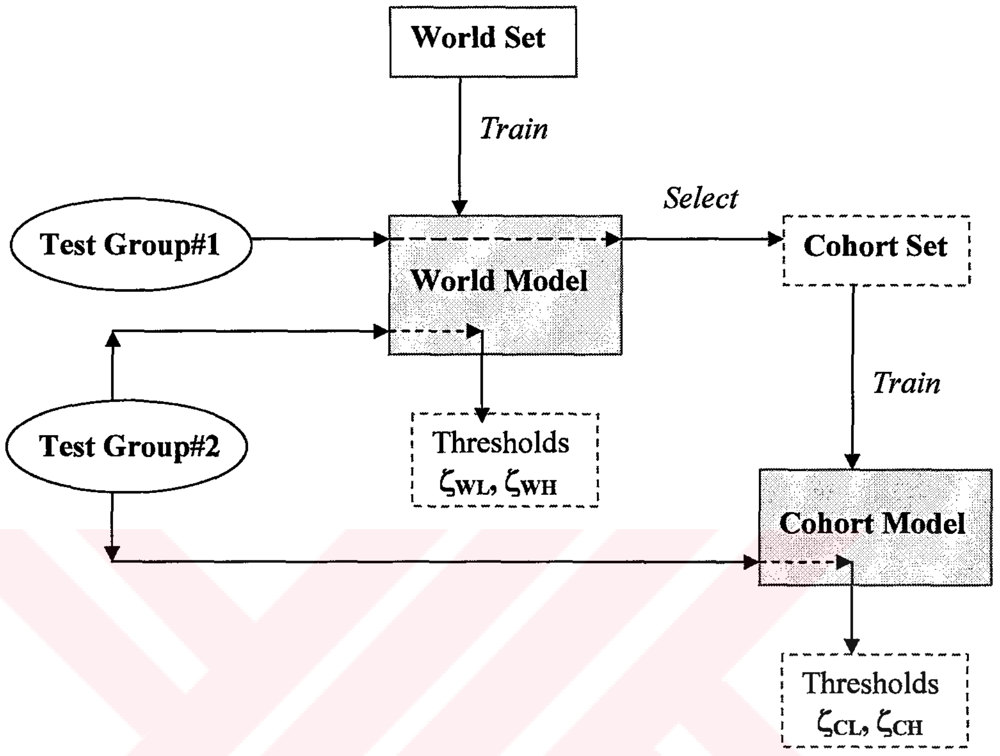

Figure 5.1 Generation of World Model, Cohort Model And Acceptance/Rejection Thresholds

5.1 Determination of Thresholds

Determination of the acceptance/rejection thresholds is an important task for generating an efficient speaker verification system.

In Figure 5.2, the blue curve is Gaussian assumption of impostor test features, while the red curve is Gaussian assumption of genuine speaker test features (Results for the Test Group #2 in Figure 5.1).

We determined the speaker rejection threshold as:

$$\zeta_{WL} = \mu_1 + k_1\sigma_1 \tag{5.1}$$

and the speaker acceptance threshold as:

$$\zeta_{WH} = \mu_2 - k_2\sigma_2 \tag{5.2}$$

Where, μ_1 is the mean of impostor scores from the network, μ_2 is the mean of genuine speaker scores from the network, σ_1 is standard deviation of impostor scores, and σ_2 is standard deviation of genuine speaker scores. The constants k_1 and k_2 are used to shift the thresholds according to the type of the verification system, e.g., if a high- security system is the case, then the value of k_2 should be kept small.

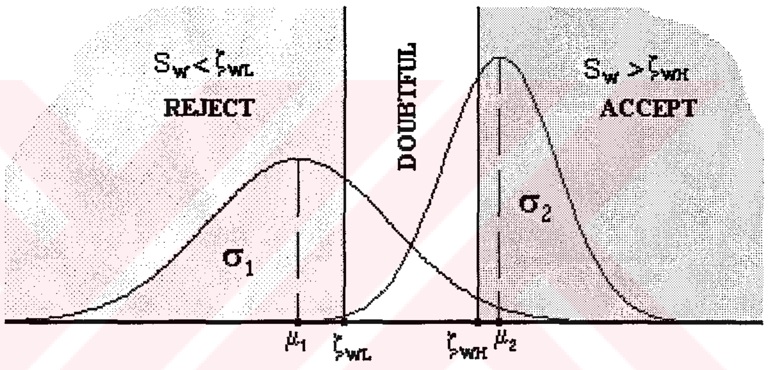

Figure 5.2 Acceptance/Rejection Thresholds for the World Model

5.2 Generation of the Cohort Model

Cohort means the group of speakers who are acoustically close to the genuine speaker. The reason for using a cohort model here is to eliminate the impostors as much as possible when they manage to pass from the first stage (from the world model) without being eliminated.

To generate the cohort model, we used a separate test group of 50 non-speakers (Test Group #1 in Figure 5.1), and we extracted 18 non-speakers who are most close to the genuine speaker. Then, these 18 non-speaker features were used to train the RBF network to form the cohort model. The acceptance/rejection thresholds were determined by using Test Group #2, and with respect to the genuine-speaker features' mean and standard deviation such that:

The speaker rejection threshold of the cohort model:

$$\zeta_{CL} = \mu_4 - k_3\sigma_4 \tag{5.3}$$

The speaker acceptance threshold of the cohort model:

$$\zeta_{CH} = \mu_4 - k_4\sigma_4 \tag{5.4}$$

Where, μ_4 is the mean of genuine speaker features, σ_4 is the standard deviation of genuine speaker features when tested with the cohort model, k_3 and k_4 are constants such that $k_3 > k_4$ (see Figure 5.3).

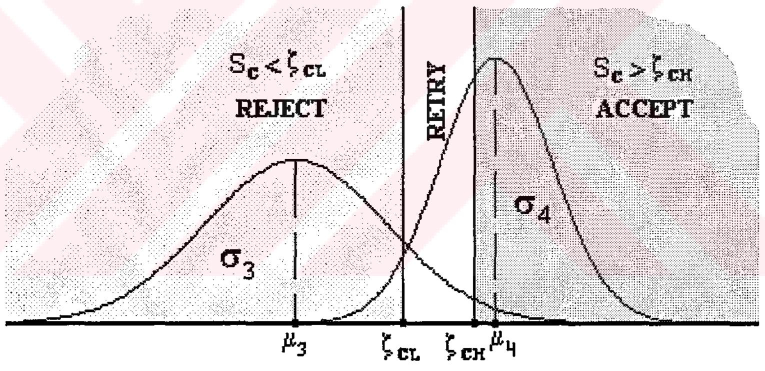

Figure 5.3 Acceptance/Rejection Thresholds for the Cohort Model

It can be seen from Equations (5.3) and (5.4) that, both the acceptance and rejection thresholds were arranged according to the genuine speaker's statistics. This is for controlling the width of the unsure area, where claimants are asked to re-utter to the verification system. In this way, the number of false acceptance and false rejections can be reduced at the expense of more some time loss.

