Chapter 1

Introduction

1.1 Motivation

Communicating with a computer or machines using nature speech, just like people do in the science fiction, has been the dream of everyone for a long time. Due to the maturity of the automatic speech recognition technologies, the dream will come true. The improvement on the computation hardware makes many complex algorithms feasible in a practical ASR with a low cost. The ASR is useful for many applications, such as automatic tickets booking, voice command, speech-to-text or text-to-speech system, etc. The ASR works very well when they are trained and tested under similar acoustic environments. However, with the deployment of ASR in real word, the input speech for recognition could not always be received in the similar acoustic conditions. The performance of the ASR will decrease as long as the training and testing environments are mismatched. The mismatch happens in many situations, such as the additive background noise, channel effects, different speaker characteristics, etc. The aim of the work presented in this thesis is to make automatic speech recognition systems robust to the additive background noises.

In past years, many approaches dealing with the additive noise have been proposed. These approaches can be roughly categorized into three classes. First is to develop a special robust feature so that it is less sensitive to the various acoustic conditions, e.g., the short-time modified coherence representation (SMC)[1], the perceptual linear prediction (PLP), and the relative spectral (RASTA) approach[2]. Second class of approaches try to modify the speech features obtained in the application environment and make them better match to the acoustic conditions for the clean speech models, e.g., the spectral subtraction (SS)[3], the code-book dependent cepstral normalization (CDCN)[4], and the probabilistic optimal filtering (POF)[5]. In the third class, the compensation is performed on the clean speech models, so that the modified models will be able to match the testing speech features collected in the application environment, e.g., speech and noise decomposition (SND) [6], vector Taylor series (VTS) [7], the model-based stochastic matching [8], and the parallel model combination (PMC) [9]. The method proposed in this thesis belongs to the third class.

1.2 Overview

The chapter of thesis is organized as follows. In chapter 2, the front-end techniques of the speech recognition system will be introduced, including the MFCC feature extraction. In chapter 3, the Hidden Markov Model and its training and recognition procedures will be described. In chapter 4, the parallel model combination method will be introduced first, and then, the method of robust speech recognition using the pre-trained noisy models will be proposed. The experiment results will be shown in the last of this chapter. The conclusions and future works will be presented in chapter 5.

Chapter 2

Speech Signal Pre-Processing and Parameterization

In general, it is difficult to process speech signals directly in time-domain due to the fact that speech signals change fast with time. Fortunately, a speech signal is known to be short time stationary, i.e., any two successive short periods of a speech signal almost have the same characteristics. Due to this property, the short-term spectral analysis can be applied to get the features useful for speech recognition.

There are several kinds of methods to obtain speech feature parameters, such as Linear Prediction Coding (LPC)[10], LPC-derived Cepstrum (LPCC)[11], Mel Frequency Cepstrum Coefficient (MFCC)[12], and Perceptual Linear Predictive analysis (PLP)[13], etc. Since the MFCC is a common and useful method to obtain the speech features, it will be introduced and used in this thesis. The procedures to get MFCCs are shown in Fig.2.1 and will be described in the following sections.

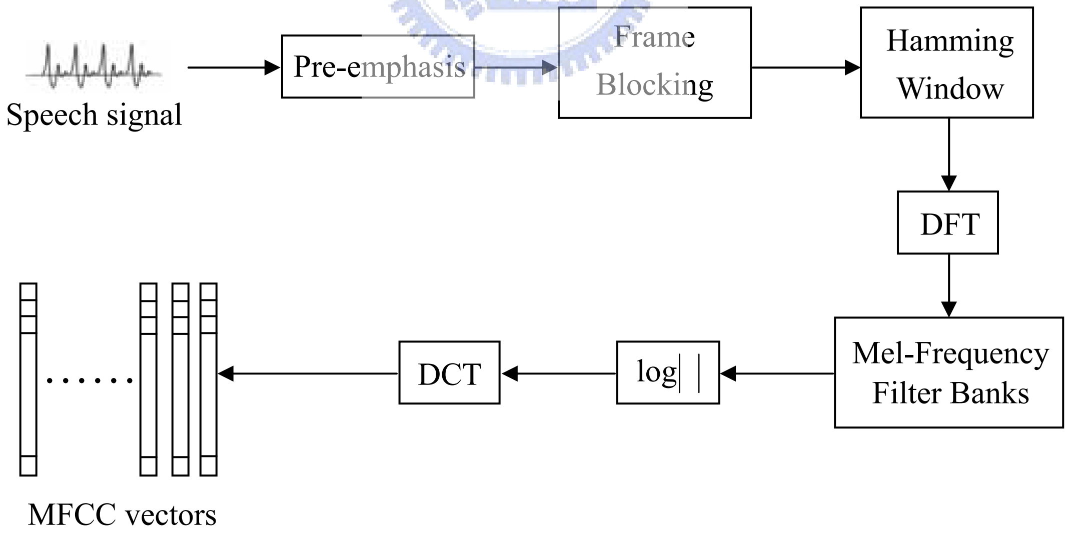


Fig. 2.1 Procedures of producing MFCCs.

2.1 Pre-emphasis

Before getting the MFCCs, speech signals has to be processed through a high-pass filter, known as the procedure of pre-emphasis. The high-pass filter is often represented as

$$F_{pre}(z) = 1 - az^{-1}, \qquad 0.9 < a < 1.0 \qquad (2\text{-}1)$$

From the production model of voiced speech, there is an overall of –6 dB/oct decay, with –12 dB/oct due to excitation source and +6 dB/oct due to the radiation compensation, in speech radiated from lips as frequency increases. Therefore, a pre-emphasis filter providing +6 dB/oct in high-frequency will be adopted to compensate the overall –6 dB/oct decay.

2.2 Frame Blocking with Hamming Window

As mentioned before, since the speech signals are of short time stationary, windows with fixed length are commonly employed to block the speech signals frame by frame. Usually, there are three factors, called frame duration, overlap, and frame period, which should be considered for utilizing a fixed length window shown in Fig.2.2.

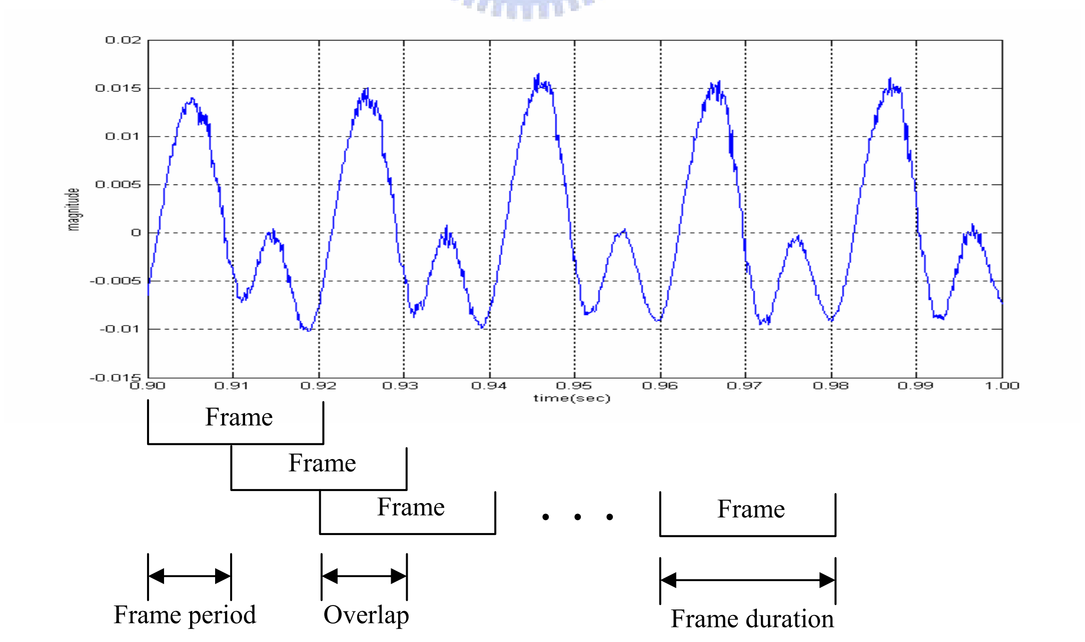


Fig. 2.2 The diagrams of the frame, overlap and frame duration.

The frame duration is the length of the window, which is often chosen as 25ms or 30ms in speech process. Commonly, the frame duration 25ms is selected for the sampling frequency 16 KHz, each frame containing 400 samples. As for the sampling frequency 8 KHz, the frame duration 30ms is adopted, and then each frame contains 240 samples.

Besides, to avoid the characteristics of two successive frames changing too rapidly, an overlap between them will be purposely added. Usually, the overlaps 15ms and 20ms are selected for the sampling frequency 16 KHz and 8 KHz, respectively. With the overlap, the frame period, or frame shift, is defined as the difference between the frame duration and the overlap.

To fulfill frame blocking, in general, a fixed window $w[n], 0 \le n \le N-1$, will be required for the speech signals $s[n]$, where N is the length of the window. Then, the blocked frame is represented as

$$f[n;m] = s[n]w[m-n] \tag{2-3}$$

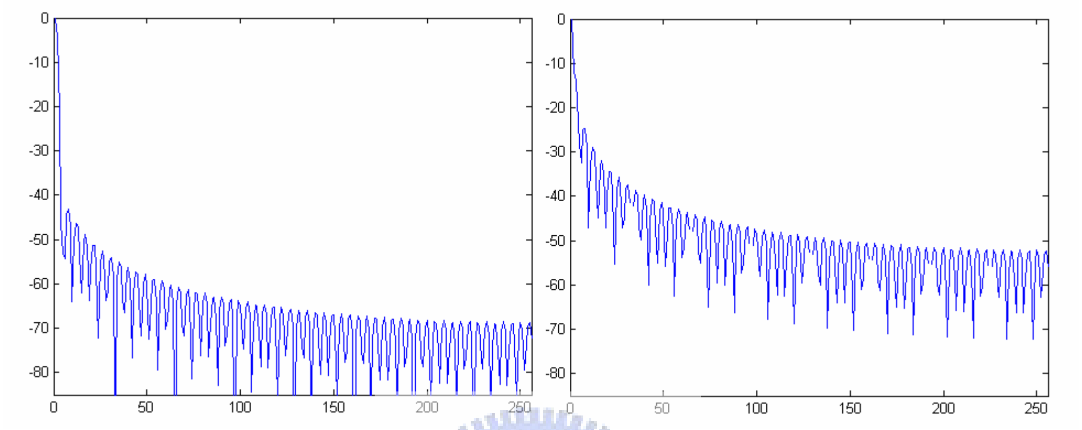
where m is the end position of the frame. Its Discrete-Fourier Transform becomes

$$\begin{aligned} F(e^{j\omega};m) &= \sum_{n=-\infty}^{\infty}[s[n]w[m-n]]e^{-jn\omega} \\ &= \sum_{n=-\infty}^{\infty}[s[n]\{\frac{1}{2\pi}\int_{-\pi}^{\pi} e^{jm\theta}W(e^{-j\theta})e^{jn\theta}d\theta\}]e^{-jn\omega} \\ &= \frac{1}{2\pi}\int_{-\pi}^{\pi}[\sum_{n=-\infty}^{\infty}s[n]e^{-jn\omega}e^{jn\theta}]W(e^{-j\theta})e^{jm\theta}d\theta \\ &= \frac{1}{2\pi}\int_{-\pi}^{\pi}S(e^{j(\omega-\theta)})W(e^{-j\theta})e^{jm\theta}d\theta \end{aligned} \tag{2-4}$$

Note that if $W(e^{j\theta})$ ideally equals to $2\pi\delta(\theta)$, $F(e^{j\omega};m)$ will be equal to $S(e^{j\omega})$, i.e., the original signal $s[n]$ is not changed after transformation. Viewing from the ideal $W(e^{j\theta})$, a good window should possess narrow main lobe and large degradation of side lobe. The simplest window is rectangular window. However, though it has a narrow main lobe in frequency domain, its degradation of side lobe is too small.

Therefore, the Hamming window (2-5) is usually used instead of rectangular window. From Fig.2.3, it is obvious that the Hamming window has much better degradation of side lobe.

$$w[n] = \begin{cases} 0.54 - 0.46\cos(\frac{2n\pi}{N-1}), & 0 \le n \le N\text{-}1 \\ 0, & \text{others} \end{cases} \tag{2-5}$$



(a) (b)

Fig. 2.3 Magnitude response of (a) Hamming window and (b) rectangular window.

Furthermore, the Fast Fourier Transformation is often adopted to substitute for the DFT for more efficient computation. To do FFT, the input data's number must be 2^N exactly; hence, in case that the number is less than or more than 2^N, it is then purposely to pad with zeros or to truncate the input data to obtain 2^N input data for FFT processing.

2.3 Cepstral Analysis

According to the speech production model, voiced speech is composed of a convolved combination of the excitation sequence $e[n]$ with the vocal system impulse response $\theta[n]$, and expressed as

$$s[n] = e[n] * \theta[n] \tag{2-6}$$

It is difficult to separate the $e[n]$ and $\theta[n]$ directly in spectral domain because the individual parts are not combined linearly. Commonly, a special domain, called the cepstral domain, will apply to speech processing. In this domain, the representatives of the component signals will be linear combined, which can be separated in the cepstrum.

The real cepstrum of a speech sequence s[n] is defined as

$$\begin{aligned} C_s[n] &= F^{-1}\{\log|F\{s[n]\}|\} \\ &= F^{-1}\{\log|S(\omega)|\} \\ &= F^{-1}\{\log|E(\omega)\Theta(\omega)|\} \\ &= F^{-1}\{\log|E(\omega)| + \log|\Theta(\omega)|\} \\ &= F^{-1}\{\log|E(\omega)|\} + F^{-1}\{\log|\Theta(\omega)|\} \\ &= C_{\mathrm{e}}[n] + C_\theta[n] \end{aligned} \tag{2-7}$$

where $F\{\cdot\}$ and $F^{-1}\{\cdot\}$ denotes FFT and IFFT operation, respectively. Usually, for real cepstrum, the IFFT operation will be replaced by Discrete-Cosine Transform. Besides, the independent variable n in cepstral domain is defined the term quefrency.

After this operation, the low-quefrency part of cepstrum represents an approximate to the cepstrum of the vocal system impulse response, $C_\theta[n]$, and the high-quefrency corresponds to the cepstrum of the excitation, $C_e[n]$. Therefore, $C_\theta[n]$, containing more information about the speech signals, can be easily extracted by a low-time lifter

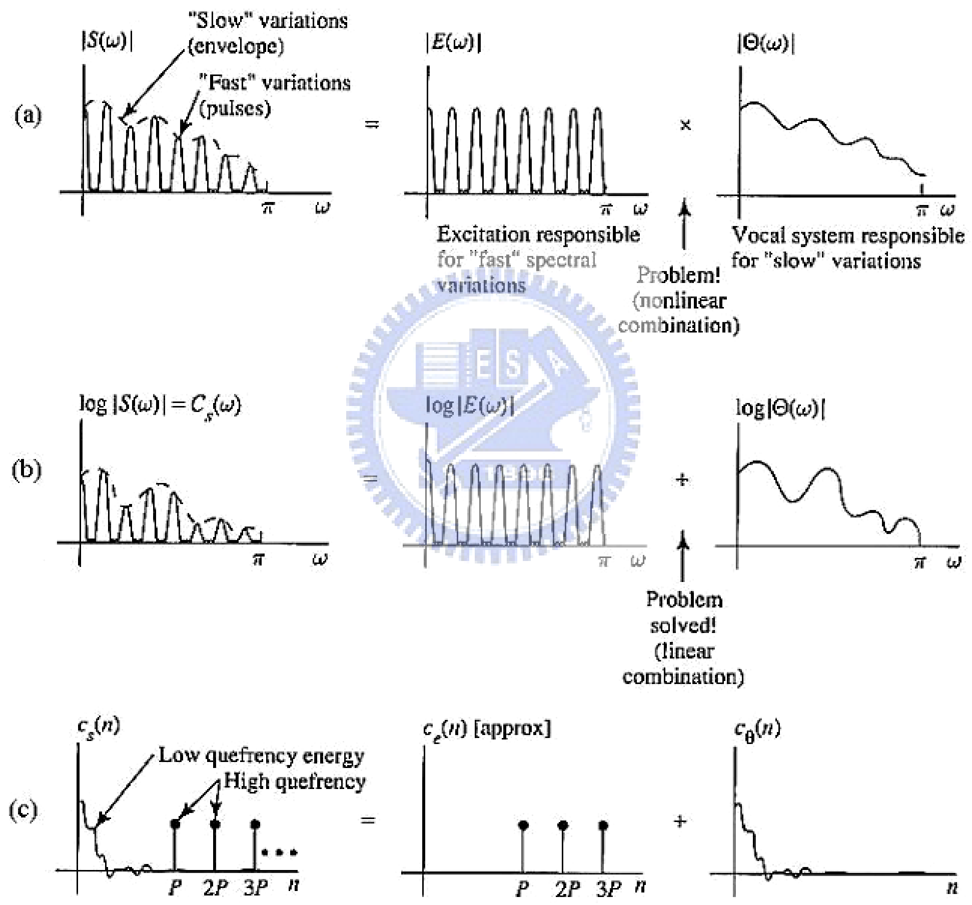
$$L[n] = \begin{cases} 1, & 0 \le n \le L \\ 0, & otherwise \end{cases} \tag{2-8}$$

and $C_s[1], C_s[2], C_s[3], \ldots\ldots, C_s[L]$ will be chosen to form a set of cepstral coefficients. Moreover, at $n = 0$, it implies the intensity of the signal, and it is usually not useful itself. The procedures of getting cepstral coefficients are shown in Fig.2.4, and some of vocabulary is illustrated in Fig.2.5.

