Speaker Recognition

Ling Feng

Kgs. Lyngby 2004
IMM-THESIS-2004-73

Speaker Recognition

Ling Feng

Kgs. Lyngby 2004

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk

IMM-THESIS: ISSN 1601-233X

Preface

The work leading to this Master thesis was carried out in the Intelligent Signal Processing group at Institute of Informatics and Mathematical Modelling, IMM, at Technical University of Denmark, DTU, from 15th of February to 22nd of September 2004. It serves as a requirement for the degree Master of Science in Engineering. The work was supervised by Professor Lars Kai Hansen, and co-supervised by Ph.D student Peter Ahrendt.

Kgs Lyngby, September 22, 2004

Ling Feng s020984

Acknowledgement

The work would not be carried out so smoothly without the help, assistance and support from a number of people. I would like to thank the following people:

Supervisor, Lars Kai Hansen, for always having moment to spare, as well as inspiring and motivating me throughout my project period.

Co-supervisor, Peter Ahrendt, for sharing knowledge without reservation, and patient guidance.

Secretary, Ulla Nørhave, for her great help and support on my speech database recording work. She gathered many staffs and Ph.D students from IMM as my recoding subjects. Her kindness always reminds me my mother.

Master students, Slimane, Rezaul and Mikkel, for discussing variance issues and idea sharing. They made the hard working period interesting and relaxing. Special thank goes to Slimane for his proofreading and inspiration.

All the people who supported my speech database building work, for their kindness, their voice messages and time dedication.

Last but not the least, I wish to thank my Wei for his boundless love, support and caretaking throughout this project. I will never forget those days he took care of me when I was sick and lying in the bed.

Abstract

The work leading to this thesis has been focused on establishing a text-independent closed-set speaker recognition system. Contrary to other recognition systems, this system was built with two parts for the purpose of improving the recognition accuracy. The first part is the speaker pruning performed by KNN algorithm. To decrease the gender misclassification in KNN, a novel technique was used, where Pitch and MFCC features were combined. This technique, in fact, does not only improve the gender misclassification, but also leads to an increase on the total performance of the pruning. The second part is the DDHMM speaker recognition performed on the 'survived' speakers after pruning. By adding the speaker pruning part, the system recognition accuracy was increased 9.3%.

During the project period, an English Language Speech Database for Speaker Recognition (ELSDSR) was built. The system was trained and tested with both TIMIT and ELSDSR database.

Keywords: *feature extraction, MFCC, KNN, speaker pruning, DDHMM, speaker recognition and ELSDSR.*

Nomenclature

Below shows the most used symbols and abbreviations in this thesis.

$\aleph(\cdot)$	Gaussian density
$d_E(\cdot)$	Euclidean distance
$Y(\cdot)$	Output of filter in mel-frequency
$\Omega_i(\cdot)$	Sampled magnitude response of the i^{th} channel filterbank
$\alpha_t(i)$	Forward variables of forward-backward algorithm
$\beta_t(i)$	Backward variables of forward-backward algorithm
a_{ij}	Transition coefficients
b_{jk}	Emission coefficients
A	State transition matrix
B	Emission probability matrix
$\pi=\{\pi_1,\ldots,\pi_N\}$	Initial start distribution vector
$O=o_1,o_{2,\ldots,}o_T$	A observations sequence
$S=\{s_1,\ldots s_N\}$	A set of states
$X(*)$	(Optimal) state sequence
x_t	Random variables for Markov sequence
$a(n;m)$	LP coefficients

pdf	Probability density function
stCC	Short-term Complex Cepstrum
stRC	Short-term Real Cepstrum $c_s(n;m)$
ANN	Artificial Neural Network
AUTOC	Autocorrelation
CDHMM	Continuous-Density HMM
CEP	Cepstrum
CC	Complex Cepstrum
DDHMM	Discrete-Density HMM
DMFCC	Delta Mel-frequency cepstral coefficients
DDMFCC	Delta-Delta Mel-frequency cepstral coefficients
DTFT	Discrete Time Fourier Transform
DTW	Dynamic Time Warping
ELSDSR	English Language Speech Database for Speaker Recognition
EM	Expectation and Maximization algorithm
FIR	Finite Impulse Response
GMM	Gaussian Mixture Model
HMM	Hidden Markov Model
HPS	Harmonic Product Spectrum
ICA	Independent Component Analysis
KNN	K-Nearest Neighbor

LLD	Low-Level audio descriptors
LPA	Linear Prediction analysis
LPCC	LP based Cepstral Coefficients
MFCC	Mel-frequency cepstral coefficients
ML	Maximum Likelihood
MMI	Maximum Mutual Information
NN	Neutral Network
PCA	Principal Component Analysis
RC	Real Cepstrum
SI(S)	Speaker Identification (System)
SR(S)	Speaker Recognition (System)
SV(S)	Speaker verification (System)
VQ	Vector Quantization

Contents

List of Figures

Chapter 1 Introduction

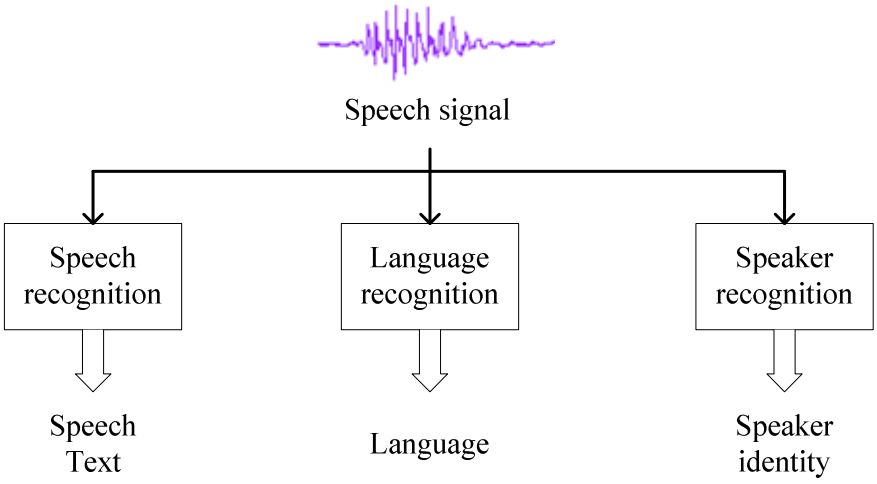


Fig. 1.1 Automatically extract information transmitted in speech signal
The main structure is taken from [1]. The speech signal contains rich messages, and three main recognition fields from speech signal, which are of most interest and have been studied for several decades, are speech recognition, language recognition and speaker recognition. In this thesis, we focus our attention on speaker recognition field.

In our everyday lives there are many forms of communication, for instance: body language, textual language, pictorial language and speech, etc. However amongst those forms speech is always regarded as the most powerful form because of its rich dimensions character. Except for the speech text (words), the rich dimensions also refer as the gender, attitude, emotion, health situation and identity of a speaker. Such information is very important for an effective communication.

From the signal processing point of view, speech can be characterized in terms of the signal carrying message information. The waveform could be one of the representations of speech, and this kind of signal has been most useful in practical applications. Extracting from speech signal, we could get three main kinds of information: Speech Text, Language and Speaker Identity [1], shown in Fig.1.1.

1.1 Elementary Concepts and Terminology

We notice from Fig. 1.1 there are three recognition systems: speech recognition systems, language recognition systems and speaker recognition systems. In this thesis, we concentrate ourselves on speaker recognition systems (*SRS*). In the mean while, for the purpose of fixing the idea about *SRS*, speech recognition will be introduced, and the distinctions between speech recognition and *SR* will be given too.

1.1.1 Speech Recognition

During the past four decades, a large number of speech processing techniques have been proposed and implemented, and a number of significant advances have been witted in this field during the last one to two decades, which are spurred by the high speed developing algorithms, computational architectures and hardware. Speech recognition refers to the ability of a machine or program to recognize or identify spoken words and carry out voice. The spoken words are digitized into sequence of numbers, and matched against coded dictionaries so as to identify the words.

Speech recognition systems are normally classified as to following aspects:

Whether system requires users to train it so as to recognize users' speech patterns;
Whether system is able to recognize continuous speech or discrete words;
Whether system is able to recognize small vocabulary or large one[1].

A number of speech recognition systems are already available on the market now. The best can recognize thousands of words. Some are speaker-dependent, others are discrete speech systems. With the development of this field speech recognition systems are entering the mainstream, and are being used as an alternative to keyboards.

1.1.2 Principles of Speaker Recognition

However nowadays more and more attention has been paid on speaker recognition field. Speaker recognition, which involves two applications: speaker identification and speaker verification, is the process of automatically recognizing who is speaking on the basis of individual information included in speech waves. This technique makes it possible to use the speaker's voice to verify their identity and control access to services such as voice dialing, banking by telephone, telephone shopping, database access services, information services, voice mail, security control for confidential information areas, and remote access to computers [2].

Speaker verification (*SV*) is the process of determining whether the speaker identity is who the person claims to be. Different terms which have the same definition as *SV* could be found in literature, such as voice verification, voice authentication, speaker/talker authentication, talker verification. It performs a one-to-one comparison (it is also called binary decision) between the features of an input voice and those of the claimed voice that is registered in the system.

Fig. 1.2 shows the basic structure of *SV* system (*SVS*). There are three main components: Front-end Processing, Speaker Modeling, and Pattern Matching. *Front-end processing* is used to highlight the relevant features and remove the irrelevant ones.

---

[1] Small vocabulary approximately includes tens or at most hundreds of words; on the contrary, large vocabulary refers thousands of words.

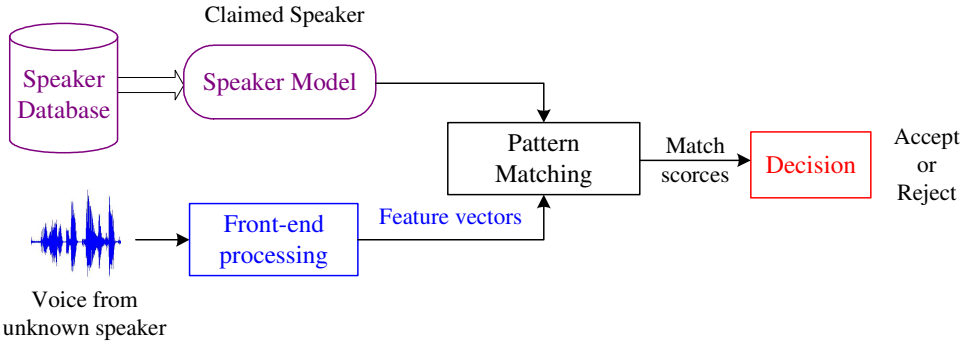


Fig. 1.2 Basic structure of Speaker Verification

Three main components shown in this structure are: Front-end Processing, Speaker Modeling, and Pattern Matching. To get the feature vectors of incoming voice, front-end processing will be performed, and then depending on the models used in Pattern Matching, match scores will be calculated. If the score is larger than a certain threshold, then as a result, claimed speaker would be acknowledged.

After the first component, we will get the feature vectors of the speech signal. Pattern Matching between the claimed *speaker model* registered in the database and the *imposter model* will be performed then, which will be described in detail in Chapter 4. If the match is above a certain threshold, the identity claim is verified. Using a high threshold, system gets high safety and prevents impostors to be accepted, but in the mean while it also takes the risk of rejecting the genuine person, and vice versa.

Speaker identification (*SI*) is the process of finding the identity of an unknown speaker by comparing his/her voice with voices of registered speakers in the database. It's a one-to-many comparison [3]. The basic structure of *SI* system (*SIS*) is shown in Fig. 1.3. We notice that the core components in *SIS* are the same as in *SVS*. In *SIS*, M speaker models are scored in parallel and the most-likely one is reported.

In different situations, speaker recognition is often classified into closed-set recognition and open-set recognition. Just as their names suggest, the closed-set refers to the cases that the unknown voice must come from a set of known speakers; and the open-set means unknown voice may come from unregistered speakers, in which case we could add 'none of the above' option to this identification system.

Moreover in practice speaker recognition systems could also be divided according to the speech modalities: text-dependent recognition, text-independent recognition. For text-dependent *SRS*, speakers are only allowed to say some specific sentences or words, which are known to the system. In the bargain, the text-dependent recognition is sub-

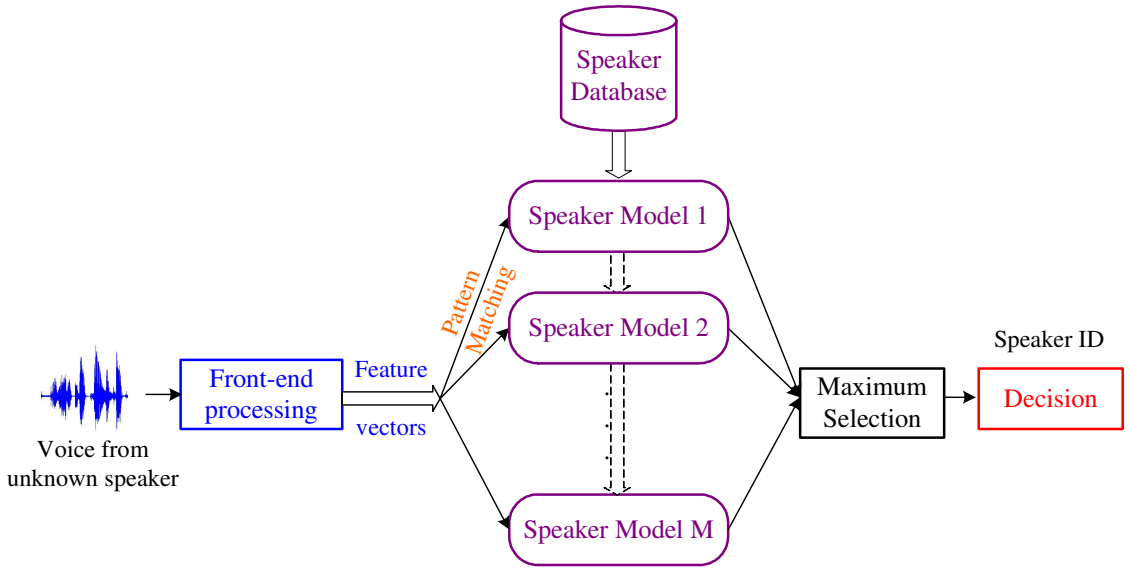


Fig. 1.3 Basic structure of Speaker Identification

The core components in *SIS* are the same as in *SVS*. In *SIS*, M speaker models are scored in parallel and the most-likely one is reported, and consequently decision will be one of the speaker's ID in the database, or will be 'none of the above' if and only if the matching score is below some threshold and it's in the case of a open-set *SIS*.

divided into fixed phrase and prompted phrase. On the contrary, as for the text-independent *SRS*, they could process freely spoken speech, which is either user selected phrase or conversational speech. Compared with text-dependent *SRS*, text-independent *SRS* are more flexible, but more complicated.

The detailed taxonomy of speech processing is shown in Fig. 1.4, so as to give a general view.

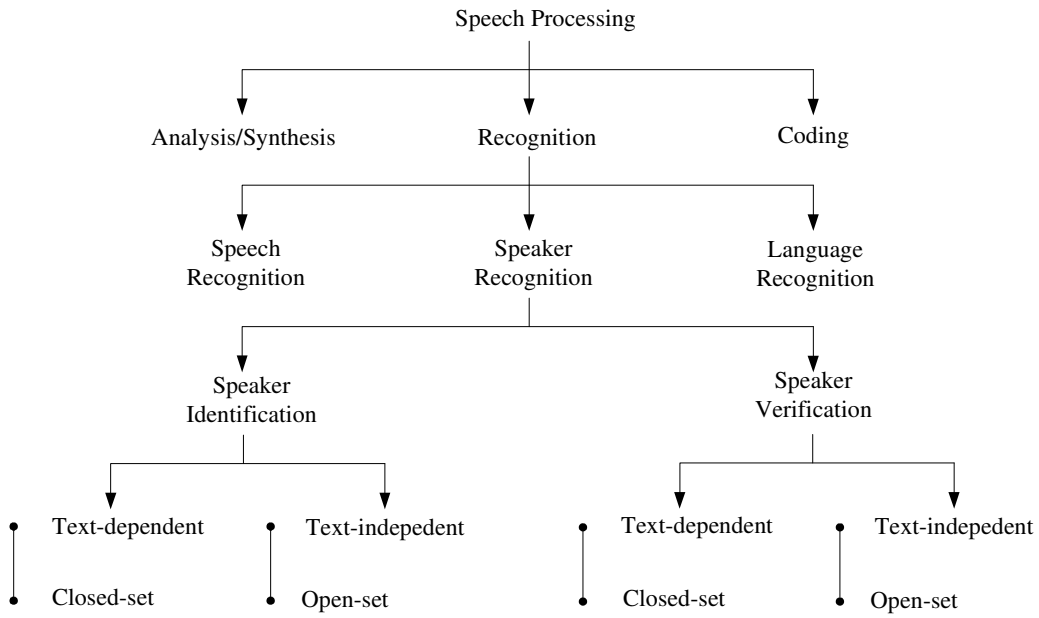


Fig. 1.4 Speech processing taxonomy

Speech signal processing could be divided into three different tasks: Analysis, Recognition and Coding. Shown in Fig. 1.1, recognition research fields could be subdivided into three parts: Speech, Speaker and Language recognition. Into the bargain, according to the different applications and situations that recognition systems work in, Speaker recognition is classified into text-dependent, -independent, closed-set and open-set.

Before processing, it's important to emphasize the difference between *SV* and speech recognition. The aim of speech recognition system is to find out what the speaker is saying, and to assist the speaker in accomplishing what he/she wants to do. However speaker verification system is often used for security. The system will ask speakers to say some specific words or numbers, but unlike speech recognition system, the system doesn't know whether the speakers have said what they are expected to say. Moreover in some literature voice recognition is mentioned. Voice recognition is ambiguous, and it usually refers to speech recognition, but sometimes it is also used as a synonym for speaker verification.

1.1.3 Phases of Speaker Identification

For almost all the recognition systems, training is the first step. We call this step in *SIS* enrollment phase, and call the following step identification phase. Enrollment phase is to get the speaker models or voiceprints for speaker database. The first phase of verification systems is also enrollment. In this phase we extract the most useful features from speech signal for *SI*, and train models to get optimal system parameters.

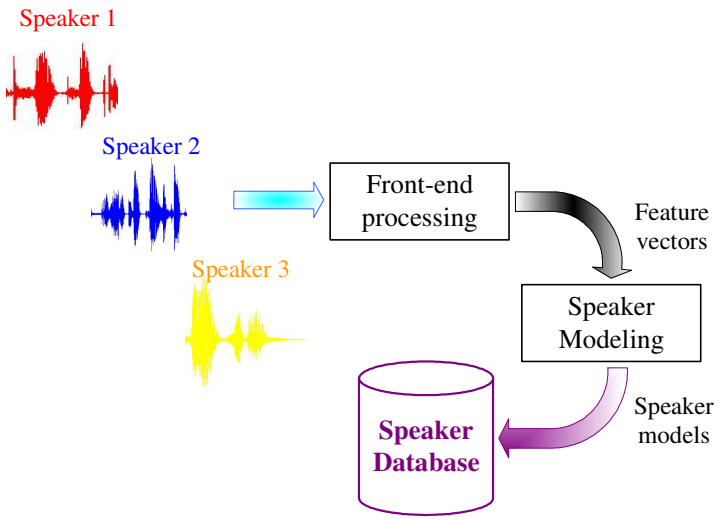


Fig. 1.5 Enrollment phase for *SI*

Enrollment phase is to get the speaker models or voiceprints to make a speaker database, which could be used later in the next phase, i.e. identification phase. The *front-end processing* and *speaker modeling* algorithms in both phases of *SIS* (*SVS*) should be consistent respectively.

In identification phase, see Fig. 1.3, the same method for extracting features as in the first phase is used for the incoming speech signal, and then the speaker models getting from enrollment phase are used to calculate the similarity between the new speech signal model and all the speaker models in the database. In closed-set case the new speaker will be assigned to the speaker ID which has the maximum similarity in the database. Even though the enrollment phase and identification phase are working separately, they are still closely related. The modeling algorithms used in the enrollment phase will also work on the identification algorithms.

1.2 Development of Speaker Recognition Systems

The first type of speaker recognition machine using spectrograms of voices was invented in the 1960's. It was called voiceprint analysis or visible speech. Voiceprint is acoustic spectrum of the voice, and it has similar definition as fingerprint. Both of them belong to biometrics[2]. However voiceprint analysis could not realize automatic recognition. Human's manual determination was needed. Until now a number of feature extraction techniques, which are commonly used in Speech Recognition field, have been used to distinguish from individuals. Since the mid-1980s, this field has been steadily getting matured that commercial applications of *SR* have been increasing, and many companies currently offer this technology.

[2] Biometrics is the technology of measuring and analyzing uniquely identifiable physical human characteristics: handwriting, fingerprint, finger lengths, eye retina, iris and voiceprint.

For speaker recognition problem, different representations of the audio signal using different features have been addressed. Features can be calculated in time domain, frequency domain [4], or in both domains [6]. [6] started from the system illustrated in [5], and used features calculated in both domains. For their own database which was extracted from Italian TV news, the system achieved 99% recognition rate when 1.5 seconds was used to identify.

Furthermore, different classification paradigms using different modeling techniques (see Fig. 1.6) for *SRS* could be found, such as *Gaussian Mixture Model* (GMM) [5] and *Hidden Markov Model* (HMM), which are prevalent techniques in *SR* field. The system in [5] has been frequently quoted. It uses Mel-scale cepstral coefficients, which is cepstral analysis in the frequency domain. Based on [5], transformations have been done. One example can be seen in [7], which transformed Mel cepstral features for compensating the noise components in the audio channel, and then formants features were calculated and used in classification. In [8], *Principal Component Analysis* (PCA) was used on the features based on [5]. PCA was to reduce the computational complexity of the classification phase, and it will be described in detail in Chapter 7. Moreover, speaker recognition applications have distinct constraints and work in different situations. Following the applications requests, recognition systems are divided into closed-set [7], open-set, text-independent [6], [7] and text-dependent [8]. According to the usage of applications, systems are designed for single speaker, and also for multi-speaker [10]. As a part of information included in spoken utterance, emotions get more and more attention at present, and vocal emotions have been studied as a separate topic. [13] shows that the average *fundamental frequency* increased and the range of *fundamental frequency* enlarged when the speaker was involved in a stressful situation.

Until now, MPEG-7 as a new technique is used for speaker recognition [11]. MPEG-7, formally named “Multimedia Content Description Interface”, is a standard for describing the multimedia content data that supports some degree of interpretation of the information’s meaning, which can be passed onto, or accessed by, a device or a computer code. MPEG-7 is not aimed at any one application in particular; rather, the elements that MPEG-7 standardizes support as broad a range of applications as possible [12]. In [11] for speaker recognition problem, MPEG-7 Audio standard were used. MPEG-7 Audio standard comprises descriptors and description schemes. They are divided into two classes: generic low-level tools and application-specific tools. There are 17 low-level audio descriptors (LLD). [11] used a method of projection onto a low-dimensional subspace via reduced-rank spectral basis functions to extract speech features. Here two LLDs were used: *AudioSpectrumBasisType* and *AudioSpectrumProjectionType*. Using *Independent Component Analysis* (ICA) in [11], the speaker recognition accuracy for small set is 91.2%, for large set is 93.6%; and the gender recognition accuracy for small set is 100%.

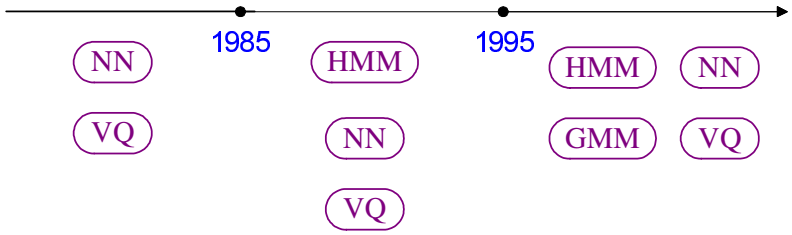


Fig. 1.6 Classification paradigms used in *SRS* during the past 20 years (taken from CWJ's presentation slides [31]) VQ, NN, HMM and GMM represent Vector Quantization, Neutral Network, Hidden Markov Model and Gaussian Mixture Model respectively. It has been shown that a continuous ergodic HMM method is superior to a discrete ergodic HMM method and that a continuous ergodic HMM method is as robust as a VQ-based method when enough training data is available. However, when little data is available, the VQ-based method is more robust than a continuous HMM method [9].

1.3 Project Description

Although a lot of work has been done in *SRS* field, many realistic problems still need to be solved, and as far as we know, no work has been done in hearing aid application. This research work is to make general overview of the techniques have been utilized in practice, and to design a *SIS* and do preparation work in lab for people with hearing loss. As long as we know the speakers ID, the task for speech enhancement will become comparably easy.

For most people in their everyday lives, the number of people whom they contact with is limited. For example the number is approximately around 30-50 for people with many social doings, and it's probably around 10 to 20 in regular cases. For the people with hearing loss, those 10 to 20 people could be the ones whom the patient is most familiar with. Therefore according to our purpose, a speech database will be built, which enrolls 22 people's voice messages, for details see Chapter 6.

In this project we concentrate ourselves on Speaker Identification System (*SIS*) since a large work has been done in *SV* field. For detailed condition, we will work in the closed-set, text-independent situations.

The report is organized in the following chapters:

- Chapter 2 gives the general overview of human speech production, and consequently introduces the speech model and the estimated model.
- Chapter 3 mainly describes the *front-end processing*. Before going to the main topic, short-term spectral analysis is introduced with some basic concepts of framing and windowing, etc. The description of *front-end processing* is divided

into three parts: preemphasis; feature extraction; and channel compensation, more attention and efforts are put into feature extraction techniques commonly used in *SRS*.

- Chapter 4 presents some speaker modeling and recognition techniques and algorithms. Moreover HMM is introduced in details beginning with the basic form, Markov chain.
- Chapter 5 introduces the idea of our speaker pruning and the pruning algorithm.
- Chapter 6 describes the speech database (ELSDSR) made in the period of my project.
- Chapter 7 presents experiments and results. First preemphasis and feature extraction are executed and comparisons amongst features are made using different techniques. A new method is invented to combine the pitch information with MFCC features for calculating the similarity in KNN algorithm with the purpose of improving speaker pruning accuracy. Finally experiments on HMM modeling and recognition are given with different setups. Moreover error rate with and without speaker pruning are compared.
- Chapter 8 summarizes this project results, and discusses the improvement could be achieved in the future work.

Chapter 2 Speech Production

Front-end processing, the first component in the basic structure of *SR* system (subsection 1.1.2) is a key element of the recognition process. The main task of *front-end processing* in *SRS* is to find the relevant information from speech, which could represent speaker's individual voice characters, and help achieve good classification results. However in order to get desired features for speaker recognition task, it is crucial to understand the mechanism of speech production, the properties of human speech production model, and the *articulators* which have speaker-dependent characters.

There are two main sources of speaker-specific characteristics of speech: physical traits and learned traits [14]. Learned traits include speaking rate, timing patterns, pitch patterns, prosodic effects, dialect, idiosyncratic word/phase usage, etc. They belong to high-level cues for speaker recognition. Although the high-level cues (learned traits) are more robust and are not much affected by noise and channel mismatch, we limit our scope in the low-level cues (physical traits) because they are easy to be automatically extracted and suitable for our purpose.

2.1 Speech Production

Speech is human being's primary means of communication, and it contents essentially the meaning of information from a speaker to a hearer, individual information representing speaker's identity and gender, and also sometimes the emotions. For a complete account of speech production, the properties of both articulators, which produce the sound, and auditory organs, which perceive the sound, should be involved. Nonetheless auditory organs are beyond the scope of this paper.

Speech production process begins with a thought which shows the initial communication message. Following the rules of the spoken language and grammatical structure, words and phrases are selected and ordered. After the thought constructs into language, brain sends commands by means of motor nerves to the vocal muscles, which move the vocal organs to produce sound [16].

Speech production can be divided into three principal components: excitation production, vocal tract articulation, and lips' and/or nostrils' radiation.

2.1.1 Excitation Source Production

Excitation powers the speech production process. It is produced by the airflow from lungs, and then carried by trachea through the vocal folds, see Fig. 2.1. During inspiration, air is filled into lungs, and during expiration the energy will be spontaneously released. The trachea conveys the resulting air stream to the larynx.

