SEMI-CONTINUOUS HIDDEN MARKOV MODELS FOR SPEECH RECOGNITION

Xuedong Huang

黄学东

Thesis submitted for the degree of Doctor of Philosophy

University of Edinburgh

1989

DECLARATION

This thesis, unless explicitly noted, was composed entirely by myself, and reports original work of my own execution.

Xuedong Huang, *B.Sc.*[1] *M.Sc.*[2]

[1] *B.Sc. in Computer Science, Hunan University, 1982*

[2] *M.Sc. in Computer Science and Technology, Tsinghua University, 1984*

ABSTRACT

Hidden Markov models, which can be based on either discrete output probability distributions or continuous mixture output density functions, have been demonstrated as one of the most powerful statistical tools available for automatic speech recognition. In this thesis, a semi-continuous hidden Markov model, which is a very general model including both discrete and continuous mixture hidden Markov models as its special forms, is proposed. It is a model in which vector quantisation, the discrete hidden Markov model, and the continuous mixture hidden Markov model are unified. Based on the assumption that each vector quantisation codeword can be represented by a continuous probability density function, the semi-continuous output probability is then a combination of *discrete* model-dependent weighting coefficients with these *continuous* codebook probability density functions. In comparison to the conventional continuous mixture hidden Markov model, the semi-continuous hidden Markov model can maintain the modelling ability of large-mixture probability density functions. In addition, the number of free parameters and the computational complexity can be reduced because all of the probability density functions are tied together in the codebook. The semi-continuous hidden Markov model thus provides a good solution to the conflict between detailed acoustic modelling and insufficient training data. In comparison to the conventional discrete hidden Markov model, robustness can be enhanced by using multiple codewords in deriving the semi-continuous output probability; and the vector quantisation codebook itself can be optimised together with the hidden Markov model parameters in terms of the maximum likelihood criterion. Such a unified modelling can substantially minimise the information lost by conventional vector quantisation. Evaluation of the semi-continuous hidden Markov model was carried out in a range of speech recognition experiments and results have clearly demonstrated that the semi-continuous hidden Markov model offers improved speech recognition accuracy in comparison to both the discrete hidden Markov model and the continuous mixture hidden Markov model. It is concluded that the unified modelling theory is indeed powerful for modelling non-stationary stochastic processes with multi-modal probabilistic functions of Markov chains, and as such is very useful for automatic speech recognition.

ACKNOWLEDGEMENTS*

First of all, I would like to acknowledge with deep gratitude the support, patience, wisdom, and encouragement of Professor Mervyn Jack, my supervisor at Edinburgh University. His guidance made this work possible, his encouragement sustained it through the difficult stages, and his unending quest for excellence stimulated me to achieve many wonderful results. I would also like to thank Professor John Laver and Professor James Hieronymus for their interest in my work.

Part of the research in this thesis was carried out in the School of Computer Science, Carnegie Mellon University, USA. I would like to thank Dr. Kai-Fu Lee and Professor Raj Reddy, who made my visit to Carnegie Mellon University possible. In particular, Kai-Fu gave me invaluable advice in my research, and helped shape many aspects of this thesis.

I would like to extend my thanks to many other colleagues in Edinburgh University, Carnegie Mellon University, and Tsinghua University. I must thank in particular Dr. Yasuo Ariki, Dr. Fergus McInnes, and Mr. Hsiao-Wuen Hon, who helped me considerably in this research. I am grateful to Dr. George Duncan, Dr. Andrew Sutherland, Mr. Alan Crowe, Mr. Art Blokland, and Ms. Mei-Yuh Hwang for their help. I am also deeply indebted to Professor Ditang Fang and Professor Tong Chang, Tsinghua University, China, and Dr. Frank Soong, Bell Laboratories, USA, whose continuous support and encouragement will never be forgotten.

Finally, I wish to thank my family and friends for their support. I am especially grateful to my parents for all the sacrifices they have made over the years, and who by their teaching and example, have taught me to appreciate the importance and honour of scholarly work. My most loving gratitude is reserved for my wife, Yingzhi. She encouraged and cared for me during the long days and nights of my research.

Although I am indebted to the above people for their aid in this endeavour, I accept full responsibility for the final version of this thesis.

* This work was supported by an Edinburgh University Studentship and ORS Award.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Speech has evolved over many centuries to achieve today's rich and elaborate form. Human communications are today dominated by spoken language, whether face to face, over the telephone, or through television and radio. In direct contrast, human-machine (computer) interaction is largely dependent on keyboard strokes, or other mechanical means. As such, this interaction mode demands skill development by individuals and presents a barrier to widespread use of computer systems. Consequently, the goal of overcoming this barrier by building machines that understand spoken language has attracted the attentions of scientists over the past 50 years. Usage of a spoken language understanding interface would be invaluable since speech communication is a natural and efficient mode for the human user. Example applications include automatic dictation (especially for Chinese and Japanese), database query (such as airline reservations), command and control, and computer-assisted instruction. Achievement of this spoken language understanding goal demands integration of both speech processing and natural language processing. One of the key problems remains to be automatic speech recognition. The task of a speech recognition system is to take as input, the acoustic waveform produced by the speaker and to produce, as output, a sequence of linguistic words corresponding to the input utterance.

Many uncertainties exist in speech recognition. The uncertainty associated with words that have been spoken to a speech recognition system is compounded by the acoustic uncertainty of the different accents and speaking styles of individual speakers; by the lexical and grammatical uncertainty of the language the speaker uses; and by the semantic uncertainty of the subject the speaker wants to talk about. The speaker may inquire about flights to Beijing, or may reserve a ticket to Edinburgh, or may even be dictating an article in Chinese. Acoustic uncertainty has many components, such as the general quality of a speaker's voice; speaking speed and loudness; accent; and unusual

speaking conditions such as illness or stress. In addition, acoustic contaminants such as room noise or competing speakers constitute a problem. A successful speech recognition system must take into account all of these acoustic uncertainties. Lexical, syntactic, and semantic knowledge must then be applied in a manner which permits cooperative interaction among the various levels of acoustic, phonetic and linguistic knowledge in minimising the uncertainty. However, when compared with human performance, only very restricted speech can currently be used in existing speech recognition systems. The principle constraints include:

(1) speaker dependence rather than speaker independence,

(2) isolated word input rather than continuous speech operation,

(3) limited rather than extensive vocabulary, and

(4) artificial grammar rather than natural language.

Scientists with backgrounds in signal processing, pattern recognition, artificial intelligence, linguistics, statistics, information theory, and psychology have been attacking the many problems of speech recognition. Their efforts can be broadly classified into:

(1) Modelling of the speech signal and its variability to facilitate efficient information extraction. These variabilities include phonetic and linguistic effects, inter-speaker and intra-speaker variabilities, and environmental acoustic variabilities.

(2) Automatic acquisition and modelling of linguistic events (lexicon, syntax, semantics, discourse, pragmatics, and task structure).

(3) Developing human factors methods for the design of an effective user interface.

Research in speech recognition has followed two primary routes — those adopting a knowledge-based approach, and those adopting a statistically data-based approach. Knowledge-based approaches to speech recognition and understanding [89], have attempted to express human knowledge of speech in terms of acoustic-phonetic rules based on specified *features* of the acoustic waveform. For these approaches, the acoustic signal is usually first segmented and labelled into phoneme-like units, and the resulting phoneme string is used for lexical and syntactic analysis. Words in the lexicon are represented in terms of phonemic spellings, and syntax is usually described by

conventional linguistic means. Knowledge is represented in computer programs created by linguistic and phonetic experts [33,172]. It is known that human experts can be trained to *read* speech spectra giving strength to the proposition that distinct features exist in the speech spectrum [172]. Machine realisation of this human ability is however currently far poorer than the well-trained human expert. In addition to the absence of good understanding of the human auditory mechanism, this knowledge-based approach is also constrained by the inability of human experts to completely formalise their knowledge. Here totally reliable features are required to represent speech signals, before acoustic segmentation, phonetic labeling and lexical decoding can be carried out with any degree of accuracy. Formants are considered to be one of the most important features in speech recognition, and various methods have been developed to track formants from speech signals. None of this work to date has achieved the required accuracy for speech recognition and it can be argued that even if an excellent formant tracker were available, *a-priori* knowledge would still be needed to indicate phonetic context for formant tracking. However, without good feature representation (of formants etc.), it is extremely difficult to obtain the necessary *a-priori* phonetic knowledge based on these features. Thus some sophisticated and interactive formant tracking algorithms are necessary in order to obtain reliable formant estimation. This remains an unsolved problem for the knowledge-based approach. It should be noted, however, that the knowledge-based approach remains an important research area [66,173].

In contrast to the knowledge-based approach, alternative *data-based* statistical approaches have achieved considerable success. These are usually based on modelling the speech signal itself by some well defined statistical algorithms that can automatically extract *knowledge* from speech data. This thesis will focus on the alternative statistical approaches, and the knowledge-based approach will not be considered further in this work. The predominant class of these algorithms is the hidden Markov model (HMM) [7,15,80,96]. An HMM-based speech system depends on three key factors:

(1) a detailed modelling scheme which is capable of accommodating various uncertainties,

(2) access to sufficient speech training data, and

(3) an automatic learning algorithm to improve the recognition accuracy.

By using HMMs, the speech signal variability in parameter space and time can be modelled effectively. Unlike the knowledge-based approach, the HMM learning procedure is done by presenting speech data to HMMs and automatically improving the models by data as opposed to some heuristic rules presented by human experts. In general, the more data presented to the model, the higher the recognition accuracy achieved. Motivated by neural network research, improvements can also be obtained by incorporating classification into parameter estimation [12,43]. HMM methods have presented speech recognition with a solid theoretical basis, and have resulted in significant advances in large-vocabulary continuous speaker-independent speech recognition [96].

The HMM can be based on either discrete output probability distributions or continuous output probability density functions, which are very important to acoustic modelling. Both the discrete HMM and continuous HMM are widely used in state-of-the-art speech recognition systems [7,43,96,128,140]. For the discrete HMM, vector quantisation (VQ) makes it possible to use a nonparametric, discrete output probability distribution to model the observed speech signals. The objective of VQ is to find the set of reproduction vectors, or codebook, that represents an information source with minimum expected distortion. Under the discrete HMM framework, VQ is first used to obtain discrete output symbols. The discrete HMM then models observed discrete symbols. In contrast, the continuous mixture HMM [138] uses continuous mixture probability density functions to directly model speech parameters without using VQ and usually needs extensive training data and computation times.

In this thesis, a semi-continuous HMM, which is a very general model including both discrete and continuous mixture HMMs as its special forms, is proposed. It is a model in which VQ, the discrete HMM, and the continuous mixture HMM are unified. Based on the assumption that each VQ codeword can be represented by a continuous probability density function, the semi-continuous output probability is then a combination of *discrete* model-dependent weighting coefficients with these *continuous*

codebook probability density functions. In comparison to the conventional continuous mixture HMM, the semi-continuous HMM can maintain the modelling ability of large-mixture probability density functions. In addition, the number of free parameters and the computational complexity can be reduced because all of the probability density functions are tied together in the codebook. The semi-continuous hidden Markov model thus provides a good solution to the conflict between detailed acoustic modelling and insufficient training data. In comparison to the conventional discrete HMM, robustness can be enhanced by using multiple codewords in deriving the semi-continuous output probability; and the VQ codebook itself can be optimised together with the HMM parameters in terms of the maximum likelihood criterion. Such a unified modelling can substantially minimise the information lost by conventional VQ. The unified modelling approach leads to superior performance in comparison to both the discrete HMM and the continuous mixture HMM.

1.1. Thesis Organisation

This thesis will be concerned with the statistical modelling of speech signals. In particular, acoustic modelling problems will be studied. A unified modelling approach to hidden Markov modelling and VQ with a semi-continuous HMM is proposed.

Throughout this thesis, unless explicitly noted, discrete probability of finite symbols O will be denoted by $Pr(O)$; and continuous probability density function for the continuous observations $\mathbf{x}$ will be denoted by $f(\mathbf{x})$. Fundamental techniques and principles involved in speech recognition, and the results of experiments by other researchers with various formulations and extensions are first reviewed and discussed in Chapter 2.

Chapter 3 describes VQ techniques which have been widely used in speech processing, coding and recognition. The modelling method can be viewed as a problem of estimating parameters for a family of continuous mixture probability density functions, which pave the way for the unified modelling approach of the VQ and HMM.

Mathematical principles of the HMM and related techniques for speech recognition are described in Chapter 4. This Chapter is the foundation of the statistical modelling tool, the HMM, which will be discussed throughout this thesis. Chapter 5 describes continuous HMMs, which parallel discrete HMMs. The continuous mixture HMM is discused in detail, since it is strongly related to the semi-continuous HMM. Chapters 2-5 represent the theoretic foundation to the unified modelling theory in this thesis.

The semi-continuous HMM is presented in Chapter 6. The unified theory of VQ and hidden Markov modelling and related techniques for speech recognition, which are heavily relevant to the discussion in Chapter 4 and Chapter 5, are highlighted. The semi-continuous HMM offers modelling power similar to the continuous mixture HMM with a large number of mixture density functions, while demanding much lower computational complexity than the continuous mixture HMM. In addition, the semi-continuous output probability density function can be well smoothed in comparison to the discrete HMM. From the discrete HMM point of view, the semi-continuous HMM can minimise the information lost by VQ. From the continuous mixture HMM point of view, the semi-continuous HMM can reduce the number of free parameters and computational complexity because of tying of continuous density functions.

Chapter 7 discusses issues for designing a speech recognition system using HMMs. Topics such as choice of modelling unit, use of smoothing techniques, scaling, multiple features, and other implementation problems are included.

Chapter 8 begins the presentation of practical results and contains results of experiments with isolated digit recognition. The experimental results of semi-continuous HMM without reestimation of the VQ codebook are discussed and compared with other techniques. Thereafter, Chapter 9 contains results of experiments in phoneme classification. The experimental results of semi-continuous HMM with simplified unified techniques are discussed and compared with other techniques.

Chapter 10 describes experiments carried out in large-vocabulary speaker-independent continuous speech recognition, exploring the effectiveness of the unified modelling theory introduced in Chapter 6. The relationships among continuous HMMs, discrete HMMs, and semi-continuous HMMs are explored by conducting various

experiments.

Finally, Chapter 11 summarises the major results in this thesis, and contributions to the field of speech recognition. Possible future work is also discussed.

CHAPTER 2

FUNDAMENTAL TECHNIQUES FOR SPEECH RECOGNITION

The task of speech recognition is to recover a linguistic message which is encoded in the acoustic speech signal of an utterance. Various systems and methods have been proposed, tested, and abandoned. That is the history of speech recognition over the past fifty years. Advances in computing technology, the availability of extensive, characterised speech database, and the creation of powerful new statistical algorithms have greatly renewed interest in the field. A typical speech recognition system will consist of three major components — signal processing, acoustic pattern matching, and language modelling. In the signal processing stage, a speech signal is converted into a sequence of information-bearing analysis frames. The acoustic pattern matching stage then interprets these frame sequences into possible linguistic words. The language modelling stage determines valid linguistic words or sentences spoken. It should be pointed out here that these components may well not be separable in practice, especially in HMM-based speech recognition systems. This Chapter will briefly review those techniques that are important for state-of-the-art speech recognition systems.