In Fig.2.5(a), two components in the speech magnitude spectrum can be identified：a "slowly varying" part due to the speech system, $|\Theta(\omega)|$, and a "quickly varying" part due to the excitation $|E(\omega)|$. These two components are combined by multiplication. In Fig.2.5(b), two components in log-spectral domain are combined by addition. When the DCT is taken, two components in Fig.2.5(c) are approximately separated into two parts, and then can be easily extracted.

$$\begin{tikzcd} s[\mathrm{n}] \arrow[r] & \boxed{\text{FFT}} \arrow[r] & \boxed{\log|\cdot|} \arrow[r] & \boxed{\text{DCT}} \arrow[r] & \boxed{L[n]} \arrow[r] & C_\theta[n] \end{tikzcd}$$

Fig. 2.4 Computation of cepstral coefficients.



(a)
$|S(\omega)|$
"Slow" variations (envelope)
"Fast" variations (pulses)
=
$|E(\omega)|$
×
$|\Theta(\omega)|$
π ω
Excitation responsible for "fast" spectral variations
Problem! (nonlinear combination)
Vocal system responsible for "slow" variations
(b)
$\log|S(\omega)| = C_s(\omega)$
$\log|E(\omega)|$
+
$\log|\Theta(\omega)|$
Problem solved! (linear combination)
(c)
$c_s(n)$
Low quefrency energy
High quefrency
$c_e(n)$ [approx]
$c_\theta(n)$
P 2P 3P n

Figure 2.5 Motivation behind the real cepstrum and some of the accompanying vocabulary.[14]

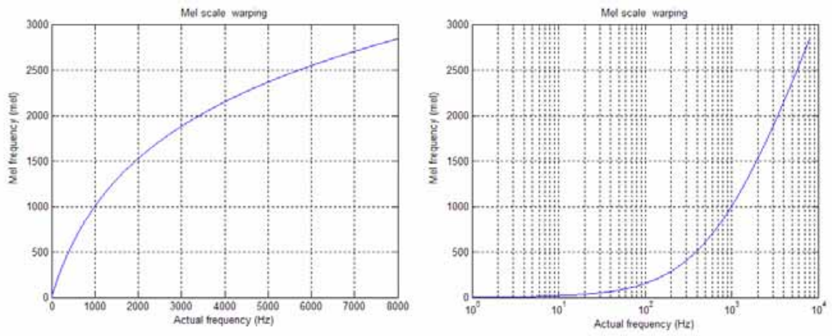
2.4 Mel-Frequency Cepstrum Coefficient

After doing FFT processing, the output will be a set of 2^N-point data. For example, the case of 16 KHz, 25ms per frame, will contain 256 point data (from 0 to π). Then, M filter banks will be utilized to reduce the computational complexity from 256 points to M points, which could be simply achieved by equally dividing 256 points data into *M* groups. Each group is represented by its average and thus, *M* points are obtained. However, it is not suitable for our perceptual hearing.

It is known that human perception of the frequency content of sounds, either for pure tones or for speech signals, does not follow a linear scale. Among several kinds of nonlinear transformation, Mel-scale has been widely used in modern speech recognition systems. The Mel-scale, obtained by Stevens and Volkman[15,16], is a perceptual scale and it attempts to mimic the human ear in terms of the manner that the frequencies are sensed and resolved. Mel is a unit of measure of perceived pitch or frequency of a tone. The precise meaning of the Mel scale becomes clearly by examining the following experiment. In the experiment, the reference frequency was selected as 1 KHz and set to be equal to 1000 mels. Then, by increasing the frequency, the subjects were asked to tell when they perceived a pitch twice of the reference, i.e., a pitch of 2000 mels. Once the subjects confirm and then the corresponding frequency will be recorded. For instance, if the pitch of 2000 mels they perceived is at 3.5 KHz, then, the frequency 3.5 KHz is mapped to 2000 mels. With the same procedure, the frequencies related to the pitches of 10 times, half, 1/10, etc. could be obtained and recorded. The formulation of Mel scale is approximated by

$$B(f) = 2595 \log_{10}(1 + \frac{f}{700}) \tag{2-9}$$

where $B(f)$ is a function mapping the actual frequency to the Mel frequency, shown in Fig.2.6.



(a) (b)

Fig. 2.6 Frequency warping according to Mel scale (a) linear frequency scale (b) logarithm frequency scale.

According to the Mel scale warping, the Mel filter bank is then designed by placing *M* triangular filters non-uniformly along frequency axis to simulate the human hearing. The *m*-th triangular filter is represented as

$$H_m(k)=\begin{cases} 0 & k<f(m-1) \\ \dfrac{(k-f(m-1))}{(f(m)-f(m-1))} & f(m-1)\le k<f(m) \\ \dfrac{(f(m+1)-k)}{(f(m+1)-f(m))} & f(m)\le k<f(m+1) \\ 0 & k\ge f(m+1) \end{cases} \qquad (2\text{-}10)$$

$$1\le k\le \frac{N}{2}, 1\le m\le M$$

where *N* is the number of FFT points, and *M* is the number of Mel filter bank. The boundary frequency *f*(*m*) in (2-10) can be calculated as

$$f(m)=\left(\frac{N}{F_s}\right)B^{-1}\left(B(f_l)+m\frac{B(f_h)-B(f_l)}{M+1}\right), \qquad 1\le m\le M \qquad (2\text{-}11)$$

where f_l and f_h are the lowest and highest frequencies of the filter bank, F_s is the sampling rate of the input speech signal, and $B^{-1}(\cdot)$ is an inverse function of (2-9), represented as

$$B^{-1}(b)=700(10^{b/2595}-1) \qquad (2\text{-}12)$$

with the Mel frequency *b*. It's noted that if $f(m)$ is not an integer, then transform it to the nearest integer towards infinity.

In this thesis, *M* is set as 20 or 24 when the sampling rate of the input speech signal is 8 KHz or 16 KHz. The Mel filter banks are shown in Fig. 2.7 and it is obvious that pass-band in low frequency is narrower than high frequency. It is because that our perceptual hearing is more sensitive to low frequency and some important information on the vocal tract, such as the first formant, hides in the low frequency. The narrow pass-band in low frequency can protect such information during compression; on the other hand, for high frequency, a wide pass-band can be used to reduce the data complexity and not to influence the original characteristics.

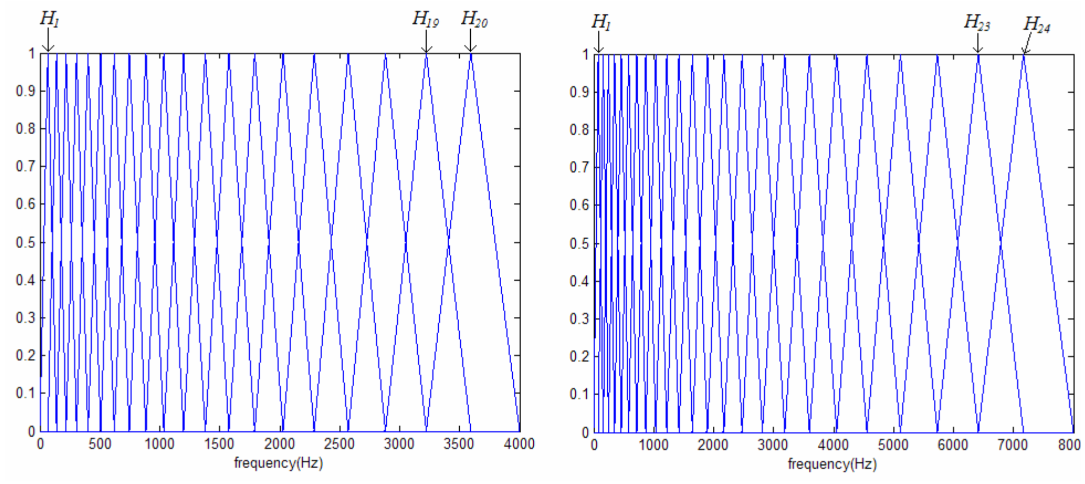


Fig.2.7 The Mel filter banks for (a) 8 KHz (b) 16 KHz.

The log-magnitude Mel spectrum is then derived by multiplying each FFT magnitude coefficients with corresponding Mel filter gains and taking logarithm as

$$Y(m) = \log\left(\sum_{k=0}^{N/2-1} |S(k)| \cdot H_m(k)\right) \tag{2-13}$$

where $S(k)$ is the FFT of the input speech $s[n]$. Next, the discrete-cosine transform will be applied to derive the Mel frequency cepstrum $c_t[i]$ as

$$c_t[i] = \sqrt{\frac{2}{M}} \sum_{m=1}^{M} Y(m) \cos\left(\frac{i\pi}{M}\left(m - \frac{1}{2}\right)\right), \quad i = 0,1,\cdots,L \tag{2-14}$$

For speech recognition, typically only the first 13 cepstrum coefficients, including the log-energy term, are used, and these coefficients are defined as Mel Frequency Cepstral Coefficients (MFCC). In addition to the MFCC, the dynamic features, delta-MFCC and delta-delta-MFCC, are usually employed in practical speech recognition to obtain the dynamic evolution of the speech signal, i.e. the temporal information of feature vector $c_t[i]$, and to cancel the channel effect. These two features are represented as (2-15) and (2-16).

$$\Delta c_t[i] = \frac{\sum_{p=1}^{P} p \cdot \left(c_{t+p}[i] - c_{t-p}[i]\right)}{2\sum_{p=1}^{P} p^2}, \qquad i = 0,1,\ldots,L \tag{2-15}$$

$$\Delta^2 c_t[i] = \frac{\sum_{p=1}^{P} p \cdot \left(\Delta c_{t+p}[i] - \Delta c_{t-p}[i]\right)}{2\sum_{p=1}^{P} p^2}, \qquad i = 0,1,\ldots,L \tag{2-16}$$

where P represents the maximum number of frames shifted for the reference frame**.** It should be properly chosen, because too small P may imply too close frames and therefore the dynamic characters may not be properly extracted; too large P may imply frames describing too different states, i.e. different acoustic phenomena. Typically, P is usually chosen as 2. Since (2-15) and (2-16) relies on past and future speech parameter values, some modification is needed at the beginning and end of the speech. A simple first order differences will be used to solve this problem, that is

$$\begin{cases} \Delta c_t[i] = c_{t+1} - c_t, & t < P \\ \Delta c_t[i] = c_t - c_{t-1}, & t \geq T - P \end{cases} \tag{2-17}$$

$$\begin{cases} \Delta^2 c_t[i] = \Delta c_{t+1} - \Delta c_t, & t < P \\ \Delta^2 c_t[i] = \Delta c_t - \Delta c_{t-1}, & t \geq T - P \end{cases} \tag{2-18}$$

where T is the total number of frames.

In this thesis, each feature used for recognition contains 13 MFCCs, 13 Delta-MFCCs and 13 Delta-Delta MFCCs, and the feature vector will be represented as $\left[c_t[0]\, c_t[1] \ldots c_t[12]\, \Delta c_t[0]\, \Delta c_t[1] \ldots \Delta c_t[12]\, \Delta^2 c_t[0]\, \Delta^2 c_t[1] \ldots \Delta^2 c_t[12]\right]^T$. A stochastic model will be introduced later, and the feature vectors extracted in this chapter will be treated as the observation data to that model in training and recognition phase.

Chapter 3

Speech Modeling and Recognition

Speech recognition is typically a problem of pattern recognition. The basic concept is to compare the incoming test speech signals with the reference signals trained before in database, and find the most possible signal as the recognition result. In stead of using the signals directly, the features extracted by the front-end process described in chapter 2 are utilized. These features will be the inputs of the recognizer, and some methods, such as Dynamic Time Warping (DTW)[17] and Hidden Markov Model (HMM)[18,19], will be employed to determine what the recognition result is.

The DTW method is to find the optimal projection with respect to time for the test speech data to each trained speech model in database, and the most similar model will be regarded as the recognition result, and the result may not be adopted if the score doesn't exceed the threshold. The speech models applied to DTW will be word-level or sentence-level, and the dynamic programming algorithm will be used to find the optimal path. DTW has been successfully employed in applications with small vocabulary size[20,21,22]. However, it is not an efficient method for large vocabulary size, because too large database will be needed.

Contrary to DTW, HMM is a statistic method using probability to determine what the recognition result is. HMM has been widely applied as speech model in ASR (automatic speech recognition) in past several years because of its wonderful ability of characterizing the speech signal in a mathematically tractable way and better performance comparing to other methods. The underlying assumption of HMM is that the speech signal can be characterized as a parametric random process, and the parameters of the stochastic process can be estimated in a precise and well-defined scheme. In the fallowing sections, HMM and how to use it to do speech recognition will be described in detail.

3.1 Definition of Hidden Markov Model

Before describing the Hidden Markov Model, the Markov chain will be introduced first. The Markov chain is a class of random process that incorporates a minimum amount of memory instead of the completely memory. For example, let $\{x_1, x_2, ..., x_N\}$ be an observation sequence of random variables from a finite discrete set O, then based on Bayes' rule, the probability of observing the sequence is

$$P(x_1, x_2, ..., x_N) = P(x_1) \cdot P(x_2 \mid x_1) \cdot P(x_3 \mid x_1, x_2) \cdot ... \cdot P(x_N \mid x_1, x_2, ..., x_{N-1}) \qquad (3\text{-}1)$$

Under the assumption that $\{x_1, x_2, ..., x_N\}$ forms a first-order Markov chain, (3-1) will be reduced as

$$P(x_1, x_2, ..., x_N) = P(x_1) \cdot P(x_2 \mid x_1) \cdot P(x_3 \mid x_2) \cdot ... \cdot P(x_N \mid x_{N-1}) \qquad (3\text{-}2)$$

If x_i is associated to a state, the Markov chain can be represented by a finite state process, which is also called the observed Markov model and with parameters described as fallows

$$a_{ij} = P(x_t = S_j \mid x_{t-1} = S_i), \;\; 1 \le i, j \le \textit{total number of states} \qquad (3\text{-}3)$$

$$\pi_i = P(x_1 = S_i), \; 1 \le i \le \textit{total number of states} \qquad (3\text{-}4)$$

where a_{ij} is the transition probability from state *i* to *j*, and π_i is the initial probability that the Markov chain will start at state *I*. Besides, the notion $x_t = S_i$ means that the observed variable is in state *i* at time *t*. Both parameters must be under the following constraints

$$\sum_{j=1}^{M} a_{ij} = 1, \; 1 \le i \le M \qquad (3\text{-}5)$$

$$\sum_{i}^{M} \pi_i = 1 \qquad (3\text{-}6)$$

where *M* is the total number of states.