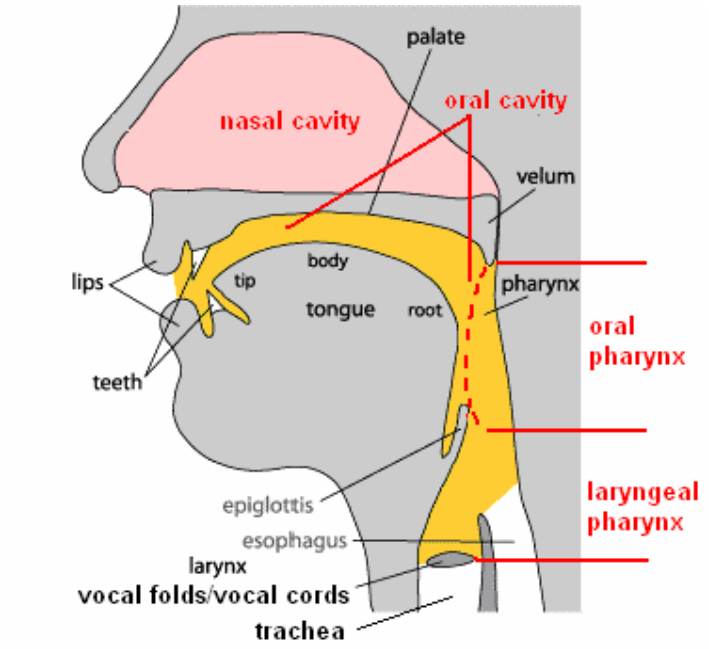


Fig. 2.1 Anatomical structure of human vocal system
(Adapted from 'How language works', Indiana University and Michael Gasser, Edition 2.0 2003 www.indiana.edu/~hlw/PhonUnits/vowels.html)
This figure was made according to the human vocal system introduced in [14]. The organs are (from the bottom up): **lungs** (not shown in this picture) which is the source of air; **trachea** (also called windpipe); **vocal folds/vocal cords** at the base of larynx is the most important part of larynx, and the area between vocal folds is glottis; **epiglottis**; **pharynx**; **velum** (also called soft palate) which allows air passing through the nasal cavity; **nasal cavity** (nose); **oral cavity**; **palate** (hard palate) which enables consonant articulation; **tongue**; **teeth**; **lips**.

Larynx refers as an energy provider to serve inputs to the vocal tract, and the volume of air determines the amplitude of the sound. The vocal folds at the base of larynx, and glottis triangular-shaped space between the vocal folds are the critical parts from speech production point of view. They separate the trachea from the base of vocal tract. The types of sounds are determined by the action of vocal folds, and we call it excitation. Normally excitations are characterized as phonation, whispering, friction, compression, vibration, or a combination of these. Speech produced by phonated excitation is called voiced, produced by the cooperation between phonation and frication is called mixed voiced, and produced by other types of excitation is called unvoiced [14].

Voiced speech is generated by modulating the air stream from the lungs, and the generation is performed by periodically open and close vocal folds. The oscillation frequency of vocal folds is called the *fundamental frequency*, F_0, and it depends on the physical characters of vocal folds. Hence *fundamental frequency* is an important

physical distinguishing factor, which has been found effective for automatic speech and speaker recognition. Vowels and nasal consonants belong to voiced speech.

Mixed voiced speech is produced by the phonation plus frication. Actually unlike the phonation that is placed in vocal folds (the vibration of vocal folds), the place of frication is inside the vocal tract (subsection 2.1.2).

Unvoiced speech is generated by a constriction of the vocal tract narrow enough to cause turbulent airflow, which results in noise or breathy voice [15]. It includes fricatives, sibilants, stops, plosives and affricates. Unvoiced speech is often regarded and modeled as white noise.

2.1.2 Vocal Tract Articulation

Vocal tract is generally considered as the speech production organ above the vocal folds, which is formerly known as vocal cords, and its shape is another important physical distinguishing factor. Fig. 2.1 pictures the anatomical structure of human vocal system. It includes both the excitation organs and vocal tract organs. Lungs, trachea and vocal folds are regarded as organs responsible for excitation production. The combination of Epiglottis, pharynx, velum (soft palate), hard palate, nasal cavity, oral cavity, tongue, teeth and lips in the picture is referred to the vocal tract. The articulators included in vocal tract are group into: [14]

- Laryngeal pharynx (beneath the epiglottis);
- Oral pharynx (behind the tongue, between the epiglottis and velum);
- Oral cavity (forward of the velum and bounded by the lips, tongue and palate);
- Nasal pharynx (above the velum, rear end of nasal cavity);
- Nasal cavity (above the palate and extending from the pharynx to the nostrils).

While the acoustic wave produced by excitations is passing through the vocal tract, depending on the shape of the vocal tract, wave will be altered in a certain way and interferences will generate resonances. The resonances of vocal tract are called *formants*. Their location largely determines the speech sound which is heard [15].

The vocal tract works as a filter to shape the excitation sources. The uniqueness of speaker voice not only depends on the physical features[3] of the vocal tract, but the speaker's mental ability to control the muscles of the organs in the vocal tract. It is not easy for speaker to change the physical features intentionally. However, these physical features are possible to be changed with ageing.

[3] Physical features of the vocal tract normally refer to vocal tract length, width and breadth, size of tongue, size of teeth and tissue density, etc [13].

2.2 Discrete-time Filter Modeling

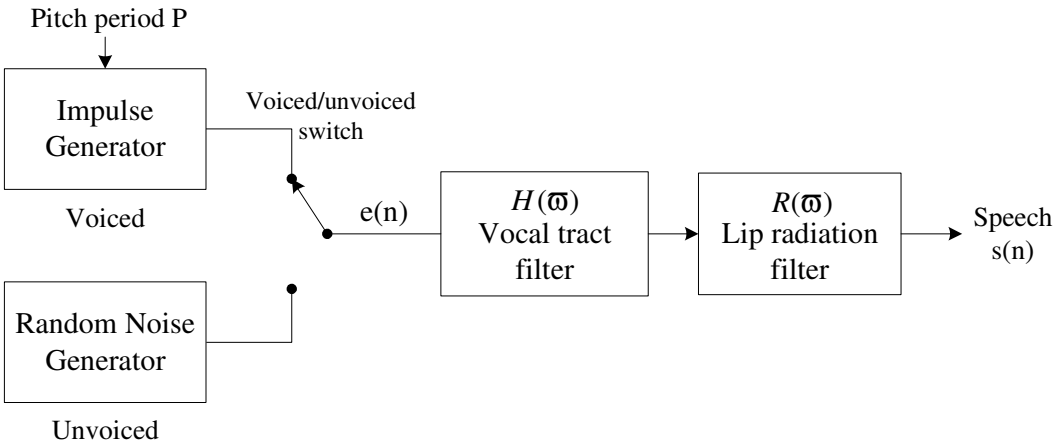


Fig. 2.2 Discrete-time speech production model (based on [16] Chapter 3)
Assuming speech production can be separated into excitation production, vocal tract articulation and lips' and/or nostrils' radiation three linear and planar propagation components, discrete-time speech production model was built.

Mentioned before, speech production is normally divided into three principal components: excitation production, vocal tract articulation and lips' and/or nostrils' radiation. As we separate speech production process into three individual parts, which have no coupling between each other, we assume that these three components are linear, separately and planar propagation[4] [16]. Further more let's think about speech production in terms of an acoustic filtering operation. Consequently we could construct a simple linear model, discrete-time filter model, for speech production, which consists of excitation production part, vocal tract filter part and radiation part separately [16], shown in Fig. 2.2. The excitation part corresponds to the vibrating of the vocal cords (glottis) causing voiced sounds, or to a constriction of the vocal tract causing a turbulent air-flow and thus causing the noise-like unvoiced excitation.

By using this model, a voiced speech, such as vowel, can be computed as the product of three respective (Fourier) transfer functions:

$$S(\varpi) = E(\varpi)H(\varpi)R(\varpi) \tag{2.1}$$

where the excitation spectrum $E(\omega)$ and radiation $R(\omega)$ are mostly constant and well known a priori, the vocal tract transfer function $H(\omega)$ is the characteristic part to determine articulation [15]. Therefore it deserves our special attention on how it can be modeled adequately.

[4] Planar propagation assumes that when the vocal folds open, a uniform sound pressure wave is produced that expands to fill the present cross-sectional area of the vocal tract and propagates evenly up through the vocal tract to the lips [16].

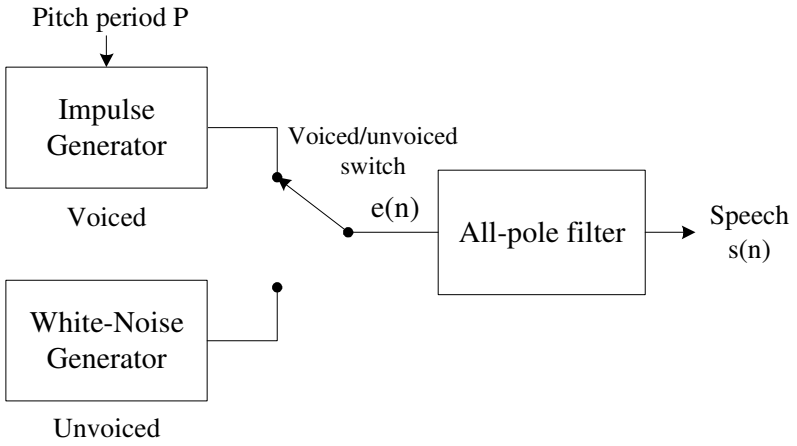


Fig. 2.3 Estimated speech production model (based on [16] Chapter 5)

By using all-pole filter to replace the vocal tract filter and lip radiation models, corrected magnitude spectrum could be achieved, but phase information in speech signal will be lost. Since human ear is fundamentally 'phase deaf', the LP estimated model (all-pole model) could also work well.

In time-domain, relation 2.1 will be presented as a convolution[5] combination of excitation sequence, the vocal system impulse response, and the speech radiation impulse response:

$$s(n) = e(n) \otimes h(n) \otimes r(n) \tag{2.2}$$

where excitation sequence has the following definition:

$$e(n) = \begin{cases} \sum_{q=-\infty}^{\infty} \delta(n - qP), & \text{voiced case} \\ Exp(e(n)) = 0, \quad Var(e(n)) = 1, & \\ \sum_{k=-\infty}^{\infty} e(n)e(n-k) = 0 & \text{unvoiced case} \end{cases} \tag{2.3}$$

where *Exp* is the expectation operator, *Var* is the variance operator, and P is the pitch period.

As we know, the magnitude spectrum can be exactly modeled with stable poles, and the phase characteristics can be modeled with zeros. However with respect to speech perception, the speech made by a speaker walking around 'sounds the same' given sufficient amplitude since the human ear is fundamentally 'phase deaf' [16].

---

[5] A convolution is an integral that expresses the amount of overlap of one function *g* as it is shifted over another function *f*. It therefore "blends" one function with another. The mathematical expression for the convolution of two discrete-time functions *f*(n) and *g*(n) over an infinite range is given by:

$$f(n) \otimes g(n) = \sum_{k=-\infty}^{\infty} f(k)g(n-k)$$

Hence as an estimation for the true speech production model shown in Fig. 2.2, an all-pole model is valid and useful. LP model (all-pole model) has the correct magnitude spectrum, but minimum-phase characteristic compared with true speech model. Fig. 2.3 shows the estimated model using LP analysis, which is also called the source-filter model.

The transfer function of an all-pole filter is represented by:

$$\Theta(\varpi) = \frac{1}{\sum_{i=0}^{p} a_i (j\varpi)^{-1}} \tag{2.4}$$

where p is the number of poles; a_0=1; and a_i are the Linear Prediction Coefficients [16] chosen to minimize the mean square filter prediction error summed over the analysis window.

As a result of this estimation, speech signal then can be presented as the product of two transfer functions:

$$S(\varpi) = E(\varpi)\Theta(\varpi) \tag{2.5}$$

where $E(\omega)$ is the excitation spectrum, and $\Theta(\omega)$ is represented by (2.4). Consequently in time domain, the speech signal is as follows:

$$s(n) = e(n) \otimes \theta(n) \tag{2.6}$$

Chapter 3 Front-end Processing

In Chapter 2, we discussed the human speech production with the purpose of finding the speaker-specific characteristics for speaker recognition task. To make the human speech production processable from the signal processing point of view, discrete-time modeling was discussed to model the process as a source-filter model, where the vocal tract is viewed as a digital filter to shape the sound sources from vocal cords. The speaker-specific characteristics, as we introduced in Chapter 2, include two main sources: physical (low-level cues) and learned (high-level cues). Although high-level features have been recently exploited successfully in speaker recognition, especially in noise environments and channel mismatched cases, our attention is on the low-level spectral features because they are widely spread, easy to compute and model, and are much more related to the speech production mechanism and source-filter modeling. With an overview of the mechanism of speech production, the aim of the *front-end processing* becomes explicit, which is to extract the speaker discriminative features.

We begin Chapter 3 with an introduction to *short-term spectral analysis* which will be used throughout our project. Then the sub-processes of *front-end processing* will be presented. More strength will be put in feature extraction sub-process. We start from the theoretical background of each feature extraction technique. Brief discussion on selection of appropriated features will then be given.

3.1 Short-term Spectral Analysis

Speech signal changes continuously due to the movements of vocal system, and it is intrinsically non-stationary. Nonetheless, in short segments, typically 20 to 40ms, speech could be regarded as pseudo-stationary signal. Speech analysis is generally carried out in frequency domain with short segments across which the speech signal is assumed to be stationary, and this kind of analysis is often called *short-term spectral analysis*, for detailed explanation, see [16] Chapter 4.

Short-term speech analysis could be summarized as following sequences:

1. Block the speech signal into frames with the length of 20 to 40ms, and overlap of 50% to 75% (the overlap is to prevent lacking of information);
2. Windowing each frame with some window functions (windowing is to avoid problem brought by truncation of the signal);
3. Spectral analyzing frame by frame to transfer speech signal into short-term spectrum;
4. Features extraction to convert speech into parameter representation.

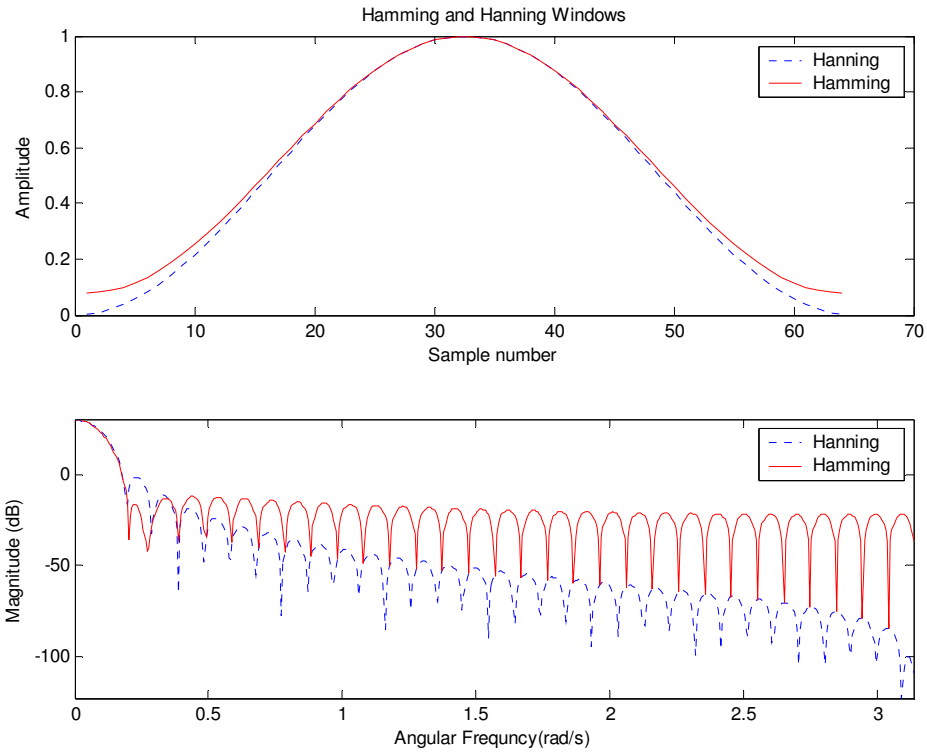


Fig. 3.1 Hamming and Hanning windows

Fig. 3.1 shows the waveforms and magnitude responses of Hamming window (red and solid line) and Hanning window (blue and dash line) with 64 samples. In time domain, Hamming window does not get as close to zero near the edges as the Hanning window does. In frequency domain, the main lobes of both Hamming and Hanning have the same width which is $8\pi/N$; whereas the Hamming window has lower side lobes adjacent to the main lobe than the Hanning window has, and side lobes farther from the main lobe are lower for the Hanning window.

3.1.1 Window Functions

Windowing is to reduce the effect of the spectral artifacts from framing process [17]. In time domain, windowing is a pointwise multiplication between the framed signal and the window function. Whereas in frequency domain, the combination becomes the convolution between the short-term spectrum and the transfer function of the window. A good window function has a narrow mainlobe and low sidelobe levels in their transfer function [17]. The windows commonly used during the frequency analysis of speech sounds are Hamming and Hanning window. They both belong to raised cosine windows family. These windows are formed by inverting and shifting a single cycle of a cosine so that to constrain the values in a specific range: [0, 1] for Hanning window; [0.054, 1] for Hamming window. Based on the same function, shown as follow:

$$W(n) = \varphi - (1-\varphi) \cdot \cos(\frac{2\pi n}{N-1}) \tag{3.1}$$

Hamming window chooses φ=0.54, and Hanning window, instead, chooses φ=0.5.

Fig. 3.1 shows the waveforms and magnitude responses of Hamming and Hanning window function. Notice that the Hamming window does not get as close to zero near the edges as the Hanning window does, and it could be seen effectively as a raised Hanning window. In magnitude response, the main lobes of both Hamming and Hanning have the same width which is 8π/N; whereas the Hamming window has lower side lobes adjacent to the main lobe than the Hanning window has, and side lobes farther from the main lobe are lower for the Hanning window.

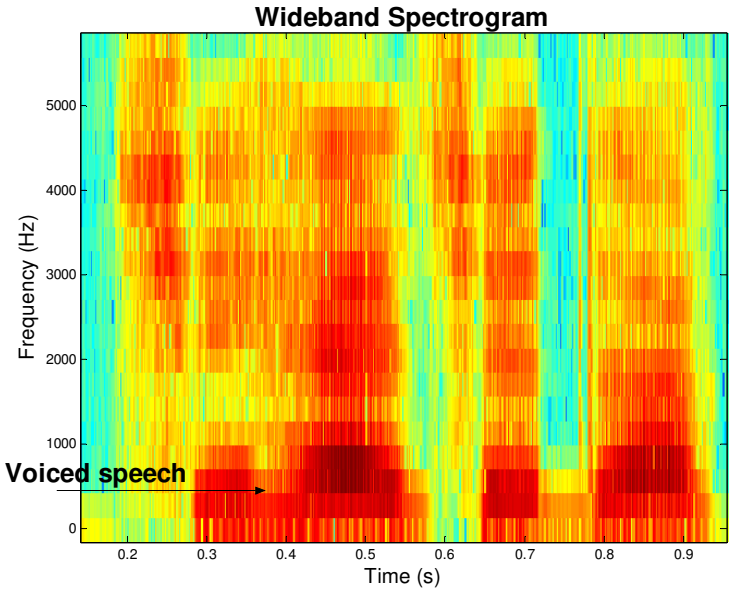
3.1.2 Spectrographic Analysis

Spectrogram for speech signal, the sonogram, is a visual representation of acoustic signal in frequency domain. It belongs to time-dependent frequency analysis. Spectrogram computes the windowed discrete-time Fourier transform (DTFT) of a signal using a sliding window. The mathematical representation of windowed DTFT is

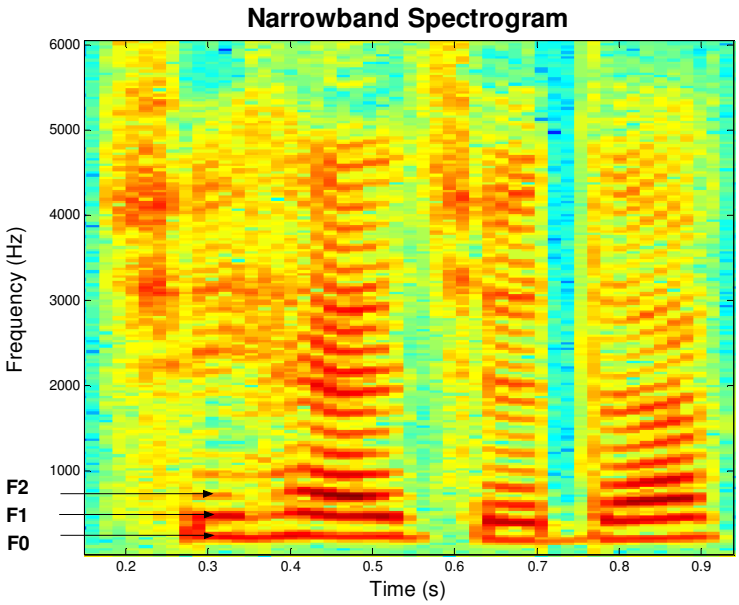
$$S_n(\omega) = \sum_{m=-\infty}^{\infty} s(m) w(n-m) e^{-j\omega m} \tag{3.2}$$

where $\omega \in (-\pi, \pi)$ denotes the *continuous* radian frequency variable, s(m) is the signal amplitude at sample number m, and $w(n-m)$ is the window function [17]. Spectrogram is a two dimensional plot of frequency against time, where the magnitudes at each frequency is represented by the grey scale darkness or of color in position (t, f) in the display, and the darker regions correspond to higher magnitudes.

Because of the inverse proportional relation between time and frequency resolution, trade-off exists. If the time resolution is high, then as a result, the frequency resolution will be poor. Depending on the size of the Fourier analysis window, there are two types of spectrograms: wideband and narrowband spectrograms [18], shown in Fig. 3.2. A long window results a narrowband spectrogram, which reveals individual harmonics, shown as red horizontal bars in voiced portions in Fig. 3.2 (b). On the contrary, a small window results a wideband spectrogram with better time resolution, but smeared adjacent harmonics.



(a)



(b)

Fig. 3.2 Wideband and narrowband spectrograms

(a) (b) show the spectrogram of the same utterance with different size of windows. (a) is the wideband spectrogram with 56 samples at 16 kHz, corresponding the time spacing is 3.5 ms. (b) is the narrowband spectrograms with 512 samples at 16 kHz, corresponding the time spacing is 32 ms. The pointed part in (a) shows the voice speech. Therefore wideband spectrograms can be used to track the voiced speech. Whereas in (b) harmonics, the red horizontal bars, can be clearly identified. The three arrows from bottom up in (b) point out the *fundamental frequency* F_0, the first formant F_1, and the second formant F_2. Thus narrowband spectrograms can be used to reveal individual harmonics, and to estimate F_0.

3.2 Sub-processes of Front-end Processing

Front-end processing is the first component in *SRS*, therefore the quality of the *front-end processing* will greatly determine the quality of the later two components: speaker modeling and pattern matching. In a word, features extracted from speech signals are vital for *SRS*.

Front-end processing generally consists of three sub-processes:

- **Preemphasis** is to compensate for the spectral damping of the higher frequencies w.r.t. the lower frequencies;
- **Feature extraction** is to convert speech waveform to some type of parametric representation. This sub-process is the key part in *front-end processing,* and always be viewed as a 'replacer' of *front-end processing*.
- **Channel compensation** is to compensate the different spectral characteristics on the speech signal induced by different input devices.

For the case of our own database created for specific purpose, channel compensation sub-process is not necessary because the recording situation and devices are the same for all the speakers, training, and test data set.

3.3 Preemphasis

Due to the characters of the human vocal system introduced in Chapter 2, glottal airflow and lip radiations make the higher frequency components of the voiced sounds dampened. For the voice sound, the glottal source has approximately -12db/ octave slope [19]. When the sound wave is radiated from lips, spectrum will be boosted +6db/octave. As a result, a speech signal has -6db/octave slope downward compared to the spectrum of vocal tract [18]. To eliminate this effect and prevent lower frequency components from dominating the signal, preemphasis should be performed before feature extraction. By preemphasizing, dynamic range will be decreased so as to let spectral modeling methods capture details at all frequency components equally.

In human auditory systems, the frequency response at a given spot along the cochlear membrane is like a high pass filter, which is tuned to a particular frequency that increases as speaker moves along the membrane [17]. This works just like the preemphasis processing.

3.3.1 The First Order FIR Filter

Generally preemphasis is performed by filtering the speech signal (original signal) with the first order FIR filter, which has the form as follow:

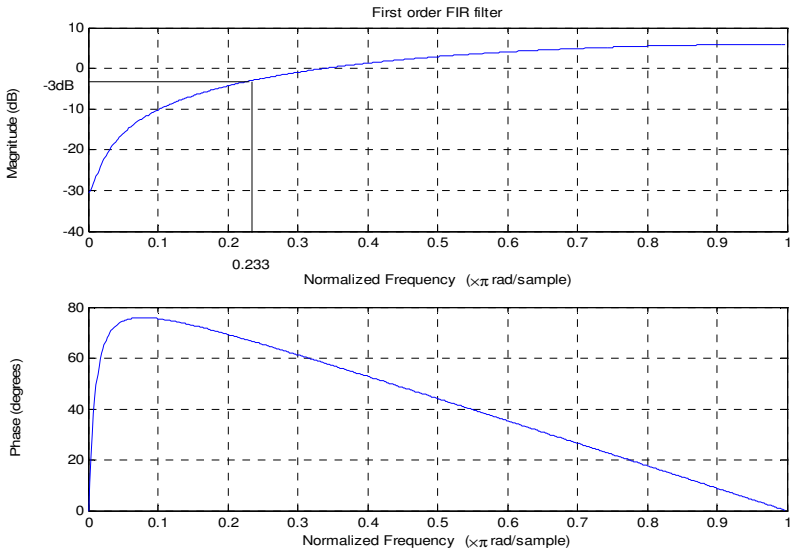


Fig. 3.3 Magnitude response and Phase response of a first order FIR filter

The first order FIR filter with 0.97 preemphasis factor works as a high pass filter. The cut-off frequency can be calculated when magnitude goes –3dB. Notice the magnitude beyond the 0db frequency in the upper panel, it shows that the high frequency components of the filtered signal will be enhanced a little bit. Notice the lower panel, the phase response, it shows that FIR filters with odd order cannot get linear phase.

$$F(z) = 1 - kz^{-1} \qquad (0 < k < 1) \tag{3.3}$$

where k is the preemphasis factor, and the recommended value is 0.97 [18, 19]. The magnitude response and phase response of the first order FIR filter with 0.97 preemphasis factor is shown in Fig. 3.3.

Consequently the output is formed as follow:

$$y(n) = s(n) - k \cdot s(n-1) \tag{3.4}$$

where s(n) is the input signal and y(n) is the output signal from the first order FIR filter.

3.3.2 Kaiser Frequency Filter

Notice the magnitude beyond the 0db frequency in the upper panel in Fig. 3.3 shows that the high frequency components of the filtered signal will be enhanced a little bit. To avoid this problem, we tried to use a frequency filter to achieve the high-pass effect. Frequency filtering is based on the Fourier Transform. Instead of doing convolution between signal and filter as spatial filter does, the operator does the multiplication between the transformed signal and the filter transfer function:

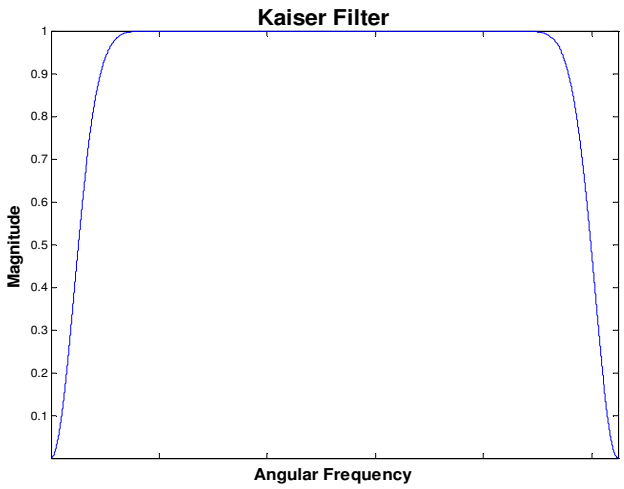


Fig. 3.4 Magnitude response of Kaiser frequency filter
Notice frequency part higher than the cut-off frequency f_0 remains unchanged since the magnitude is 1.

$$Y(\varpi) = S(\varpi)F(\varpi) \tag{3.5}$$

where $S(\omega)$ is the transformed signal, $F(\omega)$ is the filter transfer function, and $Y(\omega)$ is the filtered signal. To obtain the resulting signal in the time domain, inverse Fourier Transform needs to be done on $Y(\omega)$.
Since the multiplication in the frequency domain is identical to the convolution in the time domain, theoretically all frequency filters can be implemented as a spatial filter. However, in practice, the frequency filter function can only be approximated by the filtering mask in real space. The straight forward high pass filter is the ideal high pass filter. It suppresses all frequencies lower than the cut-off frequency $f_0 = \omega_0 / 2\pi$ and leaves the higher frequencies unchanged:

$$F(\varpi) = \begin{cases} 0 & \varpi < \varpi_0 \\ 1 & \varpi > \varpi_0 \end{cases} \tag{3.6}$$

However it has many drawbacks which make it impossible to realize and use in practice. The ringing effect which occurs along the edges of the filtered time domain signal is one drawback. Due to the multiple peaks in the ideal filter in the time domain, the filtered signal produces ringing along edges in the time domain.