2.1. Signal Processing for Speech Recognition

The purpose of signal processing is to derive a set of parameters to represent speech signals in a form which is convenient for subsequent processing. Both time domain and frequency domain approaches can be used [49,135]. Time domain approaches, such as parameters of energy and zero crossing rate, deal directly with the waveform of the speech signal and are usually simple to implement. Frequency domain approaches involve some form of spectral analysis and usually display characteristics that are not directly evident in the time domain: these are the most widely used signal analysis techniques in speech recognition. Associated with the signal analysis method is a distortion measure which calculates the distortion dissimilarity between two specific speech frames. In statistical modelling, the distortion measure is actually based on the probability density function created from a large number of characterised speech frames. Various techniques of signal processing and feature extraction for speech recognition have been reported. Most of these techniques highlight reliable and tractable representation of speech signal spectra, notably those based on linear predictive coding (LPC) [4,78,108,135] analysis and those based on short-time Fourier analysis [35,135], often in combination with vector quantisation or formant frequency estimation.

2.1.1. Short-time Fourier analysis of speech signals

Short-time Fourier (spectral) analysis is a method for analysing time-varying waveforms. Components of the speech signal are time-varying at the *articulator rate*, so the speech waveform is suited to short-time analysis. Short-time analysis depends on *windowing* of the speech waveform and the results depend on the properties of the specific window function employed. With a window of finite time duration, the window can move progressively along the speech signal to select short sections for analysis. A number of fundamental concepts and definitions of short-time Fourier analysis can be found in [49,135], and details of a typical speech recognition system based on short-time Fourier analysis can be found in [82].

Bandpass filter analysis methods can be considered as a special form of short-time Fourier analysis [135], and such bandpass filter methods have been widely used in speech recognition [24,35]. The bandpass filter system may use a variety of different frequency spacing [35,36,168], and use of frequency spacing defined from auditory modelling, such as mel-scale, or bark-scale, may improve system performance in terms of recognition accuracy.

2.1.2. LPC analysis of speech signals

Linear predictive coding (LPC) can provide a complete description for a speech production model at the vocal tract level. The basic idea underlying LPC is that each speech sample, x_n, can be represented as a linear combination of previous samples, and prediction errors can then be minimised according to the mean-square value of the prediction error, e_t, which is defined by

$$e_t = x_t + \sum_{i=1}^{p} \alpha_i x_{t-i}. \tag{2.1.1}$$

where p is the order of LPC analysis; and α_i are LPC coefficients. The LPC coefficients which minimise the mean-squared prediction error can be obtained by setting the partial derivative of the mean-squared prediction error (with respect to each α_i) equal to zero. Minimising the prediction error is equivalent to minimising the integrated ratio of the speech spectrum to the estimated all-pole model spectrum [135]. This implies that given the spectrum of a speech signal, the LPC technique models the spectrum as a smooth spectrum of an order-p all-pole filter such that the integrated sum of the ratio between the two spectra is minimised. The value of p required for adequate modelling of the vocal tract depends on the sampling frequency used in digitisation of the signal: the higher the sampling frequency, the larger the analysis order p should be. It has been suggested that when the sampling frequency in kHz is n the analysis order should be at least $n+4$ [108]. Detailed description of such LPC details can be found in [49,108,135].

From the stochastic process point of view, a speech signal, x_t, can be considered as a time series of a stationary Gaussian process. An N-sample segment of a speech signal, $\mathbf{x}=(x_{c+1},x_{c+2},\cdots,x_{c+N})^t$, (here c is a constant, which may be assumed to be

zero for simplicity) can be assumed to be a segment of a process with a spectral density of the all-pole rational form. The *maximum likelihood* method can then be used to estimate the unknown parameters, $\{\alpha\}$, of the process density [78], which result in the same formulation of minimisation of the mean-square of prediction error over the period of time N. Maximum likelihood estimation is the most commonly used criterion in parameter estimation. The purpose is to use the information provided by known samples to obtain good estimates for the unknown parameter. Intuitively, good estimates should correspond to the value that in some sense best agrees with the actually observed samples. The *likelihood* can then be defined as a function of parameters, $\{\alpha\}$, with respect to the set of samples, namely, $Pr(\mathbf{x}|\alpha)$. The maximum likelihood estimate of $\{\alpha\}$ is then the value that maximises the likelihood function. From such a statistical point of view, LPC analysis can be closely welded into hidden Markov modelling [85,131] to provide a computationally efficient model for speech recognition.

2.1.3. Cepstral analysis of speech signals

The basic model of speech production can be considered as a vocal tract filter excited by a periodic excitation function for voiced speech or white noise in the case of unvoiced speech. The observed speech sequence is thus a convolution of the excitation and the vocal tract filter impulse response in the time domain or the product of the excitation and the filter spectra in the frequency domain. Short-time spectra comprise a slowly varying envelope corresponding to the vocal tract filter and, in the case of voiced speech, a rapidly varying fine structure corresponding to the periodic excitation frequency and its harmonics [135]. If, in the frequency domain, the product of the excitation and filter spectra is transformed to the summation of these two spectra (logarithm operation), the transformation from the frequency domain back to the time domain results in the *cepstrum*, which has a number of properties well suitable to the deconvolution of speech [135,148].

Cepstral coefficients which can also be obtained from LPC analysis [135], have been widely used in speech recognition. The cepstral coefficients, c_n, of the spectra obtained from LPC analysis can be computed recursively from the LPC coefficients, α_i, as:

$$c_n = -\alpha_n - \sum_{k=1}^{n-1} \frac{n-k}{n} \alpha_k c_{n-k}, \quad n \geq 1 \tag{2.1.2}$$

where $\alpha_k = 0$ when k>p.

A variety of speech recognition systems using cepstral analysis have been reported [31,36,96,140]. A distinctive advantage of the cepstral analysis is that correlation between coefficients is extremely small such that simplified modelling assumptions can be applied. This is a very useful property as will be discussed in Chapter 10.

2.1.4. Distortion measures

The distortion measure, also known as distance or dissimilarity measure, between two speech frames has played a key role in speech coding, analysis, and recognition. Several studies have been conducted to investigate the properties of distortion measures from both theoretical and practical points of view [59,60,124].

Because of the uncertainty and randomness of speech signals, the distortion measure can be generally replaced by employing a continuous probability density function, in which the larger the density, the smaller the distortion. The parameters of a template probability density function can be estimated from extensive training data, which will usually lead to a more robust representation in comparison to conventional distortion measures.

The most widely used density function is the Gaussian density function which can be written as:

$$f(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} e^{-1/2(\mathbf{x}-\mu)^t \Sigma^{-1}(\mathbf{x}-\mu)} \tag{2.1.3}$$

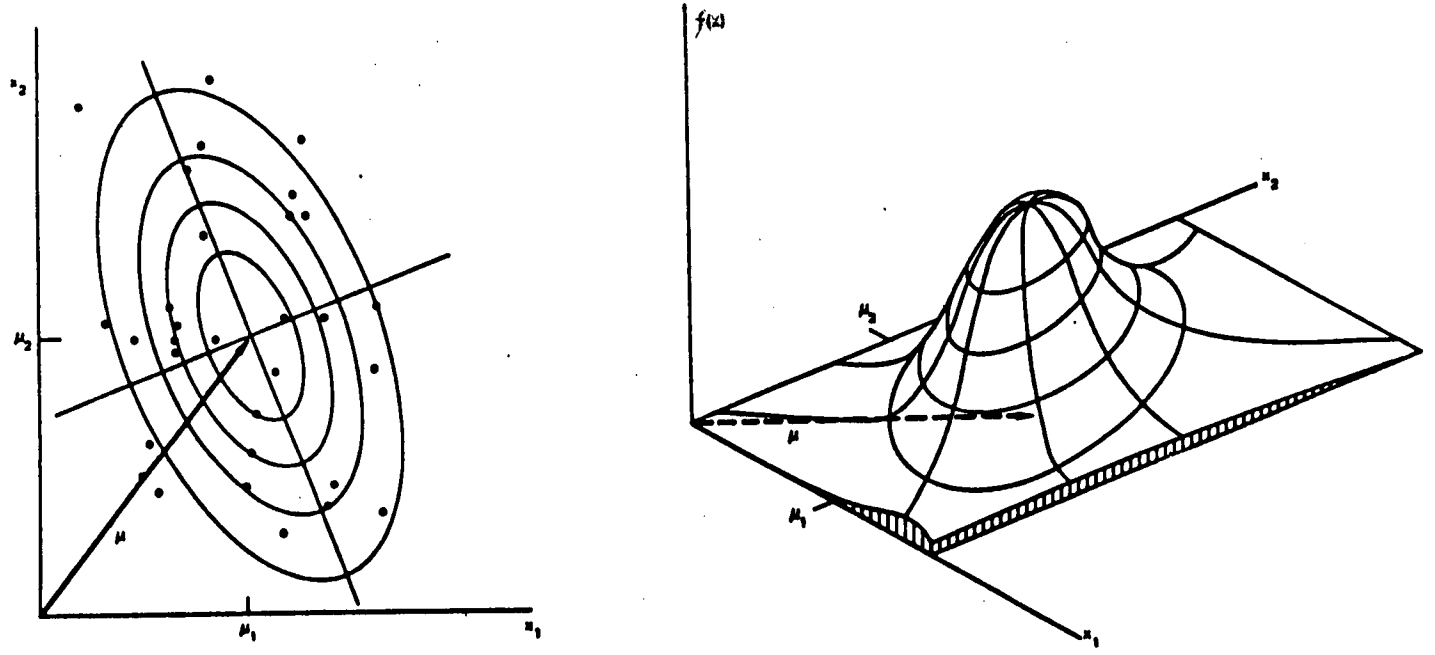
where $\mathbf{x}$ is a d-component vector, μ is the d-component mean vector, Σ is the d-by-d covariance matrix. $(\mathbf{x}-\mu)^t$ is the transpose of $\mathbf{x}-\mu$, Σ^{-1} is the inverse of Σ, and $|\Sigma|$ is the determinant of Σ. For simplicity, Eq. (2.1.3) is often abbreviated as $N(\mu,\Sigma)$. The covariance matrix Σ is always symmetric and positive semidefinite. Σ is restricted to be positive definite. Let x_i denote the ith component of $\mathbf{x}$, and σ_{ij} denote the $i-j$th component of Σ. The diagonal element σ_{ii} is the variance of x_i, and the off-diagonal

element σ_{ij} is the covariance of x_i and x_j. If x_i and x_j are statistically independent, $\sigma_{ij}=0$. This results in diagonal covariance matrices. If all the variance σ_{ii} are assumed to be the same, Eq. (2.1.3) will deteriorate to the Euclidean distortion measure. Samples drawn from a Gaussian population tend to fall in a single cluster as Figure 2.1.1. The centre of the cluster is determined by the mean vector, and the shape of the cluster is determined by the covariance matrix. It follows from Eq. (2.1.3) that the loci of points of constant density are hyperellipsoids for which the quadratic form $(\mathbf{x}-\mu)^t\Sigma^{-1}(\mathbf{x}-\mu)$ is constant. The principal axes of these hyperellipsoids are given by the eigenvectors of Σ, and the eigenvalues determine the lengths of these axes. The volume of these hyperellipsoids measures the scatter of the samples about the mean, which varies directly with $|\Sigma|^{1/2}$. As the Euclidean-like distortion measure can be viewed as a special form of Gaussian density, parameters suitable to a Euclidean-like distortion measure, such as LPC cepstral coefficients, can also be modelled by Gaussian density functions.

2.1.5. Vector quantisation

Vector quantisation (VQ) is an information-theoretic data compression principle introduced by Shannon [153]. For a specified transmission rate, the objective of VQ is to find the set of reproduction vectors, or codebook, that represents an information source with minimum expected distortion. The data compression is achieved by using the codebook vector index rather than the original source vector. VQ has been successfully used in speech coding, image coding, and speech recognition [62,96,99,101,107,115]. In hidden Markov modelling, VQ makes it possible to use a discrete probability distribution to model the observed speech signals.

As conventional VQ involves the classification of data into a discrete number of categories, some information may be lost. An alternative approach for VQ is to avoid partitioning of acoustic space into separate regions. Here, the codebook can be represented by a parametric family of mixture probability density functions with each codeword represented as a probability density function. These distributions are



(a) Scatter diagram

(b) Bivariate Gaussian density

Figure 2.1.1. Two representations of a Gaussian density function

overlapped in acoustic space. Multiple codewords with similar acoustic properties can then be used together to model acoustic events. Although parameter estimation of such a family can be obtained from various techniques [38,141], the method of maximum likelihood estimation has been studied as the most widely preferred approach to mixture density estimation problems. Associated with obtaining maximum-likelihood estimates, computational difficulties arise because of the complex dependence of the likelihood function on the parameters to be estimated. The customary way of finding a maximum-likelihood estimate is first to determine a system of equations called the likelihood equations which are satisfied by the maximum-likelihood estimate, and then to attempt to find the maximum-likelihood estimate by solving these likelihood equations. The maximum likelihood estimates are usually found by differentiating the logarithm of the likelihood function, setting the derivatives equal to zero, and performing some additional algebraic manipulations. Unfortunately, for mixture density problems, the likelihood equations are almost certain to be nonlinear and beyond hope of analytic solution. Consequently, it is necessary to seek an approximate solution via some iterative procedure. The EM algorithm [38] is such a distinctive iterative procedure with which VQ and subsequent hidden Markov modelling can be unified together. VQ and its relation to hidden Markov modelling will be discussed in detail throughout this thesis.

2.2. Acoustic Pattern Matching

With a given speech representation, acoustic pattern matching will detect and classify possible acoustic patterns, which can be phonemes, syllables, words, or sentences, from speech signals. Acoustic pattern matching forms the central issues in speech recognition research. The most important progress has been achieved using techniques based on the dynamic time warping (DTW) algorithm, the hidden Markov model (HMM), and neural networks.

2.2.1. Dynamic time warping algorithm

Dynamic time warping (DTW) [79,136,146], also known as Dynamic Programming (DP) matching, was introduced for nonlinear time alignment of speech patterns. DTW can effectively minimise errors occurring during time alignment of two speech sequences, and can significantly improve speech recognition accuracy in comparison to other non-aligned matching techniques. The basic idea of DTW, nonlinearly *stretching* or *compressing* a signal in time, has been used in various speech recognition systems, including HMM-based speech recognition systems where it is better known as the Viterbi decoding algorithm [53,163].