5.3 The Proposed Speaker Verification Algorithm

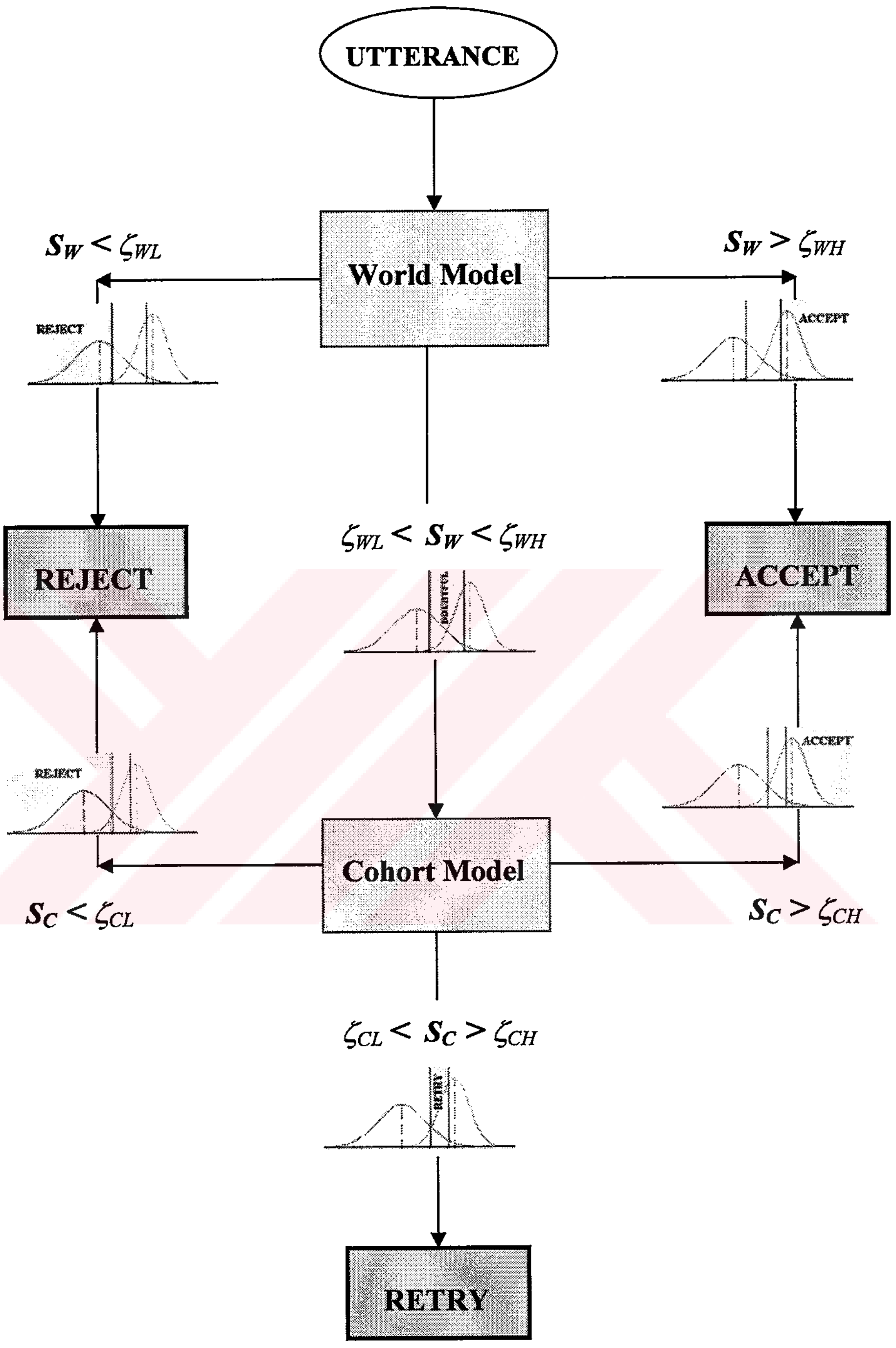

Figure 5.4 The Proposed Speaker Verification Algorithm

The algorithm of our proposed verification system is represented in block diagram in Figure 5.4. The output of the system, as seen in the figure, may be one of three possibilities: Accept, Reject or Retry. If a genuine speaker is rejected, this will be a false rejection (FR) error. If an impostor is accepted, this will be a false acceptance (FA) error.

Test results for the proposed decision-making system is given in Section 5.4.

5.4 Test Results for the Proposed System

After the world model features and the test group features #1 and #2 (see Figure 5.1) were collected as database, generation of the proposed verification system for a speaker, including the training phase of the world and cohort models, lasts for about 30 minutes. We selected the threshold determination constants k_1, k_2, k_3, *and* k_4 (see Equations 5.1 to 5.4) as follows:

$k_1 = 1$,

$k_2 = 0.2$,

$k_3 = 0.5$,

$k_4 = 0.2$.

These values were selected according to the best test results that keep the false acceptance error at a considerably low value.

Tests were done with 288 non-speaker utterances and 26 true speaker utterances. The results for the proposed two-stage automatic speaker verification system are given in Table 5.1.

Table 5.1 Test Results for the Proposed Verification System

Speaker Model	% False Acceptance Error	% Impostor Retried	% False Rejection Error	% Speaker Retried
wsc	0.0	7.2	7.7	15.4
wkt	2.1	9.3	3.8	11.5
mmm	0.0	10.4	7.7	11.5
cmf	1.4	8.0	11.5	7.7
wlh	0.0	3.1	0.0	3.8
wer	1.7	7.2	3.8	7.7
mpm	0.0	2.8	3.8	11.5
cmc	0.0	3.5	7.7	7.7
Average	**0.65**	**6.44**	**5.75**	**9.6**

As seen in Table 5.1, for most of the speakers the false acceptance error is zero, and the average false acceptance error is 0.65%, while the false rejection error and the percentage of retried speakers are kept at a non-disturbing value.

CHAPTER SIX
CONCLUSIONS

6.1 Discussion

The recent researches on automatic speaker verification (ASV) studies are focused mainly on feature selection and pattern matching problems. For the experiments of this thesis mel-frequency cepstral coefficients (MFCC) are selected as features, and radial basis function (RBF) neural networks are selected as the pattern-matching model, which are very popular and are proven to give high performance in ASV applications.

The experiments in this thesis were first focused on optimising the training set in terms of the speaker genders and the ratios of speakers and non-speakers. These investigations gave the result that; the maximum performance is achieved when the system is trained with equal amount of true speakers and non-speakers, while the 90% of the non-speaker set was composed of same gender speakers as the true speaker (e.g. if the model belongs to a male speaker, then %90 percent of the non-speakers will be composed of male speakers in the training set; if the model belongs to a female speaker, then %90 percent of the non-speakers will be composed of female speakers in the training set).

In the last part of the thesis a two-stage decision making algorithm which involves determination of accept/reject thresholds and generation of cohort set (see Figure 5.4) is introduced. The aim of this two-stage decision making algorithm is to eliminate the impostors in the second stage by means of the cohort model. The final output from the system is either accept, or reject, or retry. The proposed system indeed decreased the average false acceptance error to 0.65 percent while the false rejection error average is 5.75 percent.

6.2 Future Works

In this thesis, we focused on optimizing the training set and generating an efficient decision making algorithm. The first future work would be utilization of the proposed decision-making model as a real system and secondly would be focusing on modulating the features (we used only 11 MFCC features in this thesis), such as adding delta and delta-delta MFCC features, or linear prediction cepstral coefficients (LPCC's). Moreover, a vowel-locating algorithm can be used to implement a text-dependent speaker verification system.