Better results can be achieved with a Kaiser Filter. The advantage is that it does not incur as much ringing effect in the real space of the filtered signal as the ideal high pass filter does. Moreover it doesn't enhance the high frequency part as the first order FIR does. The magnitude response of the Kaiser Filter is shows in Fig. 3.4.

3.4 Feature Extraction

As we know feature extraction influences the recognition rate greatly, it is vital for any recognition/ classification systems. Feature extraction is to convert an observed speech signal (speech waveform) to some type of parametric representation for further analysis and processing. Features derived from spectrum of speech have proven to be the most effective in automatic systems [1]. However it is widely known that direct spectrum-based features are incompatible with recognition applications because of their high dimensionality and their inconsistency. Therefore the goal of features extraction is to transform the high-dimensional speech signal space to a relatively low-dimensional features subspace while preserving the speaker discriminative information to application. For example, during feature extraction, the features of the same pronunciations will be unified by removing the irrelevant information, and the features of different pronunciations will be distinguished by highlighting relevant information.

Another issue worth attention is the dimensionality of extracted features. We may think that the more relevant features, the better the recognition results. Unfortunately, things are always not as simple as we thought. The phenomenon, the curse of dimensionality [27], should cost our attention. The curse of dimensionality shows that the needed data for training and testing grow exponentially with the dimensionality of the input space, otherwise the representation will be very poor.

The desirable features for *SIS* should possess the following attributes: [1], [13]

- Easy to extract, easy to measure, occur frequently and naturally in speech
- Not be affected by speaker physical state (e.g. illness)
- Not change over time, and utterance variations (fast talking vs. slow talking rates)
- Not be affected by ambient noise
- Not subject to mimicry

Nevertheless, no feature has all these attributes. One thing we are sure about is that spectrum based features are the most effective in automatic recognition systems.

Before our own sound database becomes available, we will use TIMIT database which has been designed for the development and evaluation of automatic speech recognition systems. It contains 6300 sentences, 10 sentences spoken by each of 630 speakers from 8 major dialect regions of the United States. (In our case we neglect the dialect regions' influence.) Although TIMIT database was primarily designed for speech recognition in the noiseless environment, we could still use its' voice messages to perform different feature extraction methods so as to have a general idea about which methods are superior than others for the purpose of speaker recognition. Noise robustness is an important issue in real applications, however it is out of the scope of this thesis.

3.4.1 Short-Term Cepstrum

According to the source-filter model introduced in Section 2.2, speech signal s(n) can be represented as the convolved combination of the quickly varying part--excitation sequence and the slowly varying part--impulse response of the vocal system model, [16] shown as follow:

$$s(n) = e(n) \otimes \theta(n) \tag{3.7}$$

where e(n) denotes the excitation sequence, and θ(n) denotes the impulse response of the vocal system model.

It is always desired for engineers to work with linearly combined signals. The cepstral analysis appeared to resolve this problem, in addition the representatives in cepstrum are separated. The definition of real cepstrum (RC) of speech signal s(n) is:

$$c_s(n) = \Im^{-1}\left\{\log\left|\Im\{s(n)\}\right|\right\} = \frac{1}{2\pi}\int_{-\pi}^{\pi} \log\left|S(\omega)\right| e^{j\omega n} d\omega \tag{3.8}$$

where

$$S(\omega) = E(\omega)\Theta(\omega) \tag{3.9}$$

$$\log\left|S(\omega)\right| = \log\left|E(\omega)\right| + \log\left|\Theta(\omega)\right| = C_e(\omega) + C_\theta(\omega) \tag{3.10}$$

and $\Im\{\cdot\}$denotes the DTFT, $\Im^{-1}\{\cdot\}$ denotes IDTFT.

Fig. 3.5 shows the motivation behind RC. By transferring the time domain speech signal into frequency domain, the convolved combination of e(n) and θ(n) changes to multipliable combination. Moreover, by logarithming the spectral magnitude, the multipliable combination changes to additive combination. Because the inverse Fourier transform is a linear operator and would operate individually on two additive components, $c_s(n)$ can be rewritten as the linear combination:

$$c_s(n) = c_e(n) + c_\theta(n) \tag{3.11}$$

where

$$c_e(n) = \Im^{-1}\{C_e(\omega)\} \tag{3.12}$$

$$c_\theta(n) = \Im^{-1}\{C_\theta(\omega)\} \tag{3.13}$$

The domain of the new signal $c_e(n)$ and $c_\theta(n)$ is named as *quefrency* to describe the 'frequencies' in this new 'frequency domain' [16]. More detailed explanation can be found in [16].

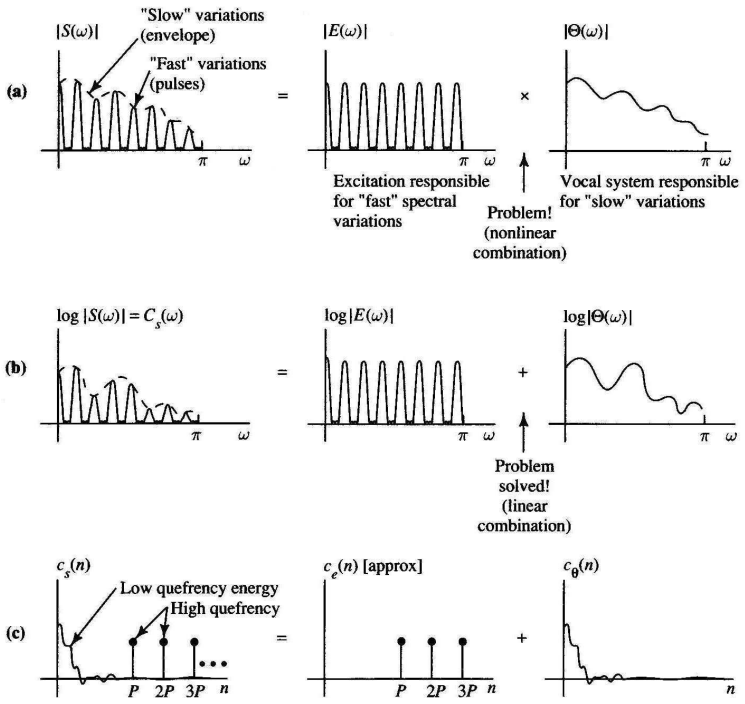


Fig. 3.5 Motivation behind RC (taken from [16] Fig. 6.3)

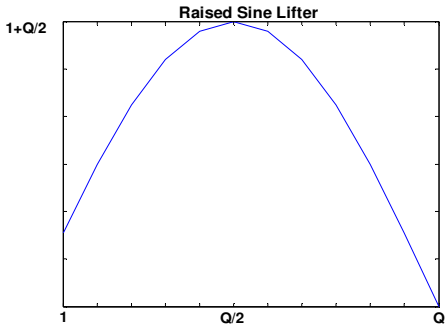


Fig. 3.6 Raised sine lifter

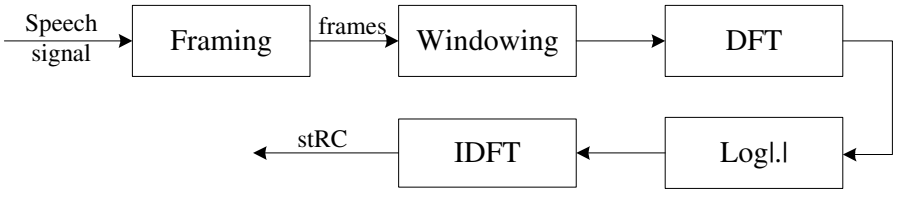


Fig. 3.7 Computation of the stRC using DFT

Following the *short-term spectral analysis* introduced in Section 3.1, speech signal should be framed and windowed into short time periods. In trun, as we described above the processure for RC, the DFT, logorithm and IDFT will be performed.

To eliminate one of the components, linear filtering can be used in the *quefrency* domain, where the filter is called lifter. Some low-time lifters are usually used, such as rectangular lifter and raised sine lifter. The analytical form of the second lifter is:

$$w(n) = 1 + \frac{Q}{2}\sin(\frac{\pi n}{Q}), n = 1,...,Q \tag{3.14}$$

Fig. 3.6 shows the figure of the raised sine lifter function. By adjusting the value of Q, which determines the length of lifter and is also the number of coefficients, we could separate the low *quefrency* part which represents vocal system impulse response and the high *quefrency* which represents excitation part. Later in Chapter 7 we will show how Q effects the features and recognition results.

As we described in Section 3.1, in practice the speech processing is performed on short terms. For the short-term Real Cepstrum (stRC), framing and windowing becomes the first step. The procedure of stRC is shown in Fig. 3.7.

Before processing, the author wants to emphasize that by real cepstrum (RC) we mean the Bogert-Tukey-Healy cepstrum, and it is equivalent to the even part of the Complex Cepstrum (CC), or homomorphic cepstrum, on the region over which the RC is defined. By simplified the CC into RC, we discard the phase information in homomorphic cepstrum. This simplification does no harm in phase-insensitive applications, and as we mentioned before that human ear is fundamentally 'phase deaf', the phase information becomes less important in speech signal. The relation between RC and CC is:

$$c_s(n) = \gamma_{s,even}(n) \tag{3.15}$$

where $c_s(n)$ is the RC, and $\gamma_{s,even}(n)$ is the even part of CC.

3.4.2 LP based Cepstrum

Linear Prediction analysis (LPA) has been commonly used as a spectral representation of speech signal for years. However, since LPA does not represent the vocal tract characteristics from the glottal dynamics [16], Linear Prediction (LP) coefficients are seldom used as features for *SRS*, and even for speaker-independent speech recognition systems. Efforts for improvement on LPA have been offered, and success has been achieved, which was to convert the LP coefficients to the cepstral coefficients. From the theoretical point of view, it is comparably easy to convert LP coefficients to stCC, and then convert stCC to stRC, in the case of minimum-phase signals. Given the LP coefficients $a(n;m)$, where m denotes the short-term analysis with m length window, the converting relation is presented by the recursive formula:

$$\gamma(n;m) = a(n;m) + \sum_{k=1}^{n-1}\left(\frac{k}{n}\right)\gamma(k,m)a(n-k;m), n = 1:Q \tag{3.16}$$

Given the stCC, stRC can be calculated by (3.15) in the short-term analysis case.

3.4.3 Mel-frequency Cepstrum

Until now, Mel-frequency cepstral coefficients (MFCC) are the best known and most commonly used features for not only speech recognition, but speaker recognition as well. The computation of MFCC is based on the *short-term analysis* introduced in Section 3.1, and it is similar to the computation of Cepstral Coefficients described in subsection 3.4.1. The significant difference lays on the usage of critical bank filters to realize mel-frequency warping. The critical bandwidths with frequency are based on the human ears perception. The block diagram for computing MFCC is given in Fig. 3.8.

A *mel* is a unit of measure based on the human ear's perceived frequency. According to [16], it is defined as the following way:

- First 1000 Hz is defined as 1000 Mels as a reference,
- Secondly listeners are asked to change the physical frequency until they perceive it is twice of the reference, or 10 times or half or one tenth of the reference, and so on.
- Thirdly those frequencies are then labeled as 2000 mels, 10,000 mels, 500 mels, 100 mels, and so on.

The mel scale is approximately a linear frequency spacing below 1000Hz, and a logarithmic spacing above 1000Hz. The approximation of Mel from frequency can be expressed as:

$$mel(f) = 2595 \cdot \log_{10}(1 + f/700) \tag{3.17}$$

where f denotes the real frequency, and $mel(f)$ denotes the perceived frequency.

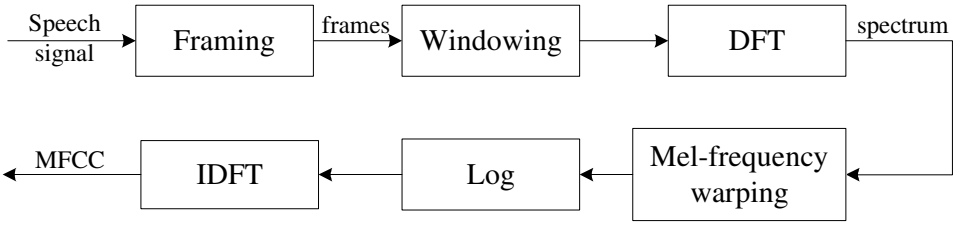


Fig. 3. 8 Computation of MFCC

The computation of Mel-frequency cepstrum is similar to that of Cepstral Coefficients. The difference lays on mel-frequency warping before doing logarithm and inverse DFT. The warping transfers the real frequency scale to the human perceived frequency scale called mel-frequency scale. The new scale spaces linearly below 1 kHz, and logarithmically above 1kHz.

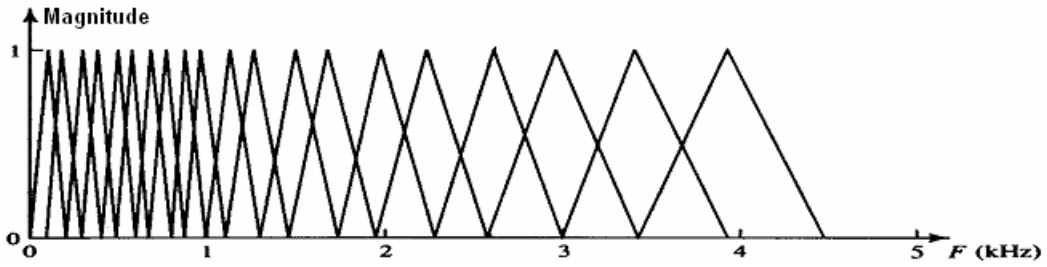


Fig. 3.9 The triangular Mel-frequency scaled filter banks
Notice the spacing of triangles: below 1 kHz, they are linearly distributed; and above 1 kHz they become logarithmically distributed.

The Mel-frequency Warping is normally realized by Filter banks. Filter banks can be implemented in both time domain and frequency domain. For the purpose of MFCC processor, filter banks are implemented in frequency domain before the logarithm and inverse DFT, see Fig. 3.8. The center frequencies of the filters are normally evenly spaced on the frequency axis. However, in order to mimic the human ears perception, the warped axis according to the non-linear function (3.17), is implemented. The most commonly used filter shaper is triangular, and in some cases the Hanning filter can be found. The triangular filter banks with *mel*-frequency warping is given in Fig.3.9.

The output of the *i*th filter *Y(i)* is given by [18]:

$$Y(i) = \sum_{j=1}^{N} S(j)\Omega_i(j) \tag{3.18}$$

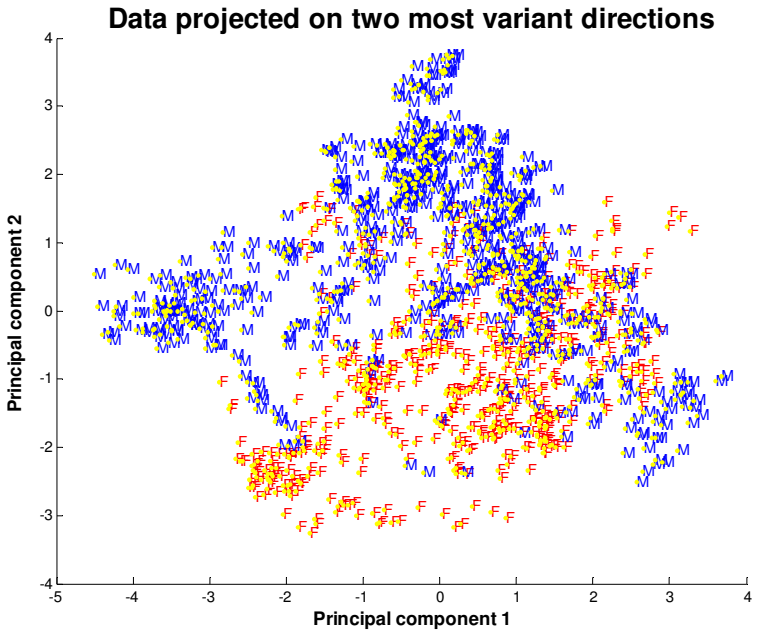
where $S(j)$ is an N-point magnitude spectrum (j=1:N), and $\Omega_i(j)$ is the sampled magnitude response of an M-channel filterbank (i=1:M). The output of the ith filter can be regarded as the magnitudes response of speech signal in that frequency region weighted by the filter response.

The last step before getting mel-cepstral coefficient is the IDFT. Normally the Discrete Cosine Transform (DCT) will be performed instead of IDFT since the output of the second last block in Fig. 3.8, $\log(Y(i))$, is symmetrical about the Nyquist frequency. Therefore MFCC is calculated as:

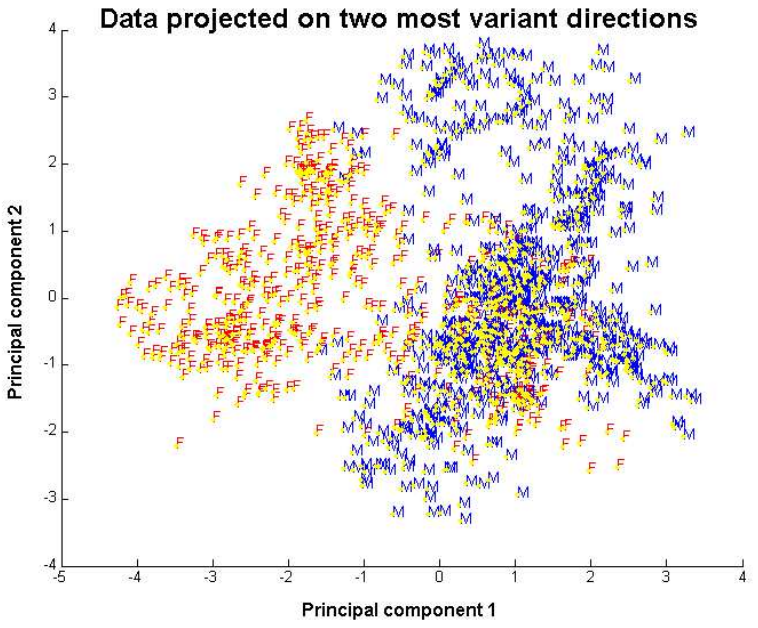
$$c_s(n,m) = \sum_{i=1}^{M} (\log Y(i)) \cos\left[i \frac{2\pi}{N'} n \right] \tag{3.19}$$

where N' is the number of points used to compute the stDFT.

Notice that zeroth coefficient (when n=0) is often excluded since it represents the average log-energy of the input signal, which only carries little speaker-specific information.



(a)



(b)

Fig. 3.10 LPCC vs. MFCC for speaker separation using TIMIT

FCJF0_SA1.wav and MGSHO_SA1.wav from TIMIT were experimented. (a) shows the LPCC. (b) shows MFCC. In both cases, frame size = 320 samples (20ms @ 16 kHz), overlap = 240 samples. No. of LPCC = 12 converted from 12 LP coefficients. No. of MFCC = 12. Then high dimensional data were projected into two principal components in order to observe the separation between two speakers. F denotes the female speaker, M denotes the male.

Fig. 3.10 gives the comparison between LP based Cepstral Coefficients (LPCC) and MFCC for speaker separation. The experimental samples are selected from TIMIT database: FCJF0_SA1.wav and MGSHO_SA1.wav, which represents one female and one male speaker saying the same sentence SA1. In both the LPCC and MFCC cases, the size for each frame is 320 samples (20ms @ 16 kHz), overlap is 240 samples. 12 cepstral coefficients converted from 12 LP coefficients were calculated. As well, 12 MFCC were computed to keep the number of coefficients consistent and comparable. 12 dimensional data were then projected into two most variant directions (two principal components) in order to observe the separation between two speakers. We notice from this figure that MFCC works better than LPCC on speaker separation in text-dependent case. More comparison experiments will come in Chapter 7.

3.4.4 Delta and Delta-Delta Coefficients

Until now no time evolution information is included in either cepstral coefficients or MFCC. However dynamic information in speech signal is also different from speaker to speaker. This information is often included in the feature set by cepstral derivatives. The first order derivative of cepstral coefficients is called Delta coefficients, and hereby the second order derivative of cepstral coefficients is called Delta-Delta coefficients. Delta coefficients tells us somehow the speech rate, and Delta-Delta coefficients gives us something similar to acceleration of speech.

Suppose $c_l(n;m)$ are the cepstral coefficients or MFCC after liftering, the Delta coefficients can be calculated as:

$$\Delta c_l(n;m) = \frac{1}{2}(c_l(n;m+1) - c_l(n;m-1)) \tag{3.20}$$

Delta-Delta coefficients could use the same Equation as (3.20) applied to Delta coefficients.

Fig.3.11 gives the figures of the original signal with its first MFCC, delta-coefficient, and delta-delta-coefficient.

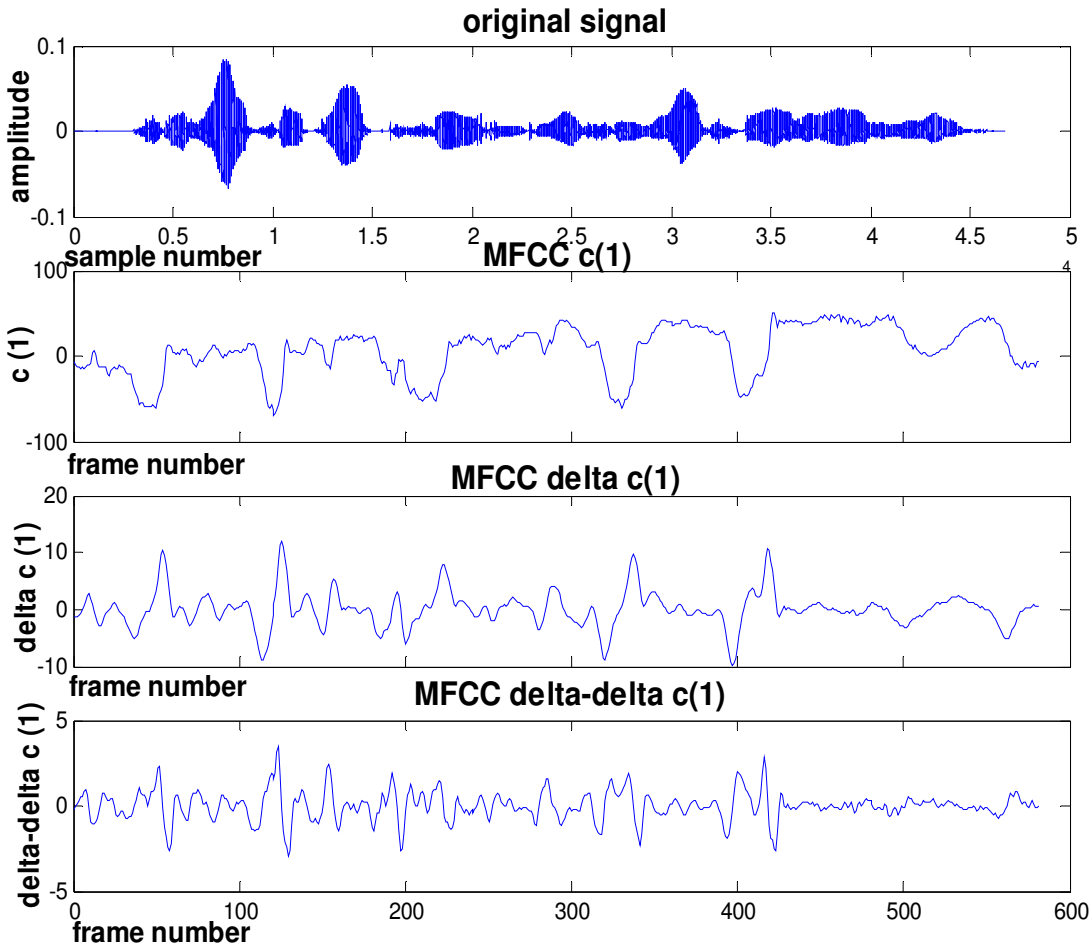


Fig. 3.11 Original Signal with MFCC, DMFCC and DDMFCC

Figure is divided into 4 panels: the first panel gives the original signal FCJF0_SA1.wav from TIMIT, which lasts around 3 seconds; the second panel shows the first MFCC from this signal: c(1); the third one shows the first DMFCC: dc(1)/dt; the last one shows the first DDMFCC: $d^{\wedge 2}c(1)/dt^{\wedge 2}$.

3.4.5 Fundamental Frequency

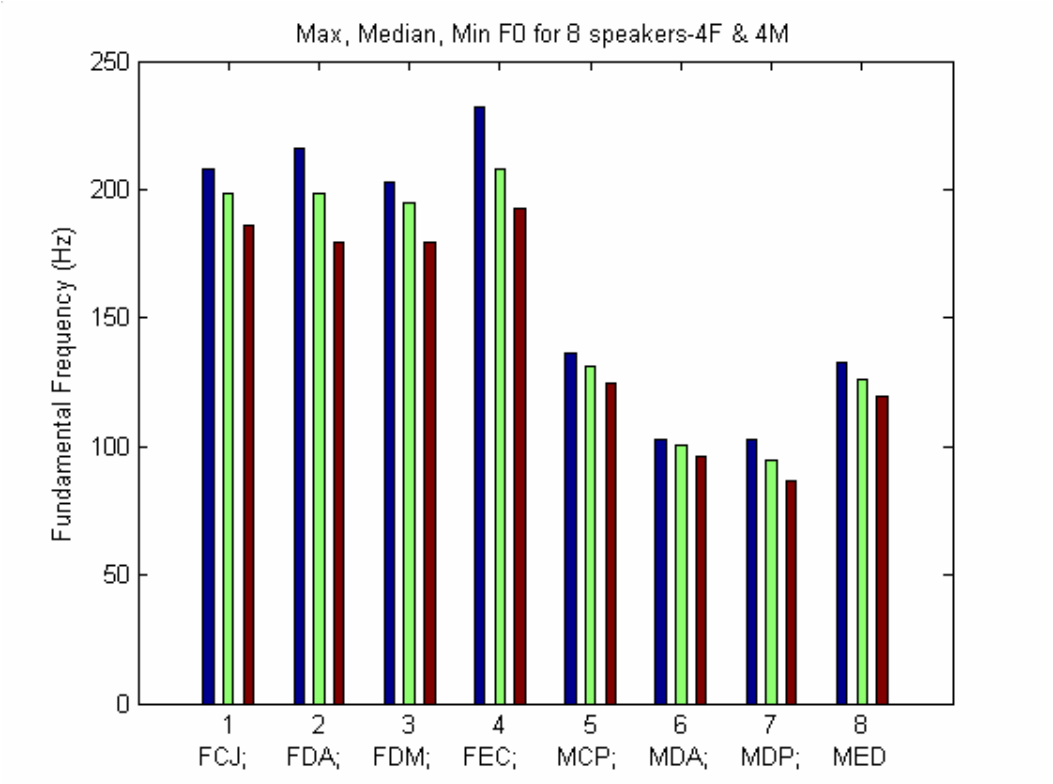


Fig. 3.12 F_0 information for eight speakers from TIMIT
Speaker FCJ FDA FDM and FEC are 4 female; MCP, MDA, MDP and MED are 4 male. This figure shows the maximum (blue bar), median (green bar) and minimum (red bar) F_0 values for each speaker. The F_0 range for female lays on approximately 180Hz to 232 Hz; and the F_0 range for male lays on 86.5Hz to136Hz.

As introduced in subsection 2.1.1, the vibration frequency of vocal folds is defined as *fundamental frequency*, an important feature for automatic speech and speaker recognition. *Fundamental frequency*, or pitch period is robust to noise and channel distortions. *SRS* based exclusively on pitch information work well with small number of speakers, but error rate increases significantly with the number of speaker increases. Therefore work has been done combining pitch information with other speaker specific features for *SR* [20].

There are many existing techniques for pitch extraction. For example: Cepstrum method (CEP); Autocorrelation method (AUTOC) which is in the time domain; Weighted Autocorrelation [21]; and Harmonic Product Spectrum (HPS) which is in the frequency domain [22]. According to [21], CEP is known as the best pitch extraction method in noiseless environments, and weighted autocorrelation method works well in noise environments.

In our cases, the CEP method was used in noiseless environment. Fig. 3.12 gives the maximum, median and minimum pitches for eight speakers: speaker No. 1 to 4 are female, and the rest are male. Notice the pitch range for female--180Hz to 232 Hz is obviously higher than that of male--86.5Hz to136Hz. Therefore pitch information is useful to classify speakers coarsely into broad groups (e.g., female, male), but is not so efficient to classify the speakers with same genders.