Consider two acoustic templates, **X**, **Y**, which consist of sequences of representations of short time segments or "frames" of the speech waveform.

$$\begin{aligned} \mathbf{X} &= \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{T_x} \\ \mathbf{Y} &= \mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{T_y} \end{aligned} \tag{2.2.1}$$

where T_x and T_y are the total frames of **X** and **Y** respectively; and $\mathbf{x}_i$ for the ith frame of **X** may be a vector of bandpass filter outputs, or a set of cepstral coefficients. In general, even for the same spoken word, the acoustic realisation of the word may vary significantly with effects such as articulatory rate, causing T_x and T_y to be different. In comparing the two templates, i.e. computing the distortion measure between **X** and **Y**, each frame of **X** is matched with a frame of **Y**, and a frame distortion, $d(\mathbf{x}_i, \mathbf{y}_j)$, is computed as described previously. Then an overall distortion, $D(\mathbf{X}, \mathbf{Y})$, is computed from the frame distortions for all the matched pairs of frames. As there are acoustic realisation variances between **X** and **Y** with time, the optimal comparison point will correspond to the matching together of the mth frame of **X** and the nth of **Y**, stretched or compressed according to the time variances of the two templates rather than simply matching points linearly. Intuitively, the matching between **X** and **Y** will be optimum in terms of temporal alignment if the time axis is transformed before the comparison starts. The sequence of matched pairs of **X** and **Y** form a time registration path, which can be depicted in Figure 2.2.1. The analysis vectors (frames) of the two utterances are positioned with their first frames in the bottom left corner of the figure with subsequent vectors following in the x-axis and y-axis direction respectively. The slope of

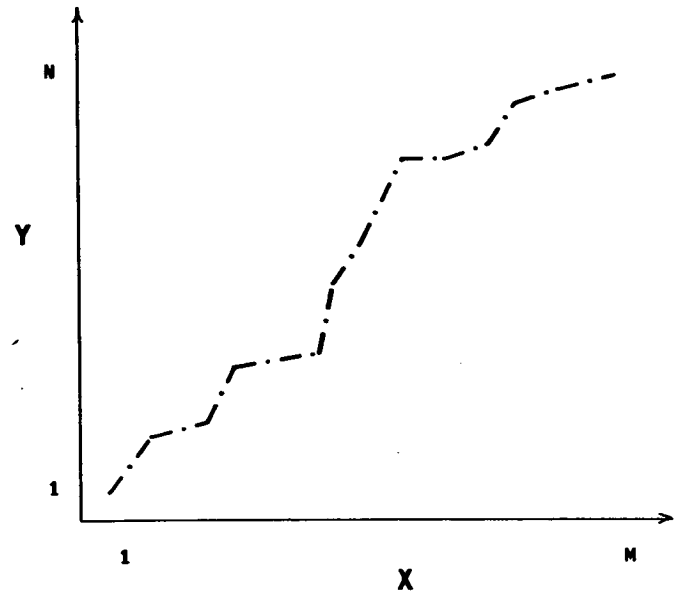


Figure 2.2.1. Sequence of matched pairs in the DTW algorithm

the path represents the degree of compression applied to **Y** in aligning it with the frames of **X**. In particular, a vertical step in the path corresponds to the matching of two successive reference frames to the same frame of **X**, and a horizontal step corresponds to the matching of the same frame of **Y** to two successive frames of **X**.

The overall distortion is a weighted sum of the individual frame distortions for pairs of frames on the path. The weight given to each frame distortion measure can be made to depend on the slope of the time registration path near the point defined by the pair of frames in question. The time registration problem involves finding the best possible time registration path for **X** and **Y**, i.e. the path which minimises the overall distortion subject to appropriate constraints, such as endpoint, continuity, and monotonicity constraints [136]. This can be done by proceeding along **X** one frame at a time and, for each successive frame $\mathbf{x}_i$, computing a frame distortion $d(\mathbf{x}_i, \mathbf{y}_j)$ and an

accumulated distortion $D(i,j)$ for each value of j permitted by the search area constraints. The accumulated distortion is the weighted sum of the frame distortions on the optimal partial path from the initial point to (i,j) and is found by optimising over the points permitted as predecessors of (i,j) on such partial paths.

Typical DTW-based speech recognition systems can be found in [23,79,111,114,121,136,146,147]. The DTW-based approach is a non-parametric technique; and many speech templates are required to accommodate various uncertainties. This results in extensive computational load in the decoding procedure, as well as an extended training procedure. It has been shown that DTW can be considered a special case of hidden Markov modelling, which is a parametric technique and offers flexibility and improved recognition accuracy [25,84].

2.2.2. Hidden Markov modelling

Statistics and probability theory have much to offer to speech recognition. First, classical multivariate statistical distributions, defined over a given pattern space, provide an adequate model for the variability of the pattern representations. Second, the question of whether or not a given pattern *belongs* in some pattern class may naturally be treated as a test of hypothesis, or as a special case of the statistical decision theory problem. For more than two decades, statistical pattern classification has been a healthy branch of pattern recognition [42,45]. Applications of basic theories of statistical pattern recognition, such as Bayesian decision [5,27,82,164], Bayesian learning [160], and feature analysis [26], can be widely found in speech recognition. The most distinctive technique used for speech recognition, however, is hidden Markov modelling, which is a technique for the study of observed items arranged in a discrete-time series. The items in the series can be countably or continuously distributed; they can be scalars or vectors. The HMM has been shown to represent one of the most powerful statistical tools available for modelling speech signals, and has been successfully used in automatic speech recognition [7,14,31,82,96,100,137,138], speech signal processing [50,92], and language modelling [87,117]. This thesis will concentrate on the HMM technique, in particular, with specific emphasis on its use in acoustic modelling.

The work of Markov [109] and Shannon [151,152] was concerned with Markov chains. In the *hidden* Markov model, the output probabilities impose a veil between the state sequence and the observer of the time series. In an effort to lift the veil, a substantial body of theory has been developed. The initial work [17,18,20] dealt with finite probability spaces and addressed the problems of tractability of probability computation; the recovery of the hidden states; iterative maximum likelihood estimation of model parameters from observed time series; and the proof of consistency of the estimators. A major development in the theory was the maximisation technique of Baum [19] that extended coverage to many of the classical distributions. This work has led to a wide range of theoretical outgrowths. They include a number of generalisations, such as variable duration HMMs [100,144], continuous mixture HMMs [86,102], autoregressive HMMs [85,131], semi-continuous HMMs [72,75], and trainable finite-state (hidden) grammars [16]. A special case of the results in [19] has been designated by Dempster [38] for maximum likelihood estimation of mixture density functions known as the EM algorithm. The HMM uses a Markov chain to model the changing statistical characteristics that exist in the actual observations of speech signals. The Markov process is therefore a double stochastic process in which there is an unobservable Markov chain defined by a state transition matrix, and where each state of the Markov chain is associated with either a discrete output probability distribution (discrete HMM) or a continuous output density function (continuous HMM). The double stochastic processes enable modelling of not only acoustic phenomena, but also timescale distortions. Unlike other non-parametric and ad hoc approaches, the parameters estimated from the Baum-Welch algorithm [19] guarantee a finite improvement on each iteration in the sense of maximisation of likelihood, and converge after only a few iterations using computationally efficient algorithms.

The HMM is a parametric modelling technique in contrast to the non-parametric DTW algorithm [25,84]. If the Viterbi algorithm is used for decoding in HMM-based speech recognition, it is actually the same as the DTW algorithm except that the probability between the test and reference model is computed in the HMM rather than the distortion measure between speech frames in the DTW system. The power of the HMM lies in the fact that the parameters that are used to model the speech signal can be well optimised, and this results in lower computational complexity in the decoding

procedure as well as improved recognition accuracy. Furthermore, other knowledge sources can also be represented with the same structure, which is one of the important advantages of hidden Markov modelling.

2.2.3. Neural networks

In the area of speech processing, besides the extensive active research work on hidden Markov modelling in recent years, the advent of new learning procedures and the availability of high speed parallel supercomputers have given rise to a renewed interest in neural net models [103,143]. Neural networks are particularly interesting for speech recognition, which requires massive constraint satisfaction, i.e., the parallel evaluation of many clues and facts and their interpretation in the light of numerous interrelated constraints. Because of the high degree of uncertainty and variability of speech, complex networks employing automatic learning algorithms [104] to discover internal abstractions for speech recognition are becoming very attractive [22,28,58,64,69,105,129,142].

The computational flexibility of the human brain comes from its large number of neurons in a mesh of axons and dendrites. The communication between neurons is via the synapse and afferent fibres. There are many billions of neural connections in the human brain. At a simple level it can be considered that nerve impulses are comparable to the phonemes of speech, or to letters, in that they do not themselves convey meaning but indicate different intensities [171] which are interpreted as meaningful units by *the language of the brain*. Artificial neural networks attempt to achieve real-time response and human-like performance using many simple processing elements operating in parallel as in biological nervous systems. Models of neural networks use a particular topology and a learning algorithm for the interactions and interrelations of the connections of the *neural units*.

Three important practical neural networks are the single-layer perceptron [45], the multi-layer perceptron [69,143], and Kohonen's feature map algorithm [91]. The most distinctive feature of neural networks is that neural classifiers compute matching scores in parallel and have parallel inputs and outputs where internal parameters (connection

weights) are typically trained adaptively using training data. With the development of the back propagation algorithm for learning [143], multi-layer perceptrons have been widely used, in feed-forward networks with one or more layers of nodes between the input and output nodes. The back propagation algorithm is a generalisation of the least-mean-square (LMS) algorithm. It uses a gradient search to minimise the difference between the desired outputs and the actual net outputs, where the optimised criterion is directly related to pattern classification. With initial parameters for the weights, the training procedure is then repeated to update the weights until the cost function is reduced to an acceptable value or remains unchanged. These weights are estimated from a large number of training observations in a manner similar to hidden Markov modelling except that here the estimation criterion is directly related to classification rather than the maximum likelihood.

Speech recognition using multi-layer perceptrons trained with back propagation has so far mostly been aimed at isolated word recognition [28,58] or isolated phoneme recognition [134,165,166] because speech signals must be segmented prior to neural network modelling. A number of these studies have reported encouraging recognition performance for isolated speech recognition [165] and limited success in continuous speech recognition [54,64].

2.3. Language Modelling

Acoustic pattern matching is only the first step in the recognition and understanding of natural continuous speech. Lexical knowledge is required (i.e. vocabulary definition) as is the syntax and semantics of the language (i.e. the rules that determine what sequences of words are grammatically well-formed and meaningful). In addition, knowledge of the pragmatics of language can be of value (i.e. knowledge of the structure of extended discourse and knowledge of what people are likely to say in particular contexts). In practical speech recognition, it may not be possible to separate the use of these different levels of knowledge. Specifically, language modelling here refers to syntax constraints.

In a speech recognition system, every string of words $\mathbf{W} = w_1, w_2, \ldots, w_n$ taken from the prescribed vocabulary can be assigned a probability, which is interpreted as the *a priori* probability that the speaker will say that string. These probabilities guide the recognition process and are a contributing factor in determination of the final transcription from a set of partial hypotheses. Given acoustic evidence observation $\mathbf{O}$, the operations of speech recognition are to find the most likely word string, $\hat{\mathbf{W}}$, satisfying

$$Pr(\hat{\mathbf{W}}|\mathbf{O}) = \max_{\mathbf{W}} Pr(\mathbf{W}|\mathbf{O}) \tag{2.3.1}$$

The right-hand side of above equation can be rewritten according to Bayes formula as

$$Pr(\mathbf{W}|\mathbf{O}) = \frac{Pr(\mathbf{W})Pr(\mathbf{O}|\mathbf{W})}{Pr(\mathbf{O})} \tag{2.3.2}$$

where $Pr(\mathbf{W})$ is the *a priori* probability that the word string $\mathbf{W}$ will be uttered. $Pr(\mathbf{O}|\mathbf{W})$ is the probability that when the speaker says word string $\mathbf{W}$ the acoustic evidence $\mathbf{O}$ will be observed (this is the output of the acoustic pattern matching model as discussed in the previous Sections), and $Pr(\mathbf{O})$ is the average probability that $\mathbf{O}$ will be observed. Since $Pr(\mathbf{O})$ is not related to $\mathbf{W}$, it is irrelevant to recognition. It follows from Eq. (2.3.1) and (2.3.2) that the purpose of the recognition operation is to find the word string $\hat{\mathbf{W}}$ that maximises the product

$$Pr(\hat{\mathbf{W}})Pr(\mathbf{O}|\hat{\mathbf{W}}) = \max_{\mathbf{W}} Pr(\mathbf{W})Pr(\mathbf{O}|\mathbf{W}) \tag{2.3.3}$$

where $Pr(\mathbf{O}|\mathbf{W})$ rely on acoustic pattern matching. The *a priori* probability $Pr(\mathbf{W})$ whose probabilities are given by the language model are thus as important as acoustic pattern matching to a speech recognition system. The performance of a speech recognition system is therefore directly related to the quality of language modelling, namely, the usable constraints $Pr(\mathbf{W})$. Without language modelling, the entire vocabulary must be considered at every decision point. With a language model, it is possible to eliminate many candidates from consideration, or alternatively to assign higher probabilities to some candidates than others, thereby considerably reducing recognition errors.

2.3.1. Language models for speech recognition

There is a large and active area of research in computational linguistics and natural language understanding that deals with language modelling. Chomsky's formal language theory [68] is widely used to specify the permissible word sequences in natural language processing. In the Chomsky hierarchy, the simplest language type is the finite-state grammar, which in fact can only generate the same set of sentences as those which have been pre-defined. A more powerful language type is the context-free grammar. For speech recognition, stochastic context-free languages [16] have also been proposed in the spirit of hidden Markov modelling. In comparison to conventional context-free parsing algorithms, such as the Earley algorithm [47] or the CYK algorithm [68], Ney [122] incorporated the DTW algorithm into the parsing algorithm. A more efficient LR parsing algorithm [161] is also adopted based on HMMs [88].

On the other hand, stochastic grammars, such as trigram or bigram [83], assign an estimated probability to any word that can follow a given word. Such a modelling approach can contain both syntactic and semantic information, but these probabilities must be trained from a large corpus. In a similar manner, word pair grammars specify the list of words that can legally follow any given word with uniform probabilities [31,96]. In general, $Pr(\mathbf{W})$ can be decomposed as

$$Pr(\mathbf{W}) = \prod_{i=1}^{n} Pr(w_i | w_1, w_2, ..., w_{i-1}) \tag{2.3.4}$$

where $Pr(w_i | w_1, w_2, ..., w_{i-1})$ is the probability that w_i will be spoken given that word sequences $w_1, w_2, ..., w_{i-1}$ were said previously; the choice of w_i thus depends on the entire past history of the input. For a vocabulary of size V there will be V^{i-1} different histories, and so to specify $Pr(w_i | w_1, ..., w_{i-1})$ completely, V^i values would have to be estimated. In reality, the probabilities $Pr(w_i | w_1, ..., w_{i-1})$ would be impossible to estimate for even moderate values of i, since most histories $w_1, \cdots w_{i-1}$ would be unique or would have only occurred only a few times. A practical solution for the above problems is to assume that $Pr(w_i | w_1, ..., w_{i-1})$ only depends on $w_{i-M+1}, \cdots w_{i-1}$. This leads to an M-gram language model, such as unigram, bigram, or trigram language models [82]. This is because most of the word strings will never occur in the language if $M > 3$ for all practical purposes. Therefore, in a trigram model, the

probability of a word depends on the two preceding words. If the training corpus is not large enough, many actually existing word successions will not be well observed leading to many extremely small probabilities. To treat the insufficient data problem, smoothing methods, such as the Turing-Good estimate [118], deleted interpolation of trigram, bigram and unigram models [41,81], as well as neural-network-based NETgram [120], can be well applied.

Because of the diversity of research activities, it is impossible to give here a comprehensive picture of language modelling, most of which may be found in [1,32,34,46,67,155,160,170].