REFERENCES

Altun, A., & Kocer, E. (2003). Guvenlik alanina yeni bir yaklasim: biyometrik sistemler. International XII. Turkish symposium on artificial intelligence and neural networks – TAINN 2003.

Bhattacharyya, S., Srikanthan, T., & Krishnamurthy, P. (2001). Ideal GMM parameters and posterior log likelihood in speaker verification. IEEE Signal Processing Magazine January 2001, pp. 471-480.

Békésy, G.V. (1953). Shearing microphones produced by vibrations near the inner and outer hair cells. J. Acoust. Soc. Am. pp. 786-790.

Békésy, G.V. (1960). Experiments in hearing. New York: Mc Graw Hill.

Beranek, L.L. (1986). Acoustics. New York: American Institute of Physics.

Bolt, R.H. et al. (1969). Identification of a speaker by speech spectrograms. Science, vol. 166.

Chen, K. (2002). Personalize mobile access by speaker authentication. School of Computer Science, The University of Birmingham.

Chen, S., Covan, C., & Grant, P. (1991). Orthogonal least squares learning algorithm for radial basis function networks. IEEE Trans. On Neural Networks, Vol. 2, pp. 302-309.

Chen, T. (1998). The past, present, and future of speech processing. IEEE Signal Processing Magazine May 1998, pp. 24-43.

Cohen, A., Zigel, Y. (2003). Feature selection in speaker verification systems. Electrical and Computer Eng. Dept., Ben-Gurion University, Beer-Sheva, Israel.

Doddington, G. (1976). Personal identity verification using voice. Proc. Electro-76, pp. 22-24.

Ehab, F.M., Badran, F., & Selim, H. (2000). Speaker recognition using artificial neural networks based on vowel phonemes. Proceedings of ICSP2000, pp. 796-802.

Helmholtz, H.V. (1954). On the sensations of tone. New York: Dover Publications.

Higgins, A. Bahler, L., & Porter, J. (1991). Speaker verification using randomized phrase prompting. Digital Signal Processing, vol. 1, no. 2, pp. 89-106.

Kamm, T., Hermansky, H., & Andreou, A.G. (1998). Learning the mel-scale and optimal VTN mapping.

Kersta, L.G. (1962). Voiceprint identification. Nature, vol. 196, no. 4861, pp. 1253-1257.

Lucey, S. (2002). Speaker verification – Tutorial for Sony. Advanced Multimedia Processing Labs, Carnegie Mellon University.

Luo, F.L., & Unbehaugen, R. (1997). Applied neural networks for signal processing. Cambridge University Press.

Mammone, R. Zhang, X., & Ramachandran, R. (1996). Robust speaker recognition, A feature based approach. IEEE Signal Processing Mag., vol. 30, no. 5, pp. 58-71.

Markowitz, J. (2002). Speaker recognition yesterday, today and tomorrow. Evanston Research Park, Northwestern University, Illinois, USA.

Matsui, T., & Furui, S. (1992). Comparison of txet independent speaker recognition methods using VQ distortion and discrete/continuous HMMs. Proc. IEEE Int. Acoust. Speech, Signal Proc., San Fransisco, pp. II-157-160.

Minh, N.D. (2000). Digital signal processing mini-project – An automatic speaker recognition system. Audio Visual Communications Laboratory, Swiss Federal Institute of Technology, Lousanne, Switzerland.

Moody, J.E., & Darken, C.J. (1989). First learning in networks of locally tuned processing units. Neural Computation, Vol. 1, pp. 281-294.

Morgan, D.P., & Scofield, C.L. (1991). Neural Networks and Speech Processing. Kluwer Academic Publishment, U.S.

Naci, U.S. (2003). Self score normalization and frame pruning techniques for speaker verification systems. Dept. of Electrical and Electronics Engineering, Bogazici University, Turkey.

Orman, O.D. (2000). Frequency analysis of speaker identification performance. Dept. of Electrical and Electronics Engineering, Bogazici University, Turkey.

Premakanthan, P., Mikhael, W.B. (2001). Speaker verification / recogni-tion and the importance of selective feature extraction: review. Dept. of Electrical Engineering, University of Central Florida, Orlando.

Rabiner, L. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. Proc. IEEE, vol. 77, pp. 257-286.

Rabiner, L., & Juang, B.H. (1993). Fundamentals of speech recognition. Prentice Hall.

Rabiner, L., & Schafer, R.W. (1978). Digital processing of speech signals. Prentice Hall.

Rodman, R. (1998). Speaker recognition of disguised voiced: A program for research. Deptartment of Computer Science, North Carolina State University, North Carolina.

Song, F.K., & Rosenberg, A.E. (1988). On the Use of Instantaneous and Transitional Spectral Information in Speaker Recognition. IEEE Trans. Acoust. Speech, Signal Processing, Vol. ASSP-36, pp. 871-879.

Stevens, K.N., Wiliams, C. E., Carbonelli, J.R., & Woods, B. (1968). Speaker authentication and identification: A comparison of spectrographic and auditory presentations of speech material. Journal of the Acoustical Society of America, vol. 43, pp. 1596-1607.

Tosi, O., Over, H., Lashbrook, W., Pedrev, C., Nicol, J., & Nash, E. (1972). Experiment on voice identification. Journal of the Acoustical Society of America, vol. 51, pp. 2030-2043.

Wolf, J.J. (1972). Efficient acoustic parameters for speaker recognition. JASA, vol. 51, no. 6 (part 2), pp. 2044-2056.

Woo, S.C. (2000). Speaker recognition using ANN. Delft University of Technology, Holland.

Xafopoulos, A. (2001). Speaker verification (an overview). TICSP (Tampere International Center for Signal Processing), Tampere University of Technology, Finland.

Xin, D., Wu, Z., & Zhang, W. (2001). Support vector domain description for speaker recognition. IEEE Signal Processing Magazine August 2001, pp. 481-488.

WEB_1. (2002). The IViE Corpus. http://www.phon.ox.ac.uk/~esther/ivyweb/, (24.07.2002).

WEB_2. (2000). Brooks, M. Voicebox, The Speech Processing Toolbox for Matlab. http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html, (31.08.2002).