Chapter 4 Speaker Modeling

In Chapter 1, we introduced the enrollment phase for Speaker Identification (Fig. 1.5). The enrollment phase can be divided into two main components: *front-end processing* and *Speaker modeling*. In other words, during enrollment the speech signals from speakers will be first passed through *front-end processing*, described in Chapter 3, and the output of the first component--feature vectors will be used to create speaker models, which will be stored into speaker database for future use.

Nowadays the most prevalent *speaker modeling* techniques are:

- *Template Matching*;
- *Artificial Neural Networks* (ANNs);
- *Vector Quantization* (VQ);
- *Hidden Markov Models* (HMMs);
- *K-Nearest Neighbor* (KNN).

For *Template Matching* technique, the model is composed of a template, which is a sequence of feature vectors from a fixed phrase. Then *Dynamic Time Warping* (DTW) is used to calculate the match score between the test phrase and the speaker templates. However, this technique is almost only for text-dependent applications [23]. ANNs consist of numbers of interconnected elements, and these elements are tied together with weights. Given the input and output training data to the networks, and the training algorithm, weights could be learned. However, there are also limitations of ANNs that it is computationally expensive for training, and models are sometimes not generalizable [23]. VQ is a process of mapping vectors from a large vector space to a finite number of regions in that space. This process may be included in discrete HMMs. HMM is a Markov chain where each state generates an observation, the observations are visible and the state sequences are hidden which need to be inferred. KNN is a simple algorithm comparing to the above algorithms. It stores all the available examples from the training set to classify the new instances based on a similarity measure.

In this project, the HMM and KNN approaches will be used. KNN is proposed to be used as a *speaker pruning* technique. The goal of adding *speaker pruning* step in our *SIS* is to increase the identification accuracy with the cost of a little growing of the identification time. The trade-off between the identification speed and the accuracy should be adjustable according to the different usages and requests. The detailed descriptions for *speaker pruning* will be given in Chapter 5. HMM, as one of the most statistically mature methods for classification, will be used in *speaker modeling*.

We begin this chapter with the original model of HMM--the discrete Markov chain, and then extend to the concept of 'hidden'. Later HMM's applications to *speaker modeling* will be presented, along with the solving of the three problems of HMM.

4.1 Markov Chain

Markov chain can be viewed as one branch of Markov sequence. Before we describe the Markov chain, let's have a look at the Markov sequence first. Markov sequence consists of a collection of random variables $\{x_t\}$, where t is the time notation. The probabilistic description of Markov sequence requires specification of the current variable (at time t), as well as the previous variable (at time t-1, t-2,…,1). For special cases we make the assumption, first-order Markov assumption [25], where the probability of the future is independent of the past and only related with the present:

$$P(x_{t+1} = j | x_t = i_t, x_{t-1} = i_{t-1}, ..., x_1 = i_1) = P(x_{t+1} = j | x_t = i_t) \tag{4.1}$$

where the process is called the Markov process.

By giving only discrete values $s_1,\ldots s_N$ to the random variables $\{x_t\}$, we convert the Markov sequence to Markov chain, which is a discrete Markov process. The discrete values $s_1,\ldots s_N$ is called states. Following the first-order Markov assumption, the probabilistic description of Markov chain goes from state s_i immediately to state s_j is denoted by a_{ij}, transition coefficients:

$$a_{ij} = P(x_{t+1} = s_j | x_t = s_i) \tag{4.2}$$

where $1 \le i, j \le N$, $a_{ij} \ge 0$ and $\sum_{j=1}^{N} a_{ij} = 1$.

The collection of the transition coefficients forms a *N*-by-*N* transition matrix A:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & \cdots & a_{NN} \end{bmatrix} \tag{4.3}$$

The outcome of the stochastic process of the first-order Markov chain is a sequence of observations $O = o_1, o_2, \ldots, o_T$, and they belong to a finite set of states $S = \{s_1,\ldots s_N\}$. In other words, if the outcome at time *t* is $o_t = s_i$, then at this time *t*, the model is in state s_i. Given the transition matrix A and initial start distribution $\boldsymbol{\pi} = \{\pi_1,\ldots, \pi_N\}$, the observation sequence can be precisely defined. To fix ideas of Markov chain, an example is shown below.

Example 4.1: We assume there are N balls with different colors (discrete values), s_1=red, s_2=green, s_3=blue,…,s_N=white, in one box. The physical process for obtaining the observation sequence is as follow. At time *t*, one ball is chosen randomly, and the color is recorded as observation. Then the ball is replaced back into the box, and another random selection will be performed to get the observation of next time *t*+1. We notice that the states are the different colors, and are observable. Given λ=(A, π), the observation sequence $O = o_1, o_2, \ldots, o_T$ will be determined.

4.2 Hidden Markov Model

In Section 4.1 we introduced the first-order Markov chain, where the states are observable. However many complex problems cannot be modeled by this simple model. Extension needs to be made where the observation is not states but the probabilistic function of the states, which gives the *Hidden Markov Model* (HMM). Therefore HMM is a doubly stochastic process: one is the stochastic process producing the sequence of observation; the other is the stochastic process describing the state evolution [24]. To set ideas of HMM, another example extended from example 4.1 is given.

Example 4.2: Same as example 4.1, there are numbers of different colored balls, and the number of distinct colors is U. Instead of putting them into one box, we separate them into N boxes randomly. Thus in each box there are numbers of different colored balls. The physical process for obtaining the observation sequence is as follow. First an initial box is chosen. From this box, one ball is chosen randomly, and the color is recorded as the observation. Then the ball is replaced back into the box. In the next round, a new box is selected based on the stochastic process describing the state evolution. In this new box, stochastic process selecting the ball is repeated. After T times processes, a finite observation sequence $O = o_1, o_2, \ldots, o_T$ will be generated as the output of HMM, and the hidden states correspond to the boxes. The selection of new boxes is determined by A, the state transition matrix of HMM; and the selection of the balls in one box (state) is determined by B, the emission probability matrix of HMM.

4.2.1 Elements and Terminology of HMM

An HMM is specified by the following:

- N, the number of states in the HMM;
- $\pi_i = P(x_1=s_i)$, the prior probability of state s_i being the first state of a state sequence. The collection of π_i forms the vector $\boldsymbol{\pi} = \{\pi_1, \ldots, \pi_N\}$;
- $a_{ij} = P(x_{t+1}=s_j | x_t=s_i)$, the transition coefficients gives the probability of going from state s_i immediately to state s_j. The collection of a_{ij} forms the transition matrix A. If any state can be reached directly from any other states, we call this model the *ergodic* HMM. If in the model the interconnection from any state only goes to itself and its followers, it's called *left-right* HMM.
- Emission probability of a certain observation o, when the model is in state s_i. The observation o can be either discrete (example 4.2) or continuous [25]:
  1. Discrete observations $o \in \{v_1, \ldots, v_U\}$, where U is the number of distinct observation symbols per state. The emission probability having the form $b_{i,u} = P(o_t = v_u \mid x_t = s_i)$ $(1 \le i \le N$ and $1 \le u \le U)$ indicates the probability to observe v_u if the current state is $x_t = s_i$. The collection of $b_{i,u}$ is a U-by-N emission probability matrix B.
  2. Continuous observations $o \in \Re^D . b_i = p(o_t \mid x_t = s_i)$ indicates the probability density function (pdf) over the observation space for the model being in state s_i. The collection of b_i is called emission pdf.

4.2.2 Three Essential Problems of an HMM

In the above section, we introduced the elements contained in an HMM. Having an HMM, there are three essential problems need to be solved:

- Evaluation Problem: Given an HMM λ=(A, B, π) and an observation sequence $O = o_1, o_2, \ldots, o_T$, compute the probability of the O generated by λ, $P(O|\lambda)$=? This problem will be solved in the *recognition* step to find out the most suitable model for the given O.
- Optimal State sequence Problem: Given an HMM λ=(A, B, π) and an observation sequence $O = o_1, o_2, \ldots, o_T$, find out the most likely state sequence for generating the observations.
- Estimation Problem: Given an observation sequence $O = o_1, o_2, \ldots, o_T$, train the model parameters A, B, π to get maximize $P(O|\lambda)$. This problem will be solved in the *training* (learning) step to get the optimal model for training set.

In this subsection we will focus on solving these three essential problems for HMMs having discrete observations, which is also called *Discrete-Density HMM* (DDHMM).

Solution for Evaluation Problem

The procedure of solving this problem is just like the procedure of using an HMM. Given λ and O, we compute the probability $P(O|\lambda)$ for all the HMMs we have, then we can find out which is the most likely model for generating observation O.

Given λ, O and a state sequence X, the naive way to calculate the probability of the observation sequence generated by model λ is to sum the probability over all possible state sequence in a model for the observation sequence, i.e. to marginalize of the hidden states X. According to Bayes rules, the probability can also be expressed by the sum of two terms, shown as follows:

$$P(O|\lambda) = \sum_s P(O, X|\lambda) = \sum_s P(O|X, \lambda) P(X|\lambda) \tag{4.4}$$

The two terms in the right-hand side of (4.4) have the forms below:

$$P(O|X, \lambda) = b_{x_1}(o_1) b_{x_2}(o_2) \cdots b_{x_T}(o_T) \tag{4.5}$$

$$P(X|\lambda) = \pi_{x_1} a_{x_1, x_2} \tag{4.6}$$

therefore we can rewrite (4.4) as:

$$P(O|\lambda) = \sum_s \pi_{x_1} b_{x_1}(o_1) a_{x_1, x_2} b_{x_2}(o_2) \cdots a_{x_{T-1}, x_T} b_{x_T}(o_T) \tag{4.7}$$

However (4.7) is computationally expensive. The alternative and more efficient solution is what we called *forward-backward* algorithm, which reduces computational complexity. Given the HMM λ, the forward variables are defined as the joint

probability of having seen the partial observation (o_1, $o_{2,...}$, o_t) and being in state s_i at time t:

$$\alpha_t(i) = P(o_1 o_2 \cdots o_t, x_t = s_i | \lambda) \tag{4.8}$$

where i in $\alpha_t(i)$ stands s_i for simplification, hereinafter we will keep the representation.

Then the probability $P(O|\lambda)$ we are looking for becomes as follow:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{4.9}$$

The detailed derivation of (4.9) is given in appendix A1.

For solving the evaluation problem, the forward recursion is enough. However we'd like to introduce the backward recursion as well for the preparation of solving problem 2 and 3. Given the model λ and the model is in state s_i at time t, the backward variables are defined as the probability of having seen the partial observations from time t+1 until the end:

$$\beta_t(i) = P(o_{t+1} o_{t+2} \cdots o_T | x_t = s_i, \lambda) \tag{4.10}$$

Solution for Optimal State Sequence Problem

The second problem is also called decoding. It is to decode (discover) the hidden components of the model in order to find out the best state sequence for the given observation sequence. Since the different definition of optimal state sequence, there are different optimal criteria and methods. The most common method is the *Viterbi* algorithm [16].

Viterbi algorithm can be viewed as a special form of the *forward-backward* algorithm where only the maximum path at each time is taken into account instead of all paths. The definition of $\delta_t(i)$ is based on the definition of forward variable, but with some constraint. It's defined as the probability of having seen the partial observation (o_1, $o_{2,...}$, o_t) and being in state s_i at time t by the most likely path, which means the $\delta_t(i)$ is the highest likelihood of the a single path among all the paths ending in state s_i at time t.

$$\delta_t(i) = \max_{x_1 x_2 \cdots x_{t-1}} P(o_1 o_2 \cdots o_t, x_1 x_2 \cdots x_t = s_i | \lambda) \tag{4.11}$$

In addition, a variable $\psi_t(i)$ is defined to keep track of the best path ending in the state s_i at time t:

$$\psi_t(i) = \arg\max_{x_1 x_2 \cdots x_{t-1}} P(o_1 o_2 \cdots o_t, x_1 x_2 \cdots x_t = s_i | \lambda) \tag{4.12}$$

The steps of *Viterbi* algorithm are similar to those of forward-backward algorithm (see appendix A1), except that instead of doing a sum of all possible sequences, the maximum is recorded at each step. By doing this optimization, the computational

expense is reduced, and the most likely state sequence can be discovered by backtracking:

$$X^* = \{x_1^* x_2^* \cdots x_T^*\} \text{ and } x_t^* = \psi_{t+1}(x_{t+1}^*) \quad t = T\text{-}1, T\text{-}2, \ldots, 1 \tag{4.13}$$

For detailed explanations of *Viterbi* algorithm, see appendix A2.

Solution for Estimation Problem

The solving of estimation problem, in other words, is the design/training procedure of an HMM. For discrete observation HMM, first the number of states N and the number of distinct observation symbols per state K should be chosen, and then normally a randomly generated model $\lambda=\{A, B, \pi\}$ should be specified. During the training procedure, the model parameters will be adjusted to maximize the $P(O|\lambda)$.

Since the different reqirements of applications, there is no one specific method to solve all kinds of optimization problems. It means several methods which focus on different optimization criteria are available. Maximum Likelihood (ML) and Maximum Mutual Information (MMI) are the two main criteria. Here we will only describe the ML method which is most commonly used in HMM training. However there is not analytically solution for adjusting the model parameters $\lambda=\{A, B, \pi\}$ to get the global maximum, $P(O|\lambda)$ can only be locally maximized by some iteration procedures, such as Baum-Welch algorithm. The Baum-Welch algorithm is a special case of the EM algorithm (Expectation and Maximization). For the explanation of EM algorithm, see [27].

Before decribing the Baum-Welch algorithm, two varibles should be defined, which will be used together with forward and backward variables. First, let's define the probability of model being in state s_i at time t, and then going to state s_j at next time period, given O and λ as follows:

$$\xi_t(i,j) = P(x_t = s_i, x_{t+1} = s_j | O, \lambda) = \frac{P(x_t = s_i, x_{t+1} = s_j, O|\lambda)}{P(O|\lambda)} \tag{4.14}$$

From the definition of forward and backward variables, (4.14) can be expressed in terms of α and β:

$$\xi_t(i,j) = \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)} \tag{4.15}$$

Secondly, let's define the probability of model being in state s_i at time t, given O and λ as follows:

$$\gamma_t(i) = P(x_t = s_i | O, \lambda) \tag{4.16}$$

Similarly, (4.16) can be rewritten in terms of α and β:

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N}\alpha_t(i)\beta_t(i)} \tag{4.17}$$

Now we are ready to introduce the Baum-Welch algorithm. Suppose we have used the last step of (A.1.3) and (A.1.8) to calculate the recursion of forward and backward variables, and have used (4.15) and (4.17) to calculate the two new variables. Then we can derivate the following equations (4.18a) to (4.18c) as re-estimation formulas:

$$\hat{\pi}_i = \gamma_1(i) \tag{4.18a}$$

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1}\xi_t(i,j)}{\sum_{t=1}^{T-1}\gamma_t(i)} \tag{4.18b}$$

$$\hat{b}_j(o_t) = \frac{\sum_{t=1}^{T-1}\gamma_t(j)}{\sum_{t=1}^{T-1}\gamma_t(i)} \tag{4.18c}$$

1) From the definition of the second new variable (4.16) we know that when t=1 it gives the expected number of times of state s_i being the first state of a state sequence, therefore $\gamma_t(i)|_{t+1}$ is used to express π_i. 2) The summation of $\gamma_t(i)$ over t, which is from t=1 to t=T-1, gives the expected number of trainsitions from state s_i. The summation of $\xi_t(i,j)$ over t, which also excludes t=T, gives the expected number of trainsitions from state s_i to state s_j. Further more, we notice that the expected number of trainsitions from state s_i to state s_j out of the expected number of trainsitions from state s_i gives the forward variables. 3) the numerator of (4.18c) gives the expected number of times being in state s_j and observing v_u. Moreover, the denominator gives the expected number of times being in state s_j. The dividsion between them gives the emission probability of model being in state s_j and observing v_u.

After the re-estimation, if the new model paramters are more likely than the old ones, which means $P(O|\lambda_{new}) > P(O|\lambda_{old})$, then the new model will replace the old one. The iteration (re-estimation) will continue until some limiting point is reached or the maximum iterations are fulfilled.

4.2.3 Types of HMM

In this subsection, some types of HMM will be introduced. More emphasis will be put into *Continuous-Density HMM* (CDHMM).

Ergodic and left-eight HMMs

As we have already mentioned in subsection 4.2.1, we can divide the type of HMMs into ergodic and left-right HMMs. The ergodic HMMs are fully connected HMMs. All the elements in the state transition matrix A have non-zero values. It means any state can be reached by any other state (including itself) directly. The left-right HMMs have different definitions according to the interconnections amongst states. The general definition is that any state is only connected to itself and its followers. If the connections are only allowed to itself and its immediate follower, then the transition matrix A of this model has only non-zero values on diagonal and one subdiagonal. A transition matrix of the above model having 4 states is shown as follow:

$$A = \begin{bmatrix} a_{11} & a_{12} & 0 & 0 \\ 0 & a_{22} & a_{23} & 0 \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{bmatrix}$$

It is obvious that for the last state of the left-right HMMs, the transition coefficients are $a_{NN}=1$ and $a_{Ni}=0 \quad (i<N)$.

DDHMM and CDHMM

The previous discussion about HMM and its three problems is limited into discret observation HMM. The features from speech signal are quantized by a vector-quantization (VQ) procedure, such as K-means algorithm. The VQ procedure aims at partitioning the acoustic space into nonoverlapping regions, and each region is representated by one *codeword* w_k. The collection of these K *codewords* makes the *codebook* $\{w_k\}$. Therefore in the DDHMM cases, the $b_j(o_t)$ is approximated by $b_j(w_k)$ where w_k is the *codeword* closest to o_t. The quality of the *codebook* is measured in terms of distortion, which has the form as follow:

$$\text{distortion} = \sum_{k=1}^{K} \sum_{t=1}^{T} (y_t - \mu_k)^2 \qquad (4.19)$$

where K is the number of *codewords*, μ_k is the centroid of the k'th *codeword* and y_t is the feature vector. The distortion depends not only on the training samples, but the number of *codeword* as well. Here the author would like to emphasis that the generation of *codebooks* is done before training and recognition processes, and they are stored for later use.

Sometimes when delta, or delta-delta coefficients (3.4.4) are used, multiple *codebooks*

will be generated (one for the basic cepstrum parameters *c*, one for Δc coefficients, and one for $\Delta\Delta c$ coefficients). The probabilities for all *codebooks* are calcultaed by simply doing the multiplication of the probabilities of each *codebook* with the independent assumption of codevbooks [28].

For CDHMM, the observations are continuous, and can be expressed by continuous pdf, where the pdf is governed by its parameters. The most commonly used pdf is Gaussian mixture density function, and the whole mixture density is:

$$b_j(o_t) = \prod_{r=1}^{R}\left[\sum_{m=1}^{M} C_{jm}\aleph(o_t;\mu_{jm},\Sigma_{jm})\right] \tag{4.20}$$

where R is the number of *codebooks*;
M is the number of Gaussian mixture components;
μ_{jm} is the mean vectors of the m'th Gaussian pdf;
Σ_{jm} is the covariance matrix of the m'th Gaussian pdf.
C_{jm} is the weighting coefficients for m'th Gaussian pdf, and satisfy the stochastic constrains: $C_{jm} \geq 0$, and $\sum_{m=1}^{M} C_{jm} = 1 \quad (1 \leq j \leq N, 1 \leq m \leq M)$

For each Gaussian component, the density is:

$$\aleph(o_t;\mu_{jm},\Sigma_{jm}) = \frac{1}{\sqrt{(2\pi)^d|\Sigma|}}\exp\left\{-\frac{1}{2}(o_t-\mu_{jm})^{\tau}\Sigma_{jm}^{-1}(o_t-\mu_{jm})\right\} \tag{4.21}$$

where d is the dimensionality of observation *o*.

Compared with DDHMM, in the CDHMM systems, no *codebooks* need to be generated and stored in advance. However during training and recognition, the probability for each observation should be calculated using (4.20). Therefore conclusions can be made that DDHMM needs more memory for storing *codebooks*, whereas it spends less computation time than CDHMM [28].

In Matsui and Furui's work [9], comparisons amongst CDHMM, DDHMM and VQ for text-independent speaker recognition have been made. It shows that an ergodic CDHMM is superior to an ergodic DDHMM. They also showed that the information on transitions between different states is ineffective for text-independent speaker recognition. Moreover the speaker recognition rates using a continuous ergodic HMM are strongly correlated with the total number of mixtures irrespective of the number of states.

Chapter 5 Speaker Pruning

In Chapter 4, we introduced the HMMs for *speaker modeling*, and we notice that for each speaker in the database, there is a corresponding HMM. Learned from Fig. 1.3, the Basic Structure of *Speaker Identification* (Identification Phase), assuming there are M speakers in the database, *Speaker Identification* has to perform M pattern matching between the unknown speaker and M known speakers. With a large number of speakers in the database, the performance of speaker recognition will decrease. Moreover since HMM is a doubly stochastic process, this models are too flexible and hard to train. As a result the high recognition accuracy is hard to be achieved with a large number of speaker models to compare with. *Speaker pruning* appears to be one solution to increase the identification accuracy with the cost of increasing a little bit recognition time.

See Fig. 1.3 again, the *speaker pruning* performs before the pattern matching to reduce amount of speaker models in the matching process, and those pruned speakers are the ones who are most dissimilar with the unknown speaker. By doing this, we notice the pattern matching between the unknown speaker's feature vectors and all the speakers in the database is reduced to the matching between the unknown feature vectors and the 'survived' candidates after pruning.

The simple algorithm KNN is used here for *speaker pruning*. In this chapter we first introduce the theory of KNN, and then its application as *speaker pruning* will be given.

5.1 K-Nearest Neighbor

K-Nearest Neighbor is a kind of non-parametric algorithm []. KNN stores all the given data examples $\{fv_i, l_i\}$, where fv_i denotes the feature vectors in our pruning system and l_i denotes the class label. It uses these examples to estimate l_{new} for the new example fv_{new}. The l_{new} is assigned to the class having the largest representatives amongst the K nearest examples, which fv_{new} are similar to. Here we use N_K to denote the number of nearest neighbors to distinguish the number of *codewords* in one codebook *K* defined in subsection 4.2.3.

Fig. 5.1 presents the KNN algorithm in the case of 5 nearest neighbors, and the procedure of KNN is as follows:

- Instead of building a model, all the training examples $\{fv_i, l_i\}$are stored;
- Calculate the *similarity* between the new example fv_{new} and all the examples in the training set fv_i;
- Determine the K-nearest examples to fv_{new};
- Assign l_{new} to the class that most of the K-nearest examples belong to.

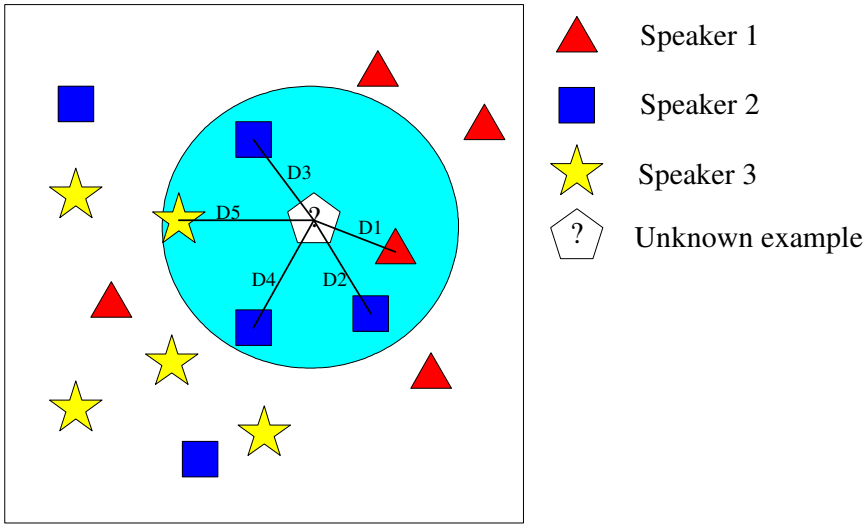


Fig. 5.1 KNN algorithm with N_K =5

First, all the examples from training speakers are stored. Red triangles stand Speaker 1, blue squares stand Speaker 2, and yellow stars stand Speaker 3. The pentagon denotes an example from an unknown speaker. Secondly, calculate the *Euclidean distances* between the unknown example and all the examples in the training set $d_E = \{d_{E1},\ldots,d_{EN}\}$. Thirdly, sort the distances and find out the N_K=5 nearest neighbors $D_1,\ldots,D_5$, which are included in the light blue circle. At last, assign unknown example to the class that most of the 5 examples belong to, which is blue square (Speaker 2) in this case.

Commonly the *similarity* between examples refers to the *Euclidean distance*, and the *Euclidean distance* between example vectors M = (m_1,m_2,…,mj) and N = (n_1,n_2,…,nj) is defined as:

$$d_E(M,N) = \sqrt{\sum_{l=1}^{j}(m_l - n_l)^2} \tag{5.1}$$

where j is the dimension of vectors.

To get a better understanding of KNN, we give one simple example as follows.

Example 5.1

Suppose we know some grades of 6 people in different subjects and their ages, and know which of them are qualified for a contest. Then according to the limited information we have to decide one new students' qualification by KNN algorithm. The data is shown in table 5.1. Now we can define the training set as $\{fv_i, l_i\}$ $(1 \le i \le 6)$, where vector fv_i= {Age_i, $Math_i$, $Physics_i$, $Chemistry_i$}, and l_i is the conclusion, either qualified or not. The information of George forms the new example x_{new} = {27, 13, 11, 11}. The *Euclidean distances* between fv_{new} and all fv_i in the training set need to be calculated using (5.1). The calculation is shown beside Table 5.1. Following the procedure of KNN algorithm, we find out the N_K=3 nearest neighbors for George, which are Lisa, Jerry and Tom. Due to the majority voting, l_{new} of George is assigned to be **Yes**, which means George is qualified.

Table 5.1 Data for KNN algorithm

Name	**Age**	**Math**	**Physics**	**Chemistry**	**Qualified**	Euclidean distances from George
Alice	18	10	10	10	**Yes**	$[(27-18)^2 + (13-10)^2 + (11-10)^2 + (11-10)^2]^{½}$ = 9.59
Tom	25	7	8	9	No	$[(27-25)^2 + (13-7)^2 + (11-8)^2 + (11-9)^2]^{½}$ = 7.28 (**3rd**)
Jerry	22	9	10	11	**Yes**	$[(27-22)^2 + (13-9)^2 + (11-10)^2 + (11-11)^2]^{½}$ = 6.48 (**2nd**)
Homer	40	5	3	6	No	$[(27-40)^2 + (13-5)^2 + (11-3)^2 + (11-6)^2]^{½}$ = 19
Lisa	23	11	13	10	**Yes**	$[(27-23)^2 + (13-11)^2 + (11-13)^2 + (11-10)^2]^{½}$ = 5 (**1st**)
Bart	35	6	7	5	No	$[(27-35)^2 + (13-6)^2 + (11-7)^2 + (11-5)^2]^{½}$ = 14.53
George	27	13	11	11	?	

One thing needs to be emphasized here is the normalization of variables in the example vectors. It is not so obvious in Example 5.1, however in some other cases, the distance between neighbors may be dominated by some variables with large variance. (Notice here the variance of age is relatively larger than the other variables.) Thus, it's necessary to normalize each variable with its largest value before executing KNN to avoid the domination. The table with normalized variables for this example is shown in appendix B.

KNN is a simple algorithm and easy to implement, and by choosing higher value of N_K, the noise vulnerability in the training set will be reduced. Nevertheless, since KNN doesn't build any model and just stores all the training data, a lot of computer storage will be needed. Moreover sometimes the *Euclidean distance* is not so suitable for finding similar examples when there are irrelevant attributes in training set.

5.2 Speaker Pruning using KNN

The proposal of inventing and introducing speaker pruning in our *SRS* is to increase the recognition accuracy. Therefore it's different from the commonly defined speaker pruning [26], where the pruning process will be continuously performed until the unknown speaker ID is found out. Our pruning process only perform once to find out the similarity between unknown speaker and all the known speakers in the database; by choosing the most similar speakers, we eliminate the dissimilar speakers. The 'survived' speakers will then be modeled in the next step—HMM *speaker modeling* to find out the unknown speaker ID.

Before going to the pruning method, some issues should be kept in our mind during implementing KNN into speaker pruning algorithm:

- Which features will be used;
- How to calculate the matching score;
- Number of nearest neighbors
- Pruning criterion (how many speaker will be pruned)
- Time consumption

The number of speakers/candidates we will keep after pruning should be adjustable with the requirements of recognition system. If more speakers are kept, for HMM more pattern recognition has to be performed, which decreases the HMM recognition accuracy. On the other hand, by keeping more speakers we ensure the accuracy rate of having the true speaker in those kept ones. The trade-off accuracy between pruning and HMM can be solved by observing the total recognition accuracy of the system with different combinations to find out the desired number of 'survived' candidates.