2.3.2. Complexity measures of language

Language can be thought of as an information source whose outputs are words w_i. The amount of information per word, i.e. *entropy* (H), in some corpus can then be approximately estimated by [83]

$$H = -\frac{1}{n}\log Pr(w_1,w_2,...,w_n) \tag{2.3.5}$$

where n is the size of the corpus [82]. To estimate H, it is necessary to know the actual probabilities $Pr(w_1,w_2,...,w_n)$ of strings of the language. These are in practice ultimately incalculable, and estimates $\hat{Pr}(w_1,...,w_n)$ are used instead. To measure the difficulty of a recognition task relative to a given language model, *perplexity* [82,157] is defined by

$$\begin{aligned} PP &= 2^{-\frac{1}{n}\log \hat{Pr}(w_1,w_2,...,w_n)} \\ &= \hat{Pr}(w_1,w_2,...,w_n)^{-1/n} \end{aligned} \tag{2.3.6}$$

Approximately, PP is a measure of the average *branching* factor of the text when presented to the language model. Therefore, in the task of continuous digit recognition, the perplexity is 10. In tasks of 5,000 word continuous speech recognition, the test set perplexity of the trigram grammar and the bigram grammar is reported to be about 128 and 176 respectively [82]. In tasks of 1,000 word continuous speech recognition, the test set perplexity of the word pair grammar and the bigram grammar is reported to be about 60 and 20 respectively [96].

As perplexity does not take any account of the acoustic model, if the main concern is the contribution of the language model to the acoustic pattern matching, other measures such as speech decoder entropy [51] can be used, though it is more expensive to compute than perplexity.

2.4. Summary

This Chapter has reviewed several basic techniques used in speech recognition. The central issue in speech recognition research is acoustic pattern matching, which has a close relation with signal processing and language modelling. Selection of signal processing methods largely depends on the subsequent distortion measures or continuous probability density functions. Cepstral analysis is widely used, partly because of its low correlation property. Language modelling helps acoustic pattern matching because it can be used to impose constraints for acoustic pattern search space. In acoustic pattern matching, DTW, HMMs, and neural networks are discussed. DTW can be considered as a simplified case of hidden Markov modelling when the Viterbi algorithm is used for decoding. More recently, neural networks have received considerable focuses, but these are difficult to apply to continuous speech recognition. It is currently techniques of hidden Markov modelling that offer state-of-the-art speech recognition. In fact, similar criteria to those used in neural networks are being developed for hidden Markov modelling [12,21,43,97]. Although the discussion here was organised through signal processing, acoustic pattern matching, and language modelling, it should be noted that acoustic pattern matching and language modelling are usually combined in the same computational framework in practical HMM-based speech recognition system design.

CHAPTER 3

VECTOR QUANTISATION AND MIXTURE DENSITIES

Quantisation, the process of approximating continuous amplitude signals by discrete signals, is an important aspect of data compression or coding, the field concerned with the reduction of the number of bits necessary to transmit or store analogue data, subject to a distortion or fidelity criterion. The independent quantisation of each signal value or parameter is termed scalar quantisation. In contrast, the joint quantisation of a block of parameters is termed vector quantisation (VQ).

The representation of the vector quantisation codeword in sample space can be the centroid of the corresponding cell as in conventional vector quantisation, or can be calculated as the probability density function for the corresponding cell. This latter approach involves computation of maximum-likelihood estimates when the observation can be viewed as incomplete data. As a data compression technique, vector quantisation has been successfully used in speech coding, image coding, and speech recognition [62,107]. In HMM-based speech recognition, vector quantisation serves an important role in describing discrete acoustic prototypes of speech signals for the discrete HMM.

This Chapter will first review the principles of conventional vector quantisation and several standard algorithms used for discrete HMMs, and then discuss maximum-likelihood estimates of mixture densities with the EM algorithm for improved performance of hidden Markov modelling. These pave the way for the unified modelling theory developed in subsequent Chapters.

3.1. Conventional Vector Quantisation

Assume that $\mathbf{x}= (x_1,x_2,...x_d)^t \in R^d$ is a d-dimensional vector whose components $\{x_k, 1 \le k \le d\}$ are real-valued, continuous-amplitude random variables. In vector quantisation, the vector $\mathbf{x}$ is mapped to another real-valued, discrete-amplitude d-dimensional vector $\mathbf{y}$. It is then said that $\mathbf{x}$ is quantised to $\mathbf{y}$.

$$\mathbf{y} = q(\mathbf{x}) \tag{3.1.1}$$

In (3.1.1) q() is the quantisation operator. Typically, $\mathbf{y}$ takes one of a finite set of values $\mathbf{Y}=\{\mathbf{y}_i, 1 \le i \le L\}$, where $\mathbf{y}_i=[y_1, y_2,...y_d]$. The set $\mathbf{Y}$ is referred to as the codebook, L is the size of the codebook, and $\{\mathbf{y}_i\}$ are the set of codewords. The size L of the codebook is also called the number of levels in the codebook.

To design a codebook, the d-dimensional space of the original random vector $\mathbf{x}$ can be partitioned into L regions or cells $\{C_i, 1 \le i \le L\}$ and associated with each cell C_i is a vector $\mathbf{y}_i$. The quantiser then assigns the codeword $\mathbf{y}_i$ if $\mathbf{x}$ lies in C_i

$$q(\mathbf{x}) = \mathbf{y}_i, \text{ if } \mathbf{x} \in C_i \tag{3.1.2}$$

This codebook design process is also known as *training* the codebook. An example of a partitioning of two dimensional space ($d=2$) for the purpose of vector quantisation is shown in Figure 3.1.1. The shaded region enclosed by the bold lines is the cell C_i. Any input vector $\mathbf{x}$ that lies in the cell C_i is quantised as $\mathbf{y}_i$. The shapes of the various cells can be different, and the positions of the codewords corresponding to the cells are determined by minimising the distortion measure associated with the corresponding cells. The positions of the codewords within each cell are shown by dots in Figure 3.1.1.

When $\mathbf{x}$ is quantised as $\mathbf{y}$, a quantisation error results and a distortion measure $d(\mathbf{x},\mathbf{y})$ can be defined between $\mathbf{x}$ and $\mathbf{y}$ to measure the quantisation quality. The distortion measure between $\mathbf{x}$ and $\mathbf{y}$ is also known as a distance measure in the speech recognition context. The measure must be tractable in order to be computed and analysed, and also must be subjectively relevant so that differences in distortion values can be used to indicate differences in speech signals. Most distortion measures discussed in Chapter 2 can be used here as distortion measures for vector quantisation. A number of perceptually based distortion measures, and others that correlate well with subjective

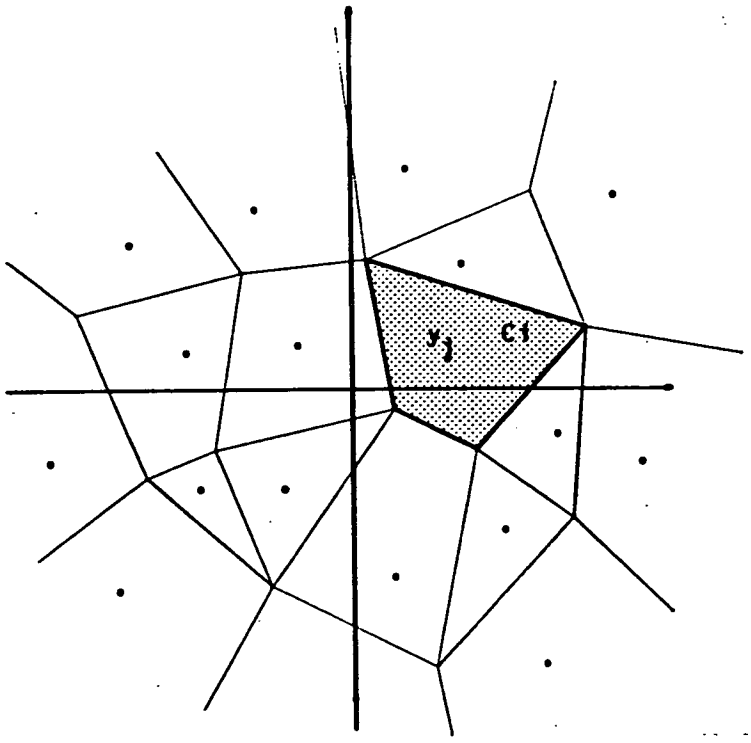


Figure 3.1.1. Partitioning of two-dimensional space

judgements have also been used in speech coding [6,125]. However, the most commonly used measure is the Euclidean distortion measure which assumes that the distortions contributed by quantising the different parameters are equal. In general, unequal weights can be introduced to render certain contributions to the distortion more important than others. One choice for weights that is popular in many practical applications is to use the inverse of the covariance matrix of $\mathbf{y}$.

$$d(\mathbf{x},\mathbf{y}) = (\mathbf{x}-\mathbf{y})^t \Sigma^{-1} (\mathbf{x}-\mathbf{y}) \tag{3.1.3}$$

This distortion measure, known as Mahalanobis distance, is actually a simplified Gaussian density representation as discussed in Chapter 2.

To design an L-level codebook, it is required to partition d-dimensional space into L cells and associate with each cell a quantised vector. One criterion for optimisation of the vector quantiser is to let the overall average distortion be minimised over all L-levels of the quantiser. The overall average distortion can be defined by

$$\begin{aligned} D &= E[d(\mathbf{x},\mathbf{y})] \\ &= \sum_{i=1}^{L} Pr(\mathbf{x} \in C_i)\, E[d(\mathbf{x},\mathbf{y}_i)|\mathbf{x} \in C_i] \\ &= \sum_{i=1}^{L} Pr(\mathbf{x} \in C_i) \int_{x \in C_i} d(\mathbf{x},\mathbf{y}_i) f(\mathbf{x}) d\mathbf{x} \end{aligned} \tag{3.1.4}$$

where $Pr(\mathbf{x} \in C_i)$ is the discrete probability that $\mathbf{x}$ is in C_i, $f(\mathbf{x})$ is the multidimensional probability density function of $\mathbf{x}$, $E[.]$ denotes the expectation, and the integral is taken over all components of the vector $\mathbf{x}$.

3.1.1. Codebook design

As mentioned above, a quantiser is optimal if Eq. (3.1.4) is minimised over all L-levels of the quantiser. There are two necessary conditions for optimality. The first condition is that the optimal quantiser is realised by using a minimum distortion or nearest neighbour selection rule

$$q(\mathbf{x}) = \mathbf{y}_i, \text{ if and } only \text{ if } d(\mathbf{x},\mathbf{y}_i) \leq d(\mathbf{x},\mathbf{y}_j),\ j \neq i,\ 1 \leq j \leq L \tag{3.1.5}$$

This means that the quantiser must choose the codeword that results in the minimum distortion with respect to $\mathbf{x}$, i.e. $\mathbf{x}$ are selected for the corresponding cell C_i. The second condition for optimality is that each codeword $\mathbf{y}_i$ is chosen to minimise the average distortion in cell C_i. That is, $\mathbf{y}_i$ is that vector $\mathbf{y}$ which minimises

$$\begin{aligned} D_i &= E[d(\mathbf{x},\mathbf{y})|\mathbf{x} \in C_i] \\ &= \int_{\mathbf{x} \in C_i} d(\mathbf{x},\mathbf{y}) f(\mathbf{x}) d\mathbf{x} \end{aligned} \tag{3.1.6}$$

Such a vector is called the centroid of the cell C_i, and is written

$$\mathbf{y}_i = cent(C_i) \tag{3.1.7}$$

Computing the centroid for a particular region (cell) will depend on the definition of the distortion measure [56]. In practice, given a set of training vectors $\{\mathbf{x}_k,\ 1 \leq k \leq T\}$, a subset of K_i vectors will be located in cell C_i. The average distortion D_i in cell C_i can

then be given by

$$D_i = \frac{1}{K_i} \sum_{\mathbf{x} \in C_i} d(\mathbf{x},\mathbf{y}_i) \tag{3.1.8}$$

The overall average distortion measure can be then computed according to:

$$D = \sum_{i=1}^{L} D_i / T \tag{3.1.9}$$

Given the distortion measure of cluster C_i as Eq. (3.1.8), and $d(\mathbf{x},\mathbf{y})$ as Eq. (3.1.3), the minimisation of D_i with respect to $\mathbf{y}_i$ is given by

$$\mathbf{y}_i = \frac{1}{K_i} \sum_{\mathbf{x} \in C_i} \mathbf{x} \tag{3.1.11}$$

$\mathbf{y}_i$ is simply the sample mean of all the training vectors, $\mathbf{x}$, contained in cluster C_i. This works well with a large class of Euclidean-like distortion measures [107].

One method for codebook design is to iteratively minimise the average distortion measure by employing a clustering algorithm. The most widely used clustering algorithm is the k-means algorithm [2,45,52], in which the basic idea is to divide the set of training vectors into L clusters C_i $\{1 \le i \le L\}$ in such a way that the two necessary conditions for optimality described above are satisfied. The k-means algorithm can be described as follows:

$k-means$ algorithm

Step 1: *Initialisation*:

Choose some adequate method [45] to derive an initial VQ codebook. ($\mathbf{y}_i$, $1 \le i \le L$)

Step 2: *Classification*:

Classify each element of training vectors $\{\mathbf{x}_k\}$ into one of the clusters C_i by the nearest neighbour rule. ($\mathbf{x} \in C_i$, iff $d(\mathbf{x},\mathbf{y}_i) \le d(\mathbf{x},\mathbf{y}_j)$ for all $j \neq i$)

Step 3: *Codebook Updating*:

Update the codeword of every cluster by computing the centroid of the training vectors in each cluster. ($\mathbf{y}_i$ = cent(C_i), $1 \le i \le L$)

Step 4: *Termination*:

If the decrease in the overall distortion D at the current iteration relative to the overall distortion at the previous iteration is below a chosen threshold, stop; otherwise go to Step 2.

In the process of minimising the average distortion measure, the k-means procedure actually breaks the minimisation process into two steps. Assuming that the centroid $\mathbf{y}_i$ (or mean) for each cluster C_i has been found, then the minimisation process is found simply by partitioning of all the vectors into their corresponding closest cluster according to the distortion measure D_i. On the other hand if all of the partitions are obtained, the minimisation process involves finding the new centroid within each cluster to minimise its corresponding within-cluster distortion D_i. By iterating over these two steps, a new value of distortion measure D_i which is smaller than that of the previous step can be obtained. However, the k-means algorithm can only converge to a local optimum [2,101]. Furthermore, any such solution is, in general, not unique [61]. Global optimality may be approximated by repeating the k-means algorithm for several sets of codebook initialisation values and then choosing the codebook that produces the minimum overall distortion, although such a criterion may not necessarily lead to optimal speech recognition accuracy [75].

Another commonly used algorithm is the LBG algorithm proposed by Linde, Buzo, and Gray [101]. The LBG algorithm is an extended k-means algorithm which iteratively splits the training data into 2, 4, ..., 2^m partitions, with a centroid for each partition. The centroid is determined by iterative refinement as for k-means clustering. The algorithm can be described as follows:

LBG algorithm

Step 1: *Initialisation*:

Set L (number of partitions or clusters) $=1$. Find centroid of all the training

frames.

Step 2: *Splitting*:

Split L into $2L$ partitions. Set $L = 2L$.

Step 3: *Classification*:

Classify the set of training data $\{\mathbf{x}_k\}$ into one of the clusters C_i according to the nearest neighbour rule.

Step 4: *Codebook Updating*:

Update the codeword of every cluster by computing the centroid in each cluster.

Step 5 *Termination* 1:

If the decrease in the overall distortion D at each iteration relative to the value D at the previous iteration is below a selected threshold, goto **Step** 6; otherwise go to **Step** 3.