In the observed Markov model, each state corresponds to a deterministically observed variable x_t, which means the output sources in any given state is not random. Fig 3.1 is an example of three state observed Markov model. There are 3 states, S_1, S_2 and S_3 in this model, and they generate R, G and B, respectively. The state-transition probability matrix is

$$A = \{a_{ij}\} = \begin{bmatrix} 0.6 & 0.2 & 0.2 \\ 0.5 & 0.3 & 0.2 \\ 0.4 & 0.1 & 0.5 \end{bmatrix} \tag{3-7}$$

and the initial state probability matrix is

$$\pi = \{\pi_i\} = [0.5 \quad 0.2 \quad 0.3]^T \tag{3-8}$$

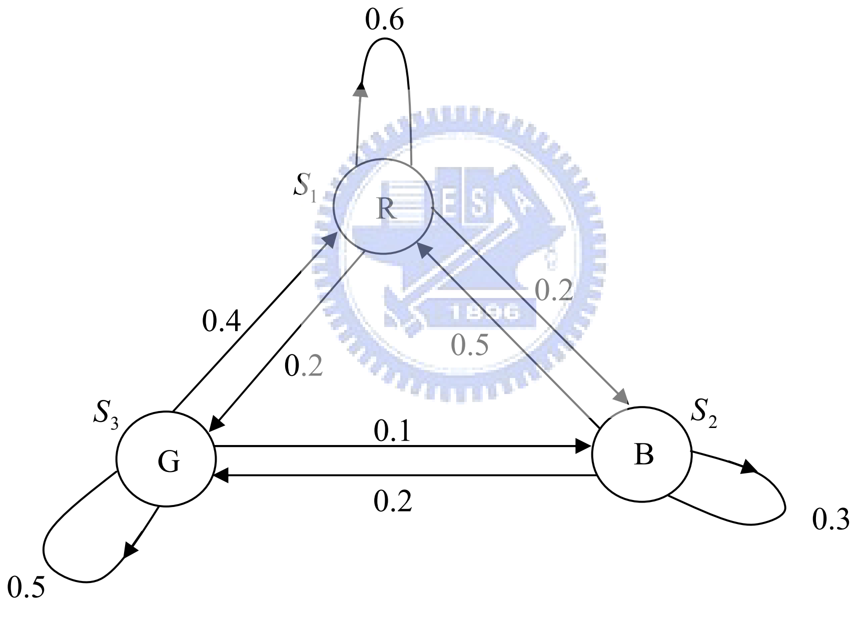


Fig. 3.1 An example for the observed Markov model.

Thus, the probability of observing the sequence $\{R, B, G, B, R, G\}$ can be calculated as $P(R,B,G,B,R,G) = \pi_1 \cdot a_{12} \cdot a_{23} \cdot a_{32} \cdot a_{21} \cdot a_{13} = 0.5 \cdot 0.2 \cdot 0.2 \cdot 0.1 \cdot 0.5 \cdot 0.2 = 2 \times 10^{-4}$

An extension to the observed Markov model introduces a non-deterministic process that generates output observation variables in any given state. Thus, depending only on the observation sequences is impossible to know the real state sequences. This new model is known as the Hidden Markov Model. The word ‘hidden’ implicitly shows that the desired state sequence is hidden behind the observation sequence, the only data that can be collected. In HMM, a new parameter is introduced as

$$b_i(x_t) = P(x_t = S_i),\ 1 \le i \le \textit{total number of states} \tag{3-9}$$

where $b_i(x_t)$ is the probability of the observed variable in state i at time t, and similar to (3-5) and (3-6), it must be under the following constraint

$$\sum_{i=1}^{M} b_i(x_t) = 1 \tag{3-10}$$

Fig 3.2 is an example of a three state hidden Markov model. The parameters A and π are the same as the observed Markov model mentioned before, and the observation matrix is

$$B = \begin{bmatrix} 0.7 & 0.1 & 0.3 \\ 0.1 & 0.6 & 0.3 \\ 0.2 & 0.3 & 0.4 \end{bmatrix} = \begin{bmatrix} P_1(R) & P_2(R) & P_3(R) \\ P_1(B) & P_2(B) & P_3(B) \\ P_1(G) & P_2(G) & P_3(G) \end{bmatrix} \tag{3-11}$$

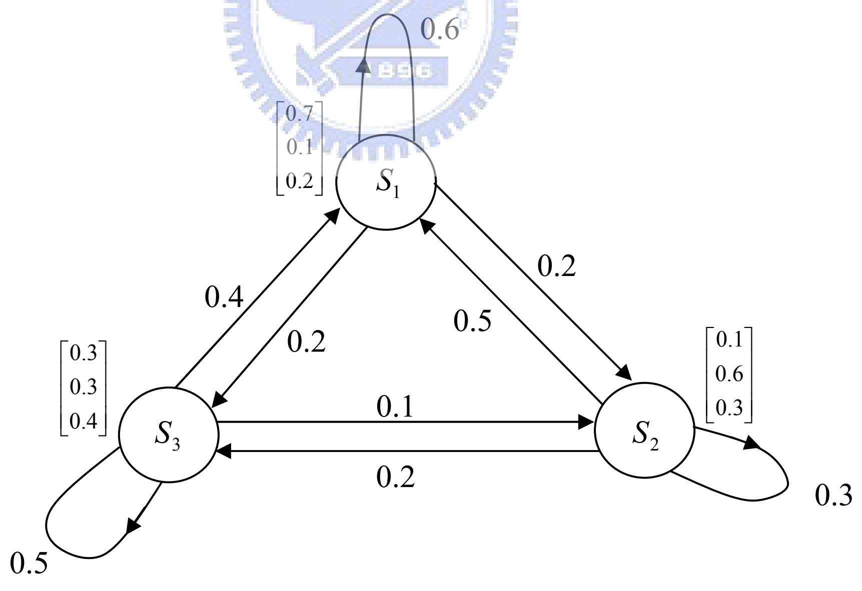


Fig. 3.2 An example for the hidden Markov model.

Thus, when we see the sequence $\{R, B, G, B, R, G\}$, the corresponding state sequence can not be uniquely determined. The possible number of state sequences will be 729, and the sequence with the largest probability will be the desired one. In this example, the corresponding state sequence will be $\{1,2,1,2,1,1\}$, and the relative probability equals to 2.1168×10^{-5}. In fact, it is not necessary to calculate all possible cases; instead, the decoding method, called Viterbi search[23], is widely employed due to its high efficiency. The Viterbi search will be introduced in Section 3.3.2.

3.2 HMM in Speech Recognition

As mentioned in Chapter 2, the speech signals are short time stationary, and several states will be generated after certain of training procedures. Signals with similar statistic properties will be classified into the same states. In traditional method, such as DTW, the speech data templates are usually of words or sentences, which usually results in an unreasonable database, especially for a task with a tremendous large of templates, such as the filed of medicine. Nevertheless, using HMM, smaller units, like syllables and phonemes, can be combined to form words or sentences which are desired to be recognized. Therefore, the dimension of database could be highly reduced since different words or sentences could be constructed by the same syllables or phonemes. There are some reasons that the DTW could not adopt the small units, like syllables or phonemes. First, DTW need to cut the unit by hand which is difficult to cut precisely. Second, even though the unit is cut precisely, it is still difficult to adjust the templates different in length to a suitable one. Therefore, HMM is superior to DTW in most speech recognition applications.

In order to execute the speech recognition, an ASR (automatic speech recognition) machine processes the sequence of speech signals extracted from speech recordings, and tries to decode their linguistic information, i.e. to recognize the speaker’s utterance. The basic ASR scheme is shown in figure 3.3. The first step of the ASR is feature extraction from the input speech signals, described in Chpter2. These feature vectors are called observations and denoted as $\boldsymbol{O}$, and $\boldsymbol{O}=(\boldsymbol{o}_1,\boldsymbol{o}_2,...,\boldsymbol{o}_T)$ is a sequence of T observations. The spoken language can be thought as a sequence of units, called linguistic units, and each unit can be trained and then modeled as a HMM model, denoted as Θ. Let S be a given sequence of linguistic units from the database, and the aim of the ASR is to find the correct linguistic units S^* from a given observation sequence. Therefore, the ASR machine may be divided into two distinct phases. One is to build HMM speech models Θ based on the correspondence between the observation sequences $\boldsymbol{O}$ for training and the known linguistic units S; the other is to recognize a speech by the trained HMM models Θ and by the observation sequences $\boldsymbol{O}$ of the speech.

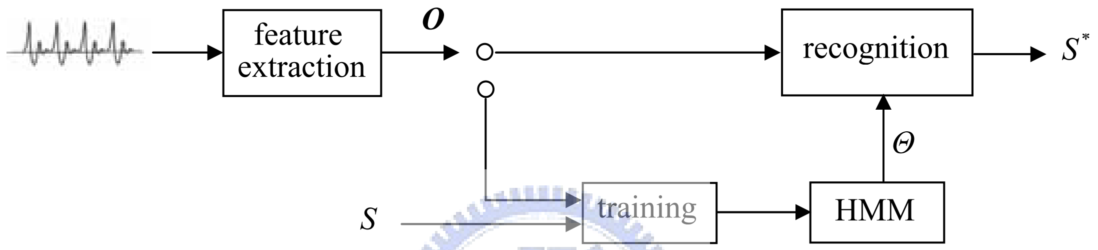


Fig. 3.3 Scheme of an ASR system functionality.

The linguistic units in Mandarin can be in word level, such as \學校\, \學生\, \學習\, syllable level, such as \學\, \校\, \生\, and phoneme level, such as \ㄒ\, \一ㄠ\, \ㄕ\. Small units can be combined to form a larger unit. For example, \ㄒ\ and \ㄩㄝ\ can be combined to represent \ㄒㄩㄝ\. Choosing a proper level is important since it can make the ASR machine work more efficiently. Though small units can reduce the size of database, it needs a lot of corpus to train proper models. Therefore, for a small task, like voice command, a word level model is enough, and it is not necessary to use phoneme level model which may complicate the ASR machine. On the other hand, a phoneme level model is better than a word level model for a large task, because using word level, even syllable level, would need a tremendous database.

A left-to-right HMM model will be applied to represent the linguistic units. Fig. 3.4 shows an example of 3-emtting state left-to-right HMM. The way to choose a

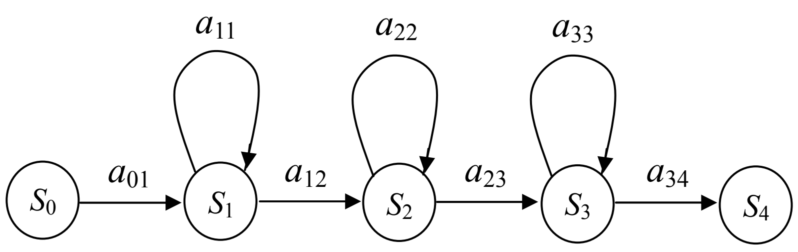


Fig 3.4 A 3 state left-to-right HMM.

reasonable number of states depends on the selected units; typically, a vowel needs 7 states (first and last are non-emitting state), and a consonant needs 5 states (first and last are non-emitting state, too). These HMM have a topology whereby transitions may only go to the same state or to the next state, i.e. no skips are allowed. Moreover, the two non-emitting state S_0 and S_N constrain the HMM to start in state S_1 and terminate in state S_N. Then, the phoneme level units can be easily combined to form the larger units by connect the state S_N of current model and the state S_0 of next model.

The HMM can be typically classified into two types, one is discrete-HMM (DHMM) and the other is continuous-HMM (CHMM). The difference between DHMM and CHMM is the way to obtain the observation probability $b_i(\boldsymbol{o}_t)$. In DHMM, a codebook will be utilized to determine the observation probability. After training procedures, all observed vectors in state i will be categorized into a finite vector sets $\boldsymbol{V} = \{\boldsymbol{V}_1, \boldsymbol{V}_2, \ldots, \boldsymbol{V}_N\}$, and the corresponding probability of each vector set $b_i(\boldsymbol{V}_n)$ will be obtained at the same time. Then, $b_i(\boldsymbol{o}_t)$ will be equal to $b_i(\boldsymbol{V}_n)$ if $\boldsymbol{o}_t$ belongs to $\boldsymbol{V}_n$. Therefore, the observation probability is discrete distributed. Contrary to DHMM, a continuous probability distribution will be utilized to calculated the observation probability $b_i(\boldsymbol{o}_t)$ in CHMM. The mixture Gaussian distribution[24] will be applied to mimic the continuous distribution, and $b_i(\boldsymbol{o}_t)$ could be figured out as

$$
\begin{aligned}
b_i(\boldsymbol{o}_t) &= \sum_{m=1}^{M} w_{jm} N(\boldsymbol{\mu}_{jm}, \boldsymbol{\Sigma}_{jm}, \boldsymbol{o}_t) \\
&= \sum_{m=1}^{M} w_{jm} \left[\frac{1}{\left(\sqrt{2\pi}\right)^L \left|\boldsymbol{\Sigma}_{jm}\right|^{\frac{1}{2}}} exp\left(-\frac{1}{2} \left(\boldsymbol{o}_t - \boldsymbol{\mu}_{jm}\right)^T \boldsymbol{\Sigma}_{jm}^{-1} \left(\boldsymbol{o}_t - \boldsymbol{\mu}_{jm}\right) \right) \right]
\end{aligned}
\tag{3-12}
$$

where $N(\cdot)$ is a Gaussian function, L is the dimension of the observation vector and M is the number of mixtures. As for $\boldsymbol{\mu}_{jm}$, $\boldsymbol{\Sigma}_{jm}$ and w_{jm}, they respectively indicate the mean vector, the covariance matrix and the weighting coefficient of the m_{th} mixture component in state S_j. The observations are assumed to be independent to each other, so the covariance matrix can be reduced to a diagonal form as

$$
\boldsymbol{\Sigma}_{jm} = \begin{bmatrix} \sigma_{jm}(1) & 0 & \cdots & 0 \\ 0 & \sigma_{jm}(2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_{jm}(L) \end{bmatrix} \tag{3-13}
$$

Then, the observation probability $b_i(\boldsymbol{O}_t)$ can be rewritten as

$$
b_i(\boldsymbol{o}_t) = \sum_{m=1}^{M} w_{jm} \left[\frac{1}{\left(\sqrt{2\pi}\right)^L \left[\prod_{l=1}^{L} \sigma_{jm}(l)\right]^{\frac{1}{2}}} \prod_{l=1}^{L} exp\left(-\frac{\left(\boldsymbol{o}_t(l) - \boldsymbol{\mu}_{jm}(l)\right)^2}{2\sigma_{jm}(l)} \right) \right] \tag{3-14}
$$

As for the weighting coefficient w_{jm}, it must satisfy

$$
\sum_{m=1}^{M} w_{jm} = 1 \text{ and } w_{jm} \geq 0,\ 1 \leq m \leq M \tag{3-15}
$$

In this thesis, the CHMM is adopted, whose model Θ contains the following parameters: $\boldsymbol{\pi}$, $\boldsymbol{A}$, $\boldsymbol{w}$, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. The problems of probability calculation, decoding and parameter estimating for HMM will be described in next section.

3.3 The Three Basic Problems for HMM

The three basic problems for HMM are probability evaluation, decoding and parameter estimating, and their descriptions, as follows:

1. Given a observation sequence $\boldsymbol{O}$ = $(\boldsymbol{o}_1,\boldsymbol{o}_2,\ldots,\boldsymbol{o}_T)$, and a HMM model $\Theta=(\boldsymbol{A},\boldsymbol{B},\pi)$, how to efficiently compute $P(\boldsymbol{O}|\Theta)$?
2. Given a observation sequence $\boldsymbol{O}$ = $(\boldsymbol{o}_1,\boldsymbol{o}_2,\ldots,\boldsymbol{o}_T)$, and a HMM model $\Theta=(\boldsymbol{A},\boldsymbol{B},\pi)$, how to determine the state sequence $\boldsymbol{S}$ = $(S_1,S_2,\ldots,S_T)$ such that $P(\boldsymbol{O},\boldsymbol{S}|\Theta)$ is maximum?
3. How to adjust a new model Θ' such that $P(\boldsymbol{O}|\Theta') > P(\boldsymbol{O}|\Theta)$ until $P(\boldsymbol{O}|\Theta')$ is maximum?

The solutions to these problems will be described more explicitly in the following sub sections.