APPENDIX A

LIST OF ABBREVIATIONS

ASR.	Automatic Speaker Recognition
ASV.	Automatic Speaker Verification
DET.	Detection Error Tradeoff
EER.	Equal Error Rate
FAR.	False Acceptance Rate
FIR.	Finite Impulse Response
FRR.	False Rejection Rate
GMM.	Gaussian Mixture Model
HMM.	Hidden Markov Model
LogCC.	Log-Cepstrum Coefficients
LPC.	Linear Prediction Coefficients
LPCC.	Linear Prediction Cepstral Coefficients
MFCC.	Mel-Frequency Cepstral Coefficients
MLP.	Multilayer Perceptron
PCA.	Principal Component Analysis
RBF.	Radial Basis Function
ROC.	Receiver Operating Characteristic
VQ.	Vector Quantization

APPENDIX B

MATLAB CODES USED IN THE THESIS

B.1 Code For MFCC Features Extraction

The function "MFCC" and its subfunctions, which do the pre-processing, are created for extraction of MFCC features from a digitized speech signal. The main function "MFCC" is inspired from the "Voicebox" Speech Processing Toolbox for Matlab (WEB_2, June 2000).

```
function c=mfcc(s,fs,w,nc,p,n,inc,fl,fh)

%MELCEPST Calculate the mel cepstrum of a signal c.
%
% Inputs:
%    s:  speech signal
%    fs:  sample rate in Hz (default 11025)
%    nc:  number of cepstral coefficients excluding 0'th coefficient (default 12)
%    n:   length of frame (default power of 2 <30 ms))
%    p:   number of filters in filterbank (default floor(3*log(fs)) )
%    inc: frame increment (default n/2)
%    fl:  low end of the lowest filter as a fraction of fs (default = 0)
%    fh:  high end of highest filter as a fraction of fs (default = 0.5)
%
%                w   any sensible combination of the following:
%
%                       'R'  rectangular window in time domain
%                       'N'      Hanning window in time domain
%                       'M'      Hamming window in time domain (default)
%
%                   't'  triangular shaped filters in mel domain (default)
%                   'n'  hanning shaped filters in mel domain
%                  'm'  hamming shaped filters in mel domain
%
%                      'p'     filters act in the power domain
%                      'a'     filters act in the absolute magnitude domain (default)
%
%                      '0'  include 0'th order cepstral coefficient
%                      'e'  include log energy
%                      'd'     include delta coefficients (dc/dt)
%                      'D'     include delta-delta coefficients (d^2c/dt^2)
%
%                  'z'  highest and lowest filters taper down to zero (default)
%                  'y'  lowest filter remains at 1 down to 0 frequency and
%                               highest filter remains at 1 up to nyquist frequency
%
%                   If 'ty' or 'ny' is specified, the total power in the fft is preserved.
%
% Outputs:
%   c:    mel cepstrum output (one frame per row).


% ----Preemphasis to boost high frequency components-----
% ---- y(n)= x(n)-0.95x(n-1)

s=filter([1 -0.95],1,s);
s=s/max(abs(s));
s=s-mean(s);
% ---------------------------------------------------

if nargin<2 fs=11025; end
if nargin<3 w='M'; end
if nargin<4 nc=12; end
if nargin<5 p=floor(3*log(fs)); end
if nargin<6 n=pow2(floor(log2(0.03*fs))); end
if nargin<9
  fh=0.5;
  if nargin<8
    fl=0;
    if nargin<7
      inc=floor(n/2);
    end
  end
end

if any(w=='R')
  z=enframe(s,n,inc);
elseif any (w=='N')
  z=enframe(s,hanning(n),inc);
```

```
else
  z=enframe(s,hamming(n),inc);
end

f=rfft(z.');
[m,a,b]=melbankm(p,n,fs,fl,fh,w);
pw=f(a:b,:).*conj(f(a:b,:));
pth=max(pw(:))*1E-6;
if any(w=='p')
  y=log(max(m*pw,pth));
else
  ath=sqrt(pth);
  y=log(max(m*abs(f(a:b,:)),ath));
end
c=rdct(y).';
nf=size(c,1);
nc=nc+1;
if p>nc
  c(:,nc+1:end)=[];
elseif p<nc
  c=[c zeros(nf,nc-p)];
end
if ~any(w=='0')
  c(:,1)=[];
end
if any(w=='e')
  c=[log(sum(pw)).' c];
end

if any(w=='D')
  vf=(4:-1:-4)/60;
  af=(1:-1:-1)/2;
  ww=ones(5,1);
  cx=[c(ww,:); c; c(nf*ww,:)];
  vx=reshape(filter(vf,1,cx(:)),nf+10,nc);
  vx(1:8,:)=[];
  ax=reshape(filter(af,1,vx(:)),nf+2,nc);
  ax(1:2,:)=[];
  vx([1 nf+2],:)=[];
  if any(w=='d')
    c=[c vx ax];
  else
    c=[c ax];
  end
elseif any(w=='d')
  vf=(4:-1:-4)/60;
  ww=ones(4,1);
  cx=[c(ww,:); c; c(nf*ww,:)];
  vx=reshape(filter(vf,1,cx(:)),nf+8,nc);
  vx(1:8,:)=[];
  c=[c vx];
end

if nargout<1
   [nf,nc]=size(c);
   t=((0:nf-1)*inc+(n-1)/2)/fs;
   ci=(1:nc)-any(w=='0')-any(w=='e');
   imh = imagesc(t,ci,c.');
   axis('xy');
   xlabel('Time (s)');
   ylabel('Mel-cepstrum coefficient');
         map = (0:63)'/63;
         colormap([map map map]);
         colorbar;
end

% ------------------------------------------------------

function f=enframe(x,win,inc)

nx=length(x);
nwin=length(win);
if (nwin == 1)
   len = win;
else
   len = nwin;
end
if (nargin < 3)
   inc = len;
end
nf = fix((nx-len+inc)/inc);
f=zeros(nf,len);
indf= inc*(0:(nf-1)).';
inds = (1:len);
f(:) = x(indf(:,ones(1,len))+inds(ones(nf,1),:));
f=slcremove(f);
if (nwin > 1)
    w = win(:)';
    nf = size(f,1);
    f = f .* w(ones(nf,1),:);
end
return

% ------------------------------------------------------
```

```
function y=rfft(x,n)

if nargin < 2
  y=fft(x);
else
  y=fft(x,n);
end
if size(y,1)==1
  m=length(y);
  y(floor((m+4)/2):m)=[];
else
  m=size(y,1);
  y(floor((m+4)/2):m,:)=[];
end
return

% -----------------------------------------------------------

function y=rdct(x,n)

fl=size(x,1)==1;
if fl x=x(:); end
[m,k]=size(x);
if nargin<2 n=m;
elseif n>m x=[x; zeros(n-m,k)];
elseif n<m x(n+1:m,:)=[];
end

x=[x(1:2:n,:); x(2*fix(n/2):-2:2,:)];
z=[sqrt(2) 2*exp((-0.5i*pi/n)*(1:n-1))].';
y=real(fft(x).*z(:,ones(1,k)));

if fl y=y.'; end
return

% -----------------------------------------------------------

function [x,mn,mx]=melbankm(p,n,fs,fl,fh,w)

if nargin < 6
  w='tz';
  if nargin < 5
    fh=0.5;
    if nargin < 4
      fl=0;
    end
  end
end
```

```
f0=700/fs;
fn2=floor(n/2);
lr=log((f0+fh)/(f0+fl))/(p+1);
% convert to fft bin numbers with 0 for DC term
bl=n*((f0+fl)*exp([0 1 p p+1]*lr)-f0);
b2=ceil(bl(2));
b3=floor(bl(3));

if any(w=='y')
  pf=log((f0+(b2:b3)/n)/(f0+fl))/lr;
  fp=floor(pf);
  r=[ones(1,b2) fp fp+1 p*ones(1,fn2-b3)];
  c=[1:b3+1 b2+1:fn2+1];
  v=2*[0.5 ones(1,b2-1) 1-pf+fp pf-fp ones(1,fn2-b3-1) 0.5];
  mn=1;
  mx=fn2+1;
else
  b1=floor(bl(1))+1;
  b4=min(fn2,ceil(bl(4)))-1;
  pf=log((f0+(b1:b4)/n)/(f0+fl))/lr;
  fp=floor(pf);
  pm=pf-fp;
  k2=b2-b1+1;
  k3=b3-b1+1;
  k4=b4-b1+1;
  r=[fp(k2:k4) 1+fp(1:k3)];
  c=[k2:k4 1:k3];
  v=2*[1-pm(k2:k4) pm(1:k3)];
  mn=b1+1;
  mx=b4+1;
end

if any(w=='n')
  v=1-cos(v*pi/2);
elseif any(w=='m')
  v=1-0.92/1.08*cos(v*pi/2);
end

if nargout > 1
  x=sparse(r,c,v);
else
  x=sparse(r,c+mn-1,v,p,1+fn2);
end
return

% ------------------------------------------------------
```

```
function [fr,thr,e]=slcremove(frames)

[nf,len]=size(frames);
emax=0;
emin=0;
e=sum(frames.^2,2);
emax=max(e);
emin=min(e);
I1=0.03*(emax-emin)+emin;
I2=4*emin;
thr=25*min(I1,I2);
k=0;
for i=1:nf
   if e(i)>thr
      k=k+1;
      fr(k,:)=frames(i,:);
   end
end
return
```