Step 6: *Termination* 2:

If L equals the VQ codebook size required, **Stop**; otherwise go to **Step** 2.

Step 3 and Step 4 are the same as for the k-means clustering algorithm. Various heuristic methods can be adopted in the splitting step to find two vectors that are far apart in each partition. In practice, both the k-means and the LBG algorithm work well in producing a VQ codebook according to the given distortion measure.

3.2. Modelling the VQ Codebook as Mixture Densities

In HMM-based speech recognition, the goal of VQ is to generate a number of acoustic prototype vectors (VQ codewords) from a large sample of training vectors such

that the codewords can represent the distribution of the training vectors; and minimise the total distortion over all training vectors. The VQ partitions the acoustic space into separate regions according to some distortion measure regardless of the probability distributions of the original data. This introduces errors as the partition operations may destroy the original signal structure. As an alternative, the VQ codebook can be modelled as a family of finite mixture Gaussian density functions such that each cell will be represented as a probability density function as shown in Figure 3.2.1, where dotted lines show conventional VQ partitions. These probability density functions can be overlapped, rather than partitioned, to represent speech parameter space. The centroid obtained via such a representation may be quite different from that obtained

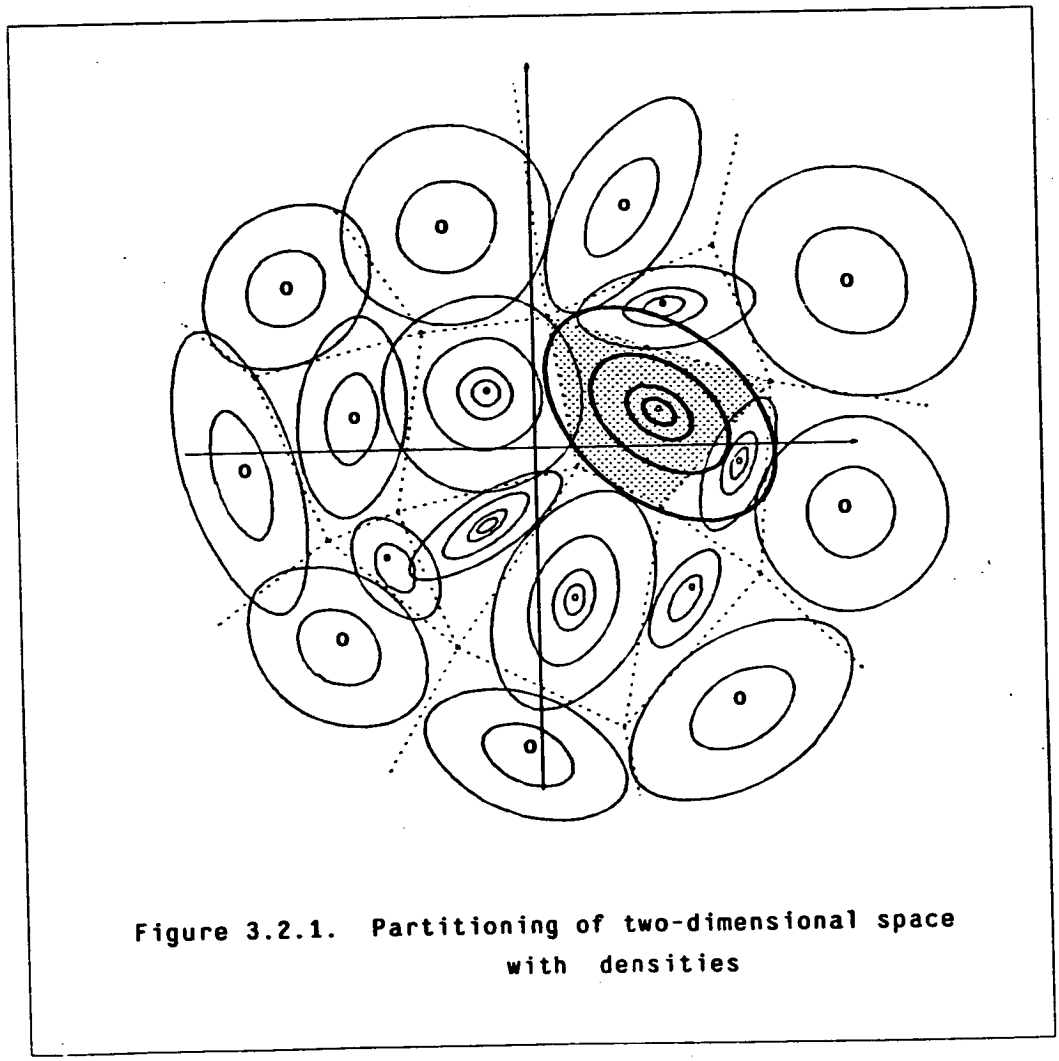


Figure 3.2.1. Partitioning of two-dimensional space with densities

using the conventional k-means algorithm since the distribution property of signals can be taken into consideration. Another advantage is that the use of a parametric family of finite mixture densities within the VQ operations can be closely combined with the HMM methodology leading to the unified modelling framework of this thesis.

Problems of estimating the parameters which determine a mixture density have been the subject of a large, diverse body of literature [141]. The most distinctive solution to the problem is the EM algorithm [38]. This technique has in fact been defined in an earlier publication by Baum [19] and has been widely used in HMM-based speech recognition.

3.2.1. Estimation of mixture densities

A parametric family of finite mixture densities, i.e. a family of probability density functions can be written as

$$f(\mathbf{x}|\Phi) = \sum_{i=1}^{L} p_i f_i(\mathbf{x}|\varphi_i), \quad \mathbf{x} \in R^d \tag{3.2.1}$$

where each p_i is the probability of being distribution i, and where $f_i(\mathbf{x}|\varphi_i)$ is itself a density function of the ith prototype parametrised by φ_i; and we denote $\Phi = (p_1,\cdots,p_L,\varphi_1,\cdots,\varphi_L)$. Finite mixture densities arise naturally, and can be interpreted as densities associated with a statistical population which is a mixture of L component populations with component densities $\{f_i\}$ and mixing proportions $\{p_i\}$ (weighting coefficients). Such densities appear as fundamental models in areas of applied statistics such as statistical pattern recognition [45]. In Eq. (3.2.1), a sample observation on the mixture is termed to be *labeled* if its component population of origin is known with certainty; otherwise, it is *unlabeled*.

A maximum likelihood estimate associated with an observation sample is a choice of parameters which maximises the probability density function of the sample, known as the *likelihood function*. It is assumed that a parametric family of mixture densities of the form Eq. (3.2.1) is specified and that a particular $\overline{\Phi}$ is the *true* parameter value to be estimated. Samples can be thought of as independent samples of unlabeled observations on the mixture (although samples need not be so represented [141]). The

likelihood function of an observation sample is the probability density function of the random sample evaluated by the observations at hand. It is usually convenient to deal with the logarithm of the likelihood function, called the *log−likelihood function*, rather than with the likelihood function itself. The log-likelihood function can be represented as

$$\begin{aligned} L(\Phi) &= \sum_{k=1}^{T} \log f(\mathbf{x}_k|\Phi) \\ &= \sum_{k=1}^{T} \log(\sum_{i=1}^{L} p_i f_i(\mathbf{x}_k|\varphi_i)) \end{aligned} \tag{3.2.2}$$

for samples $\{\mathbf{x}_k\}$ $(1 \leq k \leq T)$. If X is a sample of observations under consideration, then a maximum likelihood estimate of $\bar{\Phi}$ provides any choice of Φ in Ω at which the log-likelihood function of X attains its largest local maximum in Ω. Here Ω denote a set of parameters of finite mixture densities in Eq. (3.2.1). In practice, it is difficult to take Ω to be a set in which the log-likelihood function is bounded. Also, mixture problems are very often such that the log-likelihood function attains its largest local maximum at several different choices of Φ.

The traditional general approach to determining a maximum-likelihood estimate is first to arrive at a system of likelihood equations satisfied by the maximum-likelihood estimate and then to try to obtain a maximum-likelihood estimate by solving the likelihood equations. Basically, the likelihood equations are found by considering the partial derivatives of the log-likelihood function with respect to the components of Φ. If $\bar{\Phi}$ is a maximum-likelihood estimate, it will satisfy

$$\nabla_{\varphi_i} L(\bar{\Phi}) = 0, \quad i = 1, 2, \cdots L, \tag{3.2.3}$$

determined by the unconstrained parameters φ_i. For mixture density problems, the likelihood equations are almost certain to be nonlinear and beyond analytic solution. Consequently, an approximate solution via some iterative procedure is often used instead. There are many general iterative procedures which are suitable for finding an approximate solution of the likelihood equations, such as Newton's method, various quasi-Newton methods, and conjugate gradient methods [57]. A special iterative method, known as the EM algorithm (E for *expectation* and M for *maximisation*), has been applied to a wide variety of mixture problems [38], and has been found in most instances to have the advantages of reliable global convergence and low computational

complexity.

3.2.2. The EM algorithm

The EM algorithm has been derived and studied by a number of researchers [37,65,169]. It arises naturally from the particular forms taken by the partial derivatives of the log-likelihood function. A quite different point of view towards the algorithm was presented by Dempster and *et al.* [38], where they interpreted the mixture density estimation problem as an estimation problem involving incomplete data by regarding an unlabeled observation on the mixture as *missing* a label indicating its component population of origin. In doing so, they not only related the mixture density problem to a broader class of statistical problems but also showed that the EM algorithm for mixture density problems is really a specialisation of a more general algorithm for approximating maximum-likelihood estimates from incomplete data. Earlier, the same algorithm was actually defined independently in a different task to solve the problem of hidden Markov models by Baum *et al* [19].

To understand the EM algorithm, let us first consider a simple example with two univariate Gaussian densities.

Example 3.2.1. Consider observations generated randomly from two Gaussian densities, *class* 1 and *class* 2. Let p_1 and p_2 denote the probability of choosing *class* 1 and *class* 2 respectively. The probability for a given observation x is:

$$f(x|\Phi) = p_1 N(x, \mu_1, \sigma_1) + p_2 N(x, \mu_2, \sigma_2)$$

where $\Phi = (p_1, p_2, \mu_1, \mu_2, \sigma_1, \sigma_2)$. Given some observations, we want to estimate parameter Φ. If we know that when *class* 1 and *class* 2 are used in generating the observations, we could use maximum likelihood estimation for each of the Gaussian densities, which is just the sample mean and covariance. The problem is that we do not know which Gaussian density is used in generating the observations. This information is *hidden*. An intuitive solution is to assume some parameter Φ, and compute when and how often we expect each distribution is used, as conditioned by the given data. These expected statistics can then be used to compute new estimates of the parameters. This

procedure can be started with new estimates and can be iterated. Given observation x, the posterior probability that it belongs to *class* 1 is:

$$Pr(class\ 1|x,\Phi) = \frac{Pr(class\ 1,x|\Phi)}{Pr(class\ 1,x|\Phi) + Pr(class\ 2,x|\Phi)} = \frac{p_1 N(x,\mu_1,\sigma_1)}{p_1 N(x,\mu_1,\sigma_1) + p_2 N(x,\mu_2,\sigma_2)}$$

The expected number of times *class* 1 and *class* 2 are used can be written as:

$$\gamma_1 = \sum_i Pr(class\ 1|x_i,\Phi)$$

$$\gamma_2 = \sum_i Pr(class\ 2|x_i,\Phi)$$

The maximum likelihood estimates for p_1 and p_2 shall be:

$$\bar{p}_1 = \frac{\gamma_1}{\gamma_1+\gamma_2}$$

$$\bar{p}_2 = \frac{\gamma_2}{\gamma_1+\gamma_2}$$

The expected values of the first moment for *class* 1 is

$$\gamma_1^{\mu} = \sum_i x_i Pr(class\ 1|x_i,\Phi),$$

and the maximum likelihood estimate is

$$\bar{\mu}_1 = \frac{\gamma_1^{\mu}}{\gamma_1}$$

The results for μ_2 and σ are similar. These reestimated parameters can be used iteratively to improve the likelihood.

The iterative solution in Example 3.2.1. is well founded. In general, suppose that a measure space Y of *complete data* exists together with a measurable map $\mathbf{y}\rightarrow\mathbf{x}(\mathbf{y})$ of Y to a measure space of X of *incomplete data*. Here, incomplete data is easy to observe and measure, while complete data is hidden and unobservable. Let $f(\mathbf{y}|\Phi)$ be a member of a parametric family of probability density functions defined on Y for Φ, and suppose that $f(\mathbf{x}|\Phi)$ is a probability density function on X induced by $f(\mathbf{y}|\Phi)$. For a given $\mathbf{x}\in X$, the purpose of the EM algorithm is to maximise the incomplete data log-likelihood $L(\Phi) = \log f(\mathbf{x}|\Phi)$ over Φ by exploring the relationship between $f(\mathbf{y}|\Phi)$ and $f(\mathbf{x}|\Phi)$ because the maximisation of the complete data log-likelihood $\log f(\mathbf{y}|\Phi)$ over Φ is

particularly easy.

For $\mathbf{x} \in X$, set $Y(\mathbf{x}) = \{\mathbf{y} \in Y \mid \mathbf{x}(\mathbf{y}) = \mathbf{x}\}$. The conditional density $f(\mathbf{y}|\mathbf{x},\Phi)$ on $Y(\mathbf{x})$ is given by $f(\mathbf{y}|\Phi) = f(\mathbf{y}|\mathbf{x},\Phi)f(\mathbf{x}|\Phi)$. For $\bar{\Phi}$ and Φ in Ω, then

$$L(\bar{\Phi}) = Q(\bar{\Phi}|\Phi) - H(\bar{\Phi}|\Phi). \tag{3.2.4}$$

where

$$Q(\bar{\Phi}|\Phi) = E(\log f(\mathbf{y}|\bar{\Phi})|\mathbf{x},\Phi)_{\mathbf{y}}$$

and

$$H(\bar{\Phi}|\Phi) = E(\log f(\mathbf{y}|\mathbf{x},\bar{\Phi})|\mathbf{x},\Phi)_{\mathbf{y}}.$$

The general EM algorithm can be described as: Given a current Φ that is a maximiser of $L(\Phi)$, obtain a next approximation $\bar{\Phi}$ as follows:

1. Choose an initial estimate Φ.
2. E-step. Compute $Q(\bar{\Phi}|\Phi)$ based on the given Φ
3. M-step. Choose $\bar{\Phi} \in \underset{\Phi \in \Omega}{argmax} Q(\bar{\Phi}|\Phi)$. Here, $\underset{\Phi \in \Omega}{argmax} Q(\bar{\Phi}|\Phi)$ denotes the set of values $\bar{\Phi}$ which maximise $Q(\bar{\Phi}|\Phi)$ over Ω.
4. Set $\Phi = \bar{\Phi}$, goto 2 until convergence.

The EM algorithm is used in applications which permit the easy maximisation of $\log f(\mathbf{y}|\Phi)$ over $\Phi \in \Omega$ instead of maximising $L(\Phi)$ directly. In such applications, the M-step maximisation of $Q(\bar{\Phi}|\Phi)$ over $\bar{\Phi} \in \Omega$ is usually carried out with corresponding ease. The basis of the EM algorithm lies in the fact that if $Q(\bar{\Phi}|\Phi) \geq Q(\Phi|\Phi)$, $L(\bar{\Phi}) \geq L(\Phi)$, since it follows from Jensen's inequality that $H(\bar{\Phi}|\Phi) \leq H(\Phi|\Phi)$ [38]. This fact implies that L increases monotonically on any iteration sequence generated by the EM algorithm via maximisation of the Q-function.