By using KNN algorithm, the pruning matching score becomes the *Euclidean distances* between unknown speaker examples and all the examples in the training database. The features we are going to use as data points are MFCC. In Chapter 7, we will show the experimental verification of choosing MFCC. In order to improve the performance of KNN in speaker recognition, we invented a new method to introduce the pitch information into KNN algorithm. Here we briefly introduce the invented method, for details see Chapter 7.

First using pitch estimation technique introduced in subsection 3.4.5, we can find out the probability of the unknown speaker being a female. According to the gender probability, we set a parameter κ:

$$\kappa = (P_f - 50\%) \times 0.4 \quad P_f \in [0, \quad 100\%], \quad \kappa \in [-0.2, \quad 0.2] \tag{5.2}$$

where P_f is the probability of unknown speaker being a female speaker. The range of parameter κ is decided by the experimental experience.

Secondly we introduce the parameter into the *Euclidean distances* calculated by using MFCC as follows:

$$d_{new} = (1 \pm \kappa) \cdot d_{MFCC} \qquad 0 \le \alpha < 1 \tag{5.3}$$

d_{MFCC} is the *Euclidean distance* calculated by only using MFCC, and d_{new} is the distance after adding pitch influence. This method depends on the accuracy of pitch in separating genders. The (1- κ) factor will be multiplied with the *Euclidean distances* between the new speaker and female speakers in database, and for the rest male speakers the *Euclidean distances* will multiply a (1+ κ) factor. Therefore if the probability is less than 50%, which means speaker is more possible to be a male, κ becomes negative, and the distance between the new speaker and female speakers will be increased, vise versa. If the probability of being a female and male are equal, in this

case, pitch doesn't help and will not be taken into account.

The time issue in our pruning is critical since by introducing the speaker pruning step, we decrease a little bit the recognition speed. Therefore we should keep the time consumption as low as possible. The factors which have effect on time are training set size, test set size, feature dimensionality and the number of nearest neighbors N_K. However in the mean while we also expect to have higher accuracy after pruning. Here the accuracy refers to the probability of including the true speaker into the 'survived' candidates.

Chapter 6 Speech Database-ELSDSR

English Language Speech Database for Speaker Recognition

ELSDSR corpus of read speech has been designed to provide speech data for the development and evaluation of automatic speaker recognition system. ELSDSR corpus design was a joint effort of the faculty, Ph. D students and Master students from department of Informatics and Mathematical Modeling (IMM) at Technical University of Denmark (DTU). The speech language is English, and spoken by 20 Dane, one Icelander and one Canadian. Due to the usage of this database and some realistic factors, perfect or even correct pronunciation is not required and necessary for getting the specific and uniquely identifiable characteristics for individual.

6.1 Recording Condition and Equipment Setup

The recording work has been carried out in a chamber (room 133) in building 321, 1st floor at DTU. The chamber is an 8.82*11.8*3.05 m^3 (width*length*height) computer room. The recording is manipulated in, approximately, the middle of this chamber, with one microphone, one 70*120*70 cm^3 table in front of speakers. In order to deflect the reflection, two deflection boards with measure of 93*211.5*6 cm^3 were placed at tilted angles facing each other, and were infront of the table and speakers. For details see the setup drawing, drawing of the room and position of recording, etc., in appendix C2.

The equipment for recording work is MARANTZ PMD670 portable solid state recorder. PMD670 can record in a variety of compression algorithm, associated bit rate, file format, and recording type (channels recorded) parameters. It supports two kinds of recording format: compressed recording, which includes MP2 and MP3; uncompressed recording, which includes linear pulse code modulation (PCM). The recording type can be stereo, mono or digital, and the file can be recorded into .wav .bwf .mpg or .mp3 format according to user need. In this database, the voice messages are recorded into the most commonly used file type--.wav. The algorithm used is PCM. Since nearly all information in speech is in the range 200 Hz-8 kHz, and according to Nyquist theorem, the sampling frequency is chosen to be 16 kHz, and the bit rate is 16. Table 6.1 shows the initial setup for the recorder, for details see PMD670 user guide.

Table 6.1: Recorder Setup

Input	**Setup**								
	Auto Mark	**Pre Rec**	**Analog Out**	**MIC Atten**	**Repeat**	**ANC**	**EDL Play**	**Level Cont.**	**S. Skip**
MIC (MONO)	OFF	ON	OFF	20dB	OFF	FLAT	OFF	MANUAL	ON 20dB

6.2 Corpus Speaker Information

ELSDSR contains voice messages from 22 speakers: 10 female, 12 male, and the ages are covered from 24 to 63. Most of them are faculty and Ph. D students working at IMM, and 5 of them are Master students including 1 international Master student.

Due to the practical problem of uneven gender distribution at the experiment site, the average age of female subjects is higher than that of male, and around half of the female subjects are secretaries in IMM. The detailed information about the speakers ID, speakers' ages with average for each gender, and nationalities is included in appendix C1.

Actually even though the subjects of this database are from different countries and different places of one country, the dialect of speaking or reading English language in this database plays a very slim role for the purpose of speaker recognition, since the features which are interesting for this particular intention are language independent.

6.3 Corpus Text & Suggested Training/Test Subdivision

Part of the text, which is suggested as training subdivision, was made with the attempt to capture all the possible pronunciation of English language, which includes the vowels, consonants and diphthongs. And with the suggested training and test subdivision, seven paragraphs of text are constructed and collected for training, which includes 11 sentences; and fourth-four sentences for test (each speaker reads two of these sentences) from NOVA Home [32] were collected for test text. In a word, for the training set, 154 (7*22) utterances were recorded; and for test set, 44 (2*22) utterances were provided.

Table 6.2: Duration of reading training text and test text

No.	Male		Train(s)	Test(s)	Female		Train(s)	Test(s)
1		MASM	81.2	20.9		FAML	99.1	18.7
2		MCBR	68.4	13.1		FDHH	77.3	12.7
3		MFKC	91.6	15.8		FEAB	92.8	24.0
4		MKBP	69.9	15.8		FHRO	86.6	21.2
5		MLKH	76.8	14.7		FJAZ	79.2	18.0
6		MMLP	79.6	13.3		FMEL	76.3	18.2
7		MMNA	73.1	10.9		FMEV	99.1	24.1
8		MNHP	82.9	20.3		FSLJ	80.2	18.4
9		MOEW	88.0	23.4		FTEJ	102.9	15.8
10		MPRA	86.8	9.3		FUAN	89.5	25.1
11		MREM	79.1	21.8				
12		MTLS	66.2	14.05				

On average, the duration for reading the training data is: 78.6s for male; 88.3s for female; 83s for all. And the duration for reading test data, on average, is: 16.1s (male); 19.6s (female); 17.6s (for all). Table 6.2 shows the time spend on reading both training text and test text individually.

6.4 ELSDSR Directory and File Structure

The voice messages are organized according to the following hierarchy:

CORPUS := = ELSDSR
USAGE := = train | test
SPEAKER ID := = FXXX | MXXX |
where,
F or M indicates the speaker's gender;
XXX indicate the speakers' initials

Sentence ID := = XXXX_SM or XXXX_SrN
where,
XXXX indicate speaker ID;
S indicates training sentence, M indicates the alphabetic number of paragraphs in training text, which is from a to g;
Sr indicates test sentence N indicates sentences number in test text, from 1 to 44.
The associated documentation is located in the 'ELSDSR /DOC' directory:
where,
training text.pdf
test text.pdf
phonetic alphabet.pdf[6]
readme.pdf

[6] Phonetic alphabet.pdf shows the captured vowels, consonants and diphthongs in each paragraph of training data.

Chapter 7 Experiments and Results

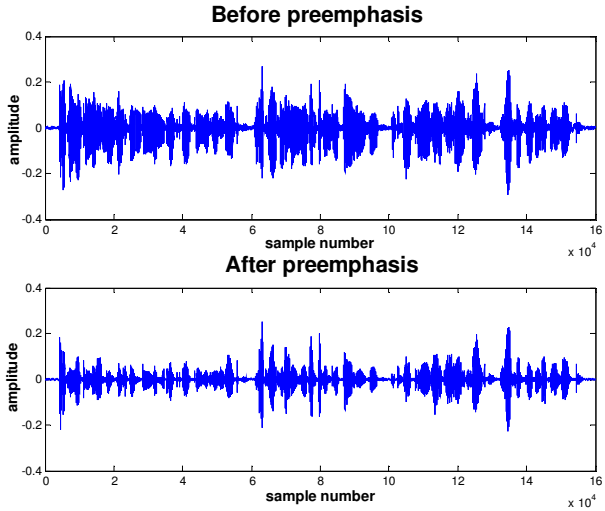


Fig. 7.1 Before and after preemphasis
Notice the envelope of the original signal (upper panel), which represents the low frequency, was removed.

In this chapter experiments will be presented with details. It is divided into 4 sections: preemphasis; feature extraction; speaker pruning; and speaker modeling and recognition. More effects have been put on feature extraction, since features is closely related with the performance of whole system. Moreover speaker pruning as an introduced step in our recognition system needs special attentions. For improving the pruning performance, a new method will be invented to make two features: pitch and MFCC working together in *Euclidean distance* calculation for KNN algorithm. Finally HMM will be implemented on the candidates after speaker pruning to fulfill the speaker recognition task.

7.1 Preemphasis

As we introduced in Section 3.3, speech signals should be processed by high-emphasis filter before performing feature extraction. The preemphasis is prevalent due to the well-known fact that higher frequency components contain more speaker-dependent information than lower frequency components [33]. Generally, the first order FIR filter will be used to execute the high pass filtering. However, here we used the Kaiser frequency filter to filter out the low frequency components, while keeping the high frequency components unchanged according to the magnitude response of this Kaiser filter, shown in Fig. 3.4.

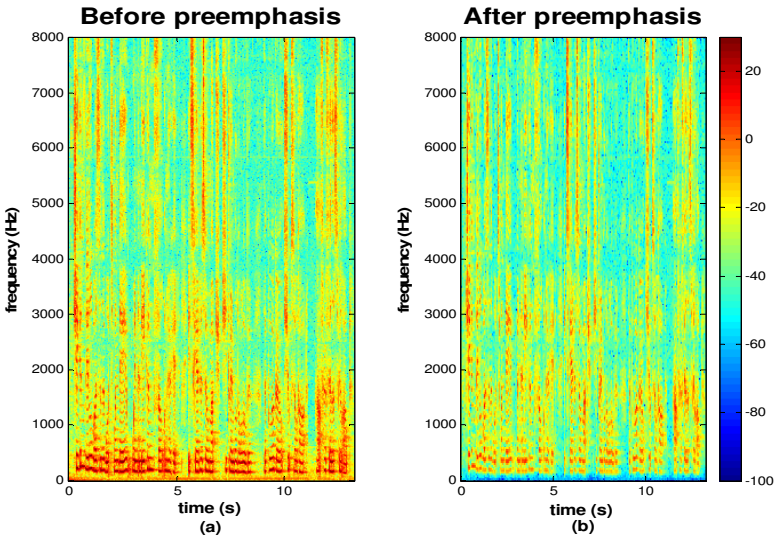


Fig. 7.2 Spectrogram before and after Preemphasis

(a) (b) show the spectrogram of the same utterance FAML_Sa.wav. (a) gives the spectrogram of the original signal; whereas (b) shows the spectrogram of the preemphasized signal. The two spectrograms are scaled equally.

By performing the preemphasis, we filtered the low frequency components. Hence the damped higher frequency components, which is -6db slope downward, can be modeled equally as the lower frequency part. The signal before and after preemphasis is shown in Fig. 7.1, where we see that the envelope of the original signal, which represents the low frequency, was removed.

Fig. 7.2 gives the spectrograms of a speech signal from the ELSDSR database before and after performing preemphasis. In order to see the difference, the two spectrograms are scaled equally. Let's focus our attention on the lower frequency part of (a) and (b), we notice the magnitude for the low frequency components in (b) was decreased compared with that in (a). However the magnitude for higher frequency components stays almost unchanged.

7.2 Feature Extraction

In Chapter 3, we introduced three types of features: Cepstral Coefficients (and LP based), Mel-frequency Cepstral Coefficients and Fundamental Frequency (Pitch). As we know Pitch doesn't work well alone for classifying a large group of people. Thus our experiments were focused on comparing LPCC and MFCC for extracting speaker-specific features. After selecting the desired feature, the dimension of features will be discussed and decided.

For comparison, two techniques were used: PCA and KNN. Mentioned in Section 3.4 that directly working with data in high (n) dimensional spaces arise problems due to the

curse of dimensionality. Generally, mapping the high (n) dimensional input data into lower (m) dimensionality is one improvement (n>m). However without exception, this reduction will discard some of the information. Thus the problem becomes to find a method for obtaining the m dimensions, which guarantees the projection on those directions contents as much the relevant information as possible. PCA appears to be a good method for this task [27].

7.2.1 Feature Selection

Comparison between LPCC and MFCC using PCA

Principal component analysis (PCA) is a mathematical procedure that transforms a number of correlated variables into a number of uncorrelated variables called principal components. This is performed by projecting the n-dimensional signal into its m largest principal components. The objective of PCA is to reduce the dimensionality of the data, while retaining most of the original variability in the data, which are the eigenvectors with the largest eigenvalues of the covariance matrix. The PCA transformation is expressed as follow [27]:

$$J = U_1^T I \tag{7.1}$$

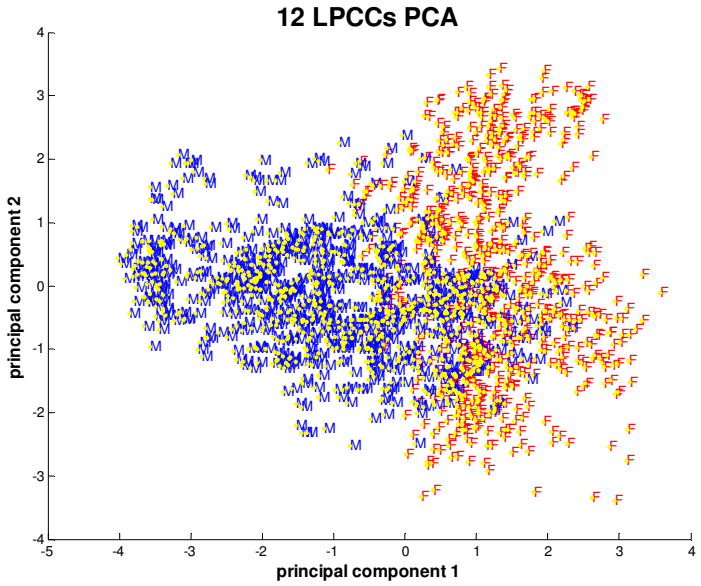
where I is the high dimensional input data, J is the low dimensional data, and U_1 is the projection matrix containing the m eigenvectors (principal components) corresponding to the largest eigenvalues.

A way to find the principal components U_1 is by decomposing the (n x n) covariance matrix G of signal I into its eigenvectors and eigenvalues as follows:

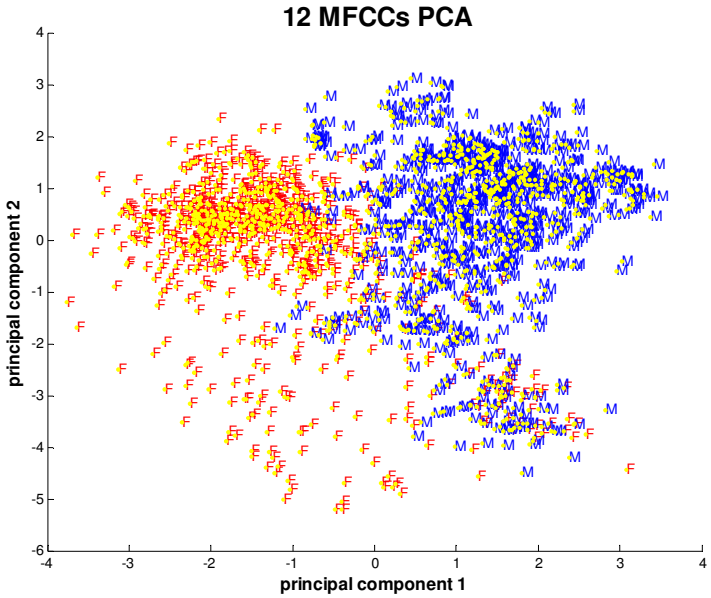
$$G = U \Lambda U^T \tag{7.2}$$

where U is a (n x n) column-orthogonal matrix which contains eigenvectors corresponding to the eigenvalues in the (n x n) diagonal matrix Λ, U_1 in (7.1) contains m columns (eigenvectors) with m largest eigenvalues. The decomposition in (7.2) is a special case of Singular Value Decomposition (SVD) [36], where G is a (r x c) matrix, and r is generally not necessary to be the same as c.

We have illustrated in Fig. 3.9 that 12 MFCC gives better separation between two speakers from TIMIT database than 12 LPCC does in text-dependent case (two speaker saying same text). More experiments were done with our own database. Fig 7.3 shows the experimental results with ELSDSR. Same as before, 12 MFCC and 12 LPCC were compared in speaker separation task with text-dependent constrain. The signal pieces from FAML_Sa.wav and MASM_Sa.wav were both 0.5s. In this case, we can say that MFCC separate speakers better than LPCC.



(a)



(b)

Fig. 7.3 LPCC vs. MFCC for speaker separation using PCA

Fig. 7.3 shows an example of PCA. Speech messages of FAML_Sa.wav and MASM_Sa.wav from ELSDSR database were used. We extracted the first 0.5s (800 samples @ 16 kHz) of both signals. (a) shows 12 LP based cepstral coefficients projected into the largest two principal components; whereas (b) shows 12 MFCCs projection. F denotes FAML, and M denotes MASM. In both cases frame size=320, overlap=240.

Comparison between LPCC and MFCC using KNN

From the above experiments we have the impression that MFCC work a little bit better than LPCC for separating two speakers' speech messages. In order to fix the conclusion, more experiments for comparing MFCC and LPCC in different situations need to be done. KNN algorithm introduced in Chapter 5 was implemented in the following experiments for classification.

Before going to the comparison experiments, the *front-end processing* of speech signals should be presented:

1. According to Section 3.2, the sub-processes of the *front-end processing*, preemphasis should be implemented first to original signals with Kaiser Frequency filter.
2. Then following the sequences of *short-term* analysis, signals were blocked into frames of 320 samples, which is 20ms @ 16 kHz, and the consecutive frames were spaced 80 samples (75% overlap) apart. For the sake of avoiding truncation of signals, we used Hamming window to multiply each frame. It's hard to say whether Hanning or Hamming window is better than the other. The choice depends on the applications. Here the most popular window function in speaker recognition field was chosen. In order to make it consistent, we used Hamming window for both MFCC and LPCC.
3. After performing the above two steps, the frames undergo the feature extraction process using MFCC and LPCC techniques individually. As for LPCC, 12 LP coefficients were calculated [13], and converted into Q lowest cepstral coefficient using (3.14). To make these two features comparable, Q lowest MFCC were calculated (excluding the 0'th coefficient since it corresponds to energy of the frame). The value of Q is related to the length of the raised sine lifter function as shown in Fig. 3.5. To include some dynamic information of speech signal, Q delta-LPCC and Q delta-MFCC were also calculated, and concatenated with the Q lowest LPCC and MFCC separately. Hence the dimension of both LPCC and MFCC features are $2Q*N_F$, where N_F is the total number of frames.

Brief introduction about the experiment setup in our implementation of KNN algorithm is necessary. For this comparison task, a binary KNN was implemented, which means that the training set and test set only include two different speakers' speech signals. The assigning for new example becomes a binary decision: the new example either belongs to class 1 or class 2. To avoid bias for majority voting algorithms, we stored the equal amount of speech information for those two speakers, which was about 4 seconds (6400 samples @ 16 kHz) for each speaker for training, and 3 seconds signals were used for testing. Moreover, we named the training examples input1, and the test examples input2, correspondingly the class labels for training and test examples are Label 1 and Label 2.

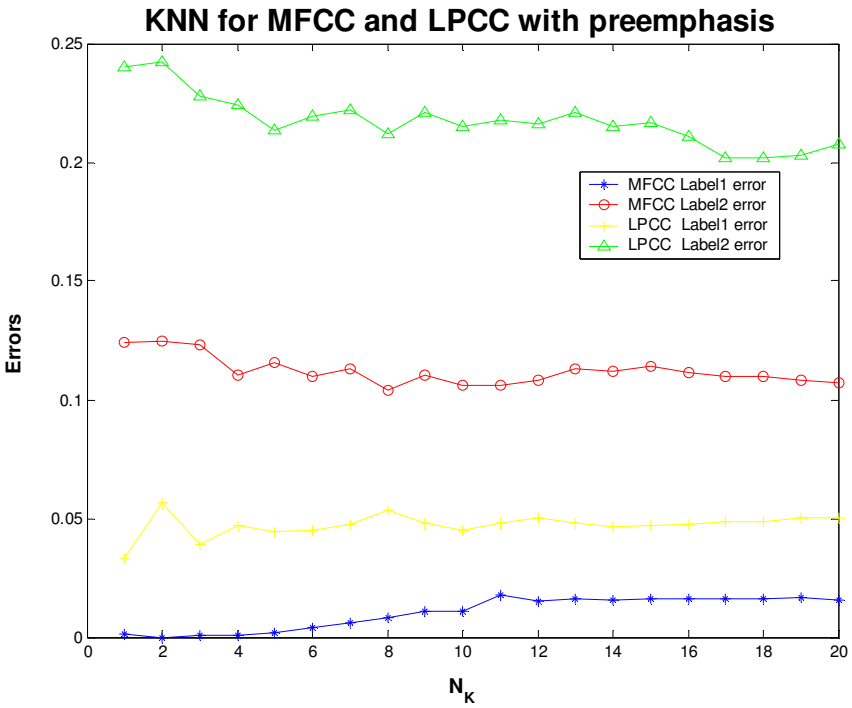


Fig. 7.4 LPCC vs. MFCC using KNN

24 MFCC and 24 LPCC features were extracted from FJAZ_Sa, MKBP_Sb, FJAZ_Sc and MKBP_Sd. They were used respectively for binary KNN algorithm in text-independent case. 20 iterations were done with N_K changing from 1 to 20. Red (o) and blue (*) curves gave the Label1 and Label2 (test) errors from MFCC. It shows MFCC can achieve smaller errors, and when N_K=8 minimum test error 0.1039 was achieved, and corresponding Label1 error was around 0.0082.

Now we will show the data source and construction for training and test set. First a piece of signal with 4s was cut from FJAZ_Sa.wav (FJAZ stands the speaker ID, Sa stands the sentence ID), and another 4s signal was cut from MKBP_Sb.wav. Afterwards the features extracted from these two signals were concatenated together to work as training examples $Fv=\{fv_1, fv_2, ..., fv_{NF}\}$. N_F is the total number of frames, which depends on the number of samples (2*6400), the frame size (320), and the overlap (240); and fv_i is a 2*Q dimensional vector. Following the 'almost standard' in speaker recognition field [29], Q was set to 12. Next we need to find out the labels for training data. For each example vector we gave one label, thus there are totally N_F labels for our training data, and the first half are class1's label, the rest are for class2: $L=\{l_1, l_2, ..., l_{NF}\}$. Now we got the training data set $\{fv_i, l_i\}$. For the test set, 3s signal from FJAZ_Sc.wav and 3s signal from MKBP_Sd.wav were concatenated, and then extracted into 24 dimensional features, the collection of them gives the new examples waiting for assigning labels: $Fv_{new}=\{ fv_{new1}, fv_{new2}, ..., fv_{newM}\}$. Notice this experiment was done in text-independent situation. For text-dependent case, see appendix D1.

Fig. 7.4 gives the Lable1 and Label2 errors of using MFCC and LPCC for classification. The Lable1 error means that we vote the label for each example from training set

without itself included, and finally compare the new voting labels with true labels to get the training error. It shows with the same dimensional features that using MFCC can achieve lower errors than using LPCC for KNN classification.

Conclusion: Using PCA and KNN, we got the same conclusion that MFCC perform better than LPCC in speaker separation or recognition. The following experiments are meant to find out the desired dimension of MFCC features.

7.2.2 Feature Dimensionality

From the previous experiments we choose the MFCC as features for our speaker recognition system since it can achieve smaller test errors on speaker classification. Now let's decide the value of Q which determines the dimensionality of features. First we compare Q=12 with Q=24 using PCA. (Q=24 is also commonly used in MFCC.) Then we will use multi-KNN algorithm to do more detailed experiments. Finally Q iterations in KNN will be implemented to find out the optimal Q and confirm our previous experimental results.

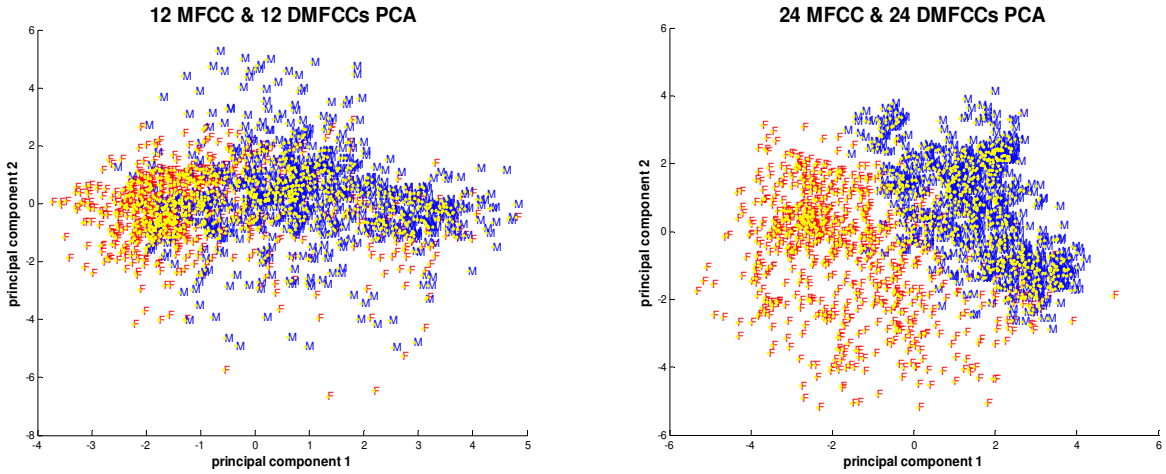
PCA

First the PCA was performed with 24 dimensional MFCC features (Q=12) and 48 dimensional MFCC features (Q=24) including the delta coefficients. The results are given in Fig. 7.5. Same as before, signal pieces lasting 0.5s were cut from ELSDSR database: FAML_Sa.wav and MASM_Sb.wav. This time the experiment was done in text-independent situation. Using PCA technique, we got the conclusion that 48 dimensional MFCC gave better data separation.

Multi-KNN

Now we use the multi-KNN algorithm to test the superiority of 48 dimensional MFCC. The training and test sets for the multi-KNN case include 6 speakers' speech messages: FAML_Sa, FDHH_Sb, FEAB_Sc, MASM_Sd, MCBR_Se and MFKC_Sf. For training set 3s signals were cut from each of these 6 messages, and for test set 2s signals were cut. Hence we got 3*6=18s signal for training, and 2*6=12s signal for testing. Same as before, 20 iterations were fulfilled with the number of nearest neighbors N_K=1 to 20. The results are shown in Fig. 7.6. For the multi-KNN case, the number of neighbors doesn't give big influence on test errors. The blue and red curves represent the 48 MFCC[7] case. 24 MFCC gave 0.4916 smallest test error, whereas 48 MFCC gave 0.3455 smallest test error, and the improvement of test error was around 29.7% w.r.t. that of 24 MFCC. We should emphasize that from now on all speech signals are preemphasized before doing feature extraction.

[7] From now on we will use delta MFCC together with MFCC, for simplification 24 MFCC means Q=12: 12 MFCC and 12 DMFCC. Hereinafter the number xx in xx MFCC gives the dimension of coefficient instead of the Q value, unless special denotation is given.



(a) (b)

Fig. 7.5 24 MFCC vs. 48 MFCC for speaker separation using PCA

(a) shows the speech data separation between one female and one male with 24 MFCC (b) gives the result with 48 MFCC. Both of them were in text-independent situation. F denotes FAML, and M denotes MASM. In both cases frame size=320, overlap=240.