B.2 Code For World Model Training

This code is for training the RBF Network to form a world model for the speaker "wsc". The feature matrice "fem90perc" is the non-speaker set, which is composed of 90% female speakers and 10% male speakers, containing 540 feature vectors in total.

```
% input of the RBF Network
P = [c1wsc(31:60,:)
   i1wsc(31:60,:)
   q1wsc(31:60,:)
   s1wsc(31:60,:)
   w1wsc(31:60,:)
   p1wsc1(501:600,:)
   p2wsc2(501:600,:)
   r2wsc2(501:600,:)
   r6wsc2(501:590,:)
   fem90perc.'].';

% target of the RBF Network
T=[ones(1,540) zeros(1,540) ; zeros(1,540) ones(1,540)];

net=newrb(P,T,2,4,850,50);
```

```
% NEWRB(P,T,GOAL,SPREAD,MN,DF) takes these arguments,
%     P      - RxQ matrix of Q input vectors.
%     T      - SxQ matrix of Q target class vectors.
%     GOAL   - Mean squared error goal, default = 0.0.
%     SPREAD - Spread of radial basis functions, default = 1.0.
%     MN     - Maximum number of neurons, default is Q.
%     DF     - Number of neurons to add between displays, default = 25.
%   and returns a new radial basis network.
```