To discuss the EM algorithm for mixture density estimation problems, we assume as in the preceding Section that a parametric family of mixture densities of the form of Eq. (3.2.1) is specified and that a particular $\bar{\Phi}$ is the *true* parameter value to be estimated. This family of densities is regarded as being associated with a statistical

population which is a mixture of L component populations. The EM algorithm for a mixture density estimation problem associated with this family is derived by first interpreting the problem as one involving incomplete data and then obtaining the algorithm from its general formulation given above. The problem is interpreted as one involving incomplete data by regarding each unlabeled observation in the sample at hand as *missing* a label indicating its component population of origin.

It can be assumed that the sample under consideration is independent of unlabeled observations. One can regard the sample by considering each $\mathbf{x}_k$ to be the *known* part of an observation $\mathbf{y}_k=(\mathbf{x}_k,i_k)$, where i_k is an integer between 1 and L indicating the component population of origin. For $\Phi\in\Omega$, the sample variables $\mathbf{x}=(\mathbf{x}_1,\ldots,\mathbf{x}_T)$ and $\mathbf{y}=(\mathbf{y}_1,\ldots,\mathbf{y}_T)$ have associated probability density functions $f(\mathbf{x}|\Phi)=\prod_{k=1}^{T} f(\mathbf{x}_k|\Phi)$ and $f(\mathbf{y}|\Phi)=\prod_{k=1}^{T} p_{i_k} f_{i_k}(\mathbf{x}_k|\varphi_{i_k})$, respectively. Then for $\Phi\in\Omega$, the conditional density $f(\mathbf{y}|\mathbf{x},\Phi)$ is given by

$$f(\mathbf{y}|\mathbf{x},\Phi)=\prod_{k=1}^{T}\frac{p_{i_k} f_{i_k}(\mathbf{x}_k|\varphi_{i_k})}{f(\mathbf{x}_k|\Phi)} \tag{3.2.5}$$

Note that $\sum_{i_k=1}^{L} p_{i_k} f_{i_k}(\mathbf{x}_k|\varphi_{i_k}) = f(\mathbf{x}_k|\Phi)$; the function $Q(\bar{\Phi}|\Phi)$ can be determined to be

$$\begin{aligned} Q(\bar{\Phi}|\Phi) &= \sum_{i_1=1}^{L}\cdots\sum_{i_T=1}^{L}\sum_{k=1}^{T}\log\bar{p}_{i_k} f_{i_k}(\mathbf{x}_k|\bar{\varphi}_{i_k})\prod_{k=1}^{T}\frac{p_{i_k} f_{i_k}(\mathbf{x}_k|\varphi_{i_k})}{f(\mathbf{x}_k|\Phi)} \\ &= \sum_{i=1}^{L}\Big[\sum_{k=1}^{T}\frac{p_i f_i(\mathbf{x}_k|\varphi_i)}{f(\mathbf{x}_k|\Phi)}\Big]\log\bar{p}_i + \sum_{i=1}^{L}\sum_{k=1}^{T}\log f_i(\mathbf{x}_k|\bar{\varphi}_i)\frac{p_i f_i(\mathbf{x}_k|\varphi_i)}{f(\mathbf{x}_k|\Phi)} \end{aligned} \tag{3.2.6}$$

Having determined an appropriate function $Q(\bar{\Phi}|\Phi)$ for the E-step of the EM algorithm, the maximisation in the M-step can be separated into two maximisation problems, the first involving the proportions $\bar{p}_1,\ldots,\bar{p}_L$ alone and the second involving only the remaining parameters of $f_i(\mathbf{x}|\bar{\varphi}_i)$, namely, $\bar{\varphi}_1,\ldots,\bar{\varphi}_L$. Since $\log\bar{p}_i$ appears linearly in $Q(\bar{\Phi}|\Phi)$, the first solution is easily determined regardless of the functional forms of the component densities $f_i(\mathbf{x}|\bar{\varphi}_i)$. If φ_i are mutually independent, the second maximisation problem is separated into L component problems, each of which involves only one of the parameters φ_i. Before finding a solution to Eq. (3.2.6), we have the

following Theorem:

Theorem 3.2.1.

If $c_i>0$, $i=1,2,...C$, and subject to the constraint $\sum_{i=1}^{C} x_i = 1$, then the function

$$Q(x) = \sum_{i=1}^{C} c_i \log x_i$$

attains its unique global maximum when

$$x_i = \frac{c_i}{\sum_{i=1}^{C} c_i} \tag{3.2.7}$$

The proof for the above comes from the Lagrange method. We have:

$$\frac{\partial}{\partial x_i}[Q(\mathbf{x}) - \lambda \sum_i x_i] = \frac{c_i}{x_i} - \lambda = 0$$

Multiplying by x_i and summing over i gives $\lambda = \sum_i c_i$, hence the result.

Assume that Φ is a current approximate maximiser of the log-likelihood function $L(\Phi)$ given by Eq. (3.2.2); from Theorem 3.2.1, it can be seen that the next approximate maximiser $\bar{p}$ prescribed by the M-step of the EM algorithm satisfies

$$\bar{p}_i = \frac{1}{T} \sum_{k=1}^{T} \frac{p_i f_i(\mathbf{x}_k|\varphi_i)}{f(\mathbf{x}_k|\Phi)}, \quad \text{for } i=1,2,...,L \tag{3.2.8}$$

For $\bar{\varphi}$, it depends on $f_i(\mathbf{x})$, and satisfies

$$\bar{\varphi}_i \in \underset{\bar{\varphi}_i \in \Omega_i}{argmax} \sum_{k=1}^{T} \log f_i(\mathbf{x}_k|\bar{\varphi}_i) \frac{p_i f_i(\mathbf{x}_k|\varphi_i)}{f(\mathbf{x}_k|\Phi)}, \quad \text{for } i=1,2,...,L \tag{3.2.9}$$

Note that each weight $p_i f_i(\mathbf{x}_k|\varphi_i)/f(\mathbf{x}_k|\Phi)$ is the posterior probability that $\mathbf{x}_k$ originated in the ith component population, given the current approximate maximum-likelihood estimate Φ.

Example 3.2.2. When $f_i(\mathbf{x}|\varphi_i)$ is a multivariate Gaussian density function, $\varphi_i = (\mu_i, \Sigma_i)$. For a given φ_i, the unique solution $\bar{\varphi}_i$ can be computed through maximising Eq. (3.2.9). (A solution for maximisation of Eq. (3.2.9) will be developed in Chapter 5 in the same form as the continuous HMM.) $\bar{\varphi}_i = (\bar{\mu}_i, \bar{\Sigma}_i)$ of Eq. (3.2.9) can be

given by

$$\bar{\mu}_i = \frac{\sum_{k=1}^{T} \mathbf{x}_k \frac{p_i f_i(\mathbf{x}_k|\varphi_i)}{f(\mathbf{x}_k|\Phi)}}{\sum_{k=1}^{T} \frac{p_i f_i(\mathbf{x}_k|\varphi_i)}{f(\mathbf{x}_k|\Phi)}} \tag{3.2.10}$$

$$\bar{\Sigma}_i = \frac{\sum_{k=1}^{T} (\mathbf{x}_k - \bar{\mu}_i)(\mathbf{x}_k - \bar{\mu}_i)^t \frac{p_i f_i(\mathbf{x}_k|\varphi_i)}{f(\mathbf{x}_k|\Phi)}}{\sum_{k=1}^{T} \frac{p_i f_i(\mathbf{x}_k|\varphi_i)}{f(\mathbf{x}_k|\Phi)}} \tag{3.2.11}$$

Here, the posterior probability $p_i f_i(\mathbf{x}|\varphi_i)/f_i(\mathbf{x}_k|\Phi)$ can be considered as $f(\mathbf{y} \in \varphi_i|\mathbf{x}_k,\Phi)$. The information as to whether a given observation $\mathbf{x}$ should belong to distribution i (the ith Gaussian density function) is *hidden*, and can only be seen via $\mathbf{y}$. The solution of the EM algorithm is to compute when and how often a given observation x will be expected to be in each distribution. These expected statistics can then be used to compute new estimates of new parameters $\bar{\Phi}$. No assumption is imposed on how each distribution should be organised with any other. As will be discussed later, the Markov properties can be imposed on these distributions such that the temporal information can be well modelled. In such cases, the EM algorithm will be the same as the Baum-Welch algorithm, which has been used in maximum-likelihood estimation of hidden Markov model parameters.

The maximum likelihood estimation problem is well-suited for entropy techniques since the measured data do not uniquely describe the underlying stochastic process. Indeed, the maximum-likelihood estimates obtained here are identical to the entropy maximisation technique via a Q-function [112]. When the Markov properties are imposed on the mixture distributions such that the temporal information can be well modelled, the EM algorithm will be the same as the Baum-Welch algorithm used for hidden Markov modelling. It can thus be expected that mutual optimisation of the vector quantisation codebook and hidden Markov model parameters is possible. The unified modelling will be discussed in Chapter 6.

3.3. Summary

This Chapter has discussed two prerequisites to hidden Markov modelling. Conventional VQ is an application of clustering techniques to produce prototypes (codewords) of observations. A given observation can then be classified into one of such prototypes. A finite set of prototypes (codeword) can be used to represent the continuous observation such that the discrete probability distributions can be used to model the given observations. Each codeword can be represented either by the centroid of observations in the corresponding cell or by the probability density function estimated from the corresponding cell. The latter assumption leads to solution of the EM algorithm, which has essentially the same underlying characteristics for hidden Markov modelling as the complete-incomplete data problem. Only incomplete data can be measured or observed, and the iterative algorithms must be used to guess or reestimate the complete data. As the same approach can be applied to both VQ and subsequent hidden Markov modelling, the unified modelling of these can be made possible on the assumption of mixture density representation of the VQ codebook.

CHAPTER 4

HIDDEN MARKOV MODELS AND BASIC ALGORITHMS

Hidden Markov modelling is a probabilistic technique for the study of time series, which permits modelling with many of the classical probability distributions. There has been a significant growth in the number of papers reporting applications of hidden Markov modelling recently. This Chapter will discuss the main tools in hidden Markov modelling of speech signals, i.e., the forward-backward algorithm, the Baum-Welch algorithm, and the Viterbi algorithm. The basic theory of HMMs will be exemplified with the discrete HMM, in which the output probabilities are discrete probability distributions and VQ is a prerequisite to convert the continuous speech signal into a finite set of prototypes.

4.1. Markov Processes

There is often significant structure embodied in a natural language. For example, in English, the letter Q is almost always followed by the letter U. Therefore, the probability of seeing the letter U depends very much on the letter that came before it. This is a situation that often arises in practice — the previous part of a message can often greatly influence subsequent events. Such a stochastic process can be described by a jth-order *Markov* process which enables description of some of the large probability structures that arise typically in languages. These can be summarised by the Markov property, i.e. for any sequence of time domain events the conditional probability density of a current event given all the past and present events depends only on the most recent j events. A process which satisfies the Markov property is called a Markov process.

As an example of a simple first-order Markov process, suppose that there are three symbols, a, b, c in the alphabet. Let the probability that symbol a is followed by any of the three symbols a, b, c be 1/3. Let the probability that the symbol b is followed by another b be 1/2, and either of the others symbol, a or c, be 1/4. Finally, let the probability that the symbol c is followed by a c be 1/2, and either of the other two, a or b, be 1/4. We have

$$\begin{aligned} Pr(a|a) &= \frac{1}{3}, \; Pr(b|a) = \frac{1}{3}, \; Pr(c|a) = \frac{1}{3} \\ Pr(a|b) &= \frac{1}{4}, \; Pr(b|b) = \frac{1}{2}, \; Pr(c|b) = \frac{1}{4} \\ Pr(a|c) &= \frac{1}{4}, \; Pr(b|c) = \frac{1}{4}, \; Pr(c|c) = \frac{1}{2} \end{aligned}$$

It is conventional to use a transition graph to illustrate a Markov process. There are, of course, three states in this case (for a, b, and c respectively) indicated by the circles as shown in Figure 4.1.1.

Each directed line is a transition from one state to another state, whose probability is indicated by the number alongside the line. For example, $Pr(a|b)$, is the directed line from state b to state a and has the probability of transition of 1/4. In this example each state has three lines *out* and three lines *in*. Such a Markov model may be used, for example, for weather forecasting. Let state a have output, saying *sunny*, state b have

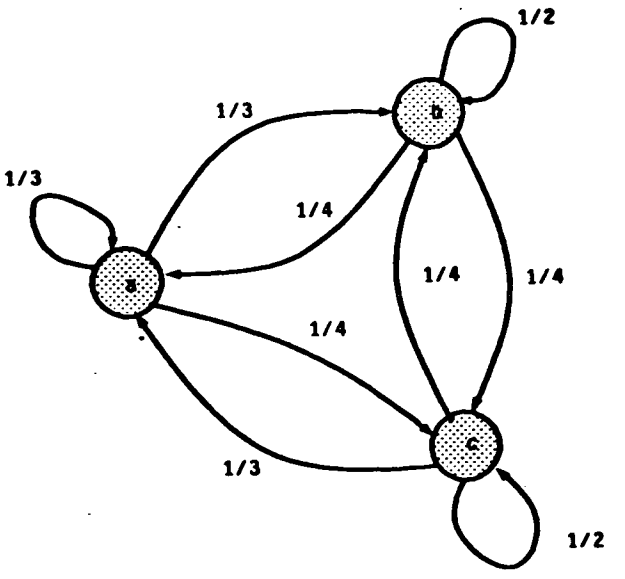


Figure 4.1.1. Example of a Markov process

output *cloudy*, and state *c* have output *rainy*. Given today is sunny, it is equally likely that tomorrow it will be any of the three states sunny, cloudy, or rainy. But if today is either cloudy or rainy it is a 50-50 chance that it will be the same tomorrow, and only one in four that it will be either of the other two. Thus, the stochastic process of weather forecasting can be roughly described by such a Markov process.

Assume states *a*, *b*, and *c* are labeled as 1, 2, and 3, then the transition graph shown in Figure 4.1.1. can be written in matrix form, where the element of the transition matrix a_{ij} denotes the transition probability from current state i to next state j.

$$A = \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/4 & 1/2 & 1/4 \\ 1/4 & 1/4 & 1/2 \end{bmatrix}$$

where in a transition matrix, the sum of the elements in any row must total exactly 1,

since the current state must necessarily go somewhere. Instead of having a definite state, we have a probability distribution for that state (π_1, π_2, π_3), where $\pi_1+\pi_2+\pi_3=1$. It can be that one $\pi_i=1$ and the other two are 0, which means that we have a definite state.

As defined, Markov models can be used for a study of observed symbols arranged in a discrete-time series. The state sequence is observed in such Markov models, for example, rainy-sunny-rainy. Nevertheless, such a model is too restrictive to be applicable to many problems of interest. In a *hidden* Markov model (HMM), the output for each state will be an output probability distribution instead of deterministic events. The output probabilities thus impose a veil between the state sequence and the observer of the time sequence, i.e. the state sequence will be *hidden* (not observable). This mechanism is more powerful in certain applications such as speech recognition. In the following Sections, we will discuss a theory to lift such a veil, i.e. the theory of hidden Markov models.