3.3.1 Solution to Problem 1-The Forward/Backward Algorithm

The most straightforward way is listing all possible state sequences and summing up their probabilities. It can be shown as

$$P(\boldsymbol{O}\,|\,\Theta)=\sum_{\text{all }\mathbf{S}}P(\boldsymbol{O},\boldsymbol{S}\,|\,\Theta)=\sum_{\text{all }\mathbf{S}}\pi_{s_1}b_{s_1}(\boldsymbol{o}_1)a_{s_1s_2}b_{s_2}(\boldsymbol{o}_2)\ldots a_{s_{T-1}s_T}b_{s_T}(\boldsymbol{o}_T) \qquad (3\text{-}16)$$

If the number of states is N and the time length is T, then, the number of possible sequences will be N^T, and will require $2TN^T$ computations. Obviously, it is not an efficiently method. A more efficient method, called forward algorithm, will be used to solve this problem. First, define the forward probability:

$$\alpha_t(i)=P(\boldsymbol{o}_1^t,S_t=i\,|\,\Theta) \qquad (3\text{-}17)$$

$\alpha_t(i)$ is the probability that the HMM is in state i at time t having generated partial observation $\boldsymbol{o}_1{}^t$ (namely $\boldsymbol{o}_1\boldsymbol{o}_2\ldots\boldsymbol{o}_t$). $\alpha_t(i)$ can be calculated inductively as follows:

Step 1: Initialization

$$\alpha_1(i)=\pi_i b_i(\boldsymbol{o}_1) \quad 1\le i\le N \qquad (3\text{-}18)$$

if under the constraint that HMM starts in state 1, (3-18) could be reduced to

$$\alpha_1(i)=\pi_1 b_1(\boldsymbol{o}_1) \qquad (3\text{-}19)$$

Step 2: Induction

$$\alpha_t(j) = \left[\sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij}\right] b_j(\boldsymbol{o}_t) \quad 2 \le t \le T, 1 \le j \le N \tag{3-20}$$

Step 3: Termination

$$P(\boldsymbol{O} \mid \Theta) = \sum_{i=1}^{N} \alpha_T(i) \tag{3-21}$$

if under the constraint that HMM end in state N, (3-21) could be reduced to

$$P(\boldsymbol{O} \mid \Theta) = \alpha_T(N) \tag{3-21}$$

It requires about N^2T computations which are much less than direct calculation. In a similar manner, define the backward probability as

$$\beta_t(i) = P(O_{t+1}^T \mid S_t = i, \Theta) \tag{3-22}$$

$\beta_t(i)$ is the probability of generating partial observation $\boldsymbol{o}_{t+1}{}^T$ given that the HMM is in state i at time t, and it can be calculated inductively as follows:

Step 1: Initialization

$$\beta_T(i) = 1/N, \quad 1 \le i \le N \tag{3-23}$$

Step 2: Induction

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(\boldsymbol{o}_{t+1}) \beta_{t+1}(j), \quad t = T-1, T-2, \ldots, 1 \quad 1 \le i \le N \tag{3-24}$$

Step 3: Termination

$$P(\boldsymbol{O} \mid \Theta) = \sum_{i=1}^{N} \beta_1(i) \tag{3-25}$$

The probability calculation is not useful itself, but it and the forward/backward algorithm are important parts of the solution to the parameter estimating.

3.3.2 Solution to Problem 2-The Viterbi Algorithm

In this problem, the aim is to find the best state sequence, which is the desired one in many applications. The forward algorithm described in the previous section can not find out such a state sequence, and the Viterbi algorithm can be applied to solve this problem efficiently. First, define the best-path probability:

$$V_t(i) = P(\boldsymbol{o}_1^t, S_1^{t-1}, S_t = i \mid \Theta) \tag{3-26}$$

$V_t(i)$ is the probability of the most likely state sequence at time t, which has generated the observation $\boldsymbol{o}_1^t$ (namely $\boldsymbol{o}_1\boldsymbol{o}_2\ldots\boldsymbol{o}_t$) and ends in state i . The best-path probability $V_t(i)$ can be calculated inductively as follows:

Step 1: Initialization

$$V_1(1) = \pi_1 b_1(\boldsymbol{o}_1) \tag{3-27}$$

$$V_1(2) = V_1(3) = \ldots = V_1(N) = 0$$

$$B_1(i) = 0 \quad (\boldsymbol{B} \text{ is the matrix to store the state})$$

Step 2: Induction

$$V_t(j) = \underset{i=j,j-1}{Max}[V_{t-1}(i)a_{ij}]\cdot b_j(\boldsymbol{o}_t) \quad 2 \le t \le T, 1 \le j \le N \tag{3-28}$$

$$B_t(j) = S_{i*} \quad \text{where } i^* = \underset{i=j,j\text{-}1}{Arg}[V_{t-1}(i)a_{ij}] \tag{3-29}$$

Step 3: Termination

$$\text{The best score} = V_T(N)$$

$$B_T(N) = S_{i*} \quad \text{where } i^* = \underset{i=N,N-1}{Arg}[V_{T-1}(i)a_{ij}] \tag{3-30}$$

Step 4: Backtracking

$$S_T^* = S_N$$

$$S_t^* = B_{t+1}(S_{t+1}^*) \quad t = T-1, T-2, \ldots, 1$$

$$\boldsymbol{S}^* = (S_1^*, S_2^*, \ldots, S_T^*) \text{ is the best sequence}$$

It is noted that the some constraints has been added, i.e., the transition has been limited to stay in current state or transited to the next state. The Viterbi algorithm is similar to the forward algorithm and the major difference is the maximum operator in (3-28), which is used in place of the summing procedure in (3-20).

3.3.3 Problem 3-Parameter Estimation(HMM training)

Given a HMM $\Theta = (\boldsymbol{A},\boldsymbol{B},\boldsymbol{\pi})$ and a set of observations $\boldsymbol{O} = (\boldsymbol{o}_1,\boldsymbol{o}_2,\ldots,\boldsymbol{o}_T)$, the purpose of estimation is to adjust the model parameters so that the $P(\boldsymbol{O}\mid\Theta)$ is local maximized by using an iterative procedure. The initial HMM model will be produced by modified K-means[25] and Viterbi algorithm. Then, the Baum-Welch algorithm[26] (or called forward-backward algorithm) will be utilized to train the HMMs.

Before applying the training algorithms, some preparing works should be done. First, a set of speech data and their associated transcriptions should be prepared and they should be transformed into the MFCC feature vectors. These feature vectors will be treated as the observations of the HMM. Second, the number of states and the number of mixtures in a HMM must be determined. Then, the first step of training is to produce proper initial HMM models. The procedures to get initial HMM models could be divided into two manners depending on whether the boundary information is available. If the boundary information is available, such as Fig.3.5, the estimation of the HMM parameters would be more precise. The transcriptions with boundary information should be saved in text files as the form in Fig.3.6 (a). It is noted that even if the boundary information are not available, the transcriptions also should be prepared and saved as in text files as the form in Fig.3.6 (b).

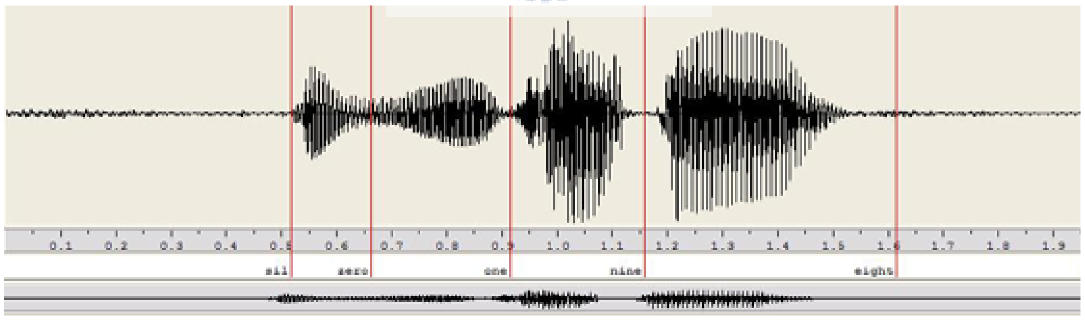
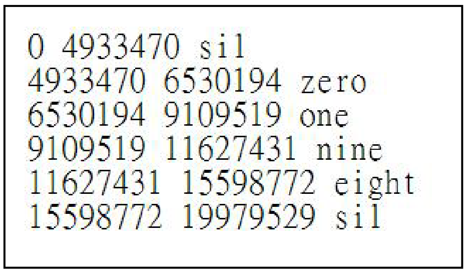
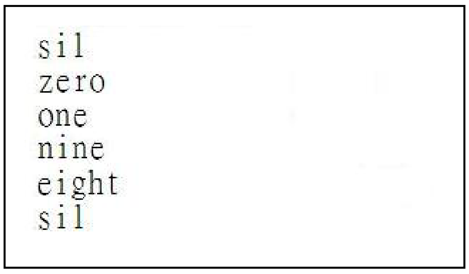


Fig. 3.5 Boundary information: the red line indicates the boundary information.



```
0 4933470 sil
4933470 6530194 zero
6530194 9109519 one
9109519 11627431 nine
11627431 15598772 eight
15598772 19979529 sil
```

(a)



```
sil
zero
one
nine
eight
sil
```

(b)

Fig 3.6 (a) transcription with boundary information. (b) transcription without boundary information.

The block diagram of getting an initial HMM model with boundary information is shown in Fig. 3.7. On the first iteration, the training data of a specific model are uniformly divided into N segments, where N is the number of states of such specific model. Then, the HMM parameters π_i and a_{ij} can be first estimated as follows

$$\pi_i = \frac{\text{number of observations in state } i \text{ at time} = 1}{\text{number of obeservations at time} = 1} \tag{3-31}$$

$$a_{ij} = \frac{\text{number of transitions from state } i \text{ to state } j}{\text{number of transitions from state } i} \tag{3-32}$$

It is noted that in implementation, the π vector will be set as 1 with leading element and 0 with others, i.e. the HMM is only allowed to start at state 1.

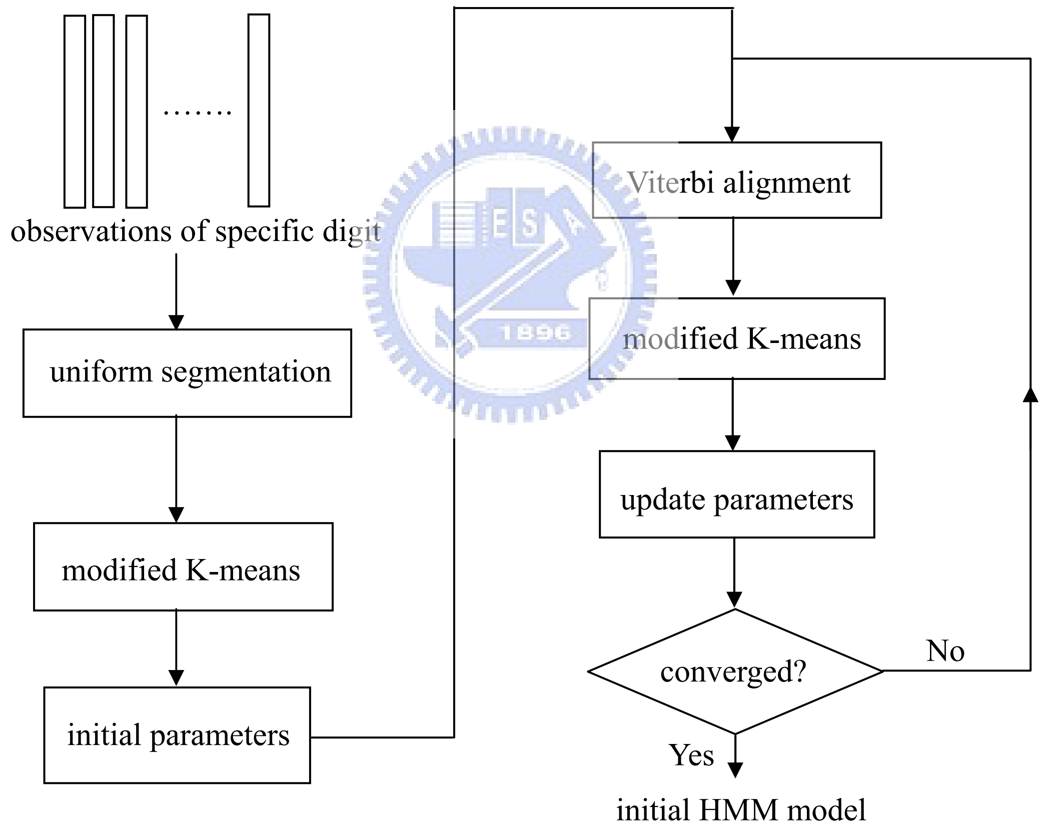


Fig. 3.7 Block diagram of getting an initial HMM model with boundary information.

Then, the modified K-means algorithm will be utilized to estimate the parameters $\boldsymbol{w}$, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. From the modified K-means algorithm, the observations will be clustered into M groups, where M is the number of mixtures in a state. The parameters can be estimated as follows

$$w_{jm} = \frac{\text{number of observations classified in cluster } m \text{ in state } j}{\text{number of observations classified in state } j} = \frac{N_{jm}}{N_j} \tag{3-33}$$

$$\begin{aligned}\mu_{jm} &= \text{mean of the observartions classified in cluster } m \text{ in state } j \\ &= \frac{1}{N_{jm}} \cdot \sum \boldsymbol{o}_n, \text{ where } n \in \text{cluster } m \text{ in state } j\end{aligned} \tag{3-34}$$

$$\begin{aligned}\boldsymbol{\Sigma}_{jm} &= \text{covariance matrix of the observations classified in cluster m in state j} \\ &= \frac{1}{N_{jm}} \cdot \sum (\boldsymbol{o}_n - \mu_{jm})(\boldsymbol{o}_n - \mu_{jm})^T, \text{ where } n \in \text{cluster } m \text{ in state } j\end{aligned} \tag{3-35}$$

With the initial parameters, next, the uniform segmentation will be replaced by the Viterbi algorithm to divide the training data into states more precisely. The iterative procedures of Viterbi alignment, modified K-means and update parameters will be repeated until the parameters are converged. Then, the initial HMM models are created.

In the case that the boundary information is not available, the method to get the initial HMM models would be easier that the parameters in each state are initialized to be identical. The mean and the covariance are set to be equal to the global mean and variance. As for the initial probability vector π, the state transition matrix $\boldsymbol{A}$ and the weighting coefficient vector $\boldsymbol{w}$, there is no information to compute these parameters; therefore, they would be set arbitrarily. Then, the initial HMM models are created. It is noted that the performance of the recognizer using the HMMs trained in this case would be worse.

After the initial HMM models have been created, the Baum-Welch algorithm will be utilized to get the final HMM models. The Baum-Welch algorithm, known as the forward-backward algorithm is the core of HMM training. Three variables, $\xi_t(i,j)$, $\gamma_t(i)$ and $\gamma_t(j,k)$, will be defined first. The variable $\xi_t(i,j)$ is defined as

$$\begin{aligned}\xi_t(i,j) &= P(q_t = S_i, q_{t+1} = S_j \mid \boldsymbol{O}, \Theta) = \frac{P(q_t = S_i, q_{t+1} = S_j, \boldsymbol{O} \mid \Theta)}{P(\boldsymbol{O} \mid \Theta)} \\ &= \frac{\alpha_t(i) a_{ij} b_j(\boldsymbol{o}_{t+1}) \beta_{t+1}(j)}{P(\boldsymbol{O} \mid \Theta)}\end{aligned} \tag{3-36}$$

which is the probability of being in state i at time t and state j at time t+1. The $\alpha_t(i)$

and $\beta_t(i)$ are the forward probability and the backward probability respectively, which have been introduced in 3.3.1, and the calculation of $P(\boldsymbol{O}|\Theta)$ is also shown in 3.3.1. The variable $\gamma_t(i)$ is defined as

$$\gamma_t(i) = P(q_t = S_i \mid \boldsymbol{O}, \Theta) = \sum_{j=1}^{N} \xi_t(i,j) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)} \tag{3-37}$$

which is the probability of being in state i at time t. The variable $\gamma_t(j,k)$ is defined as

$$\begin{aligned}\gamma_t(j,k) &= P(q_t = S_j, m_t = k \mid \boldsymbol{O}, \Theta)\\ &= \left[\frac{\alpha_t(j)\beta_t(j)}{\sum_{s=1}^{N} \alpha_t(s)\beta_t(s)}\right]\left[\frac{w_{jk} b_{jk}(\boldsymbol{o}_t)}{\sum_{k=1}^{M} w_{jk} b_{jk}(\boldsymbol{o}_t)}\right]\end{aligned} \tag{3-38}$$

which is the probability of being in state j at time t with the k-th mixture component accounting for $\boldsymbol{o}_t$. The M is the total number of mixtures in a state, and all the parameters used in above equation will be the parameters of initial HMM models created before. Then, the new parameters of HMM models could be re-estimated as follows

$$\begin{aligned}\pi_i &= \text{expected number of times in state } S_i \text{ at time } t = 1\\ &= \gamma_1(i)\end{aligned} \tag{3-39}$$

$$\begin{aligned}a_{ij} &= \frac{\text{expected number of transitions from state } S_i \text{ to state } S_j}{\text{expected number of transitions from state } S_i}\\ &= \frac{\sum_{t=1}^{T} \xi_t(i,j)}{\sum_{t=1}^{T} \gamma_t(i)}\end{aligned} \tag{3-40}$$

$$\begin{aligned}w_{jk} &= \frac{\text{expected number of times in state } S_j \text{ and mixtures } k}{\text{expected number of times in state } S_j}\\ &= \frac{\sum_{t=1}^{T} \gamma_t(j,k)}{\sum_{t=1}^{T}\sum_{k=1}^{M} \gamma_t(j,k)} = \frac{\sum_{t=1}^{T} \gamma_t(j,k)}{\sum_{t=1}^{T} \gamma_t(j)}\end{aligned} \tag{3-41}$$