B.3 Code For Determination of the Cohort Selection Threshold

This code and its sub-functions are created for determination of the cohort selection constant for the speaker “wsc”.

```
% speaker scores
perf_wsc=[perform(c4wsc,net) perform(c3wsc,net) perform(c5wsc,net)
        perform(i2wsc,net) perform(p1wsc3,net) perform(s2wsc,net)
        perform(s3wsc,net) perform(s4wsc,net) perform(s6wsc,net)
        perform(s7wsc,net)perform(w3wsc,net) perform(p1wsc4,net)
        perform(p1wsc5,net) perform(r1wsc1,net) perform(r2wsc1,net)
        perform(r2wsc3,net) perform(r3wsc1,net) perform(r3wsc2,net)
        perform(r3wsc3,net) perform(r4wsc2,net) perform(r5wsc1,net)
        perform(r2wsc5,net) perform(r2wsc6,net) perform(p2wsc3,net)
        perform(p2wsc4,net) perform(r6wsc3,net)];

% non-speaker scores
impf1=[perform9(c_w1bdh,c_w1bec,c_w1bjx,c_w1brc,c_w1clh,c_w1clp,c_w1wer,
        c_w1csm,c_w1jkj,net)]

impf2=[perform9(c_w1jmd,c_w1jte,c_w1lml,c_w1lnb,c_w1lrl,c_w1mec,c_w1mkg,
        c_w1mkm,c_w1nlc,net)]

impf3=[perform9(c_w1nrp,c_w1nsb,c_w1nvw,c_w1pki,c_w1pra,c_w1prr,c_w1psm,
        c_w1psn,c_w1slm,net)]

impf4=[perform9(c_w1slp,c_w1snt,c_w1srb,c_w1str,c_w1wkt,c_w1wkv,c_w1wnc,
        c_w1cmf,c_w1bdh,net)]

impm1=[perform9(c_w1bjm,c_w1brg,c_w1bro,c_w1bts,c_w1cma,c_w1cmc,
        c_w1cpt,c_c3ctg,c_w1jmo,net)]

impm2=[perform9(c_w1jra,c_w1jws,c_w1mdk,c_w1mdo,c_w1mmp,c_w1mpm,
        c_w1nmc,c_w1nml,c_w1nrf,net)]
```

```
impm3=[perform9(c_w1nse,c_w1pjk,c_w1prh,c_w1pta,c_w1pwa,c_w1pzm,
        c_w1sph,c_w1spx,c_w1ssb,net)]

impm4=[perform9(c_w1mmm,c_w1wlh,c_w1wmk,c_w1wxt,c_w1lmd,c_w1lrp,
        c_w1lst,c_w1lsu,c_w1bjm,net)]

impf5=[perform9(c_w1bdh,c_i1bec,c_i1bjx,c_i1brc,c_i1clh,c_i1clp,c_i1wer,
        c_i1csm,c_i1jkj,net)]

impf6=[perform9(c_i1jmd,c_i1jte,c_i1lml,c_i1lnb,c_i1lrl,c_i1mec,c_i1mkg,
        c_i1mkm,c_i1nlc,net)]

impf7=[perform9(c_i1nrp,c_i1nsb,c_i1nvw,c_i1pki,c_i1pra,c_i1prr,c_i1psm,
        c_i1psn,c_i1slm,net)]

impf8=[perform9(c_i1slp,c_i1snt,c_i1srb,c_i1str,c_i1wkt,c_i1wkv,c_i1wnc,c_i1cmf,
        c_i1bdh,net)]

impm5=[perform9(c_i1bjm,c_i1brg,c_i1bro,c_i1bts,c_i1cma,c_i1cmc,c_i1cpt,
        c_c3ctg,c_i1jmo,net)]

impm6=[perform9(c_i1jra,c_i1jws,c_i1mdk,c_i1mdo,c_i1mmp,c_i1mpm,c_i1nmc,
        c_i1nml,c_i1nrf,net)]

impm7=[perform9(c_i1nse,c_i1pjk,c_i1prh,c_i1pta,c_i1pwa,c_i1pzm,c_i1sph,
        c_i1spx,c_i1ssb,net)]

impm8=[perform9(c_i1mmm,c_i1wlh,c_i1wmk,c_i1wxt,c_i1lmd,c_i1lrp,c_i1lst,
        c_i1lsu,c_i1bjm,net)]

impf9=[perform9(c_q1bdh,c_q1bec,c_q1bjx,c_q1brc,c_q1clh,c_q1clp,c_q1wer,
        c_q1csm,c_q1jkj,net)]

impf10=[perform9(c_q1jmd,c_q1jte,c_q1lml,c_q1lnb,c_q1lrl,c_q1mec,c_q1mkg,
        c_q1mkm,c_q1nlc,net)]

impf11=[perform9(c_q1nrp,c_q1nsb,c_q1nvw,c_q1pki,c_q1pra,c_q1prr,c_q1psm,
        c_q1psn,c_q1slm,net)]

impf12=[perform9(c_q1slp,c_q1snt,c_q1srb,c_q1str,c_q2wkt,c_q1wkv,c_q1wnc,
        c_q1cmf,c_q1bdh,net)]

impm9=[perform9(c_q1bjm,c_q1brg,c_q1bro,c_q1bts,c_q1cma,c_q1cmc,c_q1cpt,
        c_c3ctg,c_q1jmo,net)]

impm10=[perform9(c_q1jra,c_q1jws,c_q1mdk,c_q1mdo,c_q1mmp,c_q1mpm,
        c_q1nmc,c_q1nml,c_q1nrf,net)]
```

```
impm11=[perform9(c_q1nse,c_q1pjk,c_q1prh,c_q1pta,c_q1pwa,c_q1pzm,c_q1sph,
        c_q1spx,c_q1ssb,net)]

impm12=[perform9(c_q1mmm,c_q1wlh,c_q1wmk,c_q1wxt,c_q1lmd,c_q1lrp,
        c_q1lst,c_q1lsu,c_q1bjm,net)]

impf13=[perform9(c_s1bdh,c_s1bec,c_s1bjx,c_s1brc,c_s1clh,c_s1clp,c_s1wer,
        c_s1csm,c_s1jkj,net)]

impf14=[perform9(c_s1jmd,c_s1jte,c_s1lml,c_s1lnb,c_s1lrl,c_s1mec,c_s1mkg,
        c_s1mkm,c_s1nlc,net)]

impf15=[perform9(c_s1nrp,c_s1nsb,c_s1nvw,c_s1pki,c_s1pra,c_s1prr,c_s1psm,
        c_s1psn,c_s1slm,net)]

impf16=[perform9(c_s1slp,c_s1snt,c_s1srb,c_s1str,c_s2wkt,c_s1wkv,c_s1wnc,
        c_s1cmf,c_s1bdh,net)]

impm13=[perform9(c_s1bjm,c_s1brg,c_s1bro,c_s1bts,c_s1cma,c_s1cmc,c_s1cpt,
        c_c3ctg,c_s1jmo,net)]

impm14=[perform9(c_s1jra,c_s1jws,c_s1mdk,c_s1mdo,c_s1mmp,c_s1mpm,
        c_s1nmc,c_s1nml,c_s1nrf,net)]

impm15=[perform9(c_s1nse,c_s1pjk,c_s1prh,c_s1pta,c_s1pwa,c_s1pzm,c_s1sph,
        c_s1spx,c_s1ssb,net)]

impm16=[perform9(c_s1mmm,c_s1wlh,c_s1wmk,c_s1wxt,c_s1lmd,c_s1lrp,c_s1lst,
        c_s1lsu,c_s1bjm,net)]

perf_imp=[impf1 impf2 impf3 impf4 impf5 impf6 impf7 impf8 impf9 impf10
        impf11 impf12 impf13 impf14 impf15 impf16 impm1 impm2 impm3
        impm4 impm5 impm6 impm7 impm8 impm9 impm10 impm11 impm12
        impm13 impm14 impm15 impm16];

speaker_mean=mean(perf_wsc)
speaker_stdev=std(perf_wsc)

imp_mean=mean(perf_imp)
imp_stdev=std(perf_imp)
c_coh=imp_mean+1*imp_stdev

far=[];
frr=[];

for t=-1:0.01:1
  e1=sum(perf_imp>t)/length(perf_imp);
```

```
   far=[far e1];
   e2=sum(perf_wsc<t)/length(perf_wsc);
   frr=[frr e2];
end

plot(far,frr)

cfar=1.2*far;
cfrr=0.8*frr;

eer=0;

dif=frr-far;
minn=min(abs(dif));

for k=1:length(far)
   if abs(dif(k))==minn
      eer=(far(k)+frr(k))/2;
   end
end
eer=eer*100

w=-1:0.01:1;
figure
plot(w,100*frr)
hold on
plot(w,100*far)

% -----------------------------------------------------


function perf = perform(x,net);
y = sim(net,x.');
f_s=mean(y(1,:));
f_w=mean(y(2,:));
perf=f_s-f_w;


% -----------------------------------------------------


function prf9 = perform9(x1,x2,x3,x4,x5,x6,x7,x8,x9,net);

pr1=perform(x1,net);
pr2=perform(x2,net);
pr3=perform(x3,net);
pr4=perform(x4,net);
pr5=perform(x5,net);
pr6=perform(x6,net);
```

```
pr7=perform(x7,net);
pr8=perform(x8,net);
pr9=perform(x9,net);

prf9=[pr1 pr2 pr3 pr4 pr5 pr6 pr7 pr8 pr9];
```

B.4 Code for Selection of Cohort Set Speakers & Determination of the Speaker Acceptance/Rejection Thresholds

This code first selects the non-speakers who are most close to the genuine speaker's vocal properties, and forms the cohort set for the genuine speaker "wsc". Then, the speaker acceptance/rejection thresholds of the world model are determined for the speaker "wsc".

```
% ----------- Determine speaker acception threshold
% ----------- Determine the cohort set
% ----------- Determine speaker rejection threshold

coh=[];

a1=[perform(p1wsc2,net) perform(p2wsc1,net) perform(q3wsc,net)
        perform(r5wsc2,net) perform(r6wsc1,net)]

a2=[perform(c2wsc,net) perform(q2wsc,net) perform(s8wsc,net)
        perform(i3wsc,net) perform(w2wsc,net) perform(s5wsc,net)]

a3=[perform(r1wsc2,net) perform(r2wsc4,net) perform(r4wsc1,net)]

mean_sp=mean([a1 a2 a3]);
std_sp=std([a1 a2 a3]);

  k1=0.6;  %speaker acception constant
thr_high = mean_sp-k1*std_sp    % speaker acception threshold

c_coh=mean_sp-3*std_sp     % cohort selection constant

[f1,coh]=per9(c_c1bdh,c_i1bec,c_q1bjx,c_s1brc,c_w1clh,c_c1clp,c_i1cmf,c_q1csm,
        c_s1jkj,net,coh,c_coh); f1

[f2,coh]=per9(c_c1jmd,c_w1jte,c_i1lml,c_q1lnb,c_s1lrl,c_w1mec,c_c1mkg,
        c_i1mkm,c_q1nlc,net,coh,c_coh); f2
```

```
[f3,coh]=per9(c_c1nrp,c_s1nsb,c_w1nvw,c_i1pki,c_s1pra,c_q1prr,c_w1psm,
        c_c1psn,c_i1slm,net,coh,c_coh); f3

[f4,coh]=per9(c_q1slp,c_s1snt,c_w1srb,c_c1str,c_i1wkt,c_s1wkv,c_q1wnc,
        c_w1wlh,c_c1bec,net,coh,c_coh); f4

[m1,coh]=per9(c_i1bjm,c_q1brg,c_s1bro,c_w1bts,c_c1cma,c_i1cmc,c_q1cpt,
        c_s1ctg,c_w1jmo,net,coh,c_coh); m1

[m2,coh]=per9(c_c1jra,c_i1jws,c_q1mdk,c_s1mdo,c_w1mmp,c_c1mpm,c_i1nmc,
        c_q1nml,c_s1nrf,net,coh,c_coh); m2

[m3,coh]=per9(c_c1nse,c_w1pjk,c_i1prh,c_q1pta,c_s1pwa,c_w1pzm,c_c1sph,
        c_i1spx,c_q1ssb,net,coh,c_coh); m3

[m4,coh]=per9(c_s1mmm,c_w1wlh,c_q1wmk,c_i1wxt,c_c1lmd,c_s1lrp,c_w1lst,
        c_i1lsu,c_c1bjm,net,coh,c_coh); m4

mean_imp=mean([f1 f2 f3 f4 m1 m2 m3 m4]);
std_imp=std([f1 f2 f3 f4 m1 m2 m3 m4]);

  k2=0.8;    %speaker rejection constant
thr_low = mean_imp+k2*std_imp    % speaker rejection threshold
```