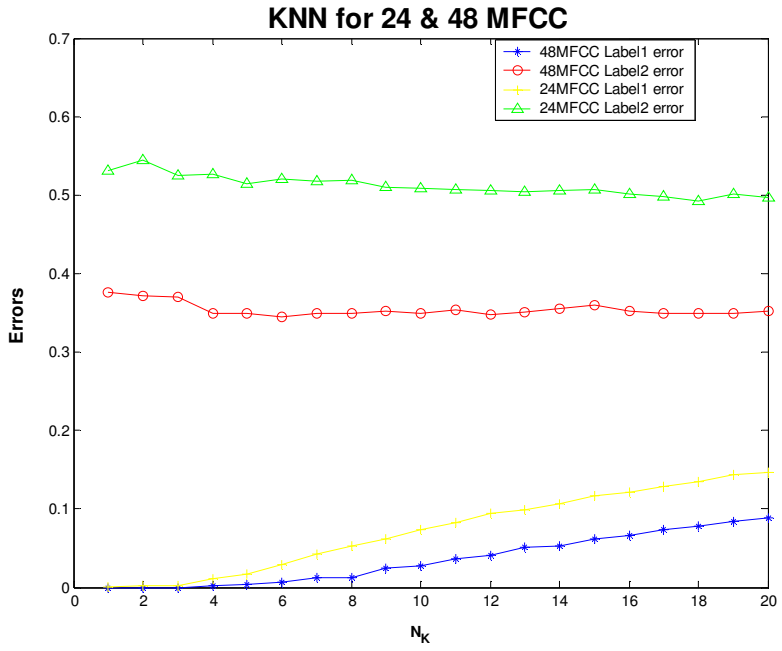


Fig. 7.6 24 MFCC vs. 48 MFCC using KNN

Figure shows the experiment results with multi-KNN algorithm using 24 MFCC and 48 MFCC respectively in text-independent case. 20 iterations were done with N_K=1:20. Red (o) and blue (*) curves gave the Label1 and Label2 (test) errors from 48 MFCC. The errors from 48 MFCC are much smaller than those from 24 MFCC, and the smallest test error was E_{Label2}=0.3455, and corresponding E_{Label1}=0.0070 when N_K=6.

- Q Iteration

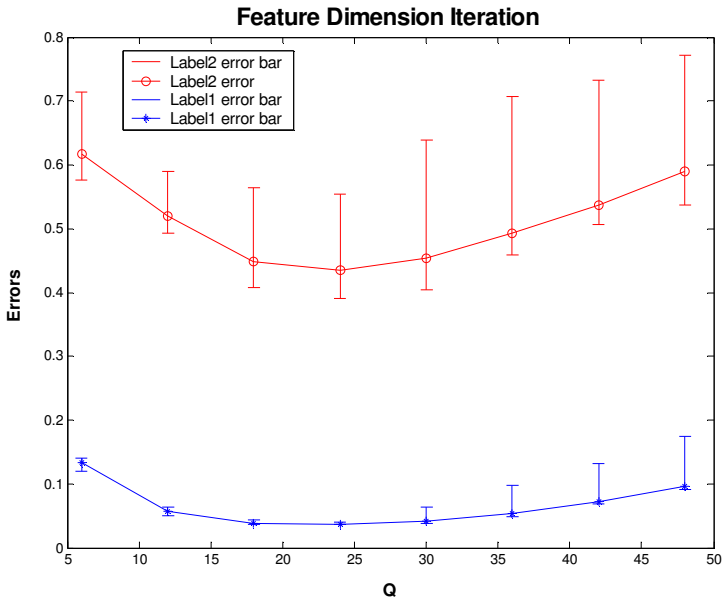


Fig. 7.7 Q iterations for searching optimal Q

Experimental signals were from 6 speakers' speech messages: FAML_Sa, FDHH_Sb, FEAB_Sc, MASM_Sd, MCBR_Se and MFKC_Sf. Different size of training set were used, test set was always 2*6s, and N_K=8. From the shape of the curve and the error bar, we found out the desired feature dimension is 48, which gave lower test error.

From the above experiments we can loosely say that 48 MFCC gives better results with smaller errors. In order to fix the conclusion Q iterations were used to illustrate the result again by using KNN algorithm. We worked on multi-KNN with the same 6 speakers as in previous experiments. The training signals from each speaker were varying from 1s to 5s, and 2s for testing. As for the number of neighbors, we choose N_K=8. Seen from Fig. 7.6, the number of neighbors doesn't give a big influence in multi-speaker cases. Since we are working in multi-KNN case, the even number of neighbors won't give bad effects. However in binary cases, using odd number of neighbors gives more precise assignments for new examples, because the majority voting is easy to get a draw when even examples are voted.

The result is given in Fig. 7.7. Observing the shape of this curve and the error bar, we see the tendency of test errors with the change of Q. It shows obviously that minimum test error was found when Q=24, and the test errors are not linearly proportional to the dimensionality of features, and models using 48 MFCC offer best recognition results.

Conclusion: Using both PCA and multi-KNN, the 48 MFCC always perform better than 24 MFCC. The Q iteration confirms the results again, and moreover shows 48 is the most desired dimension with MFCC features in speaker classification.

7.2.3 Recognition Error Improvement for KNN

Seen from Fig. 7.6, the minimum test error rate we got from this 6-speaker KNN case was around 35%, and for randomly guessing, the error rate is up to 83%. Even though KNN gives much better recognition results, the error rate is still not good enough in practice. One way to improve the recognition accuracy is to use more information from test signals for majority voting instead of only using information from one window (frame). In the previous experiments, we labeled the training data frame by frame, and used them to get the labels for new (test) signal frame by frame, too. Afterwards we perform majority voting and finally recognize the new signal as the most appearing label. Now we are still using the same way to get the frame by frame labels for new signal. However instead of performing majority voting directly to the frame by frame labels, we do the internal voting first in a small group of examples with N_G members to get a voted label for the group. After constructing the groups and getting the labels for them, we could use the grouped labels for the majority voting to recognize the speaker with the most appearing label. By this means, we increase the accuracy of voted label, since the information containing in one group is richer than one example, we expect the recognition errors to be decreased.

Since we have found out that 48MFCC outperform the 24MFCC for both binary and multi- speaker recognition, in the next few experiments we will use 48MFCC. Seen from Fig. 7.8, with N_G=10 in one group and half overlap N_{over}=5 between groups in the same multi-KNN situation as before (N_P=6), the minimum test error occurred when N_K= 4, 5, 10, and the value was 0.2586. From the experiment shown in Fig.7.6, the smallest test error from 48MFCC was 0.3455. By comparison we notice the improvement of recognition accuracy was around 13.3%. The confusion matrix of the estimation for N_K=4 is shown as follows:

Confusion Matrix =

		Estimated speakers					
		P1	P2	P3	P4	P5	P6
True Speaker	P1	49	4	3	**0**	**1**	**21**
	P2	11	62	1	**0**	**1**	**3**
	P3	10	7	41	**1**	**0**	**19**
	P4	**0**	**1**	**0**	63	8	6
	P5	**1**	**0**	**0**	5	64	8
	P6	**2**	**7**	**1**	0	0	68

Rows represent the true speakers FAML, FDHH, FEAB, MASM, MCBR and MFKC, and columns represent the estimated speakers. Notice the more voting labels lay on the diagonal of this matrix, the more accurate the recognition is.

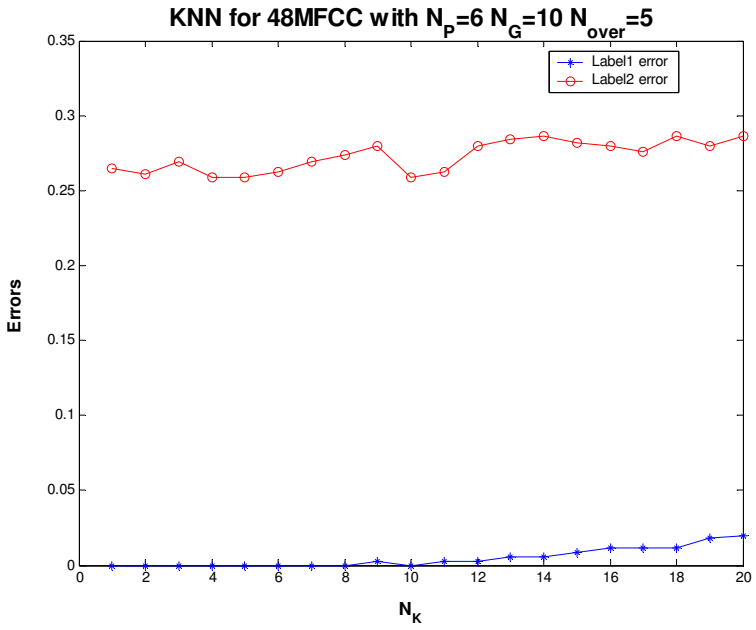


Fig. 7.8 Recognition accuracy improvement

N_K iteration was performed with 48MFCC in the same 6-speaker multi-KNN situation as before. In order to improve the voting accuracy, 10 former examples were grouped together with 5 examples overlaps in groups. The curve of test errors lies between 0.25 and 0.3, however from Fig. 7.6 the test errors for 48MFCC belong to the range [0.3, 0.4]. The minimum test error before improvement was 0.3455 (N_K =6), and now it becomes 0.2586 (N_K=4,5,10). The decrease of test error is more than 25%.

7.2.4 Combining *fundamental frequency* Information with MFCC

In this project, one challenge is to add *fundamental frequency* information into MFCC features in the application of KNN algorithm. Notice from the confusion matrix given in Subsection 7.2.3, the misclassification of labels between female and male decreased the recognition accuracy, especially for female speakers 1 and 3: 21 and 19 labels should belong to P1 and P3 respectively were misrecognized as the third male speaker. Moreover as we introduced in subsection 3.4.3, the *fundamental frequency* information is efficient to classify genders. Hence we propose to use both features: MFCC and *fundamental frequency* in KNN with the purpose of eliminating or decreasing the misrecognition between female and male speakers.

However the combination task is not as easy as we thought. First, MFCC have to follow the short-term analysis. Thus the frame size of the speech signal should be around 20ms-40ms in order to keep the speech signal pseudo-stationary in the framed period. Whereas since *fundamental frequency* (F_0) of human voice is at a much lower frequency than the formants (F_1, $F_{2,...}$), we should frame the signal into larger chunks, for extracting *fundamental frequency* information. Therefore the F_0 extracted from each

longer frame cannot just be added to the MFCC from shorter frames. Secondly, F_0 does not exist in every frame, e.g. the unvoiced parts, such as letter s, and silence parts. Therefore if we extract F_0 frame by frame, it will give us many zeros, which don't help for recognition at all.

In this subsection, we will solve the combination problem in different ways. First of all *fundamental frequency* estimation method in our project will be introduced and experimented, then comes the solutions for the combination.

***Fundamental Frequency* Estimation**

As we briefly introduced in subsection 3.4.5, there are many pitch extraction techniques fitting for different situations, e.g. noise environment, noiseless environment, etc. Since we are working with the 'pure' speech signals, Cepstrum method (CEP) technique was used to fulfill the pitch estimation.

Due to the low frequency location of *fundamental frequency*, we first blocked the speech signal into 64ms frames with half overlap (32ms), i.e. 1024 and 512 samples @16 kHz, and for each frame, real cepstral coefficients were calculated. The cepstrum turns the pitch into a pulse train with a period that equals the pitch period of the speaker [30]. The low frequency location of F_0 usually means the region [50Hz, 400Hz], which covers the F_0 of most men and women [21]. Therefore for each frame we only need to search the pitch in the range [2.5ms, 20ms] in time domain, corresponding to [40, 320] samples. Since there are unvoiced phonemes and silence in speech signals, we only calculated the F_0 with significant peak in the range of each frame, see Fig. 7.9. By averaging the F_0 from frames, we found out the median pitches for speakers. To get more information, we also included maximum and minimum pitch values for each speaker. In Fig.3.11, the fundamental frequencies for eight speakers from TIMIT database have been already given. Now the experiment has been done with our ELSDSR. All the suggested training data have been used to extract the F_0 of these 22 speakers, see Fig 7.10. The first 10 speakers are female, and the sequence follows Table 6.2 from FAML to MTLS. For our database the female F_0 belong to [150Hz, 250Hz], and the male F_0 lie in [90Hz, 177Hz]. Notice, it's hard to recognize the speaker ID only based on F_0. However from the distribution of fundamental frequencies of female and male, it's reliable to separate genders using pitches.

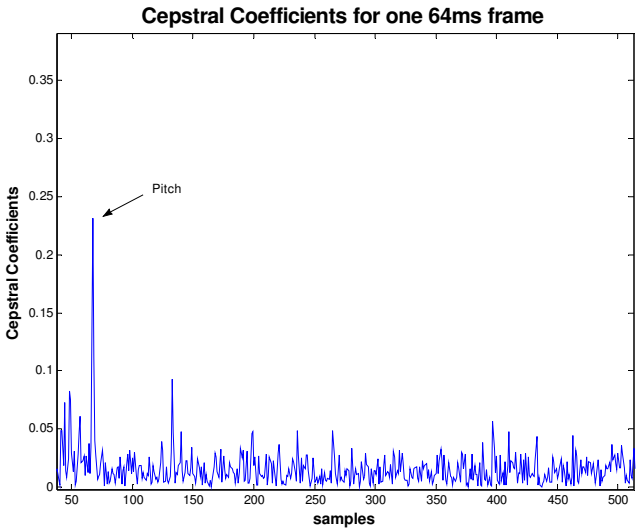


Fig. 7.9 Cepstral coefficients

This figure shows a part of cepstral coefficients in one 64ms frame with significant peak. We only need to search the peaks from samples No.40 to No.320 out of 1024 samples. The pointed pulse gives the position of pitch in this frame.

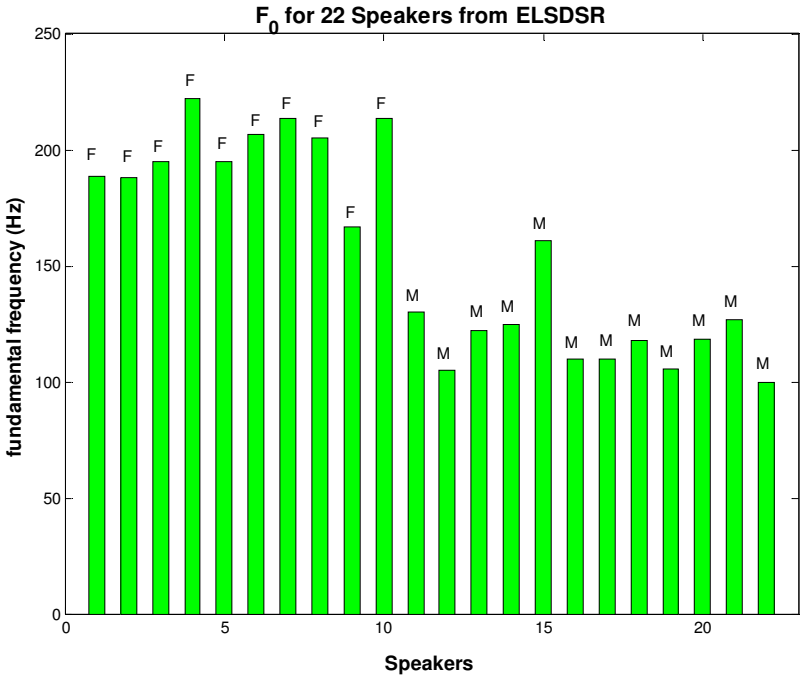


Fig. 7.10 F_0 information for 22 speakers from ELSDSR

Notice the female F_0 are comparably higher than male: female F_0 belong to [150Hz, 250Hz], and male F_0 lie in [90Hz, 177Hz].

Combining MFCC and *Fundamental Frequency*

- Static weight parameter

In order to find out a suitable way to use both MFCC and pitch features in KNN algorithm (speaker pruning technique for our system), many methods have been experimented. As mentioned before, the frame by frame combination of MFCC and pitch cannot include the correct pitch information because of the property of pitch location. Subsequently instead of trying to combine two features, we solved the combining problem from another point of view that is to modify the similarity calculation in KNN algorithm directly.

Method one:
One way to modify the similarity calculation in multi-KNN is to calculate the *Euclidean distances* by using MFCC and *Fundamental Frequency* separately. The *Euclidean distance* using only MFCC features d_{MFCC} can be computed using (5.1). Whereas the *Euclidean distance* using pitch d_{pitch} is just the simple subtraction between test signal's pitch and the pitches of 22 speakers in the database. By combining the MFCC and F_0, a weight parameter on pitch distance is introduced:

$$d_{Enew} = \sqrt{d_{MFCC}^2 + \delta \cdot d_{pitch}^2} \tag{7.3}$$

where δ is the weight for pitch information. Therefore we will then find the K-nearest neighbors with the new distance, d_{Enew}.

Since pitch is only supposed to give some help in separating genders, and it cannot recognize speakers, a weight is used in (7.3) to avoid the domination of pitch information over MFCC. However this method does not work well, since δ is a constant here, and it is also data-dependent, and have to be adjusted with new input data. No suitable way has been found out to adjust the weight since the test signal is unpredictable. If δ is not suitable to data, the recognition achieved by MFCC will be distorted by pitch information, which gives even worse recognition.

Method two:
Another similar method is also to introduce a static parameter κ. This parameter depends on the pitch detection result, but the difference is only the sign. We multiply $(1-\kappa)$ to *Euclidean distances* between the unknown speaker's examples and the examples of all female speakers in the database; and multiply $(1+\kappa)$ to the male *Euclidean distances*:

$$d_{new} = (1 \pm \kappa) \cdot d_{MFCC} \qquad -1 \le \kappa < 1 \tag{7.4}$$

First pitch detection of the test signal is performed to find out the gender of the speaker. Then according to the gender of the unknown speaker, we decide the sign of parameter κ. For instance, if the gender of unknown speaker is female, then κ is decided to be a positive value.

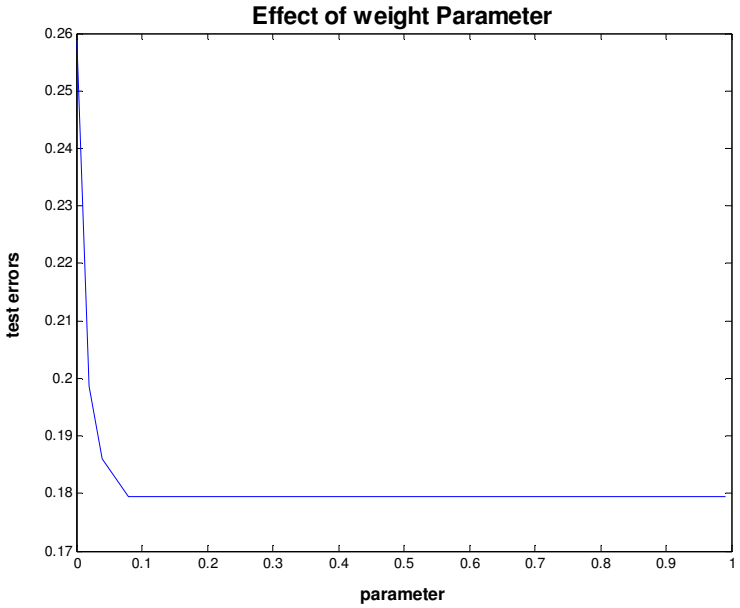


Fig. 7.11 Effect of weight parameter κ on test errors

The test errors decreased with the increasing of parameter κ. It proved that this method accomplished adding the pitch information into MFCC to decrease the recognition errors. In this case the decrease is around 31% while κ >0.1.

Consequently the d_{MFCC} between this speaker and all the female speakers in the database will be decreased to some degree, which increases the percentage of female speakers in K-nearest neighbors.

Fig. 7.11 gives the effect of using different κ. In order to get a suitable value of this parameter, we need to see its effect on test errors. The condition of this experiment was: N_K=4, N_G=10, N_{over}=5. Here we used a female test speaker. When κ=0, no parameter was used, and the test error is equal to what we got from Fig.7.8 when N_K=4, i.e. 0.2586. After κ was increased larger than 0.1, the error became stable and unchanged, i.e. 0.1795, and the improvement was around 31% compared with the case in Fig.7.8. Therefore we can choose $|\kappa|$=0.1 or 0.2.

However since the parameter is static, it cannot be suitable for all different cases. The worst case is brought by the wrong gender recognition. Since the test signal is quite short, the estimated pitch from the limited speech message may not be as reliable as pitch from long signals. Hence mistakes in gender recognition exist. In this case, the distance between unknown speaker and the true gender speakers in the database will be unfortunately increased, which decreases the speaker recognition accuracy. Therefore we found out a new method based on method two, where the weight parameter is adaptive according to the precision of pitch detection.

- Adaptive weight parameter

The new method has been briefly introduced in section 5.2. Instead of using a static weight parameter as method two, we use an adaptive parameter also called κ, which depends on the precision of pitch detection:

$$\kappa = (P_f - 50\%) \times 0.4 \quad P_f \in [0, \quad 100\%], \kappa \in [-0.2, \quad 0.2] \tag{7.5}$$

where P_f is the probability of unknown speaker being a female. That is if P_f>>50%, the weight parameter κ will get a large positive value; if P_f<50%, κ is negative; and if P_f = 50%, the unknown speaker has equal possibility to be female and male, then in this case, pitch doesn't help and will not be taken into account. The range of this weight parameter κ is determined by observing the effect of using different values on test errors, see Fig. 7.11. Since we only used a female test speaker, κ only has positive value, for both female and male speakers, the range of this parameter should be symmetric. A safety boundary is chosen to be ±0.2.

According to the accuracy of pitch in recognizing the genders, we can use (7.4) to modify this distance. The same as Method two, the (1- κ) factor is multiplied to the *Euclidean distances* between the unknown speaker and female speakers in the database; and for the rest male speakers the *Euclidean distances* are multiplied by a (1+ κ) factor.

To find out the gender probability of the new speaker, we calculated the distance between the pitch (median pitch) from test signal and the mean pitch of female speakers and male speakers. 22 F_0 from ELSDSR were used, and signals from the suggested test set were used. First we block the test signal into frames, and F_0 from each frame will be calculated, and then the median value of these pitches was found as the pitch of the new signal. By calculating the distance from the new pitch to mean female pitch, and to mean male pitch, we can find out the probability of being a female speaker P_f:

$$P_f = d_M / (d_M + d_F) \tag{7.6}$$

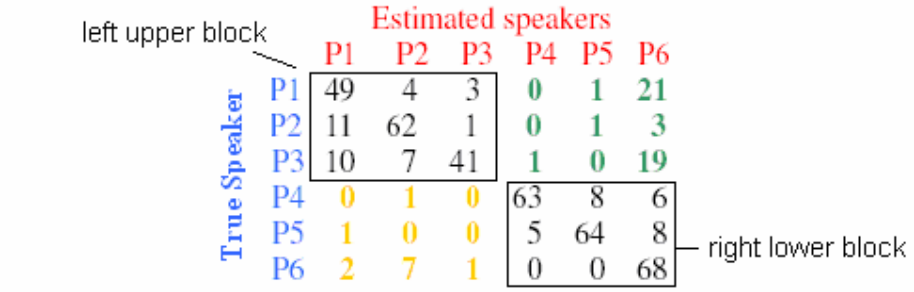
where d_M is the distance from new pitch to mean male pitch, and similarly d_F is the distance from new pitch to mean female pitch. Mean male pitch and mean female pitch were based on the speakers' pitches in our database. Our experiment proved the reliability of pitch in separating genders. The pitch accuracy for gender recognition is shown in appendix D2. We notice it can achieve high accuracy: by using 1s test signals, the accuracies were above 69% for 22 speakers in ELSDSR; and using 2s signals, the lowest accuracy was 85%, but for most of the speakers 100% accuracies were achieved.

Observing the confusion matrix, we can see the good recognition result after introducing adaptive weight parameter κ. The condition of this experiment was: N_K=4, N_G=10, N_{over}=5. Compared with the confusion matrix we got before, shown again for convenience, where the correct classification was 74.15%, the misclassification between genders all disappeared, and the diagonal of the matrix still kept the highest value of each row, which gave 82.05%. The improvement was around 10.7% w.r.t. the former classification accuracy.

Confusion Matrix (new) =

True Speaker \ Estimated speakers	P1	P2	P3	P4	P5	P6
P1	69	6	3	0	0	0
P2	15	62	1	0	0	0
P3	10	22	46	0	0	0
P4	0	0	0	64	9	5
P5	0	0	0	5	65	8
P6	0	0	0	0	0	78

Confusion Matrix =



As mentioned before method two devastates the recognition when pitch detection gives wrong gender recognition. However by introducing the adaptive weight parameter even though the gender recognition is wrong, the true speaker has still chance to be picked out in speaker pruning (details come in Section 7.3) as candidates waiting for HMM modeling and final recognition. Experiments have been done to compare the static and adaptive weight methods in the worst case. The training set includes 22 speakers speech message from ELSDSR, and test data was from one male speaker P15. The probabilities of the male speaker being each of these 22 speakers using both methods are shown in Table 7.1. Suppose that we pick out the first 6 speakers who have biggest probability to be the unknown speaker after speaker pruning. Hence when Static κ=0.1 the probability of unknown speaker being P15 (true speaker) is only 1.28%, and it's not in the first 6 candidates; when κ=0.2 the situation becomes even worse, the method two eliminates all the speakers who have different gender to the estimated gender from unknown speaker. Whereas using adaptive weight κ method, the P_f from pitch estimation was 57.07%, consequently κ became 0.0283, and the true speaker P15 was included into the first 6 candidates.

Table 7.1 Comparison between static weight and adaptive weight methods.

Probability	Static κ=0.1	Static κ=0.2	Adaptive κ=0.1
P1	0.1795[8]	0.1923	0.1667
P2	0.0769	0.0769	0.0769
P3	0.0769	0.0641	0.0513
P4	0.0641	0.0641	0.0256
P5	0.0641	0.0641	0.0256
P6	0.1154	0.1282	0.0897
P7	0.0256	0.0256	0.0256
P8	0.2436	0.2436	0.2436
P9	0.0641	0.0513	0.0513
P10	0.0769	0.0897	0.0769
P11	0	0	0
P12	0	0	0.0385
P13	0	0	0.0385
P14	0	0	0.0128
P15	0.0128	0	0.0641
P16	0	0	0
P17	0	0	0
P18	0	0	0
P19	0	0	0
P20	0	0	0
P21	0	0	0
P22	0	0	0.0128

It shows the probability of the unknown speaker (P15) being speaker P1 to P22 after KNN with static and adaptive weights respectively.

[8] The first 6 candidates who have the biggest probability being true speaker were given in red color.

7.3 Speaker Pruning

Speaker pruning in our *SRS* is designed to increase the recognition accuracy. KNN as a simple algorithm was chosen to be pruning technique in our system. As we introduced in Section 5.2, we should solve the listed issues for pruning algorithm: features selection; the matching score; number of nearest neighbors; pruning criterion; time consumption.

In the previous work, we proved that MFCC features are superior to LPCC in speaker recognition for KNN algorithm, and 48 MFCC gave better recognition than 24 MFCC (experiments have been done with both TIMIT and ELSDSR database.) However 24 MFCC and 48 MFCC will both be experimented and compared, with the intention of increasing speaker pruning speed, while keeping the pruning error rate relative low. More details come in subsection 7.3.2. In our speaker pruning, the matching score will be calculated using our invented method (7.4) with adaptive weight parameter. Obviously both MFCC and pitch features are needed here. For reducing the error rate, the recognition accuracy improvement method introduced in subsection 7.2.3 will also be used. By saying the pruning criterion, we mean the number of speaker (Ns) we are going to select for the later speaker modeling and recognition by HMM. These speakers should be the most likely ones to the unknown speaker. The pruning criterion depends on the accuracy of KNN for speaker recognition: if the accuracy is not so high, then more speakers should be retained.

7.3.1 Training Set Size

First of all, let's decide the size of training and test sets. Learned from appendix D2, gender recognition using pitch extracted from 2s speech signal can achieve almost 100% accuracy for most of the speakers. Hence the test signal is chosen to be 2s long. Remember the time consumption is also critical in our pruning task. Therefore it makes the size of training set becoming an important issue. As mentioned before, KNN algorithm does not build any models; instead, all the training data have to be retained. The more training data we have the more memory we need, and the more distances need to be calculated, consequently the slower the pruning will be. On the other hand, the less training data, the lower recognition accuracy. The trade-off [27] between achieving lower error rate, and consuming less time is an important point in our pruning task.

For the purpose of finding the desired size of training set, we first set the other parameters as constants: the number of nearest neighbor N_K=10; the number of group members in accuracy improved KNN N_G=10, overlap amongst groups N_{over}=5; the number of retained speakers after pruning N_S=4; and the feature dimensionality is 48. By using different size of training data set, we calculated the recognition time consumption, while at the same time the test error of recognition.