4.2. Definition of the Hidden Markov Model

Signal modelling based on HMMs can be considered as a technique that extends conventional stationary spectral analysis principles to the analysis of time-varying signals. Many real world processes often exhibit a sequentially changing behaviour; the properties of the process are usually held in steady states, except for minor fluctuations, for a certain period of time prior to changing to another set of properties. In general, there is no accurate procedure to detect every such short-time segment of observation. Of course, there are many processes that do not change synchronously with every analyzing segment. If it is assumed that these periods of steady state behaviour can be identified, and the temporal variations within each of these steady periods are, in a sense, statistical, then a statistical model may be used to model these well-behaved sections of a steady signal with some characterisation of how one such steady period evolves to the next. It has to be questioned how these steadily or distinctively behaving periods can be identified and represented, and how the sequentially evolving nature of these periods can be statistically modelled. It is the HMM that successfully treats these problems

under a probabilistic or statistical framework.

HMMs use a Markov process [109] to model the changing statistical characteristics that are only probabilistically manifested through actual observations. The state sequence is *hidden*, and can only be observed through another set of observable stochastic processes. Each hidden state of the model, or transition between states, which has a parallel theory as discussed here, is associated with a set of output probability distributions, which can be characterised by either discrete probability distributions or continuous probability density functions. The veil between hidden state sequences and the observable stochastic process is imposed by the output probabilities.

To understand the concept of the HMM, consider the following example illustrated as Figure 4.2.1. A person is performing an experiment behind a veil. There are $N=3$ urns containing a large number of coloured balls, and there are $L=5$ distinct colours of the balls. An initial urn is chosen, according to some random process. A coloured ball is then chosen from this urn at random. The result of the colour can be observed in front of the veil. After the colour of the ball is observed, the coloured ball is replaced in the same urn and a new urn is selected according to a random process associated with the current urn. The ball selection process from this new urn is repeated. This experiment generates a finite observation sequence of coloured balls. Only the sequence of coloured balls can be observed in front of the veil. This simple example already possesses properties associated with a HMM: it is a generative mechanism for creating observations and the mechanism is a stochastic process with a hidden component. In the process of generating the observed sequence of coloured balls, a hidden sequence of urns is also generated. The problem of interest is how to build a stochastic model according to the observed sequence of coloured balls to explain regulations on the experiment conducted behind the veil.

To describe the HMM formally, the following model notation can be used.

T = length of the observation sequence, $O_1 O_2 \cdots O_T$ (number of coloured balls observed in experiment)

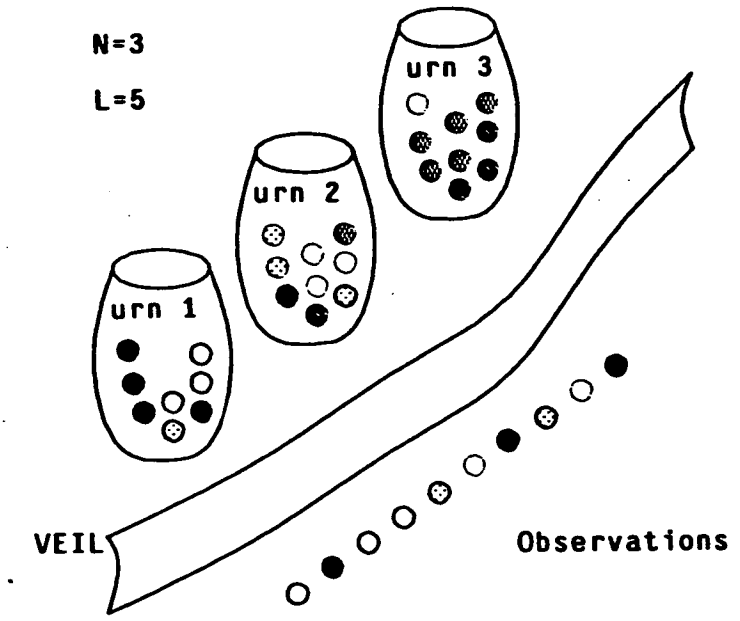


Figure 4.2.1. The urn and ball experiment

N = number of states in the model (number of urns)

L = number of observation symbols (number of colours in use)

S = $\{s\}$, a set of states (A state can be considered to possess some measurable, distinctive properties of events). For simplicity, state i at time t may be denoted by $s_t = i$ when ambiguity does not exist.

$v = \{ v_1, v_2, ...v_L\}$, a discrete set of possible symbol observations. O_t belongs to one such observation symbol.

$A = \{ a_{ij} | a_{ij} = Pr(s_{t+1} = j | s_t = i) \}$, state transition probability distribution, where a_{ij} denotes the transition probability from state i to state j.

$B = \{ b_j(O_t) \mid b_j(O_t) = Pr(O_t | s_t = j)\}$, For each state*, there is a corresponding output probability (discrete probability distributions in the discrete case and continuous probability density functions in the continuous case); and all of these output probabilities represent random variables or stochastic processes to be modelled. In the discrete HMM, it refers to the probability of generating some discrete symbol v_k in state j, which can be denoted simply by $b_j(k)$. In the continuous HMM, it denotes a probability density function for emission of observations O_t, where O_t is often denoted by $\mathbf{x}_t$. This difference between the discrete HMM and the continuous HMM leads to different reestimation algorithms for the model parameters.

$\pi = \{\pi_i | \pi_i = Pr(s_1 = i)\}$, initial state distribution.

An HMM can be represented by using the compact notation $\lambda = (A, B, \pi)$. Specification of an HMM involves choices of the number of states, N, and the number of discrete symbols L, and specification of three probability densities with matrix form A, B, and π. A set of initial states S_I and final states S_F can be defined. Thus, transitions must start from one of S_I and end at one of S_F. Let N_I and N_F denote the number of initial states and final states respectively. In practice, N_I and N_F are often chosen to be one.

The urn and ball experiment can be modelled by an HMM with the above definitions, where each state, i, corresponds to a specific urn, and output probabilities, $b_i(\mathbf{O})$, are defined for the coloured balls observable in each state, i.e. the probability distribution of coloured balls in each urn. The observation symbol, v_k, is the colour of the ball selected from the urns. The choice of urns is modelled by the initial state distribution and by the state transition probability distribution. Figure 4.2.2. shows an

* The state-dependent output probability is a special case of the transition-dependent one.

example of such an HMM. The transition probabilities are labeled in the figure; the output probabilities associated with each state are also illustrated. If the model is looked at generatively, the Markov chain synthesises a sequence of states (urns), and the output probability distributions then turn the sequence of states into a time series (observation sequence of coloured balls). The observed time series gives evidence about the hidden state sequence and the parameters of the generating model.

In a first-order HMM, there are two assumptions. The first is the *Markov assumption*, i.e. at each observation time, t, a new state is entered based on the transition probability, which only depends on the previous state. Note that the transition may allow the process to remain in the previous state. The second assumption is the *output−independence assumption*, i.e. the output probability depends only on the state at that time regardless of when and how the state is entered. Although these

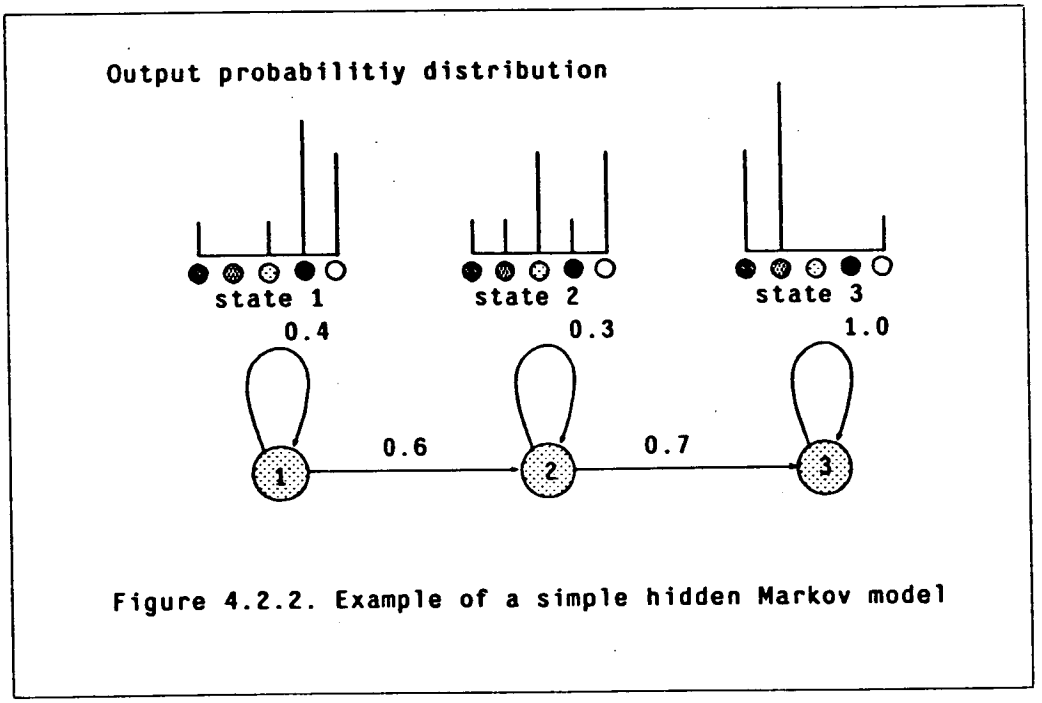


Figure 4.2.2. Example of a simple hidden Markov model

assumptions severely limit the memory of first-order HMMs, they reduce the number of *free* parameters, and also make learning and decoding algorithms extremely efficient. Efforts to explicitly model time correlation can be found in [26,167].

In statistical modelling, free parameters refer to those that will be estimated from observations, which represents one of the most important factors in statistical system design. Even though inadequate assumptions are made, it may provide better performance if such assumptions can drastically reduce the number of free parameters. Some insight into this problem can be gained from considering an analogous problem in curve fitting. Suppose we have several available data points and several candidate curves for fitting them and these data points are obtained by adding zero-mean, independent noise from a parabola. Of all the possible polynomials, a parabola should give the best fit, assuming that we are interested in fitting unknown data obtained from

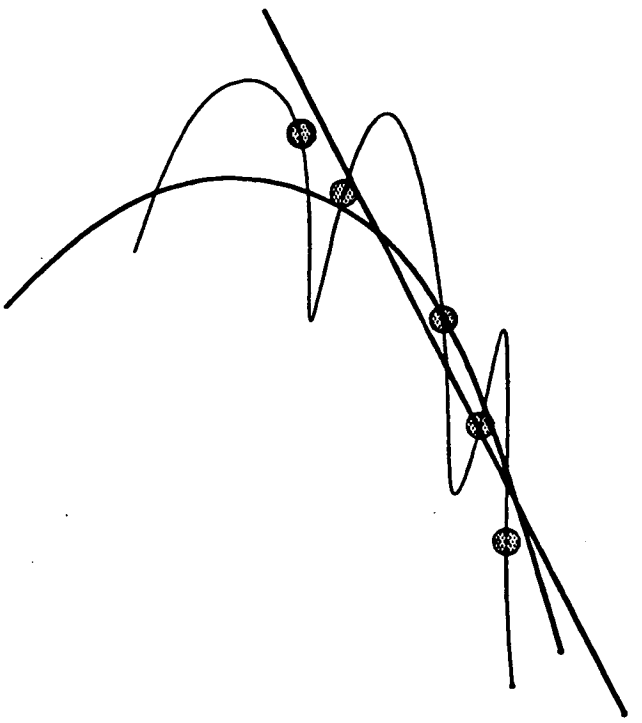


Figure 4.2.3. Example of curve fitting

the same parabola as well as the points at hand. As noise exists, if the sample points are limited, the straight line may fit the given data even better than the parabola, though fitting from a larger data set might be quite different. On the other hand, a high-degree polynomial can fit the given data perfectly, but is of no use to predict unknown data. Indeed, many more sample points would be needed to get a good fit with a high-degree polynomial than a low-degree polynomial because more parameters need to be estimated for a high-degree polynomial.

4.3. Basic Algorithms for HMMs

Given the definition of HMMs, there are three key problems:

(1) The Evaluation Problem: Given the observation sequence $\mathbf{O} = O_1,O_2,..., O_T$, and the model $\lambda=(A,B,\pi)$, how to compute $Pr(\mathbf{O}|\lambda)$, the probability that this observed sequence was produced by the model. This problem can be also viewed as: given several competing models and a sequence of observations, how to choose the model which best matches the observations for the purpose of classification or recognition.

(2) The Estimation Problem: Given the observation sequence $\mathbf{O}$, how to adjust the model parameters $\lambda = (A,B,\pi)$ to maximise $Pr(\mathbf{O}|\lambda)$. The problem concerns how to optimise the model parameters so as to best describe how the observation comes about.

(3) The Decoding Problem: Given the observation sequence $\mathbf{O}$, what is the most likely state sequence $S = s_1,s_2,...,s_T$ according to some optimality criterion. This relates to recovery of the hidden part of the model.

Formal mathematical solutions to these problems will be presented in the following Sections. It can be shown that the three problems are closely related under the same probabilistic framework.

4.3.1. Forward-Backward algorithm

The most straightforward way of computing the probability of an observation is through enumerating every possible state sequence of length T (the number of observations). For every fixed state sequence $S = s_1 s_2 \ldots s_T$, because of our assumptions, the probability of the observation sequence **O** is $Pr(\mathbf{O}|S,\lambda)$, where

$$Pr(\mathbf{O}|S,\lambda) = b_{s_1}(O_1) b_{s_2}(O_2) \ldots b_{s_T}(O_T) \tag{4.3.1}$$

The probability of such a state sequence S, on the other hand, is

$$\begin{aligned} Pr(S|\lambda) &= \pi_{s_1} a_{s_1 s_2} a_{s_2 s_3} \cdots a_{s_{T-1} s_T} \\ &= a_{s_0 s_1} a_{s_1 s_2} a_{s_2 s_3} \cdots a_{s_{T-1} s_T} \end{aligned} \tag{4.3.2}$$

where $a_{s_0 s_1}$ denotes π_{s_1} for simplicity.

The joint probability of **O** and S, i.e., the probability that **O** and S occur simultaneously, is simply the product of the above two terms.

$$Pr(\mathbf{O},S|\lambda) = Pr(\mathbf{O}|S,\lambda) Pr(S|\lambda) \tag{4.3.3}$$

The probability $Pr(\mathbf{O}|\lambda)$ is the summation of Eq. (4.3.3) over all possible state sequences:

$$\begin{aligned} Pr(\mathbf{O}|\lambda) &= \sum_{all\ S} Pr(\mathbf{O}|S,\lambda) Pr(S|\lambda) \\ &= \sum_{all\ S} \prod_{t=1}^{T} a_{s_{t-1} s_t} b_{s_t}(O_t) \end{aligned} \tag{4.3.4}$$

From Eq. (4.3.4) it can be seen that a transition starts from an initial state (at time $t=1$) with probability $a_{s_0 s_1}$ (π_{s_1}), generating the symbol O_1 with the output probability $b_{s_1}(O_1)$ in the corresponding state s_1, and a transition is then made from the initial state s_1 to state s_2 with transition probability $a_{s_1 s_2}$, and generating symbol O_2 with output probability $b_{s_2}(O_2)$ attached to the corresponding state s_2. This process continues until the last transition from state s_{T-1} to state s_T with the transition probability $a_{s_{T-1} s_T}$ and output probability $b_{s_T}(O_T)$ generating symbol O_T is reached.