B.5 Code For Cohort Model Training

This code is for training the RBF Network of the cohort model of the speaker "wsc". The feature matrice "coh" is the cohort set which is formed by the previous m-file given above.

```
% input of the Cohort Network
P = [c1wsc(31:60,:)
   i1wsc(31:60,:)
   q1wsc(31:60,:)
   s1wsc(31:60,:)
   w1wsc(31:60,:)
   p1wsc1(531:585,:)
   p2wsc2(531:585,:)
   r2wsc2(531:585,:)
   r6wsc2(531:575,:)
   coh].';
```

```
% target of the Cohort  Network
T=[ones(1,360) zeros(1,360) ; zeros(1,360) ones(1,360)];

netc=newrb(P,T,1,4,650,50);


% NEWRB(P,T,GOAL,SPREAD,MN,DF) takes these arguments,
%     P      - RxQ matrix of Q input vectors.
%     T      - SxQ matrix of Q target class vectors.
%     GOAL   - Mean squared error goal, default = 0.0.
%     SPREAD - Spread of radial basis functions, default = 1.0.
%     MN     - Maximum number of neurons, default is Q.
%     DF     - Number of neurons to add between displays, default = 25.
%   and returns a new radial basis network.
```

B.6 Code For Determination of the Acceptance/Rejection Thresholds of the Cohort Model

By this code, the speaker acceptance/rejection thresholds of the cohort model are determined for the speaker “wsc”.

```
b1=[perform(c1wsc,netc) perform(i1wsc,netc) perform(q1wsc,netc)
        perform(s1wsc,netc) perform(w1wsc,netc) perform(r6wsc2,netc)
        perform(r2wsc2,netc)]

b2=[perform(p1wsc1,netc) perform(p2wsc2,netc) perform(c4wsc,net)
        perform(c3wsc,net) perform(c5wsc,net) perform(i2wsc,net)]

mean_c_s=mean([b1 b2]);
std_c_s=std([b1 b2]);
coh_high = mean_c_s-0.2*std_c_s;    %speaker acception threshold


bi1=[perform9(c_w1mmm,c_w1cmf,c_w1wmk,c_w1wxt,c_w1lmd,c_w1lrp,c_w1lst,
        c_w1lsu,c_w1bjm,netc)]

bi2=[perform9(c_w1bdh,c_i1bec,c_i1bjx,c_i1brc,c_i1clh,c_i1clp,c_i1wer,c_i1csm,
        c_i1jkj,netc)]

bi3=[perform9(c_c1bcc,c_c1bgm,c_c1bdo,c_c1cer,c_c1chb,c_c1cje,c_c1cji,c_c1jah,
        c_c1jcr,netc)]

bi4=[perform9(c_q1bdh,c_q1bec,c_q1bjx,c_q1brc,c_q1clh,c_q1clp,c_q1wer,
        c_q1csm,c_q1jkj,netc)]
```

```
bi5=[perform9(c_s1jra,c_s1jws,c_s1mdk,c_s1mdo,c_s1mmp,c_s1mpm,c_s1nmc,
        c_s1nml,c_s1nrf,netc)]

bi6=[perform9(c_s1nrp,c_s1nsb,c_s1nvw,c_s1pki,c_s1pra,c_s1prr,c_s1psm,c_s1psn,
        c_s1slm,netc)]

bi7=[perform9(c_q1nse,c_q1pjk,c_q1prh,c_q1pta,c_q1pwa,c_q1pzm,c_q1sph,
        c_q1spx,c_q1ssb,netc)]

mean_c_i=mean([bi1 bi2 bi3 bi4 bi5 bi6 bi7]);
std_c_i=std([bi1 bi2 bi3 bi4 bi5 bi6 bi7]);
coh_low = mean_c_i+1*std_c_i;    %speaker reject threshold
```