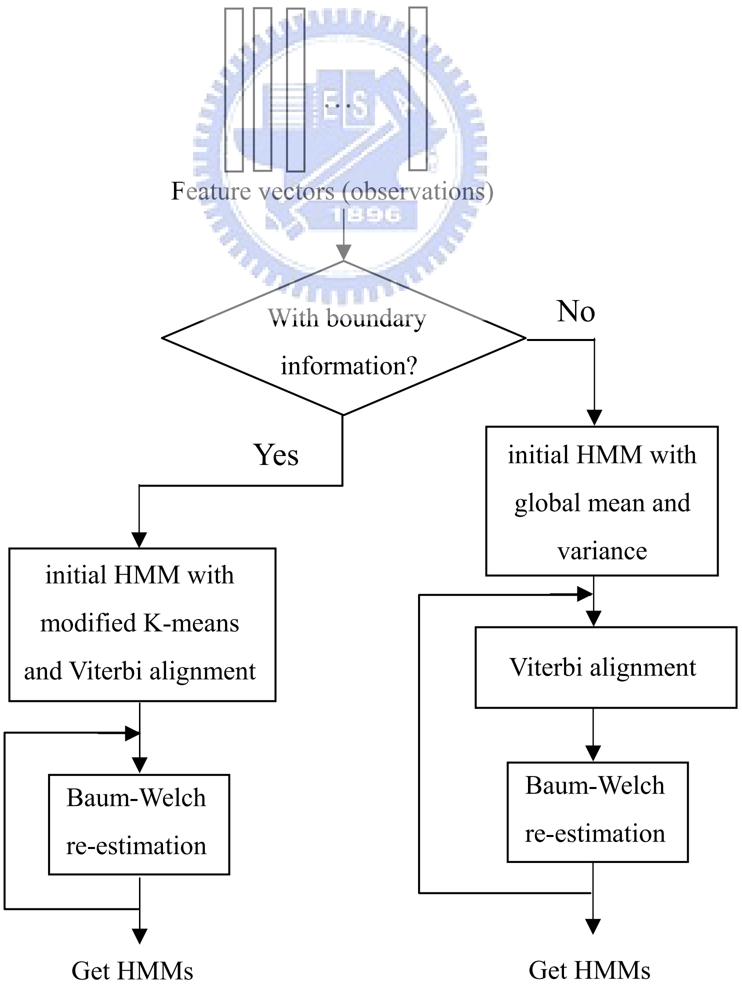
$\boldsymbol{\mu}_{jk}$ = mean of the observations at state S_j and mixture k

$$= \frac{\sum_{t=1}^{T} \gamma_t(j,k)\boldsymbol{o}_t}{\sum_{t=1}^{T} \gamma_t(j,k)} \tag{3-42}$$

$\boldsymbol{\Sigma}_{jk}$ = covariance matrix of the observations at state S_j and mixture k

$$= \frac{\sum_{t=1}^{T} \gamma_t(j,k)\left[(\boldsymbol{o}_t - \boldsymbol{\mu}_{jk})(\boldsymbol{o}_t - \boldsymbol{\mu}_{jk})^T\right]}{\sum_{t=1}^{T} \gamma_t(j,k)} \tag{3-43}$$

These parameters, then, would be updated, and go back to calculate the variables $\xi_t(i,j)$, $\gamma_t(i)$ and $\gamma_t(j,k)$. These operations will be repeated until the parameters $\boldsymbol{\pi}$, $\boldsymbol{A}$, $\boldsymbol{w}$, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are converged, and the HMM models are finally determinant. The complete flow chart of training procedures is shown in Fig.3.8.



...
Feature vectors (observations)
With boundary information?
No
Yes
initial HMM with global mean and variance
initial HMM with modified K-means and Viterbi alignment
Viterbi alignment
Baum-Welch re-estimation
Baum-Welch re-estimation
Get HMMs
Get HMMs

Fig.3.8 Training procedures of the HMM.

3.4 Recognition Procedures

Given the HMMs and the observation sequence $\boldsymbol{O} = \{\boldsymbol{o}_1, \boldsymbol{o}_2, \ldots \boldsymbol{o}_T\}$, the recognition problem can be regarded as that of computing

$$\arg\max_i \{P(w_i \mid \boldsymbol{O})\} \tag{3-44}$$

where w_i is the i-th vocabulary word. By Bayes' Rule, $P(w_i|\boldsymbol{O})$ could be transformed as

$$P(w_i \mid \boldsymbol{O}) = \frac{P(\boldsymbol{O} \mid w_i) P(w_i)}{P(\boldsymbol{O})} \tag{3-45}$$

Thus, for a given set of prior probabilities $P(w_i)$, the most probable spoken word depends only on the likelihood $P(\boldsymbol{O}|w_i)$. It can be solved by assuming that

$$P(\boldsymbol{O} \mid w_i) = P(\boldsymbol{O} \mid \Theta_i) \tag{3-46}$$

where Θ_i is the corresponding HMM of w_i. The calculation of $P(\boldsymbol{O}|\Theta_i)$ is shown in 3.3.1. If not only the probability but the best state sequence is desired, i.e., $P(\boldsymbol{O},\boldsymbol{S}|\Theta_i)$, the Viterbi algorithm, introduced in 3.3.2, could be applied.

Under the condition of connected words recognition (or called continuous speech recognition), (3-44) would be transformed as

$$\arg\max_i \{P(\boldsymbol{W}_i \mid \boldsymbol{O})\} \tag{3-47}$$

where $\boldsymbol{W}_i = \{ W_1 , W_2 ,\ldots, W_n \}$ is a word sequence. Similar to (3-46), the problem could be solved by assuming that

$$P(\boldsymbol{O} \mid \boldsymbol{W}_i) = P(\boldsymbol{O} \mid \boldsymbol{\Theta}_i) \tag{3-48}$$

where $\boldsymbol{\Theta}_i$ is the connected HMMs of corresponding word sequence $\boldsymbol{W}_i$. The connection of HMMs is simple when the HMM is the type of left-to-right HMM, mentioned in 3.2. The connection is just to connected the last state of former HMM with the first state of current HMM, and Fig 3.9. shows an example of connecting the HMM of "one" and the HMM of "two".

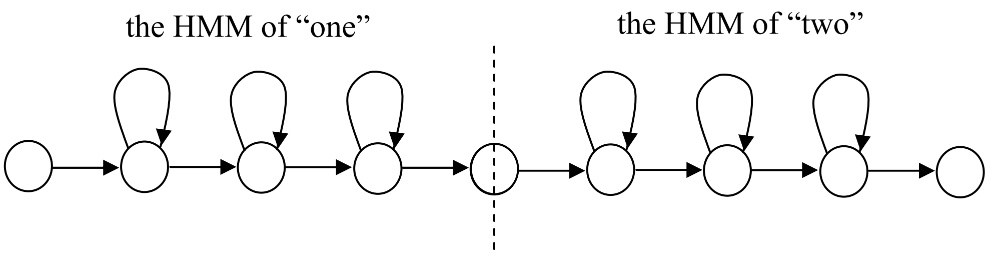


Fig.3.9 The connection of the HMM of "one" and "two".

In this thesis, all the training and recognition stages will utilize the HMM Tool Kits (HTK) [27], which is powerful tool kit dealing with the HMM. This tool kit is developed by the Speech Research Group of the University of Cambridge.

Chapter 4

Speech Recognition with Additive Noise

The state-of-the-art speech recognition system works very well when they are trained and tested under similar acoustic environments. The performance of the recognizer will decrease as long as the training and testing environments are mismatched. However, when a recognizer works in a real-world, it has to face the environment distortions which cause mismatch between pre-trained models and testing data. Various sources cause the distortions, such as the channel effects, the additive background noise, the different speaker characteristics, the different speech modes, etc. In this thesis, only the additive noise is considered and solved.

In this chapter, first, the most popular method in model-based class, PMC, will be introduced. Then, the disadvantages of using PMC will be discussed. Final, the method of noisy speech recognition using the pre-trained noisy models will be proposed, and the effect of mismatched noise and mismatched signal-to-noise ratio (SNR) will be shown by some experiments.

4.1 The Parallel Model Combination Method

In chapter 3, the statistical model HMM has been introduced to do the speech recognition. This model generally consists of the state transition probability, observation probability and initial state probability. However, these probabilities, especially the observation probability, are usually altered by environmental noise. Because the observation probability is expressed by mixture Gaussian distribution, their means and variances can be adapted so that they can represent the observation probability of noisy speech. In PMC method, the mean and variance combination is performed in linear spectral domain to obtain the mean and variance of noisy speech. The scheme of PMC method is illustrated in Fig.4.1.

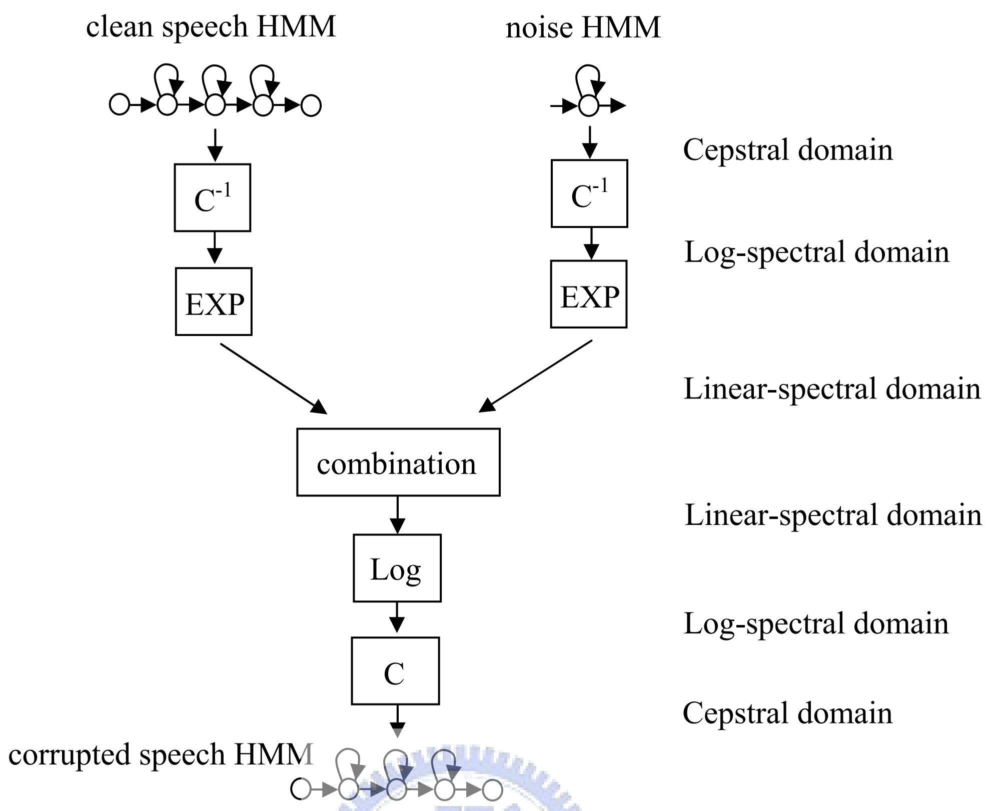


Fig 4.1 The scheme of PMC.

The inputs to the scheme are clean speech models and a noise model. The combination of the clean speech and noise is most naturally expressed in linear-spectral domain, i.e., it is simplest to model the effects of the additive noise on linear-spectral domain. The function to approximate this will be defined as the mismatch function and formed as follows

$$y[\tau] = g \cdot S[\tau] + N[\tau] \qquad (4\text{-}1)$$

where g is the gain matching term introduced to account for level difference between the speech and the noisy speech. In general, g will be set as 1. $S[\tau]$ is the clean speech 'observations' in linear-spectral domain, $N[\tau]$ is the noise 'observations' in linear-spectral domain, and $y[\tau]$ is the noisy speech 'observations' in linear-spectral domain. However, the HMM model are trained by using the observations in the cepstral domain, so some domain transformation must be applied before doing the combination.

Some assumptions have been made in PMC method. First, the speech and noise are independent. Second, the frame/state alignment used to generate the speech models from the clean speech data is not altered by the additive noise, i.e. the state transition matrix $\boldsymbol{A}$ would not be changed. Third, a single Gaussian or multiple Gaussian mixtures model contains sufficient information to represent the distribution of observation vectors in the cepstral domain and log-spectral domain. Therefore, just the mean μ and variance Σ in the observation probability distribution should be adapted.

The first stage in the scheme of PMC is to transform the μ and Σ from the cepstral domain to the log-spectral domain. This is simply achieved by using the inverse DCT, and the mapping is given by

$$\boldsymbol{\mu}^l = \boldsymbol{C}^{-1}\boldsymbol{\mu}^c \tag{4-2}$$

$$\boldsymbol{\Sigma}^l = \boldsymbol{C}^{-1}\boldsymbol{\Sigma}^c(\boldsymbol{C}^{-1})^T \tag{4-3}$$

where μ^c and Σ^c are the mean vectors and full variance matrix, respectively, for clean speech models in cepstral domain, μ^l and Σ^l those in log-spectral domain, and $\boldsymbol{C}$ is the matrix representing the DCT and the elements of it are given by

$$C_{ij} = \cos(i(j-0.5)\pi / B) \tag{4-4}$$

where B is the number of Mel filter banks.

The second stage in scheme of the PMC is to transform the μ and Σ from the log-spectral domain to the linear-spectral domain. Since the transformation between the log-spectral domain and linear-spectral domain is nonlinear, different approximations are made. Here, three popular methods, log-normal approximation, log-add approximation and data-driven PMC (DPMC), will be introduced.

The log-normal approximation assumes that the sum of two log-normally distributed variables is itself approximately log-normally distributed. Given the assumption that the speech and noise are independent and additive in the linear-spectral domain the corrupted speech static parameters in the linear-spectral domain are

$$\hat{\boldsymbol{\mu}} = \boldsymbol{\mu} + \tilde{\boldsymbol{\mu}} \tag{4-5}$$

$$\hat{\boldsymbol{\Sigma}} = \boldsymbol{\Sigma} + \tilde{\boldsymbol{\Sigma}} \tag{4-6}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are the mean vectors and full variance matrix, respectively, for clean speech models in linear-spectral domain. The notation '^' and '~' indicates that the parameters for corrupted speech and noise, respectively. The parameters of the clean speech in the linear-spectral and log-spectral domains are related by

$$\mu_i = \exp(\mu_i^{\ l} + {\Sigma_{ii}^{\ l}}/{2}) \tag{4-7}$$

$$\Sigma_{ij} = \mu_i \mu_j \left[\exp(\Sigma_{ij}^{\ l} - 1)\right] \tag{4-8}$$

the derivation could be found in [28]. Then, the mean vectors and full variance matrix for corrupted speech models in log-spectral domain could be obtained by

$$\hat{\mu}_i^{\ l} = \log(\hat{\mu}_i) - \frac{1}{2}\log\left(\frac{\hat{\Sigma}_{ii}}{\hat{\mu}_i^2} + 1\right) \tag{4-9}$$

$$\hat{\Sigma}_{ij} = \log\left(\frac{\hat{\Sigma}_{ij}}{\hat{\mu}_i \hat{\mu}_j} + 1\right) \tag{4-10}$$

Finally, the mean vectors and full variance matrix for corrupted speech models in cepstral domain could be easily obtained by

$$\hat{\boldsymbol{\mu}}^c = \boldsymbol{C} \cdot \hat{\boldsymbol{\mu}}^{\boldsymbol{l}} \tag{4-11}$$

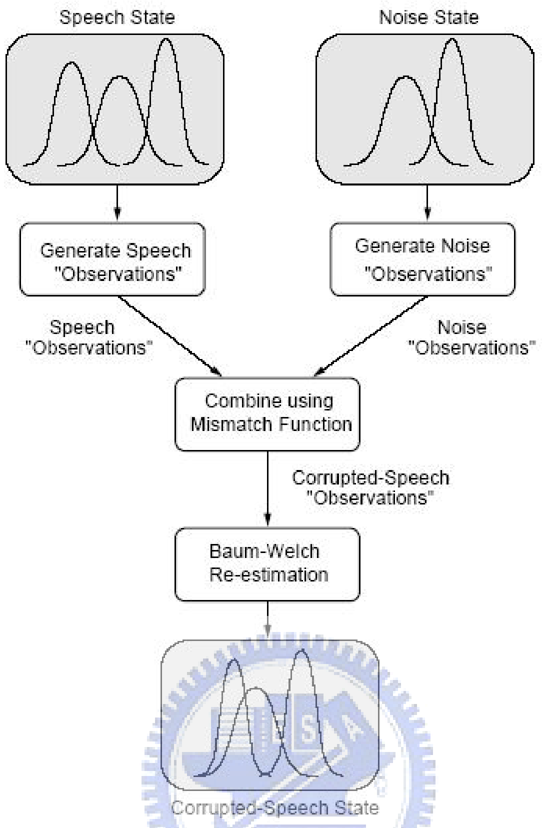
$$\hat{\boldsymbol{\Sigma}}^c = \boldsymbol{C} \cdot \hat{\boldsymbol{\Sigma}}^{\boldsymbol{l}} \cdot \boldsymbol{C}^{\boldsymbol{T}} \tag{4-12}$$

The second method is log-add approximation. In the log-add approximation, the variances are assumed to be small, so for the static parameters it is possible to write

$$\hat{\mu}_i^{\ l} = \log(\exp(\mu_i^{\ l}) + \exp(\tilde{\mu}_i^{\ l})) \tag{4-13}$$

This approximation is the simplest method, but the performance will be not good.