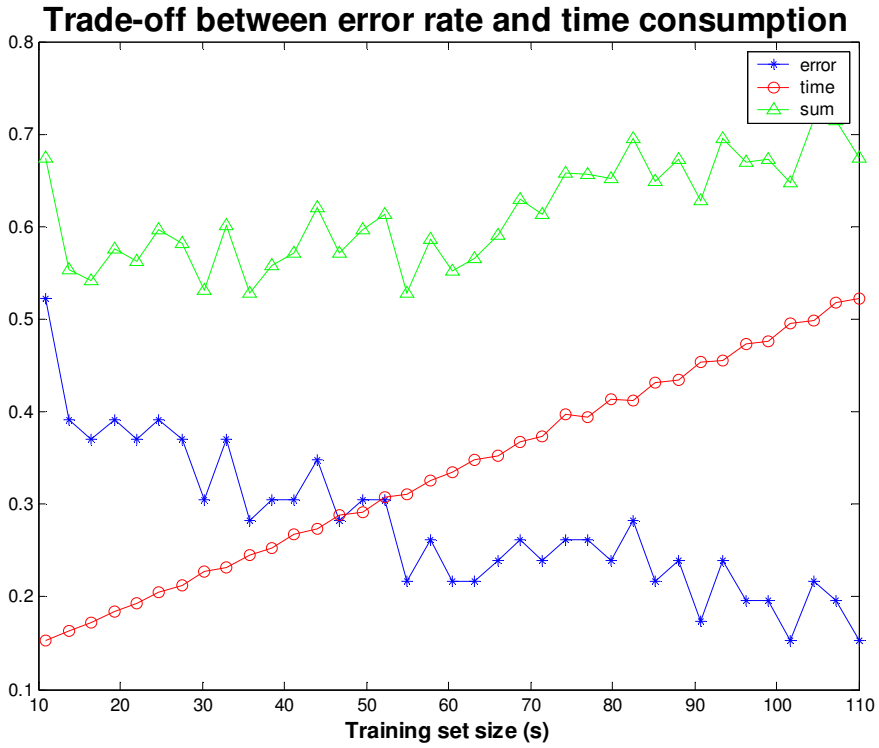


Fig. 7.12 Searching desired training set size

This figure shows the trade-off between the time and error rate, and the desired size for training set is appeared at the lowest point in the sum curve of these two factors, which is 55s. Here the normalization of time and error is necessary.

Here the error is defined as the probability of failing to include the true speaker into the N_S retained speakers. The normalization of the time and error is necessary since these two variables do not have the same scale and have totally different definition and property. The normalization is to make the range of both two factors the same; and consequently equal the influence of both factors on training set size. The normalization is as follow:

$$x_{aft} = \frac{x_{bef} - \min(x_{bef})}{\max(x_{bef}) - \min(x_{bef})} \tag{7.7}$$

where x_{bef} represents either time or error before normalization, and x_{aft} stands the values after normalization. Notice that the subtraction of $\min(x_{\text{bef}})$ in the nominator of (7.7) is optional, since it will not affect the shape of the sum in Fig. 7.12.

After normalizing the time and error rate, the trade-off between the time and error rate was found, see Fig. 7.12. The desired size for training set appeared at the lowest point in the sum curve of these two factors: the time consumption and the test error, which is 55s. It means for each speaker 2.5s (55/22=2.5) signal is desired for training.

7.3.2 Feature Dimensionality in Speaker Pruning

In the previous experiments, we found out the desired training set size for this pruning task, i.e. 55s. Next step is to find out the feature dimensionality. We know from the experiments we did before that 48MFCC is superior to 24 MFCC in speaker recognition. However since the most time consumption part is the distance calculation, the feature dimensionality becomes another important issue for speaker pruning. Comparison between 24 MFCC and 48 MFCC in pruning task will be made to find out the suitable feature dimensionality from the perspective of recognition accuracy and time consumption. Table 7.2 gives the time consumption and test errors of pruning task using 24 and 48 dimensional features respectively. Learned from the previous experiments, the training set was set to be 55s with 2.5s for each speaker, and test set was 2s. Four different training set were used. The other parameters were set to: N_K=10, N_G =10, N_{over}=5 and N_S=6.

Table 7.2 Time consumption and Test Errors using 24MFCC & 48MFCC

Training Set	48MFCC		24MFCC	
	Time (s)	Error	Time (s)	Error
1	19.6947	0.1522	11.1824	0.3478
2	19.9877	0.1522	10.9864	0.3261
3	20.2772	0.1087	11.0795	0.3261
4	19.8863	0.2074	10.8906	0.3043
Average	19.9600	0.1551	11.0347	0.3261

As we did in Fig. 7.12, we should find out the smallest sum of time and error. However since time and error have different scale and variance, directly summation doesn't follow the assumption of time and error having equal weight/influence on results. Therefore we will compare the percentage of time changing and error changing to decide the dimension of features:

Time: (19.96-11.0347) /19.96=44.72%

Error: (0.3261-0.1551) /0.1551=110.3%

Notice the time changing was 110.3%, whereas error changing was 52.44%. It means even though we saved a little bit time by using lower dimensional features, we lost much more recognition accuracy. Therefore it is better to choose 48 MFCC in our study. However the choice of feature dimension does really depend on the applications, whether time is more critical or error is.

7.3.3 Determining the Other Parameters

After determining the two most important issues in our speaker pruning, the other parameters, such as N_K, N_G and N_{over} need to be decided. Based on our experience from the experiments, the changing of the number of nearest neighbors, the number of group members and overlap are not critical to the time consumption. Thus we put our attention on the test error again to find out the optimal values.

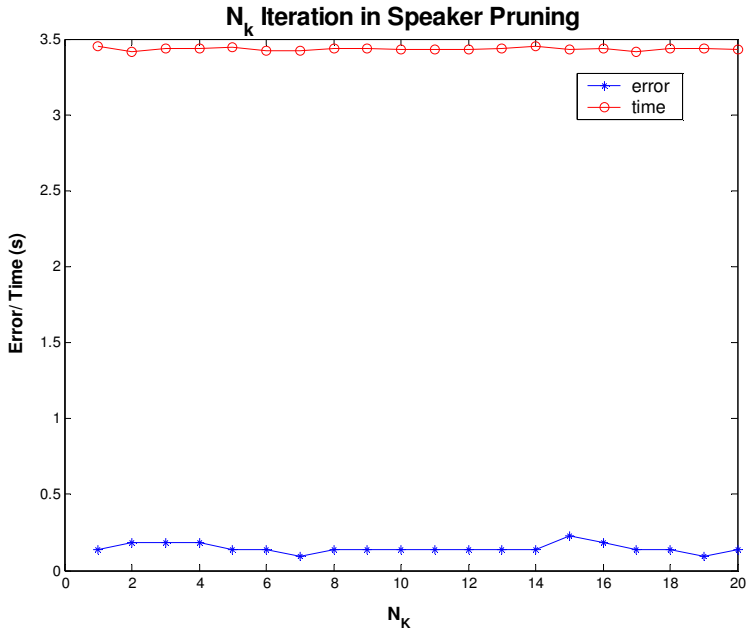


Fig. 7.13 N_K iteration for finding optimal N_K

This figure gives the changing of the errors with different nearest neighbors. N_K is from 1:20. When 7 neighbors were used, we got the minimum error, i.e. 0.0709. Here we proved that the number of N_K doesn't give a big influence on the time consumption, since the time consuming calculation in pruning/KNN is the distance computation. (the time here doesn't refer to total time consumption of speaker pruning, and it just shows the variance of using different N_K.)

The number of group members in accuracy improved KNN depends on the data very much, and it's hard to say which value is desired. The more members in one group, the more accurate the group label will be; but too many group members will cause the lack of data problem, hence decrease accuracy. From our experimental experience with different N_G and N_{over}, we can loosely say that when N_G=10 and N_{over}=5, the group label is accurate enough, while the data are also sufficient.

For finding optimal number of nearest neighbors, we do the N_K iterations as before. The setup of this experiment is following the results of previous experiments: N_G=10, N_{over}=5, training set size=55s, test set size=2s, feature dimension=48 and retained models N_S=6. Fig. 7.13 shows the results. Notice with the iteration of N_K the time consumption is very equable (expect the case with only one nearest neighbor). While N_K=7, the minimum error rate was found to be 7.09%.

In this section, we introduced speaker pruning technique into our recognition system. The important issues of speaker pruning are the size of training set and the dimension of features from the error and time consumption perspective. The optimal values for training set size and feature dimension are 55s and 48 respectively. Afterwards with the optimal values of these two issues, we decided the number of nearest neighbors N_K, the number of group members N_G and overlap between groups N_{over}.

In the next section, we will use HMM on the survived speakers to fulfill speaker modeling and recognition. According to the recognition accuracy of HMM within the survived speakers, we will adjust the pruning criterion.

7.4 Speaker Modeling and Recognition

Following the steps of enrollment phase in *SI*, shown in Fig. 1.5, after *front-end processing* the feature vectors will be processed by some speaker modeling methods to create speaker models. Here the statistically mature method, HMM, were used for speaker modeling. In detail, the Discrete-Density HMMs introduced in Chapter 4 were used. For DDHMM, the number of *codewords K*, the number of states *N*, and the model parameters $\lambda = (A, B, \pi)$ should be set before and during training/ learning procedure.

As for the identification phase in *SI*, it is slightly different from the steps shown in Fig. 1.3 due to the introduction of speaker pruning process. The speaker pruning should be done right after the *front-end processing*, and the candidates are picked out of all the speakers in the database. The feature vectors of those candidates will then be input to the DDHMM for pattern recognition, finally the decision will be made according to the maximum selection. Introduced in subsection 4.2.3, DDHMM needs more memory for storing *codebooks*, whereas spends less computation time than CDHMM [28]. Therefore in our system, DDHMM is chosen.

In the following subsections, we will give the experimental results on speaker modeling and speaker recognition respectively.

7.4.1 Speaker Modeling

Speaker modeling refers to training hidden Markov model for each speaker, and get a set of the model parameters. The speaker modeling follows the solution of the third essential problem introduced in subsection 4.2.2.

The first task is to derive codebooks. The training feature vectors from all the speakers in the database will be gathered together to create one or more codebooks. Normally only one codebook is derived from one training set, however since we introduced the delta coefficient, two codebooks were created: one is for the lowest 24 MFCC; the other is for the rest feature vectors [28]. The algorithm for vector quantization was chosen to be K-means. K-means is to partition the data points (feature vectors $\{fv_i\}$) into *K* disjointed subsets S_j containing numbers of data points in each subset. According to the commonly used version of K-means, random cluster centers have been selected from the data points, and then adjusted according to the space spreading of feature vectors and the subset members of each cluster. The adjustment of cluster centers is evolved in a way to minimize the sum-of-squares clustering function as follows:

$$J = \sum_{j=1}^{K} \sum_{n \in S_j} \left\| fv^n - \mu_j \right\|^2 \tag{7.8}$$

where fv^n represents the members in subset S_j, and μ_j is the center (mean) of the data points in subset S_j. After one evolution, each point is re-assigned to a new set according to the nearest mean vector, and the means of each subset are then recomputed. The evolution stops until there's no big change in the grouping of the data points [27].

K-means turns the continuous feature vectors into a sequence of discrete observation symbols which have lower dimensions than the original feature vectors. Those quantized vectors regarded as discrete observations will then be input into DDHMM. By adjusting the model parameters using Baum-Welch algorithm, we are trying to get the maximum likelihood $\lambda = \arg\max_{\lambda} P(O \mid \lambda)$ for observing the quantized feature vectors from each speaker. For each speaker there exists one HMM. The codebooks derived by the training set will also be used to quantize the test feature vectors.

Finding desired *N* and *K*

The following experiments have been done to find out the desired number of states and number of *codewords* (discrete observation per state) for speaker recognition and for our own database ELSDSR. For the K-means clustering algorithm, the number of centers (*K*) must be decided in advance [27]. Moreover for DDHMM, the number of states (*N*) should also be decided before executing training and testing. However we had no idea what the value of these two parameters should be for our recognition task and for our database. Therefore *K* and *N* were set to different values for observing their influence on the test errors for recognizing speakers.

The training set has been set to be 22s per speaker, i.e. 22*22=484s long signals. Before vector quantization, the *front-end processing* was performed with preemphasis and feature extraction described in 7.2.1. Subsequently, feature vectors were input into K-means algorithm, and then discrete observations were generated. Subsequently, the discrete observations were used to train the HM models for each speaker. The model parameters $\lambda = (A, B, \pi)$ were initialized randomly. Afterwards, the Baum-Welch *EM* reestimation algorithm was performed to update and devise the model parameters until the local maximum of the likelihood $P(O|\lambda)$ was reached. The adjustment or update of the model parameters utilizes equations (4.18) derived in subsection 4.2.2.

With the purpose of finding out the desired number of states *N* and number of *codewords K*, experiments with different combination of *N* and *K* have been done. For these experiments, 10 different test sets were used. All the test sets have the same amount of speech messages from Ns=6 candidates, i.e. 5s. Table 7.3 shows part of the results which are representative. Two codebooks were derived: first one was generated by quantizing the lowest 24 MFCC vectors into *K codewords*; the second one was from the 24 delta MFCC vectors converted into *K codewords* as well. Finally two sets of

likelihood $\{P_1(O|\lambda)\}$ and $\{P_2(O|\lambda)\}$ were calculated. Err1 represents the test error using only the first codebook to test the lowest 24 MFCC from test signals. Similarly Err2 stands the test error using the second codebook only. ERR gives the final test error using both codebooks. The final $P(O|\lambda)$ was calculated as follow:

$$P(O|\lambda_i) = P_1(O|\lambda_i) \cdot P_2(O|\lambda_i) \qquad 1 \le i \le 22$$

$$\text{or equivalently} \quad \log P(O|\lambda_i) = \log P_1(O|\lambda_i) + \log P_2(O|\lambda_i) \tag{7.9}$$

where i is the model sequence.

Take a look at Table 7.3, when *K* is smaller than 128, the errors (Err1, Err2 and ERR) from one state DDHMM were always smaller or at least not bigger than multi-state DDHMM. The result proves and accords with the summary of [9] and [23] about text-independent speaker recognition using DDHMM. The desired number of observations per state in this study is 32 with two codebooks, which gave 3.7% error rate. When *K* is too small, the *codewords* (clusters) are not enough for quantizing and separating 22 speakers speech messages; whereas when *K* is too big the data space was divided into too many clusters, and each cluster has few data points, as a result the model was easy to be over trained, which took the noise or speaker-independent information into account. The extreme case is that for each data point there is a cluster whose center is the position of data point.

Table 7.3 Test Errors from Codebook1, 2 separately and together with different *N* and *K*.

States	*K*=16			*K*=24			*K*=32		
	Err1	Err2	ERR	Err1	Err2	ERR	Err1	Err2	ERR
1	0.3704	0.2037	0.3704	0.0926	0.2407	0.0741	0.0556	0.1296	0.0370
2	0.3704	0.2963	0.4259	0.2778	0.7222	0.2963	0.3889	0.5000	0.3704
3	0.3704	0.6296	0.3704	0.3704	0.8148	0.3889	0.2407	0.7222	0.2778
4	0.3704	0.5185	0.3704	0.2407	0.7593	0.3148	0.0926	0.4815	0.1296
5	0.5185	0.6111	0.5000	0.1481	0.7037	0.2222	0.0926	0.5741	0.0926
States	*K*=64			*K*=72			*K*=128		
	Err1	Err2	ERR	Err1	Err2	ERR	Err1	Err2	ERR
1	0.2963	0.2037	0.2963	0.4259	0.0926	0.4259	0.4630	0.0370	0.4630
2	0.2963	0.4259	0.3519	0.5741	0.2037	0.5741	0.4630	0.1481	0.4444
3	0.3519	0.5556	0.3519	0.5000	0.6481	0.5370	0.4630	0.3148	0.4444
4	0.2963	0.6667	0.2963	0.4259	0.4444	0.4259	0.4630	0.2593	0.4444
5	0.3148	0.5185	0.3519	0.4444	0.3519	0.4444	0.4630	0.3333	0.4630

Note: the errors shown above were the average of 10 times experiments; Ns=6.

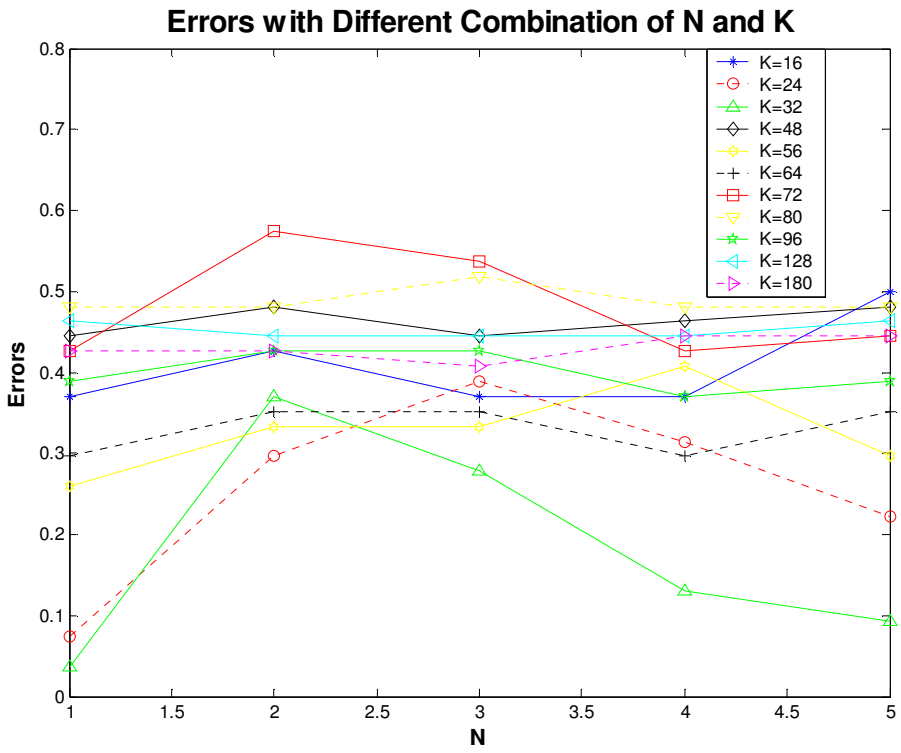


Fig. 7.14 Test errors with different combination of *N* and *K*

Notice in almost all the cases, *N*=1 gave lower test errors than multi-states cases; while for *K*=32 *N*=1, the test error got the lowest value. The errors shown in the figure were from both codebooks which correspond to ERR in table 7.3.

Since we used two codebooks in our system, we found out an interesting result happened when *K* is quite big. Take a look at the data in Table 7.3 while *K*=72 and 128, using the codebook from the first 24 feature vectors gave very low recognition accuracy, on the contrary using the second codebook generated by the first time derivative features gave much high accuracy while only one state was used. Therefore we can grossly say that with small *K*, using both codebook1 and 2 gives better recognition accuracy for one state DDHMM; whereas for large *K*, using only codebook2 gives better result for one state DDHMM.

Fig. 7.14 gives the test recognition errors using both codebooks with different combination of *K* and *N*. The states were from 1 to 5, and *K* was from 16 to160. From this experiment we proved again that one state models give higher accuracy than more state models, and 32 *codewords* for each codebook, in other words, 32 distinct observations per state gives the best recognition in this study. The desired number observations per state are different from database to database. For example, for TIMIT database with only one codebook for 48 dimensional features, the desired *K* was 128. However for the number of state, it was the same no matter whether TIMIT or ELSDSR was used, i.e. *N*=1 offered the best results.

7.4.2 Speaker Recognition

With desired N=1 and K=32, we can build 22 discrete density hidden Markov models for each known speaker. In this subsection we will show the recognition accuracy we achieved with different size of test set and different number of candidates survived from speaker pruning.

Table 7.4 provides the accuracy rate in different experiment setups. The test set for HMM recognition was changing from 2s to 6s with the same training set 22*22=484s (22s for each speaker). The speaker pruning was performed only on 2s test signal, since the distance calculation takes time and the training and test data points need amount of computer storage. The candidates for further DDHMM recognition were from 4 to 8. Notice the more speaker retained, the lower HMM recognition accuracy becomes, whereas the more test data used, the higher HMM recognition accuracy achieved. With Ns=4 and 6s test set, the HMM recognition reached 100%. However since few candidates survived from speaker pruning, the accuracy of including the true speaker into the candidates becomes lower. Therefore we should not only pay attention on the individual recognition accuracy, but the total accuracy as well. It was calculated as follows:

$$\text{Accuracy}_{\text{total}} = \text{Accuracy}_{\text{sp}} \times \text{Accuracy}_{\text{HMM}} \tag{7.10}$$

where Accuracy = 1-Error. In table 7.4, A_1 represents $\text{Accuracy}_{\text{HMM}}$, A_2 represents $\text{Accuray}_{\text{sp}}$, and A is the total accuracy of the system.

The highest recognition accuracy, 92.07%, appeared when 8 candidates were remained, and 6s test set was used for HMM recognition. However direct speaker recognition using DDHMM for 22 speakers was not so reliable, and the highest accuracy was only 84.21%. The improvement brought by speaker pruning was 9.3% w.r.t the former accuracy. We can definitely say that the introduction of speaker pruning increases the total recognition accuracy. Whereas the recognition speed is slowed down a little bit. For details on time consumption, see appendix D3.

It is necessary to mention one practical problem occurred in the procedure of codebook deriving. As we mentioned the codebook is formed by K-means algorithm, the feature vectors should be all stored and quantized into *K codewords*. When more data will be used for training, more memory and time will be spent on quantizing the feature vectors into *codewords*. Here for training set we only used two seventh of suggested training data in our database ELSDSR, since large training set brought out of memory problems to Matlab implementation. It's one reason that the highest recognition accuracy of our speaker recognition system is not as high as some other systems. The comparison of our work to others will be given in Chapter 8.

Table 7.4 Summary of Recognition accuracy

Test set / Accuracy(%)	2s	3s	4s	5s	6s
HMM with Ns=4	A_1= 83.33%	A_1= 88.89%	A_1= 94.44%	A_1= 97.22%	A_1=100 %
Speaker pruning (2s)	A_2= 86.57%	A_2= 86.57%	A_2= 86.57%	A_2= 86.57%	A_2= 86.57%
Total error $A=A_1*A_2$	A= 72.14%	A= 76.95%	A= 81.76%	A= 84.16%	A= 86.57%
HMM with Ns=6	A_1= 75.93%	A_1= 87.04%	A_1= 88.89%	A_1= 96.3%	A_1=97.62%
Speaker pruning (2s)	A_2= 92.91%	A_2= 92.91%	A_2= 92.91%	A_2= 92.91%	A_2=92.91 %
Total error $A=A_1*A_2$	A= 70.55%	A= 80.87%	A= 82.59%	A= 89.47%	A= 90.70%
HMM with Ns=8	A_1= 72.22%	A_1= 79.11%	A_1= 84.72%	A_1= 90.28%	A_1=96.43 %
Speaker pruning (2s)	A_2= 95.48%	A_2= 95.48%	A_2= 95.48%	A_2= 95.48%	A_2= 95.48%
Total error $A=A_1*A_2$	A= 73.73%	A= 75.53%	A= 80.86%	A= 86.20%	A= 92.07%
HMM with 22 speaker models	60.1%	68.18%	72.22%	80.81%	84.21%

Chapter 8 Conclusion and Future Work

Conclusion

In this thesis, work has been focused on establishing a text-independent closed-set speaker recognition system. Efforts have been distributed into 4 main parts:

- Database Establishment

With the purpose of collecting more speech samples, an English language speech database for speaker recognition was built during the period of this project. It contains rich speech messages and captures almost all the possible pronunciation of English language: vowels, consonants and diphthongs.

- Feature Selection

Feature is critical for any systems. During this work, cepstral coefficient, LP based cepstral coefficient, MFCC and Pitch were extracted and compared with variant techniques: PCA, binary and multi- KNN. Finally MFCC were chosen, which accorded with the suggested and commonly used features in speaker recognition [18]. Subsequently, feature dimension became another issue waiting to be decided. Testing with different dimensional features, 48 MFCC performed superior to the other dimensions.

- Performance Improvement

Several methods were investigated in increasing the performance of this system on recognizing speaker ID within 22 speakers from ELSDSR database.

1. Speaker pruning technique (Chapter 5 and 7)
   Speaker pruning technique was introduced into our system for the purpose of increasing the recognition accuracy with a little cost of speed. KNN algorithm was implemented to eliminate some of the most dissimilar known speakers with the unknown speaker in ELSDSR database. By this means, HMM recognition only needs to be performed on the 'survived' speaker models, which increased the accuracy. Due to the intention of this speaker pruning in the system, it's designed to be different from the pruning technique, which is to find out the speaker ID with only pruning method [26]. Whereas, herein it is regarded as pre-election before the final election performs. The pruning time consumption mainly depends on the training set size and feature dimensions. The pruning accuracy depends on the number of 'survived' candidates. With 6 candidates and 2s test set, the speaker pruning accuracy was 92.91%; whereas with 8 candidates, it was 95.48%.

2. Improvement of KNN algorithm (Subsection 7.2.3)
Since KNN algorithm was used as the speaker pruning technique, improving the performance of KNN in matching score calculation is necessary. The frame by frame label method was modified into group by group label method, which divided the frame by frame labels into groups with overlaps, and then find the label of each group by majority voting. By doing so, recognition accuracy of KNN was increased 13.3%, since the groups contain richer information than the frames, and the labels became more reliable.

3. Merging Pitch with MFCC in KNN similarity calculation (Chapter 5 & 7)
Several methods for combining pitch and MFCC in similarity calculation have been discussed and tested. Finally the method with adaptive weight parameter outperformed. The initial intention was to combine two features together. However problems occurred due to the low frequency location of fundamental frequency. The combination method adds the pitch estimation results into the *Euclidean distance* calculation to decrease the misclassification of the gender. It is applicable to all algorithms with distance calculations. The adaptive weight depends on the probability of unknown speaker being female. In the worst case, where gender recognition using pitch is wrong, the method can also include the true speaker into candidates after speaker pruning, since the weight becomes very small and will not bring disaster to the distances computed with MFCC, as the method one and two do (Subsection 7.2.4).

- HMM Training

For DDHMM, the number of observations per state and the number of state should be determined in advance. They were decided by experimental results: K=32 per state and one state in HMM. Actually as proved in [9], and quoted in [23], for text-independent applications the information on transitions between different states is ineffective, it means one state in HMM performs better or at least the same as multi-states. This claim was also proved in our experiments with both ELSDSR and TIMIT database. To get the discrete observations, vector quantization was performed by K-means algorithm. Two codebooks were derived: one was for the lowest 24 MFCC; the other was for the 24 first time derivative features vectors. The deriving of codebooks is the most time consuming and memory spending part, which brought practical problems in this system. When more data are used for training, more memory and time is spent on quantizing the feature vectors into *codewords*. In our experiments, generating one small codebook (with less than 64 *codewords*) took at least one hour on an Intel Pentium IV computer (2.5GHz); and for larger data set with more *codewords* it can take 3 to 4 hours. Moreover because of the Matlab memory problem, only two seventh of the suggested training data in ELSDSR database was used. With the limited training data, 22s per speaker, and 6s test set from unknown speaker, the HMM recognition achieved 97.62% accuracy with 6 ‘survived’ candidates; and the accuracy for 8 candidates was 96.43%. However with no doubt, the HM models will become more reliable trained with more training data. Another problem should be mentioned is DDHMM, as a doubly

stochastic process, is hard to train and the recognition results are different from time to time even with the same setups.

Finally we take both speaker pruning accuracy and HMM recognition accuracy within candidates into consideration, and get the accuracy for the whole system. The highest recognition accuracy the system achieved was 92.07% with 8 candidates and 6s test signal. However speaker recognition performed by HMM directly from all the speakers in the ELSDSR gave a lower accuracy, 84.21%.

The comparison of the work in this project to other researchers work in speaker recognition field is necessary. According to Reynolds's work in 1996 with HMM approach [36] in text-independent speaker verification, the identification error rate with 3s test speech recoded in telephone was 11%; and with 10s test speech error became 6%. Our work was done in generally difficult text-independent case, with 6s test speech recorded in lab, and the lowest error rate was 7.93% with speaker pruning and HMM approaches. Both of the work used MFCC with first time derivatives.

From a Master project work on text independent speaker recognition [30], the best accuracy in identifying the correct speaker is roughly 70%. The approach used in this project was weighted VQ, invented in [35], with MFCC, DMFCC, DDMFCC and Pitch. In [6], the recognition system reached 99% recognition rate with 10 speakers database and 1.5 s test speech, which is quite impressive and can be used as a real-time system.

Future Work

In our system speaker pruning was introduced. Because of this step we reduced the number of pattern matching in the next HMM recognition step, which increases the recognition accuracy since HMM does not perform well in speaker recognition from a large set of known speakers in our study. However in the mean while we slowed down a little bit of the recognition speed. The KNN algorithm was used as pruning technique. It retains all the training set and takes time to calculate the *Euclidean distances* between new examples and all the examples in the training set. Therefore, for the future research, more work need to be done for improving the KNN performance. For example weighted method introduced in [35] could be one solution. A variant of this approach calculates a weighted average of the nearest neighbors. Given a specific instance e that shall be classified, the weight of an example increases with increasing similarity to e [34].

As mentioned in subsection 4.2.3, the DDHMM stores the generated codebooks in advance, whereas CDHMM calculates the probability for each observation during training and recognition. Therefore we chose DDHMM as speaker modeling method since it spends less computation time, even it needs more memory for storing codebooks [28]. However, while saving computation time, we lose some recognition accuracy. According to [9], an ergodic CDHMM is superior to an ergodic DDHMM, so the study and implementation of CDHMM may be one of the future works to increase

the recognition accuracy for our speaker recognition system.