B.7 Code For Testing the Generated Speaker Verification Model

By this code, the two-stage speaker verification model of the speaker "wsc", which is generated by the above m-files, is tested and the results are supplied in terms of false acceptance rate (FAR), false rejection rate (FRR), retried speakers, and retried impostors.

```
%-----------------TEST--------------------------------------------------------

% speaker scores
perf_wsc=[perform(c4wsc,net) perform(c3wsc,net) perform(c5wsc,net)
        perform(i2wsc,net) perform(p1wsc3,net) perform(s2wsc,net)
        perform(s3wsc,net) perform(s4wsc,net) perform(s6wsc,net)
        perform(s7wsc,net) perform(w3wsc,net) perform(p1wsc4,net)
        perform(p1wsc5,net) perform(r1wsc1,net) perform(r2wsc1,net)
        perform(r2wsc3,net) perform(r3wsc1,net) perform(r3wsc2,net)
        perform(r3wsc3,net) perform(r4wsc2,net) perform(r5wsc1,net)
        perform(r2wsc5,net) perform(r2wsc6,net) perform(p2wsc3,net)
        perform(p2wsc4,net) perform(r6wsc3,net)]


% impostor scores
impf1=[perform9(c_w1bdh,c_w1bec,c_w1bjx,c_w1brc,c_w1clh,c_w1clp,c_w1wer,
        c_w1csm,c_w1jkj,net)]

impf2=[perform9(c_w1jmd,c_w1jte,c_w1lml,c_w1lnb,c_w1lrl,c_w1mec,c_w1mkg,
        c_w1mkm,c_w1nlc,net)]
```

```
impf3=[perform9(c_w1nrp,c_w1nsb,c_w1nvw,c_w1pki,c_w1pra,c_w1prr,c_w1psm,
        c_w1psn,c_w1slm,net)]

impf4=[perform9(c_w1slp,c_w1snt,c_w1srb,c_w1str,c_w1wkt,c_w1wkv,c_w1wnc,
        c_w1wlh,c_w1bec,net)]

impm1=[perform9(c_w1bjm,c_w1brg,c_w1bro,c_w1bts,c_w1cma,c_w1cmc,
        c_w1cpt,c_c3ctg,c_w1jmo,net)]

impm2=[perform9(c_w1jra,c_w1jws,c_w1mdk,c_w1mdo,c_w1mmp,c_w1mpm,
        c_w1nmc,c_w1nml,c_w1nrf,net)]

impm3=[perform9(c_w1nse,c_w1pjk,c_w1prh,c_w1pta,c_w1pwa,c_w1pzm,
        c_w1sph,c_w1spx,c_w1ssb,net)]

impm4=[perform9(c_w1mmm,c_w1cmf,c_w1wmk,c_w1wxt,c_w1lmd,c_w1lrp,
        c_w1lst,c_w1lsu,c_w1bjm,net)]

impf5=[perform9(c_w1bdh,c_i1bec,c_i1bjx,c_i1brc,c_i1clh,c_i1clp,c_i1wer,
        c_i1csm,c_i1jkj,net)]

impf6=[perform9(c_i1jmd,c_i1jte,c_i1lml,c_i1lnb,c_i1lrl,c_i1mec,c_i1mkg,
        c_i1mkm,c_i1nlc,net)]

impf7=[perform9(c_i1nrp,c_i1nsb,c_i1nvw,c_i1pki,c_i1pra,c_i1prr,c_i1psm,
        c_i1psn,c_i1slm,net)]

impf8=[perform9(c_i1slp,c_i1snt,c_i1srb,c_i1str,c_i1wkt,c_i1wkv,c_i1wnc,c_i1wlh,
        c_i1bdh,net)]

impm5=[perform9(c_i1bjm,c_i1brg,c_i1bro,c_i1bts,c_i1cma,c_i1cmc,c_i1cpt,
        c_c3ctg,c_i1jmo,net)]

impm6=[perform9(c_i1jra,c_i1jws,c_i1mdk,c_i1mdo,c_i1mmp,c_i1mpm,c_i1nmc,
        c_i1nml,c_i1nrf,net)]

impm7=[perform9(c_i1nse,c_i1pjk,c_i1prh,c_i1pta,c_i1pwa,c_i1pzm,c_i1sph,
        c_i1spx,c_i1ssb,net)]

impm8=[perform9(c_i1mmm,c_i1cmf,c_i1wmk,c_i1wxt,c_i1lmd,c_i1lrp,c_i1lst,
        c_i1lsu,c_i1bjm,net)]

impf9=[perform9(c_q1bdh,c_q1bec,c_q1bjx,c_q1brc,c_q1clh,c_q1clp,c_q1wer,
        c_q1csm,c_q1jkj,net)]

impf10=[perform9(c_q1jmd,c_q1jte,c_q1lml,c_q1lnb,c_q1lrl,c_q1mec,c_q1mkg,
        c_q1mkm,c_q1nlc,net)]
```

```
impf11=[perform9(c_q1nrp,c_q1nsb,c_q1nvw,c_q1pki,c_q1pra,c_q1prr,c_q1psm,
        c_q1psn,c_q1slm,net)]

impf12=[perform9(c_q1slp,c_q1snt,c_q1srb,c_q1str,c_q2wkt,c_q1wkv,c_q1wnc,
        c_q1wlh,c_q1bdh,net)]

impm9=[perform9(c_q1bjm,c_q1brg,c_q1bro,c_q1bts,c_q1cma,c_q1cmc,c_q1cpt,
        c_c3ctg,c_q1jmo,net)]

impm10=[perform9(c_q1jra,c_q1jws,c_q1mdk,c_q1mdo,c_q1mmp,c_q1mpm,
        c_q1nmc,c_q1nml,c_q1nrf,net)]

impm11=[perform9(c_q1nse,c_q1pjk,c_q1prh,c_q1pta,c_q1pwa,c_q1pzm,c_q1sph,
        c_q1spx,c_q1ssb,net)]

impm12=[perform9(c_q1mmm,c_q1cmf,c_q1wmk,c_q1wxt,c_q1lmd,c_q1lrp,
        c_q1lst,c_q1lsu,c_q1bjm,net)]

impf13=[perform9(c_s1bdh,c_s1bec,c_s1bjx,c_s1brc,c_s1clh,c_s1clp,c_s1wer,
        c_s1csm,c_s1jkj,net)]

impf14=[perform9(c_s1jmd,c_s1jte,c_s1lml,c_s1lnb,c_s1lrl,c_s1mec,c_s1mkg,
        c_s1mkm,c_s1nlc,net)]

impf15=[perform9(c_s1nrp,c_s1nsb,c_s1nvw,c_s1pki,c_s1pra,c_s1prr,c_s1psm,
        c_s1psn,c_s1slm,net)]

impf16=[perform9(c_s1slp,c_s1snt,c_s1srb,c_s1str,c_s2wkt,c_s1wkv,c_s1wnc,
        c_s1wlh,c_s1bdh,net)]

impm13=[perform9(c_s1bjm,c_s1brg,c_s1bro,c_s1bts,c_s1cma,c_s1cmc,c_s1cpt,
        c_c3ctg,c_s1jmo,net)]

impm14=[perform9(c_s1jra,c_s1jws,c_s1mdk,c_s1mdo,c_s1mmp,c_s1mpm,
        c_s1nmc,c_s1nml,c_s1nrf,net)]

impm15=[perform9(c_s1nse,c_s1pjk,c_s1prh,c_s1pta,c_s1pwa,c_s1pzm,c_s1sph,
        c_s1spx,c_s1ssb,net)]

impm16=[perform9(c_s1mmm,c_s1cmf,c_s1wmk,c_s1wxt,c_s1lmd,c_s1lrp,c_s1lst,
        c_s1lsu,c_s1bjm,net)]


perf_imp=[impf1 impf2 impf3 impf4 impf5 impf6 impf7 impf8 impf9 impf10
        impf11 impf12 impf13 impf14 impf15 impf16 impm1 impm2 impm3 impm4
        impm5 impm6 impm7 impm8 impm9 impm10 impm11 impm12 impm13
        impm14 impm15 impm16];
```

```
% ----------------false rejection----------------------
n1=length(perf_wsc);
sp_retry=0;
sp_doubt=0;

rej1=0;
rej2=0;

for x=1:n1
   if perf_wsc(x)<thr_high & perf_wsc(x)>thr_low
      sp_doubt=sp_doubt+1;
      if perf_wsc(x)<coh_high & perf_wsc(x)>coh_low
         sp_retry=sp_retry+1;
      end
      if perf_wsc(x)<coh_low
         rej2=rej2+1;
      end
   end
      if perf_wsc(x)<thr_low
      rej1=rej1+1;
   end
end



% -------------false acception rate-------------------
n2=length(perf_imp);
imp_retry=0;
imp_doubt=0;
acc1=0;
acc2=0;

for y=1:n2
   if perf_imp(y)<thr_high & perf_imp(y)>thr_low
      imp_doubt=imp_doubt+1;
      if perf_imp(y)<coh_high & perf_imp(y)>coh_low
         imp_retry=imp_retry+1;
      end
      if perf_imp(y)>coh_high
         acc2=acc2+1;
      end
   end
      if perf_imp(y)>thr_high
      acc1=acc1+1;
   end
end
```

```
% FalseAccept = acc1+acc2
% FalseReject = rej1+rej2

acc1=100*acc1/n2
imp_doubt = 100*imp_doubt/n2
acc2=100*acc2/n2
ReImpostor = 100*imp_retry/n2

rej1=100*rej1/n1
sp_doubt = 100*sp_doubt/n1
rej2=100*rej2/n1
ReSpeaker = 100*sp_retry/n1
```