The third method is the data-driven PMC. The basic concept of DPMC is shown in Fig.4.2.



Speech State
Noise State
Generate Speech "Observations"
Generate Noise "Observations"
Speech "Observations"
Noise "Observations"
Combine using Mismatch Function
Corrupted-Speech "Observations"
Baum-Welch Re-estimation
Corrupted-Speech State

Fig.4.2 The illustration of data-driven PMC [28].

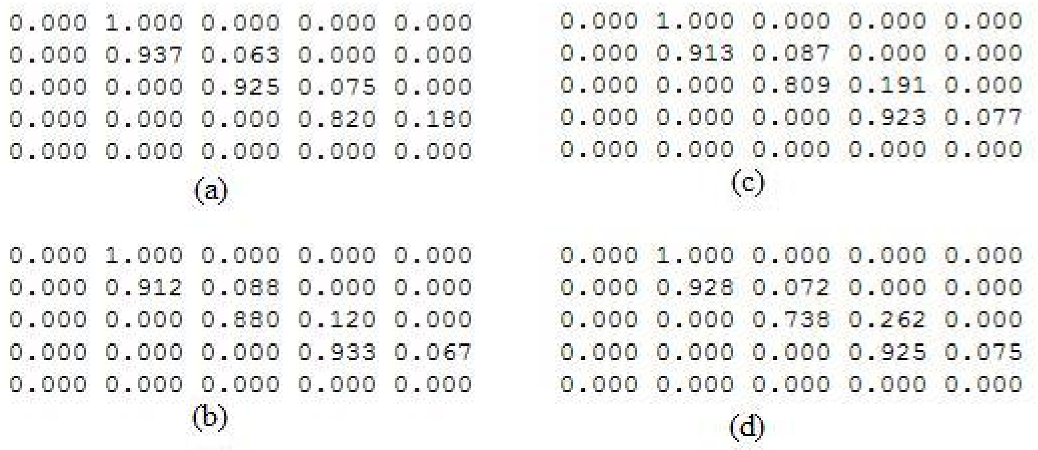
This is an iterative method; the integration is performed by generating corrupted speech observations. These are obtained by generating a speech observation and a noise observation for a particular pair of speech and noise states and combining them according to the appropriate mismatch function. Having generated a set of observations for a particular state pair, standard multiple mixture component single-emitting-state HMM training can be used to train the noisy speech model. This method could get more explicit noisy model, but would need large computation.

The PMC method has been proved that performs not bad in previously works. [29,30,31]

4.2 Robust Speech Recognition Using the Pre-Trained Noisy Models

Though the PMC method could obtain an acceptable performance, it still has some drawbacks in implementation. In the stage of transformation from the cepstral domain to the log-spectral domain, some information is lost due to the truncating of the original cepstrum coefficients described in chapter 2. After the truncating, the number of cepstrum coefficients is reduced from 20 (or 24) to 13. However, the domain transformation is based on the original number of cepstrum coefficients, i.e. the number of Mel filter banks *B*. Therefore, the mean vector μ which dimension is 13 must be padded with zeros so that the dimension could be equal to the number of Mel filter banks *B* and this may cause a bad representation in log-spectral domain.

Second, the PMC method is assumed that the additive noises will not alter the state transition matrix ***A***. Nevertheless, the state transition matrix ***A*** is indeed affected by the additive noise and the difference of ***A*** between clean speech models and noisy models are increasing as the SNR degrades. Fig. 4.3 shows the difference between the clean speech model and noisy model of "four" in low SNR. This would also cause the compensated noisy models work not well especially in low SNR. Unfortunately, there is still no efficiently method to precisely estimate the state transition matrix ***A*** of noisy models.



```
0.000 1.000 0.000 0.000 0.000
0.000 0.937 0.063 0.000 0.000
0.000 0.000 0.925 0.075 0.000
0.000 0.000 0.000 0.820 0.180
0.000 0.000 0.000 0.000 0.000
```

(a)

```
0.000 1.000 0.000 0.000 0.000
0.000 0.913 0.087 0.000 0.000
0.000 0.000 0.809 0.191 0.000
0.000 0.000 0.000 0.923 0.077
0.000 0.000 0.000 0.000 0.000
```

(c)

```
0.000 1.000 0.000 0.000 0.000
0.000 0.912 0.088 0.000 0.000
0.000 0.000 0.880 0.120 0.000
0.000 0.000 0.000 0.933 0.067
0.000 0.000 0.000 0.000 0.000
```

(b)

```
0.000 1.000 0.000 0.000 0.000
0.000 0.928 0.072 0.000 0.000
0.000 0.000 0.738 0.262 0.000
0.000 0.000 0.000 0.925 0.075
0.000 0.000 0.000 0.000 0.000
```

(d)

Fig. 4.3 The transition matrix of model "four" in (a) clean speech model (b) noisy model with 0dB (c) noisy model with 5dB (d) noisy model with 10dB.

The most difficult part in PMC method is the stage of transformation from the log-spectral domain to the linear-spectral domain because it is a nonlinear transformation. The three approaches introduced above all have some disadvantages. The most important one is the assumption that the sum of two log-normally distributed variables is itself approximately log-normally distributed. Obviously, it is not surely. Even the sum of two single Gaussian distribution is not surely still a single Gaussian distribution. Approaches used in PMC method can just "approximately" represent the transformation from the log-spectral domain to the linear-spectral domain. Besides, one state is usually not enough to represent the HMM of noise. It is complicated for PMC method under the condition that the noise HMM contains more than one state. Furthermore, single Gaussian distribution is not sufficient to represent the observation probability in HMM. If the mixture Gaussian distribution is utilized to represent the observation probability, the computation load would increase and may not estimate well. The DPMC method might overcome the drawbacks described above, because it could represent the noisy model more explicitly. However, it need too much computation and is not useful for an on-line application.

Therefore, I will propose a concept that using the pre-trained noisy models to do the robust speech recognition. Now that it is hard to estimate the distribution of the noisy model just combining the clean speech model and noise model, and it is shown that using the matched noisy model will get the best recognition result (see the experiments result in 4.3), we can pre-train various noisy models and save as a database. In recognition phase, the first 10,000 sampling data will be used to analyze that current background noise belongs to which noise in the database, and then, the corresponding pre-trained noisy model will be utilized to do the recognition. The scheme is shown in Fig.4.4.

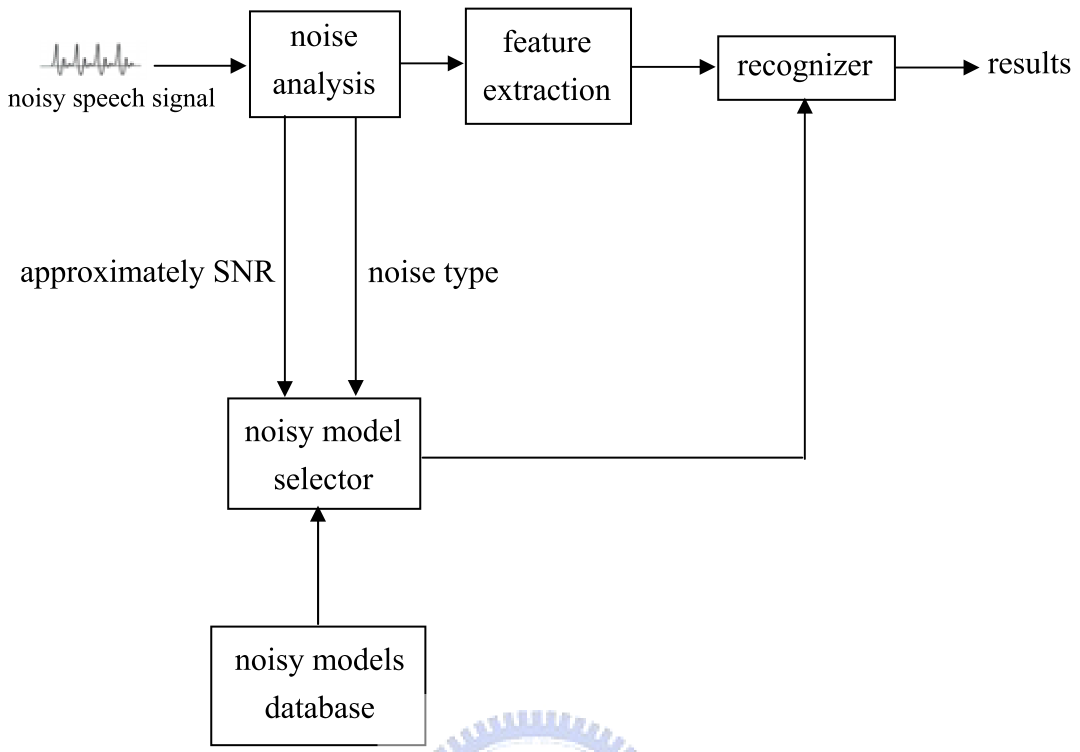


Fig. 4.4 The scheme of the method using the pre-trained noisy models to do the noisy speech recognition.

In the stage of noise analysis, the noise type is chosen based on the mean spectrum, and the SNR is approximately decided by

$$\hat{SNR} = 10 \cdot \log \frac{\left(\sum_{i-1}^{T} y_i^{\,2} - \sum_{i-1}^{N} y_i^{\,2} \right)}{\sum_{i-1}^{N} y_i^{\,2}} \tag{4-14}$$

where y_i is the corrupted speech and N is the number of samples to estimate the noise. This method will need more memory to store the pre-trained noisy models, but with the progress of the storage technology, it would be no longer the problem. The advantage of this method is that a more explicitly noisy model could be obtained without too much computation in recognition phase. All the complicated computation will be done in training phase and the pre-trained noisy models will be stored in recognizer.

4.3 Experiments Result

The experiments will be divided into two parts. In the first part, the performance of the recognizer will be compared with the condition of using matched/mismatched noisy models, the MFCC with/without dynamic features and different mixture number of GMM. In the second part, a test noise which is not in the database will be added. Then, the experiments will show the performance of the recognizer using a most likely pre-trained noisy model.

4.3.1 Experiments A

The clean speech data were collected from 16 persons with 8 males and 8 females. 40 utterances for each containing 10 connected digits are recorded by each person. The noises used in this experiment are babble, f16, factory and white noise taken from NOISE-92 database [32] and resample them to 16KHz. The recorded clean speech data were manually added with these four noises individually at different SNR (0dB, 5dB, 10dB, 15dB, 20dB, 25dB and 30dB) to produce several sets of noise-corrupted speech data. The SNR is defined as

$$SNR = 10 \cdot \log \frac{P_S}{P_N} \tag{4-15}$$

where P_s is the average power of clean speech and P_N is the average power of noise. The sampling rate is 16KHz. A 25 ms Hamming window shifted with 10 ms steps, a pre-emphasis factor of 0.97 and 24 Mel filter banks were used to evaluate 13 MFCCs (including the energy term) with its delta and delta-delta term to obtain 39-dimensional feature vectors. For the case without dynamic terms, the feature vectors just contain 15 MFCCs. The testing data were selected from the original recorded data, two males and two females, so this in a speaker dependent experiment. The statistics of training data and testing data are shown in Table 4.1, and the number of states of each digit (the first and last state are null states) is shown in Table 4.2.

	Males	Females	Total
Amounts of speakers	8	8	16
Amounts of sentences	320	320	640
Amounts of digits	3200	3200	6400

(a)

	Males	Females	Total
Amounts of speakers	2	2	4
Amounts of sentences	80	80	160
Amounts of digits	800	800	1600

(b)

Table 4.1 The statistics of (a) training speech data (b) testing speech data.

Digits	0	1	2	3	4	5	6	7	8	9	sil
Number of states	10	7	7	10	5	7	10	10	7	10	5

Table 4.2 The number of state of each digit. (sil means the silence)

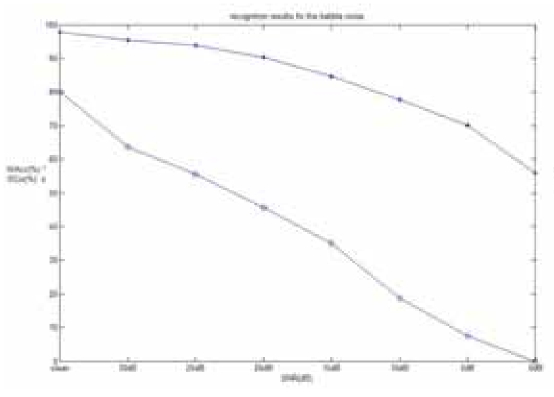
The experiments are divided into two sets. In the first experiment set, the MFCC features with its dynamic features are used, i.e. each feature vector contains 13 MFCCs, 13 delta MFCCs and 13 delta-delta MFCCs. The other experiment set uses MFCC features only, and each feature vector contains 15 MFCCs. The performances are examined by the word accuracy rate (WAcc %) and the sentence correct rate (SCor %). The calculation of WAcc (%) and SCor (%) are

$$WAcc(\%) = \frac{T_w - D_w - S_w - I_w}{T_w} \times 100 \tag{4-16}$$

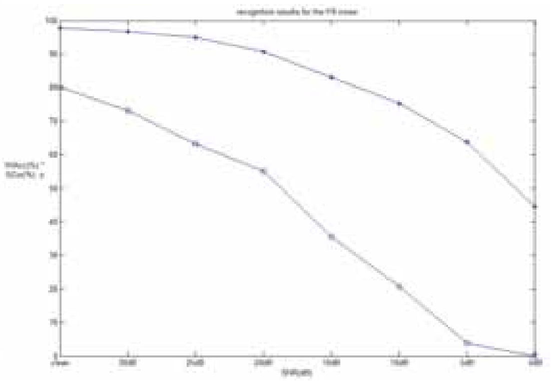
$$SCor(\%) = \frac{T_s - F_s}{T_s} \times 100 \tag{4-17}$$

where T_w and T_s are the total numbers of words and sentences. D_w, S_w and I_w represent the numbers of deletion errors, substitution errors and insertion errors, respectively. F_s means that the number of wrong sentences.

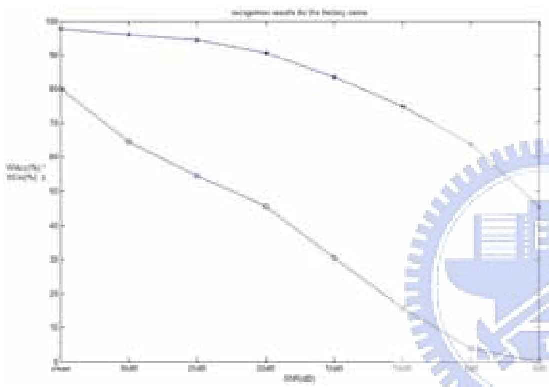
The recognition results of the noise-corrupted speech under different SNR using the clean speech models are shown in Fig. 4.5.