Finally, some work could be done in the future on feature pruning. Now we are using 48 dimensional MFCC including 24 first time derivatives. Because of the curse of dimensionality, we could try to get lower dimensional features while keeping the most important and speaker-dependent information [13].

A Essential Problems in HMM

A1 Evaluation Problem

A1.1 *Forward Algorithm*

The evaluation problem is solved by *forward-backward* algorithm, which has less computational complexity than the naive way mentioned in subsection 4.2.2. The definition of forward variables is shown in (4.8), and for convenience we repeat it here:

$$\alpha_t(i) = P(o_1 o_2 \cdots o_t, x_t = s_i | \lambda) \tag{A.1.1}$$

The *forward recursion* can be explained inductively as follows:

- Initialization:

$$\alpha_1(i) = P(o_1, x_1 = s_i | \lambda) = \pi_i b_i(o_1), \quad 1 \le i \le N \tag{A.1.2}$$

When t=1, the joint probability of initial observation o_1 and state s_i is expressed by the multiplication of the initial state distribution π_i and the emission probability of the initial observation $b_i(o_1)$.

- Induction

In this step, we will lead the forward variable through time. (A.1.1) shows the forward variable at time t, and suppose at time t+1, model goes to state s_j from N possible state s_i $(1 \le i \le N)$, and then the forward variable at time t+1 can be derived as follows:

$$\begin{aligned}
&\alpha_{t+1}(j) \\
&= P(o_1 o_2 \cdots o_t o_{t+1}, x_{t+1} = s_j | \lambda) \\
&= \sum_{i=1}^{N} P(o_1 o_2 \cdots o_t o_{t+1}, x_t = s_i, x_{t+1} = s_j | \lambda) \\
&= \sum_{i=1}^{N} P(o_1 o_2 \cdots o_t, x_t = s_i | \lambda) \cdot P(x_t = s_i, x_{t+1} = s_j | \lambda) \cdot P(o_{t+1} | x_{t+1} = s_j, \lambda) \\
&= \left[\sum_{i=1}^{N} \alpha_t(i) \cdot a_{ij} \right] b_j(o_{t+1})
\end{aligned} \tag{A.1.3}$$

$$1 \le j \le N, \quad 1 \le t \le T-1$$

We notice that the last step of (A.1.3) gives the recursion of the forward variable.

- Termination

From (A.1.1), we know the terminal forward variable (when t=T) is:

$$\alpha_T(i) = P(o_1 o_2 \cdots o_T, x_T = s_i | \lambda) \tag{A.1.4}$$

Therefore the desired $P(O|\lambda)$ is just the sum of the terminal forward variable:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{A.1.5}$$

A1.2 *Backward Algorithm*

The definition of the backward variables are: given the model λ and the model is in state s_i at time t, the probability of having seen the partial observations from time t+1 until the end:

$$\beta_t(i) = P(o_{t+1} o_{t+2} \cdots o_T | x_t = s_i, \lambda) \tag{A.1.6}$$

Same as the *forward algorithm*, *backward algorithm* could also be explained in steps as follows:

- Initialization

When t=T, the backward variable becomes 1 for all permitted final states:

$$\beta_T(i) = P(o_T | x_T = s_i, \lambda) = 1, \quad 1 \le i \le N \tag{A.1.7}$$

- Induction

When at time t, the backward variable could be expressed as follow:

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j), \quad t = T-1, T-2, \cdots, 1, \quad 1 \le i \le N \tag{A.1.8}$$

We read (A.1.8) from right to left, due to the backward recursion what we know is $\beta_{t+1}(j)$, we need to derivate the backward variable for the previous time. From the elements of an HMM, we know the probability of having observation o_{t+1} at time t+1, $b_j(o_{t+1})$, and the probability of jumping from state s_i is s_j, a_{ij} , then the variable for the previous time is just the sum of the production of those three components.

A2 Optimal State Sequence Problem

The optimal criterion is aimed at choosing the state sequence having the maximum likelihood w.r.t the given model. The task can be fulfilled recursively by the *Viterbi* algorithm.

The *Viterbi* algorithm uses two variables:

$\delta_t(i)$ is defined as the probability of having seen the partial observation ($o_1, o_2, \ldots, o_t$) and the model being in state s_i at time t by the most likely path, which means that the $\delta_t(i)$ is the highest likelihood of a single path among all the paths ending in state s_i at time t.

$$\delta_t(i) = \max_{x_1 x_2 \cdots x_{t-1}} P(o_1 o_2 \cdots o_t, x_1 x_2 \cdots x_t = s_i | \lambda) \tag{A.2.1}$$

$\psi_t(i)$ is defined to keep track of the best path ending in the state s_i at time t:

$$\psi_t(i) = \arg\max_{x_1 x_2 \cdots x_{t-1}} P(o_1 o_2 \cdots o_t, x_1 x_2 \cdots x_t = s_i | \lambda) \tag{A.2.2}$$

The procedure of *Viterbi* algorithm is summarized as follows into four steps:

- Initialization

The initialization of $\delta_t(i)$ is the same as that of forward variable $\alpha_1(i)$:

$$\delta_1(i) = \max_{x_1 x_2 \cdots x_{t-1}} P(o_1, x_1 = i | \lambda) = \pi_i b_i(o_1) \tag{A.2.3}$$

$$\psi_1(i) = 0 \tag{A.2.4}$$

- Induction

$$\delta_{t+1}(j) = \max_{1 \le i \le N} (\alpha_t(i) \cdot a_{ij}) b_j(o_{t+1}), \quad 1 \le j \le N, \quad 1 \le t \le T-1 \tag{A.2.5}$$

$$\psi_{t+1}(j) = \arg\max_{1 \le i \le N} (\alpha_t(i) \cdot a_{ij}), \quad 1 \le j \le N, \quad 1 \le t \le T-1 \tag{A.2.6}$$

Notice the difference between (A.2.5) and the last step of (A.1.3), in (A.2.5) only the path (sequence) with highest likelihood survived.

- Termination

$$P^*(O|\lambda) = \max_{1 \le i \le N} \delta_T(i) \tag{A.2.7}$$

$$x_T^* = \arg\max_{1 \le i \le N} \delta_T(i) \tag{A.2.8}$$

where x_T^* is the optimal final state, and * denotes the optimal value.

- Backtracking

In the termination step we get the optimal final state, then by doing backtracking, we can find out the optimal state sequence:

$$X^* = \{x_1^* x_2^* \cdots x_T^*\} \text{ and } x_t^* = \psi_{t+1}(x_{t+1}^*) \quad t = T\text{-}1, T\text{-}2, \ldots, 1 \tag{A.2.9}$$

B Normalized KNN

Table B Normalized Data for KNN algorithm (N_K=3)

Name	**Age**	**Math**	**Physics**	**Chemistry**	**Qualified**	Euclidean distances from George
Alice	18/40 =0.45	10/13 =0.77	10/13 =0.77	10/11 =0.91	**Yes**	$[(0.675-0.45)^2 + (1-0.77)^2 + (0.85-0.77)^2 + (1-0.91)^2]^{½}$ = 0.344 (**2nd**)
Tom	25/40 =0.625	7/13 =0.54	8/13 =0.62	9/11 =0.82	No	$[(0.675-0.625)^2 + (1-0.54)^2 + (0.85-0.62)^2 + (1-0.82)^2]^{½}$ = 0.547
Jerry	22/40 =0.55	9/13 =0.69	10/13 =0.77	*11*/11 =1	**Yes**	$[(0.675-0.55)^2 + (1-0.69)^2 + (0.85-0.77)^2 + (1-1)^2]^{½}$ = 0.344 (**2nd**)
Homer	*40*/40=1	5/13 =0.38	3/13 =0.23	6/11 =0.55	No	$[(0.675-1)^2 + (1-0.38)^2 + (0.85-0.23)^2 + (1-0.55)^2]^{½}$ = 1.038
Lisa	23/40 =0.575	11/13 =0.85	*13*/13 =1	10/11 =0.91	**Yes**	$[(0.675-0.575)^2 + (1-0.85)^2 + (0.85-1)^2 + (1-0.91)^2]^{½}$ = 0.251 (**1st**)
Bart	35/40 =0.875	6/13 =0.46	7/13 =0.54	5/11 =0.45	No	$[(0.675-0.875)^2 + (1-0.46)^2 + (0.85-0.54)^2 + (1-0.45)^2]^{½}$ = 0.855
George	27/40 =0.675	*13*/13 =1	11/13 =0.85	11/11 =1	**YES**	

Table B shows the normalization of variables for KNN algorithm. The normalization is to avoid the domination of variables with large values. It should be done before computing the Euclidean distances. Using the data set from Example 5.1, first we found out the maximum value for each variable, the italic numbers in the table. Then all the variables are divided by the maximum values, shown in the left half of the table. Afterwards the Euclidean distances can be calculated for finding the smallest N_K distances, which represents N_K nearest neighbors. Here N_K is set to 3. The 3 nearest neighbors of George are Lisa, Jerry and Alice. However, without normalization, the 3 nearest neighbors are Lisa, Jerry and Tom. It is caused by the domination of Age variable. Nevertheless, as we said before, in this example the normalization effect is not so obvious, and doesn't effect the final decision.

C Database Information

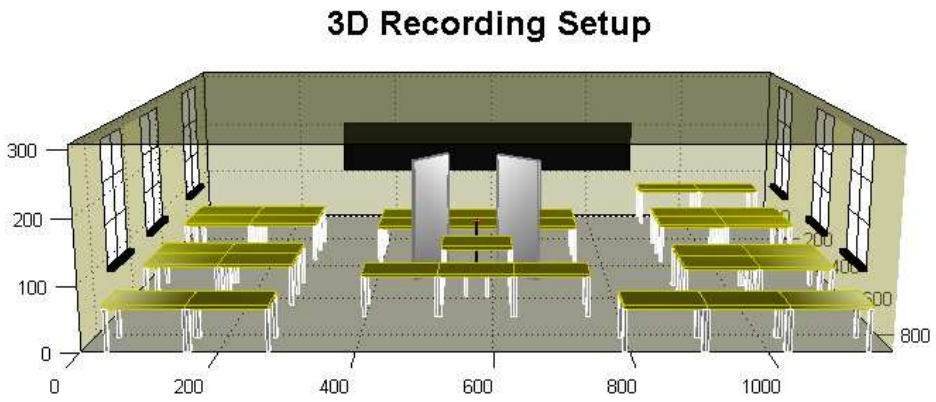
C1 Detailed Information about Database Speakers

Table B.1: Information about Speakers

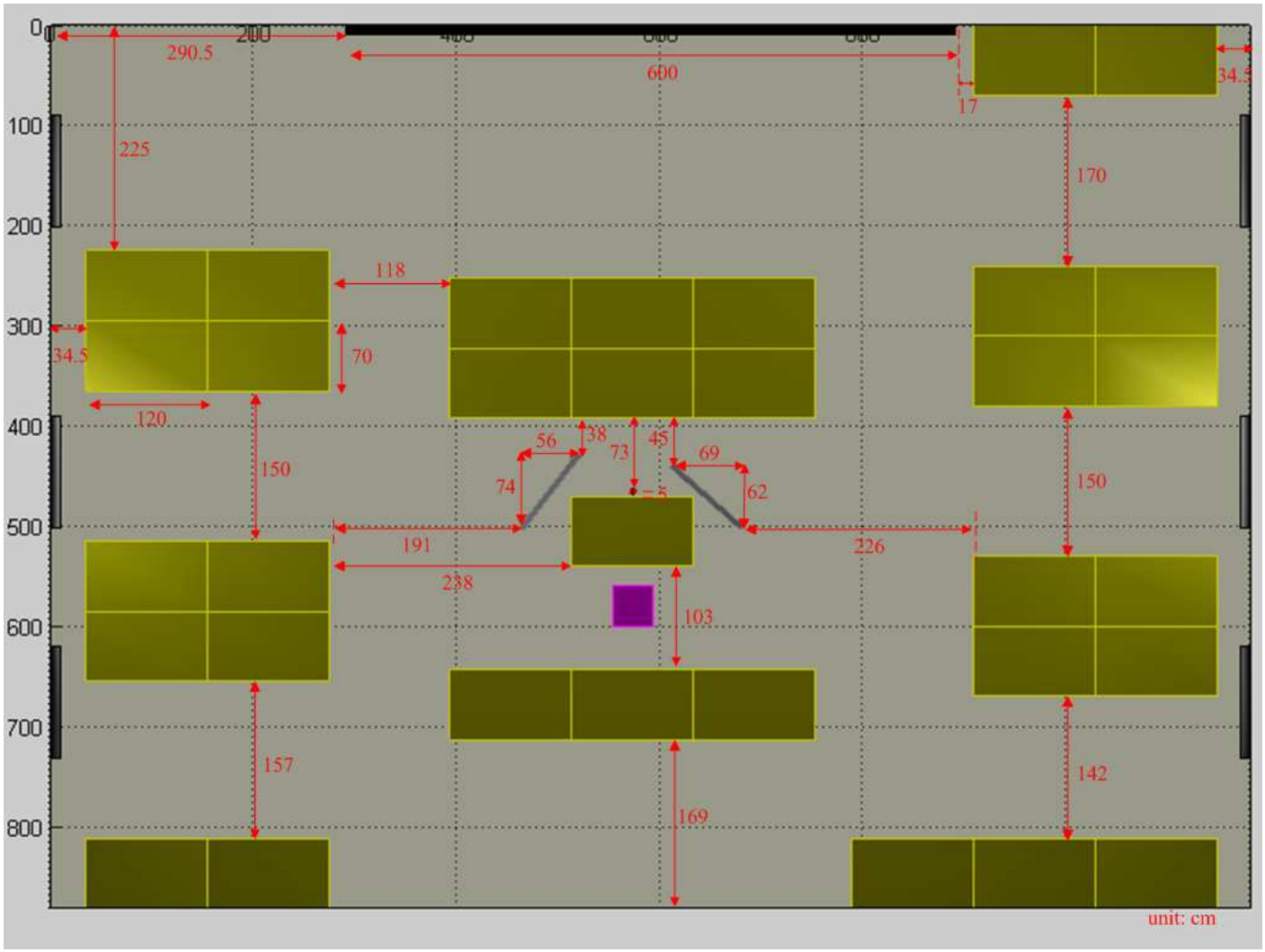
Speaker ID	**Age**	**Nationality**
FAML	48	Danish
FDHH	28	Danish
FEAB	58	Danish
FHRO	26	Icelander
FJAZ	25	Canadian
FMEL	38	Danish
FMEV	46	Danish
FSLJ	24	Danish
FTEJ	50	Danish
FUAN	63	Danish
Average	40.6	
MASM	27	Danish
MCBR	26	Danish
MFKC	47	Danish
MKBP	30	Danish
MLKH	47	Danish
MMLP	27	Danish
MMNA	26	Danish
MNHP	28	Danish
MOEW	37	Danish
MPRA	29	Danish
MREM	29	Danish
MTLS	28	Danish
Average	31.75	

C2 Recording Experiment Setup

C2.1 3D Setup



C2.2 2D Setup with Measurement



D Experiments

D1 Text-dependent Case for Binary KNN

Two 4s long signals were cut from FJAZ_Sa.wav and MKBP_Sa.wav. Then features extracted from these two signals were concatenated together to work as training examples. For the test set, two 3s signals from FJAZ_Sb.wav and MKBP_Sb.wav separately were cut and their features were then concatenated. The features we used all have 24 dimensions.

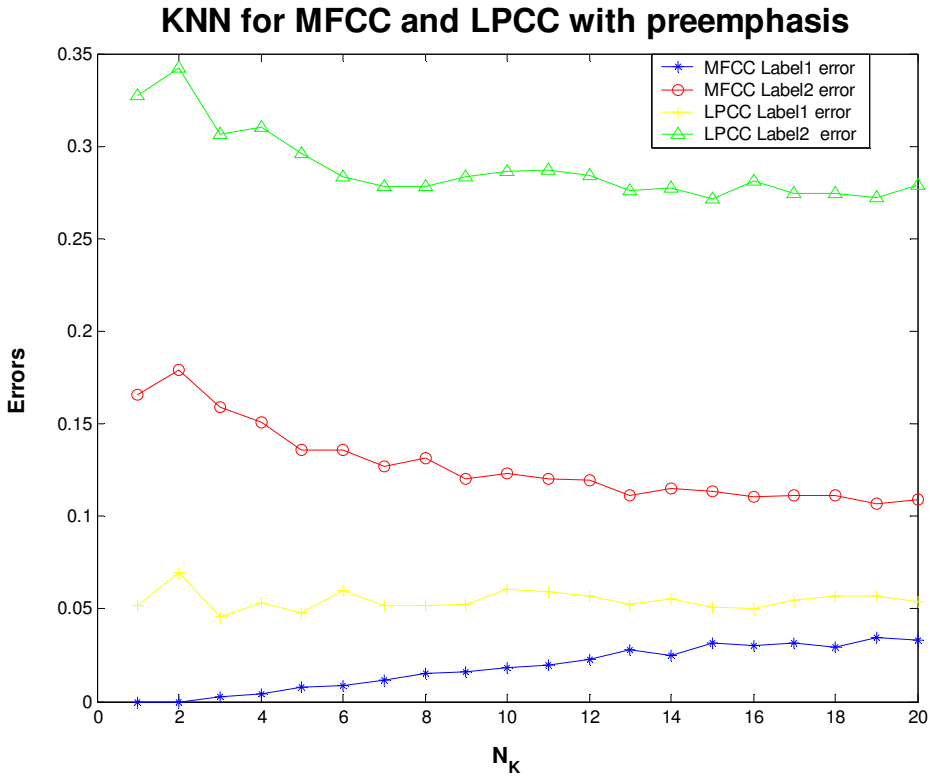


Fig C1.1 LPCC vs. MFCC using KNN in text-dependent case

24 MFCC (12 delta) and 24 LPCC (12 delta) features were extracted from FJAZ_Sa.wav, MKBP_Sa.wav, FJAZ_Sb.wav and MKBP_Sb.wav. First two were used as training data, the rest were used as test data for binary KNN algorithm in text-dependent case. 20 iterations were done with the number of neighbors changing from 1 to 20 in order to find out the minimum test error. Red (o) and blue (*) curves gave the Label1 and Label2 (test) errors from MFCC. It shows using MFCC can achieve smaller errors, and when N_K=19 minimum test error 0.1072 was achieved, and corresponding Label1 error was around 0.0345.

D2 Pitch Accuracy for Gender Recognition

Table D2.1 Pitch estimation accuracy for recognizing genders from 1s and 2s

Sentence / Speaker	Sentence 1		Sentence2	
	1s	2s	1s	2s
FAML	0.9130	1.0000	1.0000	1.0000
FDHH	1.0000	1.0000	1.0000	1.0000
FEAB	1.0000	1.0000	0.9722	1.0000
FHRO	1.0000	1.0000	1.0000	1.0000
FJAZ	1.0000	1.0000	1.0000	1.0000
FMEL	1.0000	1.0000	1.0000	1.0000
FMEV	1.0000	1.0000	1.0000	1.0000
FSLJ	1.0000	1.0000	1.0000	1.0000
FTEJ	0.8571	1.0000	1.0000	1.0000
FUAN	1.0000	1.0000	1.0000	1.0000
MASM	1.0000	1.0000	0.9859	1.0000
MCBR	1.0000	1.0000	1.0000	1.0000
MFKC	0.9000	1.0000	0.9412	1.0000
MKBP	0.7778	1.0000	0.9091	1.0000
MLKH	0.6933	0.8200	0.8500	0.8900
MMLP	1.0000	1.0000	1.0000	1.0000
MMNA	1.0000	1.0000	1.0000	1.0000
MNHP	0.9333	1.0000	1.0000	1.0000
MOEW	1.0000	1.0000	1.0000	1.0000
MPRA	1.0000	1.0000	1.0000	1.0000
MREM	0.8333	0.8750	0.9130	1.0000
MTLS	1.0000	1.0000	1.0000	1.0000

Table D2.1 gives the pitch accuracy for gender recognition for ELSDSR database with 1s and 2s test signals respectively. The detected pitches were compared with the pitches estimated from all the training signals of each speaker in the database, which is quite reliable. The lowest accuracies using 1s was 69.33%, but for most of the speakers it achieved 100%. With 2s signals, the detected pitches became more reliable, which gave 85% lowest accuracy.

Notice the accuracies for some speakers on gender recognition using Pitch information are not so high. Even though the worst situation happens, where the detected pitch gives wrong gender recognition, the adaptive weight parameter κ in the invented method can reduce the impact by adjusting the weight parameter with the probability of the speaker being certain gender. For this uncertain case, the probability becomes around 50%, which will give a very small weight parameter.

D3 Time consumption of recognition with/without speaker pruning

Table D3.1 Time consumption with/without speaker pruning

Test set / Time (s)	2s	3s	4s	5s	6s
Feature extraction	T_1= 0.11	T_1= 0.22	T_1= 0.24	T_1= 0.33	T_1= 0.45
HMM with Ns=4	T_2= 1.53	T_2=2.35	T_2=3.06	T_2=3.88	T_2= 4.05
Speaker pruning	T_3= 19.85	T_3= 29.67	T_3= 40.99	T_3= 49.36	T_3= 60.09
Total time $T=T_1+T_2+T_3$	T=21.49	T=32.24	T=44.29	T=53.57	T=64.59
HMM with Ns=6	T_2= 2.31	T_2= 3.48	T_2= 4.60	T_2=5.76	T_2= 8.30
Speaker pruning	T_3= 19.86	T_3= 28.89	T_3= 40.41	T_3= 50.81	T_3= 60.87
Total time $T=T_1+T_2+T_3$	T=22.28	T=32.59	T=45.25	T=56.90	T=69.62
HMM with Ns=8	T2=4.29	T2=6.40	T2=8.52	T2=10.57	T2= 10.38
Speaker pruning	T3=19.85	T3= 30.77	T3= 41.02	T3= 50.33	T3= 59.87
Total time T=T1+T2 +T3	T=24.25	T=37.39	T=49.78	T=61.23	T=70.7
HMM with 22 speaker models	22.8624	25.83	34.05	41.97	68.79

Table D3.1 shows that by introducing the speaker pruning, the total recognition speed was slightly slowed down. However the time consumption can be significantly decreased by optimizing the implementation.

References

[1] D.A. Reynolds, L.P. Heck, “Automatic Speaker Recognition”, AAAS 2000 Meeting, Humans, Computers and Speech Symposium, 19 Feb 2000.

[2] R. A. Cole and colleagues, “Survey of the State of the Art in Human Language Technology”, National Science Foundation European Commission, 1996. http://cslu.cse.ogi.edu/HLTsurvey/ch1node47.html

[3] J. A. Markowitz and colleagues, “J. Markowitz, Consultants”, 2003. http://www.jmarkowitz.com/glossary.html

[4] J. Rosca, A. Kofmehl, “Cepstrum-like ICA Representations for Text Independent Speaker Recognition”, *ICA2003*, pp. 999-1004, 2003.

[5] D.A. Reynolds, R.C. Rose, “Robust text-independent speaker identification using Gaussian Mixture speaker models”, *IEEE Trans. on Speech and Audio Processing*, vol. 3, no. 1, pp. 72-83, 1995.

[6] L. P. Cordella, P. Foggia, C. Sansone, M. Vento, “A Real-Time Text-Independent Speaker Identification System”, *Proceedings of the ICIAP*, pp. 632, 2003.

[7] H. A. Murthy, F. Beaufays, L. P. Heck, M. Weintraub, “Robust Text-independent Speaker Identification over Telephone Channels”, *IEEE Trans. on Speech and Audio Processing*, vol. 7, no.5, pp.554-568, 1999.

[8] C. Tanprasert, C. Wutiwiwatchai, S. Sae-tang, “Text-dependent Speaker Identification Using Neural Network On Distinctive Thai Tone Marks”, IJCNN '99 *International Joint Conference on Neural Network*, vol. 5, pp. 2950-2953, 1999.

[9] T. Matsui, S. Furui, “Comparison of text-independent speaker recognition methods using VQ-distortion and discrete/continuous HMMs”, *Proc. ICASSP*, vol. II, pp. 157-160, 1992.

[10] A. F. Martin, M. A. Przybocki, “Speaker Recognition in a Multi-Speaker Environment”, *Eurospeech 2001*, Scandinavia, vol. 2, pp. 787-790.

[11] H. G. Kim, E. Berdahl, N. Moreau. T. Sikora, “Speaker Recognition Using MPEG-7 Descriptors”, *Eurospeech 2003*, Geneva, Switzerland, September 1-4, 2003.

[12] Jose M. Martinez (UAM-GTI, ES), “MPEG-7 Overview (version 9)”, ISO/IEC JTC1/SC29/WG11N5525, March 2003, Pattaya.

[13] Ing. Milan Sigmund, CSc. “Speaker Recognition, Identifying People by their Voices”, Brno University of Technology, Czech Republic, Habilitation Thesis, 2000.

[14] J. P. Campbell, JR., “Speaker Recognition: A Tutorial”, *Proceedings of the IEEE*, vol. 85, no.9, pp. 1437-1462, Sep 1997.

[15] D. Schwarz, “Spectral Envelopes in Sound Analysis and Synthesis”, IRCAM *Institut de la Recherche et Coordination* Acoustique/Musique, Sep 1998.

[16] J. R. Deller, J. H.L. Hansen, J. G. Proakis, “Discrete-Time Processing of Speech Signals”, IEEE Press, New York, NY, 2000.

[17] J. G. Proakis, D. G. Manolakis, “Digital signal processing. Principles, Algorithms and Applications”, Third ed. Macmillan, New York, 1996.

[18] T. Kinnunen, “Spectral Features for Automatic Text-independent Speaker Recognition”, University of Joensuu, Department of Computer Science, Dec. 2003.

[19] J. Harrington, S. Cassidy, “Techniques in Speech Acoustics”, Kluwer Academic Publishers, Dordrecht, 1999.

[20] H. Ezzaidi, J. Rouat, D. O’Shaughnessy, “Towards Combining Pitch and MFCC for Speaker Identification Systems”, *Proceedings of Eurospeech 2001*, pp. 2825, Sep 2001.

[21] T. Shimamura, “Weighted Autocorrelation for Pitch Extraction of Noisy Speech”, *IEEE Transactions on Speech and Audio Processing*, vol. 9, No. 7, pp. 727-730, Oct 2001.

[22] G. Middleton, “Pitch Detection Algorithm”, Connexions, Rice University, Dec 2003 http://cnx.rice.edu/content/m11714/latest/

[23] D. A. Reynolds, “An Overview of Automatic Speaker Recognition Technology”, *Proc. ICASSP* 2002, Orlando, Florida, pp. 300-304.

[24] L. R. Rabiner, “A tutorial on hidden Markov models and selected application in speech recognition”, *Proceedings of the IEEE*, vol. 77, No. 2, pp. 257-286, Feb 1989.

[25] B. Resch, “Hidden Markov Models, A tutorial of the course computational intelligence”, Signal Processing and Speech Communication Laboratory.

[26] E. Karpov, “Real-Time Speaker Identification”, University of Joensuu, Jan. 2003.

[27] C. M. Bishop, “Neural Networks for Pattern Recognition”, *OXFORD University Press*, Oxford, UK, 1995.

[28] X. Wang, “Incorporating Knowledge on Segmental Duration in *HMM*-based Continuous Speech Recognition”, Ph. D Thesis, Institute of Phonetic Sciences, University of Amsterdam, Proceedings 21, pp.155-157, 1997.

[29] A. Cohen, Y. Zigel, “On Feature Selection for Speaker Verification”, *Proceedings of COST 275 workshop on The Advent of Biometrics on the Internet,* pp. 89-92, Nov. 2002.

[30] N. Bagge, C. Donica, “Text Independent Speaker Recognition”, ELEC 301 Signals and Systems Group Project, Rice University, 2001.

[31] C.W.J, “Speaker Identification using Gaussian Mixture Model”, Speech Processing Laboratory at National TaiWan Univeristy, May. 2000.

[32] NOVA online, WGBH Science Unit,1997 http://www.pbs.org/wgbh/nova/pyramid/

[33] T. Kinnunen, T. Kilpeläinen, P. Fränti, “Comparison of Clustering Algorithm in Speaker Identification”, Proc. IASTED Int. Conf. Signal Processing and Communications (SPC), pp. 222-227, Marbella, Spain, 2000.

[34] BSCW project group, “The Machine Learning network Online Information Service”, website supported by EU project Esprit No. 29288, University of Magdeburg and GMD. http://www.mlnet.org/

[35] T. Kinnunen, P. Fränti, “Speaker discriminative weighting method for VQ-based speaker identification”, Proc. 3rd International Conference on audio-and video-based biometric person authentication (AVBPA), pp. 150-156, Halmstad, Sweden, 2001.

[36] L. K. Hansen, O. Winther, “Singular value decomposition and principal component analysis”, Class notes, IMM, DTU, April 2003.