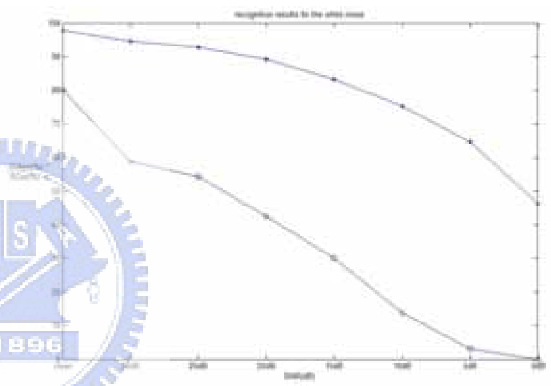
(a)



(b)



(c)



(d)

Fig. 4.5 The recognition results of the noisy speech using the clean speech models under different SNR corrupted by (a) babble noise (b) f16 noise (c) factory noise (d) white noise.

It is shown that the performance of the recognizer degrades while the SNR decreases. Table 4.3 shows the recognition results of the noise-corrupted speech in different SNR using the clean speech models with different mixtures of GMM. The results are represented by the WAcc (%) and SCor (%, the values in the brackets).

	clean	30dB	25dB	20dB	15dB	10dB	5dB	0dB
1-mixture	97.71 (80.00)	96.77 (73.13)	94.90 (63.13)	90.68 (55.00)	83.07 (35.63)	75.21 (20.63)	63.70 (3.75)	44.58 (0.00)
2-mixtures	98.02 (85.63)	96.46 (77.50)	95.73 (73.75)	94.27 (65.63)	90.10 (45.63)	82.08 (25.00)	69.22 (6.25)	49.01 (0.00)

4-mixtures	98.75 (90.00)	96.98 (80.63)	95.94 (75.63)	94.64 (68.75)	89.38 (48.13)	81.20 (28.75)	69.32 (6.88)	50.42 (1.25)
8-mixtures	99.43 (95.00)	98.49 (88.13)	96.88 (78.13)	93.28 (65.00)	84.64 (33.13)	71.41 (7.50)	58.49 (0.63)	43.02 (0.00)

(a)

	clean	30dB	25dB	20dB	15dB	10dB	5dB	0dB
1-mixture	97.71 (80.00)	95.47 (63.75)	93.96 (55.63)	90.42 (45.63)	84.74 (35.00)	77.66 (18.75)	70.10 (7.50)	55.94 (0.00)
2-mixtures	98.02 (85.63)	97.03 (80.00)	96.61 (75.63)	95.52 (68.38)	91.82 (45.63)	86.72 (28.13)	75.73 (7.50)	60.52 (0.00)
4-mixtures	98.75 (90.00)	97.24 (83.13)	96.35 (76.88)	94.84 (66.25)	91.51 (51.25)	86.25 (33.13)	74.69 (11.25)	59.27 (0.63)
8-mixtures	99.43 (95.00)	98.39 (86.88)	97.71 (81.25)	96.25 (72.50)	91.82 (48.75)	84.48 (20.63)	71.56 (1.88)	49.79 (0.00)

(b)

	clean	30dB	25dB	20dB	15dB	10dB	5dB	0dB
1-mixture	97.71 (80.00)	95.99 (64.38)	94.48 (54.38)	90.68 (45.63)	83.54 (30.63)	74.84 (15.63)	63.70 (3.75)	45.52 (0.00)
2-mixtures	98.02 (85.63)	96.25 (75.00)	95.47 (71.88)	93.91 (63.13)	90.16 (45.00)	82.97 (27.50)	70.57 (5.63)	51.41 (0.63)
4-mixtures	98.75 (90.00)	97.03 (81.25)	95.99 (75.26)	93.85 (66.88)	89.11 (47.50)	80.94 (28.13)	70.10 (8.13)	48.85 (0.63)
8-mixtures	99.43 (95.00)	98.67 (89.04)	97.19 (80.63)	94.74 (66.25)	88.44 (43.13)	77.60 (17.50)	61.82 (0.63)	41.56 (0.00)

(c)

	clean	30dB	25dB	20dB	15dB	10dB	5dB	0dB
1-mixture	97.71 (80.00)	94.64 (58.75)	92.86 (54.38)	89.22 (42.50)	83.23 (30.00)	75.26 (13.75)	64.58 (3.13)	46.41 (0.00)
2-mixtures	98.02 (85.63)	96.09 (73.13)	95.00 (68.13)	92.29 (56.88)	88.18 (41.88)	81.04 (19.38)	68.23 (1.25)	48.07 (0.00)
4-mixtures	98.75 (90.00)	96.15 (76.25)	94.79 (71.25)	91.35 (60.00)	85.89 (45.63)	77.40 (23.13)	66.20 (2.50)	49.69 (0.00)
8-mixtures	99.43 (95.00)	97.14 (80.00)	95.05 (70.00)	89.48 (53.75)	78.85 (29.38)	63.54 (0.00)	53.65 (0.00)	41.72 (0.00)

(d)

Table 4.3 The recognition results of the noisy speech under different SNR using the clean speech models with different mixtures of GMM. The noisy speech is corrupted by (a) f16 noise (b) babble noise (c) factory noise (d) white noise.

The results show that using the dynamic features can get not bad performance while the SNR is larger than 20dB. Besides, increasing the number of mixtures of GMM is not useful for low SNR, because the observation probability distribution has been altered so severely by the noise that the noisy speech can be barely recognized by the original model.

The following experiments were examined to investigate the performances of noisy speech recognition using matched noisy models. The "matched" means if the tested noisy speech is corrupted by f16 noise in 0dB, the f16 noisy model trained in 0dB condition will be utilized to do the recognition. Table 4.4 shows the recognition results of using the matched noisy models with different mixtures of GMM. The performances are presented by SCor (%), and values in the brackets are the recognition rates using clean speech models.

	30dB	25dB	20dB	15dB	10dB	5dB	0dB
1-mixture	71.25 (73.13)	68.13 (63.13)	61.88 (55.00)	61.88 (35.63)	50.63 (20.63)	39.38 (3.75)	15.63 (0.00)
2-mixtures	82.50 (77.50)	83.75 (73.75)	76.25 (65.63)	76.88 (45.63)	68.75 (25.00)	58.13 (6.25)	15.00 (0.00)
4-mixtures	89.38 (80.63)	88.75 (75.63)	88.13 (68.75)	83.75 (48.13)	77.50 (28.75)	66.25 (6.88)	20.00 (1.25)
8-mixtures	92.50 (88.13)	93.75 (78.13)	92.50 (65.00)	87.50 (33.13)	85.63 (7.50)	54.38 (0.63)	16.25 (0.00)

(a)

	30dB	25dB	20dB	15dB	10dB	5dB	0dB
1-mixture	68.75 (64.38)	67.50 (54.38)	58.75 (45.63)	49.38 (30.63)	43.13 (15.63)	11.25 (3.75)	1.88 (0.00)
2-mixtures	82.50 (75.00)	82.50 (71.88)	76.88 (63.13)	69.38 (45.00)	60.00 (27.50)	18.75 (5.63)	3.13 (0.63)
4-mixtures	88.75 (81.25)	86.88 (75.26)	85.00 (66.88)	79.38 (47.50)	65.63 (28.13)	16.25 (8.13)	3.75 (0.63)
8-mixtures	91.88 (89.04)	92.50 (80.63)	88.13 (66.25)	82.50 (43.13)	67.50 (17.50)	17.50 (0.63)	2.50 (0.00)

(b)

	30dB	25dB	20dB	15dB	10dB	5dB	0dB
1-mixture	58.13 (58.75)	55.63 (54.38)	48.13 (42.50)	35.63 (30.00)	25.00 (13.75)	18.13 (3.13)	11.25 (0.00)

2-mixtures	72.50 (73.13)	72.50 (68.13)	68.13 (56.88)	61.25 (41.88)	56.88 (19.38)	33.13 (1.25)	20.00 (0.00)
4-mixtures	84.38 (76.25)	83.75 (71.25)	80.00 (60.00)	72.50 (45.63)	63.13 (23.13)	45.63 (2.50)	26.88 (0.00)
8-mixtures	91.25 (80.00)	86.25 (70.00)	85.00 (53.75)	80.00 (29.38)	70.00 (0.00)	50.63 (0.00)	30.00 (0.00)

(c)

	30dB	25dB	20dB	15dB	10dB	5dB	0dB
1-mixture	78.13 (63.75)	71.25 (55.63)	65.00 (45.63)	52.50 (35.00)	43.75 (18.75)	19.38 (7.50)	5.00 (0.00)
2-mixtures	86.88 (80.00)	85.63 (75.63)	79.38 (68.38)	75.00 (45.63)	55.00 (28.13)	28.13 (7.50)	7.50 (0.00)
4-mixtures	89.38 (83.13)	85.63 (76.88)	86.25 (66.25)	75.00 (51.25)	58.13 (33.13)	29.38 (11.25)	6.25 (0.63)
8-mixtures	92.50 (86.88)	94.38 (81.25)	93.75 (72.50)	86.25 (48.75)	55.00 (20.63)	26.88 (1.88)	8.13 (0.00)

(d)

Table 4.4 The recognition results of the noisy speech under different SNR using the clean speech models (the values in the brackets) and matched noisy models with different mixtures of GMM. The noisy speech is corrupted by (a) f16 noise (b) factory noise (c) white noise (d) babble noise.

The results show that the matched noisy models indeed promote the performance of noisy speech recognition. The performances are good when SNR is larger than 5 dB, and the mixture number of the GMM is 8. Besides, the matched noisy models are fairly good for the noisy speech corrupted by white noise in low SNR; the SCor (%) for white noisy speech is 30 in SNR 0dB.

The next experiment will show the performance of using the same noisy models but with different SNR, i.e. the training and testing environment will be incongruity in SNR. The SNR of tested noisy speech is set 15dB, and the same kinds of noisy models trained in different SNR with the 8 mixtures GMM will be utilized to do the recognition. The experiment results are shown in Table 4.5, and the performances will be presented by SCor (%).

	0dB	5dB	10dB	15dB	20dB	25dB	30dB
F16	25.63	61.88	86.25	**87.50**	88.75	77.50	70.63
factory	15.63	31.25	71.88	**82.50**	79.38	75.00	62.50
white	11.88	51.25	71.25	**80.00**	81.25	70.63	41.88
babble	25.00	58.13	69.38	**86.25**	86.25	81.88	76.88

Table 4.5 The recognition results of the noisy speech under 15dB SNR using the same kind of noisy models (f16) in different SNR with 8 mixtures of GMM. The highlighted values are the results using the matched noisy models.

The results show that the SNR does not affect the performance very much, because the recognition rate would not change too large even use the noisy models with SNR varying 5dB from the matched SNR. Therefore, the SNR of the tested noisy speech can be roughly determined and still obtain good enough recognition results.

The above experiments will be examined again, but the features used are all replaced by the MFCC features without its dynamic terms, i.e. each feature contains only 15 MFCCs. These experiments are examined to see which features, with or without dynamic terms, are efficient to the noisy speech recognition using matched noisy models. Table 4.6 shows the recognition results of the noise-corrupted speech in different SNR using the clean speech models with different mixtures of GMM. The results are represented by the WAcc (%) and SCor (%, the values in the brackets).

	clean	30dB	25dB	20dB	15dB	10dB	5dB	0dB
1-mixture	94.53 (60.00)	91.77 (48.75)	88.80 (37.50)	83.91 (25.63)	73.65 (9.38)	57.14 (0.00)	41.46 (0.00)	23.39 (0.00)
2-mixtures	91.98 (50.00)	89.48 (36.88)	89.43 (39.38)	86.93 (27.50)	79.32 (10.63)	62.08 (0.63)	40.42 (0.00)	24.53 (0.00)
4-mixtures	95.05 (67.50)	93.54 (52.50)	92.55 (49.38)	89.53 (38.75)	82.14 (21.25)	67.34 (5.63)	46.04 (0.63)	24.48 (0.00)
8-mixtures	96.30 (71.25)	93.23 (52.50)	88.80 (37.50)	86.09 (32.50)	77.45 (10.63)	60.05 (0.63)	39.38 (0.00)	21.93 (0.00)

(a)

	clean	30dB	25dB	20dB	15dB	10dB	5dB	0dB
1-mixture	94.53 (60.00)	93.59 (57.50)	92.08 (50.00)	86.30 (29.38)	74.79 (9.38)	61.72 (0.63)	46.61 (0.00)	29.17 (0.00)
2-mixtures	91.98 (50.00)	92.08 (47.50)	92.19 (48.13)	88.75 (33.13)	79.58 (12.50)	62.14 (1.25)	48.44 (0.00)	31.04 (0.00)
4-mixtures	95.05 (67.50)	95.47 (65.00)	94.53 (58.75)	90.99 (44.38)	83.33 (24.38)	69.01 (6.88)	51.72 (0.00)	34.27 (0.00)
8-mixtures	96.30 (71.25)	95.47 (63.75)	93.85 (54.38)	89.38 (40.63)	78.44 (11.88)	61.61 (0.00)	45.47 (0.00)	29.01 (0.00)

(b)

	clean	30dB	25dB	20dB	15dB	10dB	5dB	0dB
1-mixture	94.53 (60.00)	92.03 (50.00)	89.43 (41.25)	83.07 (21.25)	72.71 (5.00)	58.54 (0.63)	40.42 (0.00)	20.89 (0.00)
2-mixtures	91.98 (50.00)	91.35 (45.63)	90.63 (41.88)	85.68 (25.00)	77.19 (8.13)	58.54 (0.00)	38.65 (0.00)	22.60 (0.00)
4-mixtures	95.05 (67.50)	93.91 (53.75)	92.86 (45.63)	88.54 (30.63)	79.79 (13.13)	63.91 (2.50)	42.60 (0.00)	23.13 (0.00)
8-mixtures	96.30 (71.25)	94.64 (58.13)	91.77 (46.25)	86.72 (30.63)	75.94 (6.25)	57.24 (0.63)	38.18 (0.00)	21.51 (0.00)

(c)

	clean	30dB	25dB	20dB	15dB	10dB	5dB	0dB
1-mixture	94.53 (60.00)	88.33 (35.63)	82.50 (21.25)	74.27 (11.88)	63.96 (0.00)	54.01 (0.00)	40.10 (0.00)	20.05 (0.00)
2-mixtures	91.98 (50.00)	85.94 (27.50)	81.41 (18.75)	76.20 (8.75)	68.33 (1.25)	55.21 (0.00)	36.72 (0.00)	19.22 (0.00)
4-mixtures	95.05 (67.50)	90.21 (43.13)	84.22 (28.13)	78.33 (17.50)	68.02 (1.88)	52.81 (0.00)	38.65 (0.00)	19.17 (0.00)
8-mixtures	96.30 (71.25)	89.01 (37.50)	83.70 (26.25)	76.93 (9.38)	66.04 (0.63)	53.02 (0.00)	31.61 (0.00)	15.78 (0.00)

(d)

Table 4.6 The recognition results of the noisy speech under different SNR using the clean speech models with different mixtures of GMM. The noisy speech is corrupted by (a) f16 noise (b) babble noise (c) factory noise (d) white noise.

Comparing Table 4.6 with Table 4.3, we can find that the recognition results using the features without dynamic terms degrade seriously as the SNR decreasing, especially the SCor (%). The performances are bad even in 30dB SNR. Table 4.7 shows the recognition results of using the matched noisy models with different mixtures of GMM. The performances are presented by SCor (%), and values in the brackets are the recognition rates using clean speech models.

	30dB	25dB	20dB	15dB	10dB	5dB	0dB
1-mixture	53.13 (48.75)	44.38 (37.50)	43.75 (25.63)	45.63 (9.38)	30.63 (0.00)	16.25 (0.00)	5.63 (0.00)
2-mixtures	50.63 (36.88)	53.75 (39.38)	50.00 (27.50)	47.50 (10.63)	43.75 (0.63)	17.50 (0.00)	9.38 (0.00)
4-mixtures	55.63 (52.50)	56.25 (49.38)	54.38 (38.75)	52.50 (21.25)	43.75 (5.63)	19.38 (0.63)	12.50 (0.00)
8-mixtures	56.88 (52.50)	53.75 (37.50)	53.75 (32.50)	51.88 (10.63)	41.25 (0.63)	16.25 (0.00)	13.75 (0.00)

(a)

	30dB	25dB	20dB	15dB	10dB	5dB	0dB
1-mixture	51.25 (50.00)	50.00 (41.25)	40.63 (21.25)	33.75 (5.00)	19.38 (0.63)	5.63 (0.00)	1.25 (0.00)
2-mixtures	51.25 (45.63)	35.63 (41.88)	50.00 (25.00)	43.75 (8.13)	20.63 (0.00)	11.25 (0.00)	4.38 (0.00)
4-mixtures	56.25 (53.75)	36.88 (45.63)	48.13 (30.63)	39.38 (13.13)	23.13 (2.50)	12.50 (0.00)	0.63 (0.00)
8-mixtures	56.25 (58.13)	51.88 (46.25)	46.25 (30.63)	40.63 (6.25)	21.25 (0.63)	10.00 (0.00)	0.63 (0.00)

(b)

	30dB	25dB	20dB	15dB	10dB	5dB	0dB
1-mixture	36.88 (35.63)	33.75 (21.25)	26.88 (11.88)	28.75 (0.00)	22.50 (0.00)	15.00 (0.00)	4.38 (0.00)
2-mixtures	32.50 (27.50)	48.75 (18.75)	41.25 (8.75)	36.88 (1.25)	35.00 (0.00)	21.88 (0.00)	6.25 (0.00)
4-mixtures	32.50 (43.13)	41.25 (28.13)	36.25 (17.50)	31.25 (1.88)	30.00 (0.00)	25.63 (0.00)	10.00 (0.00)
8-mixtures	39.38 (37.50)	49.38 (26.25)	41.88 (9.38)	41.88 (0.63)	39.38 (0.00)	31.88 (0.00)	9.38 (0.00)

(c)

	30dB	25dB	20dB	15dB	10dB	5dB	0dB
1-mixture	49.38 (57.50)	49.38 (50.00)	40.00 (29.38)	24.38 (9.38)	18.13 (0.63)	9.38 (0.00)	1.88 (0.00)

2-mixtures	59.38 (47.50)	57.50 (48.13)	47.50 (33.13)	27.50 (12.50)	22.50 (1.25)	13.13 (0.00)	5.00 (0.00)
4-mixtures	55.63 (65.00)	57.50 (58.75)	45.00 (44.38)	25.00 (24.38)	26.88 (6.88)	16.88 (0.00)	3.75 (0.00)
8-mixtures	56.88 (63.75)	58.75 (54.38)	49.38 (40.63)	26.88 (11.88)	26.25 (0.00)	12.50 (0.00)	2.50 (0.00)

(d)

Table 4.7 The recognition results of the noisy speech under different SNR using the clean speech models (the values in the brackets) and matched noisy models with different mixtures of GMM. The noisy speech is corrupted by (a) f16 noise (b) factory noise (c) white noise (d) babble noise.

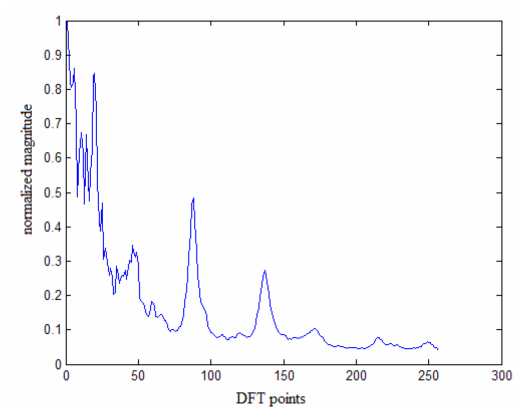
Comparing Table 4.7 with Table 4.4, we can find that the recognition results using the features without dynamic terms were worse than those using the features with dynamic terms. The reason is that the dynamic terms are relatively not altered by the add noise. From (2-15), the term of $c_{t+p}[i]-c_{t-p}[i]$ can diminish the influence of the additive noise. Assuming the noise is added equally to each frame, then

$$\tilde{c}_{t+p}[i]-\tilde{c}_{t-p}[i] \approx (c_{t+p}[i]+n[i])-(c_{t-p}[i]+n[i]) = c_{t+p}[i]-c_{t-p}[i] \qquad (4\text{-}18)$$

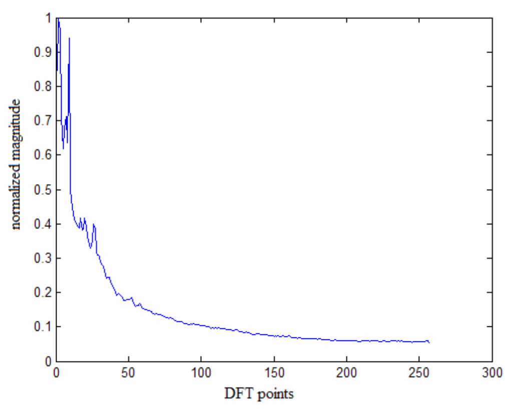
where $\tilde{c}[i]$ is the noise-corrupted cepstrum coefficient and $n[i]$ is the noise cepstrum coefficient. Therefore, the recognizer using MFCC with its dynamic features will obtain better results.

4.3.2 Experiments B

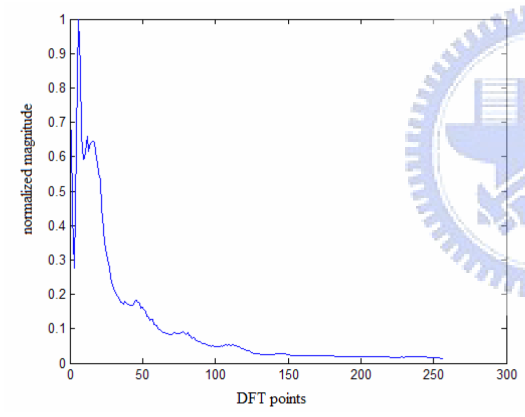
In this experiment, a test noise which is not in the database will be added. The noise is the sound of drilling machine recorded in a construction site. A most likely noisy model should be selected first to do the recognition. The selection is based on the mean square error of the normalized mean spectrums between the tested noise and noises in the database. The noise which has minimum one will be selected and its model will be used as the noisy model. The mean spectrums of the f16, factory, babble, white and drilling machine are shown in Fig 4.6. The babble noisy model is selected after comparing all four kinds of noise with the drilling machine noise. Therefore, the recognition will use the babble noisy models, and the results are shown in Table 4.8. The performances are presented by SCor (%).



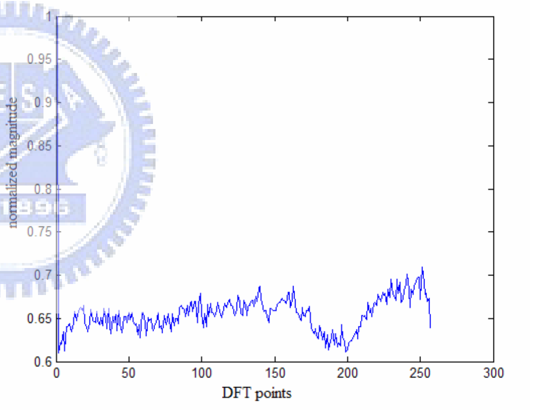
(a)



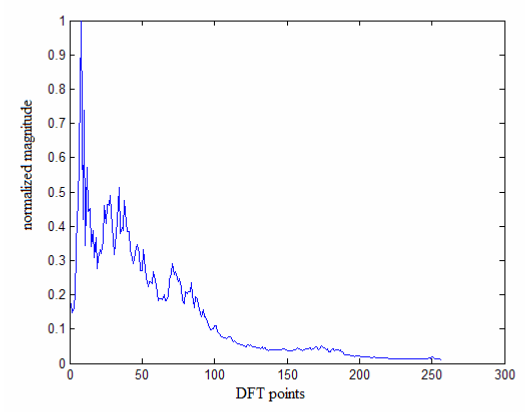
(b)



(c)



(d)



(e)

Fig. 4.6 Normalized mean spectrums of (a) f16 (b) factory (c) babble (d) white (e) drilling machine.

	30dB	25dB	20dB	15dB	10dB	5dB	0dB
1-mixture	75.00 (58.13)	67.50 (51.88)	53.13 (42.50)	46.88 (31.88)	36.88 (10.00)	12.50 (0.63)	0.00 (0.00)
2-mixtures	80.00 (78.13)	82.50 (70.00)	73.75 (58.75)	68.13 (34.48)	47.50 (5.00)	10.63 (0.00)	1.88 (0.00)
4-mixtures	83.75 (76.88)	81.88 (70.63)	81.25 (61.25)	72.50 (42.50)	43.13 (16.88)	11.88 (1.88)	1.25 (0.00)
8-mixtures	89.38 (80.63)	88.75 (73.75)	86.88 (57.50)	81.88 (28.75)	48.75 (1.25)	14.38 (0.00)	1.88 (0.00)

Table 4.8 The recognition results of the drilling noisy speech under different SNR using the clean speech models (the values in the brackets) and babble noisy models with different mixtures of GMM.

The results show that the performances are promoted obviously for the cases of SNR larger than 10 dB, and it shows again that using the noisy model with 8 mixtures GMM is the best choice.

Chapter 5

Conclusions and Future Works

In this thesis a method using the pre-trained noisy models to do the noisy speech recognition has been proposed. Previously experiments have proved that it can improve the performance of the recognizer in unknown noisy environments. The best results are obtained by using the matched noisy models trained with the dynamic MFCC features and 8 mixtures GMM. The advantages of this method are summarized as follows:

1) More explicit noisy models can be obtained easily. In this method, the mixture numbers of GMM can be increased to mimic the true noisy speech observation probability distribution. The state number of noise can be not limited as 1 to catch the behavior of the noise more precisely, and can obtain more exactly state transition matrix, because the noisy model are really trained using the truly noise-corrupted speech data.
2) In on-line application, the system needs little computation to get an acceptable noisy model. The only operation needed to do is comparing with all the noise type in the database, and then, find a most likely noisy model. All the complicated computations, training the noisy models, are performed off-line.

There are three further things should be done in the future. First is to modify the approach of selecting the most likely noisy model. In experiment 4.3.2, the babble noise is the closest to the tested noise, drilling machine noise, and it indeed can improve the performance. However, the babble noisy model is not the best selection but the f16 noisy model to do the recognition. Table 5.1 shows the recognition results of using babble noisy model and f16 noisy model. The performances are presented by SCor (%).

	30dB	25dB	20dB	15dB	10dB	5dB	0dB
babble	89.38	88.75	86.88	81.88	48.75	14.38	1.88
f16	90.00	87.50	84.38	80.00	64.38	33.75	12.50

Table 5.1 The recognition results of using babble noisy model and f16 noisy model.

It can be shown that using the f16 noisy model can obtain better results than babble noisy model. It means that the approach of selecting suitable noisy model should be modified. The noise types can not be identified only using the normalized mean spectrum.

Secondly, the environment existing varying noises should be considered. In this thesis, it is assumed that the environment exist only one kind of noise. The approach to estimate the noise is to analyze the first *N* samples of the noisy speech, which is regarded as the noise. Therefore, while the original noise is replaced with another kind of noise, this approach could not be used to estimate the newly noise. Another sufficient noise estimating approach should be used to overcome this problem.

The last work is to continue establishing the noisy models database. In this thesis, only four kinds of noisy models are established, and it is obviously not enough. In the future, more noises should be collected and these collected noises should be analyzed to classify them into groups. The noises in the same group are trained to produce one representative noisy model. In this way, the size of the noisy models database would be reduced, and the selecting of suitable noisy models might be easy and accurate.

Reference

[1] D. Mansour and B. H. Juang, "The Short-Time Modified Coherence Representation and Noisy Speech Recognition," *IEEE Transitions on Acoustics, Speech, and Signal Processing, Volume 37, Issue 6, Jun. 1989 Page(s): 795-804.*

[2] H. Hermansky and N. Morgan, "RASTA processing of speech," *IEEE Transitions on Speech, and Audio Processing, Volume 2, Issue 4, Oct. 1994 Page(s): 578-589.*

[3] D. V. Compernolle, "Noise adaptation in a hidden Markov model speech recognition system," *Comput. Speech Lang., volume 3, 1989 Page(s): 151–167.*

[4] A. Acero, "Acoustical and environmental robustness in automatic speech recognition," *Ph.D. Dissertation, Carnegie Mellon University. 1990.*

[5 L. Neumeyer and M. Weintraub, "Probabilistic optimum filtering for robust speech recognition," *In Proceedings ICASSP, 1994 Page(s) 417-420.*

[6] A. P. Varga and R. K. Moore, "Hidden Markov model decomposition of speech and noise," *In Proceeding ICASSP, 1990 Page(s) 845–848.*

[7] P. J. Moreno, B. Raj and R. M. Stern, "A vector Taylor series approach for environment-independent speech recognition," *In Proceeding ICASSP, 1996 Page(s) 733–736.*

[8] A. Sankar and C. H. Lee, "Robust speech recognition based on stochastic matching," *In Proceedings ICASSP, 1995 Page(s) 121-124.*

[9] M. K. Gales and S. J. Young, "Robust continuous speech recognition using parallel model combination," *IEEE Transitions on Speech, and Audio Processing, Volume 4, Issue 5, Sep 1996 Page(s): 352-359.*

[10] J. Makhoul, "Spectral analysis of speech by linear prediction," *IEEE Transitions on Audio and Electroacoustics, Volume 21, Issue 3, Jun. 1973 Page(s): 140-148.*

[11] S. Furui, "Cepstral analysis technique for automatic speaker verification," *IEEE*

Transitions on Acoustics, Speech, and Signal Processing, Volume 29, Issue 2, Apr. 1981 Page(s): 254-272.

[12] S. B. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Transitions on Acoustics, Speech, and Signal Processing, Volume 28, Issue 4, Aug. 1980 Page(s): 357-366.*

[13] J. C. Junqua, H. Wakita and H. Hermansky, "Evaluation and optimization of perceptually-based ASR front-end," *IEEE Transitions on Speech, and Audio Processing, Volume 1, Issue 1, Jan. 1993 Page(s): 39-48.*

[14] J. R. Deller, J. G. Proakis and, John H. L. Hansen, "*Discrete-time processing of speech signals*," Macmillan Publishing Co., 1993.

[15] S. S. Steven, "On hearing by electrical stimulation," *Journal of the Acoustic Society of America, Volume 8 1937 Page (s): 191-195.*

[16] S. S. Steven and J. Volkman, "The relation of pitch to frequency," *Journal of psychology, Volume 53 1940 Page(s): 329-353.*

[17] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Transitions on Acoustics, Speech, and Signal Processing, Volume 26, Issue 1, Feb. 1978 Page(s): 43-49.*

[18] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceeding of the IEEE, Volume 77, Issue 2, Feb. 1989 Page(s) 257-286.*

[19] L. R. Rabiner and B. H. Juang, "An introduction to hidden Markov models," *ASSP magazine, IEEE, Volume 3, Issue 1, Jan. 1986 Page(s): 4-16.*

[20] C. S. Myers and L. R. Rabiner, "Connected digit recognition using a level-building DTW algorithm," *IEEE Transitions on Acoustics, Speech, and Signal Processing, Volume 29, Issue 3, Jun. 1981 Page(s): 351-363.*

[21] L. R. Rabiner and C. E. Schmidt, "Application of dynamic time warping to connected digit recognition," *IEEE Transitions on Acoustics, Speech, and Signal Processing, Volume 28, Issue 4, Jun. 1980 Page(s): 377-388.*

[22] Y. Ishikawa and K. Nakajima, "A real time connected word recognition system," *Pattern Recognition, Proceeding of 10th International Conference on Volume 2 Jun. 1990 Page(s):215 – 217.*

[23]A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transitions on Information Theory, Volume 13, Issue 2, Apr. 1967 Page(s) 260-269.*

[24] D. A. Reynolds, T. F. Quatieri and R. B, Dunn, "Speaker Verification Using Adapted Gaussian Mixture Models," *Digital Signal Processing Volume 10, 2000 Page(s) 19-41.*

[25] J. G. Wilpon and L. R. Rabiner, "A modified K-means clustering algorithm for use in isolated word recognition," *IEEE Transitions on Acoustics, Speech, and Signal Processing, Volume 33, Issue 3, Jun. 1985.*

[26] A. P. Dempster, N. M. Laird and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Roy. Stat. Soc. 39(1):1-38, 1977.*

[27] "Hidden Markov model Toolkit", http://htk.eng.cam.ac.uk/.

[28] M. K. Gales, "Model based techniques for noise robust speech recognition," *Ph.D. Dissertation, University of Cambridge. 1995.*

[29] M. K. Gales and S. J. Young, "Robust speech recognition in additive and convolutional noise using parallel model combination," *Comput. Speech Lang., vol. 9, 1995 Page(s) 289-307.*

[30] M. K. Gales and S. J. Young, "A fast and flexible implementation of parallel model combination," *In Proceeding ICASSP, 1995 Page(s) 131–136.*

[31] J. W. Hung, J. L. Shen and L. S. Lee, "New approaches for domain transformation and parameter combination for improved accuracy in parallel model combination (PMC) techniques," *IEEE Transitions on Speech and Audio Processing, Volume 9, Issue 8, Nov. 2001 Page(s): 842-855.*

[32] http://spib.ece.rice.edu/spib/select_noise.html