It should be noted that the computational load for such a process involves on the order of $O(N^T)$ if such a direct definition is used without careful consideration. At every time $t=1,2,...T$, there are N possible states to go through. Fortunately, considering that there are only N states, any possible state sequences have to be remerged into these N states no matter how long the observation sequence is. Thus a more efficient algorithm can be derived based on these characteristics. Such an algorithm is called the forward-backward algorithm [133].

The forward variable can be first defined as:

$$\alpha_t(i) = Pr(O_1 O_2 \cdots O_t, s_t = i | \lambda) \tag{4.3.5}$$

This is actually the probability of the partial observation sequence to time t and state i which is reached at time t, given the model λ. This probability can be calculated inductively, as follows:

Forward algorithm

Step 1: $\alpha_1(i) = \pi_i b_i(O_1)$, for all states i. (if $i \in S_I$ $\pi_i = \frac{1}{N_I}$; otherwise $\pi_i = 0$)

Step 2: Calculating $\alpha()$ along the time axis,

for $t=2,...,T$, and all states j, compute:

$$\alpha_t(j) = [\sum_i \alpha_{t-1}(i) a_{ij}] b_j(O_t) \tag{4.3.6}$$

Step 3: Final probability equals:

$$Pr(\mathbf{O}|\lambda) = \sum_{i \in S_F} \alpha_T(i) \tag{4.3.7}$$

In the above forward iteration, Step 1 initialises the forward probabilities with the initial probability for all states. Eq. (4.3.6) illustrates that state j can be reached at time t from all the possible states i at time t-1. Note that α_{t-1}(i) is the probability that the joint event that $O_1, O_2, ...O_{t-1}$ are observed and state stops at i, thus the product $\alpha_{t-1}(i)a_{ij}$ is then the probability of the joint events $O_1 O_2 \cdots O_{t-1}$ are observed and

state j is reached at time t through state i at time $t-1$. Summing this product over all possible states i, at time $t-1$ results in the probability of state j being reached at time t through all the previous partial observations. Multiplication by $b_j(O_t)$, the observation probability attached to state j which produces O_t results in $\alpha_t(j)$, the probability of the new observation sequence $O_1 O_2 \cdots O_{t-1} O_t$ at time t and state j.

Step 3 gives the desired calculation of $Pr(\mathbf{O}|\lambda)$ as the sum of the final forward variables $\alpha_T(i)$ at final states. This is so because $\alpha_T(i) = Pr(O_1 O_2 \cdots O_T, s_T = i|\lambda)$ and transitions must end at one of S_F.

The computation in the calculation of $\alpha_t(j)$ requires only on the order of $O(N^2T)$. An example of the recursive computation for the forward variable using the model given in Figure 4.2.2. is illustrated in Figure 4.3.1. The computation leads to a lattice

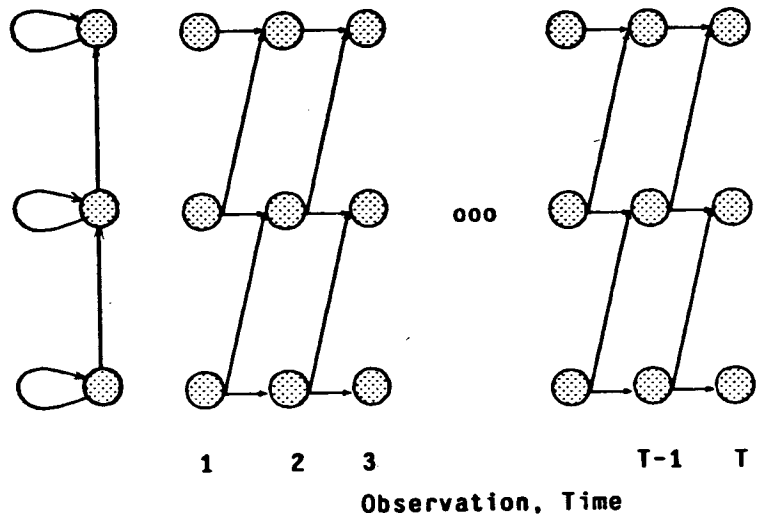


Figure 4.3.1. Illustration of forward computation

structure in which only legal transitions from an originating state i to a destination state j is allowed. Each column of states for time $t-1$ is completely computed before going to time t, the next column. When the states in the last column have been swept, the final state in the final column contains the probability of generating the given observation sequence **O**.

In a similar way a backward variable $\beta_t(i)$ can be defined as:

$$\beta_t(i) = Pr(O_{t+1}, O_{t+2}, \cdots O_T | s_t = i, \lambda) \tag{4.3.8}$$

i.e. the probability of the partial observation sequence from $t+1$ to the final observation T, given state i at time t and the model λ. This backward variable can be also solved inductively in a manner similar to the forward variable $\alpha()$ as follows:

Backward algorithm

Step 1: $\beta_T(i) = \frac{1}{N_F}$, for all states $i \in S_F$, otherwise $\beta_T(i) = 0$;

Step 2: Calculating $\beta()$ along the time axis,

for $t = T-1, T-2, \ldots, 1$ and all states j, compute:

$$\beta_t(j) = [\sum_i a_{ji} b_i(O_{t+1}) \beta_{t+1}(i)] \tag{4.3.9}$$

Step 3: Final probability equals:

$$Pr(O|\lambda) = \sum_{i \in S_I} \pi_i b_i(O_1) \beta_1(i) \tag{4.3.10}$$

Step 1 arbitrarily defines β_T(i) to be $\frac{1}{N_F}$ for all final states. Step 2 shows that in order to have been in state j at time t, and to account for the rest of the observation sequence, a transition from state j to every one of the possible states at time $t+1$ must be made, which accounts for the observation symbol O_{t+1} in the corresponding state, and then account for the rest of the observation sequence. The computation complexity of β_t(i) is similar to that of $\alpha_t(i)$, which also produces a lattice with observation length

and state number.

As mentioned above, both the forward and backward algorithms can be used to compute $Pr(\mathbf{O}|\lambda)$ for the evaluation problem. They can also be used together to formulate a solution to the problem of model parameter estimation as discussed Section 4.3.3.

4.3.2. Viterbi algorithm

The hidden part of HMMs, i.e. the state sequence, cannot be uncovered, but can be interpreted in some meaningful way. A typical use of the recovered state sequence is to learn about the structure of the model, and to get average statistics, behaviour, etc. within individual states. There are several possible ways to find the optimal state sequence associated with the given observation sequence. One possible optimality criterion is to choose the states, s_t, which are in the best path with highest probability, i.e. with maximum $Pr(\mathbf{O},S|\lambda)$. A formal technique for finding this single best state sequence is called the Viterbi algorithm [163]. This is very similar to the DTW algorithm discussed in Chapter 2, where the transition information is neglected and speech data such as LPC cepstrum is usually stored without further parameterisation.

Viterbi Algorithm

Step1: Initialisation, for all states i,

$$\delta_1(i)=\pi_i b_i(O_1)$$

$$\Psi_1(i)=0;$$

Step 2: Recursion, from time at $t=2$ to T, for all states j,

$$\delta_t(j)=\underset{i}{Max}[\delta_{t-1}(i)a_{ij}]b_j(O_t)$$

$$\Psi_t(j)=\underset{i}{argmax}[\delta_{t-1}(i)a_{ij}]$$

Step 3: Termination

$$P^*=\underset{s\in S_F}{Max}[\delta_T(s)]$$

$$s_T^*=\underset{s\in S_F}{argmax}[\delta_T(s)]$$

Step4: Path (state sequence) backtracking, from time at $T-1$ to 1

$$s_t^* = \Psi_{t+1}(s_{t+1}^*)$$

The Viterbi algorithm can be also used in score evaluation. As has already been explained in the previous Sections, the forward-backward algorithm can be used in obtaining the probability $Pr(\mathbf{O}|\lambda)$. This probability is the summation of $Pr(\mathbf{O},S|\lambda)$ over all possible state sequences S, while the Viterbi algorithm only efficiently finds the maximum of $Pr(\mathbf{O},S|\lambda)$ over all S. Therefore, the Viterbi algorithm can be viewed as a special case of the forward-backward algorithm. For speech signals, experiments show that $\underset{S}{Max}[Pr(\mathbf{O},S|\lambda)]$ may not well represent the summation for $Pr(\mathbf{O}|\lambda)$, specially in the HMM parameter estimation procedure, though the probabilities obtained from the forward and Viterbi algorithm may be very close [139]. In such cases, the forward-backward algorithm may work more robustly than the Viterbi algorithm. This is achieved at the sacrifice of the increase of computational complexity of the forward-backward algorithm since all the paths must be taken into account. On the other hand, the Viterbi algorithm is extremely efficient since it can operate in the logarithm domain using only additions. In addition, it is possible to obtain the state sequence at the same time. Because of its advantages, it has been widely used in many speech recognition systems [96,139].

4.3.3. Baum-Welch reestimation algorithm

The most difficult problem in HMM is how to adjust the model parameters (A,B, π) to maximise the probability of the observation sequence given the model. There is no known way to solve for a maximum likelihood model analytically as discussed in Chapter 3. Therefore an iterative algorithm or gradient technique for optimisation is used. The iterative algorithm used in the HMM-based speech recognition is known as the Baum-Welch algorithm [19]. This has the same optimisation techniques as the EM algorithm for the mixture density problems discussed in Chapter 3. The iterative algorithm will be discussed from an intuitive point of view here. A formal proof from an

information-theoretic point of view will be given in the next Section.

The purpose here is to obtain parameters of the model from observations. If the model parameters are known, the forward-backward algorithm can be used to evaluate probabilities produced by given model parameters for given observations. We can then make a guess about original model parameters based on current probabilities.

Consider any model whose parameters λ contain no zeros. Probability computations are, for the moment, to be based on this model, as if it were the true model. By using the forward-backward algorithm on such a model, the posterior probability of transitions γ_{ij}, from state i to state j, conditioned on the observation sequence and the model can be computed as:

$$\begin{aligned}\gamma_t(i,j) &= Pr(s_t=i, s_{t+1}=j \mid \mathbf{O}, \lambda) \\ &= \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{Pr(\mathbf{O}|\lambda)} \\ &= \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_{k \in S_F} \alpha_T(k)}\end{aligned} \tag{4.3.11}$$

As illustrated in Figure 4.3.2., $\gamma_t(i,j)$ is the probability of a path being in state i at time t and making a transition to state j at time $t+1$, given the observation sequence and the model. Obviously, this joint event occurs with probability $\alpha_t(i)$, which accounts for the path terminating in state i at time t, times $a_{ij} b_j(O_{t+1})$, which accounts for the local transition from state i, times $\beta_{t+1}(j)$, which accounts for the path being in state j at time $t+1$.

Similarly, the posterior probability of being in state i at time t, $\gamma_t(i)$, given the observation sequence and model, is

$$\begin{aligned}\gamma_t(i) &= Pr(s_t=i \mid \mathbf{O}, \lambda) \\ &= \frac{\alpha_t(i) \beta_t(i)}{\sum_{k \in S_F} \alpha_T(k)}\end{aligned} \tag{4.3.12}$$

From Eq. (4.3.12), it can be observed that $\gamma_t(i)$ can be computed from $\sum_j \gamma_t(i,j)$ if $t<T$. Since such a computation involves only additions, it is generally better to compute $\gamma_t(i)$ from $\gamma_t(i,j)$ rather than from the forward and backward variables directly.

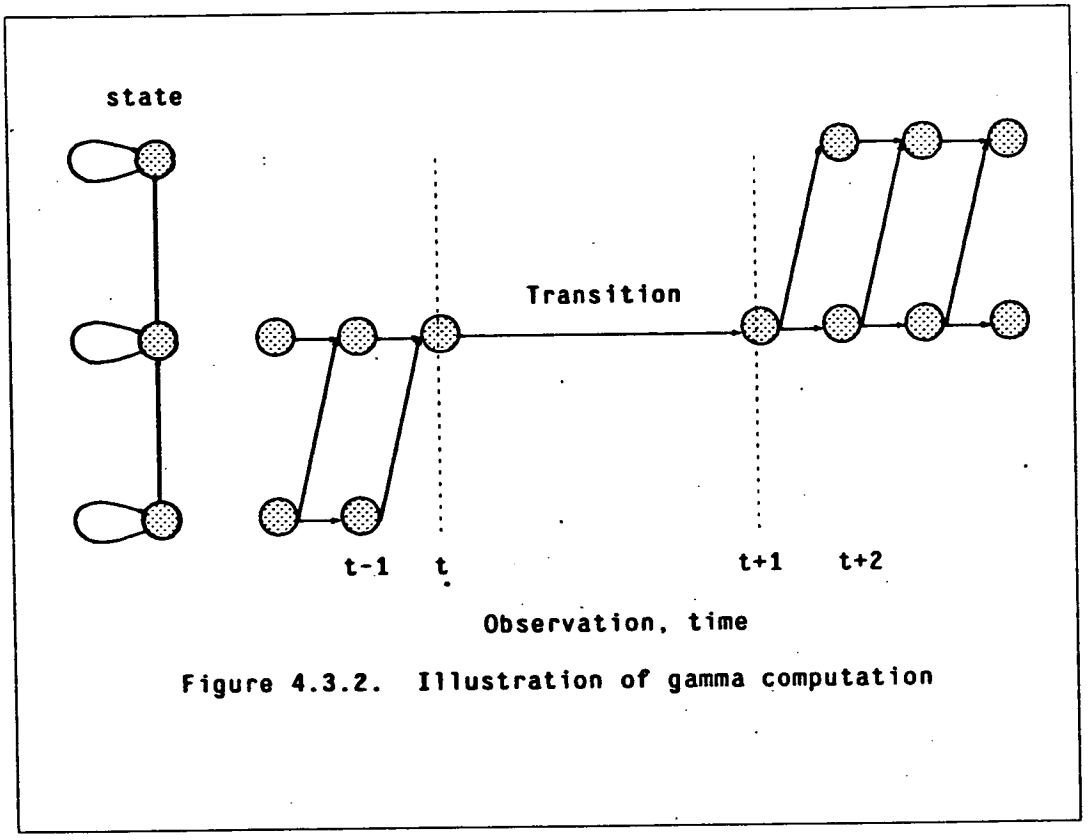


Figure 4.3.2. Illustration of gamma computation

Recalling the urn and ball experiment, (state corresponding to the urn and output probabilities corresponding to the colour ball distributions), it can be seen that $\sum_{t \in O_t = black} \gamma_t(1)$ is the expected number of draws from urn 1 (state 1) that yield the ball with colour *black*, given the observation and model. Similarly, $\sum_{t=1}^{T} \gamma_t(1)$ is the expected number of draws from urn 1, again conditioned on the observations and the model. It is intuitively appealing to use the evidence to replace original output probabilities, $b_1(black)$, by $\sum_{t \in O_t = black} \gamma_t(1) / \sum_{t=1}^{T} \gamma_t(1)$. A new model $\bar{\lambda}$ can then be created in such a manner to iteratively improve our guessing.

In general, the physical meaning of a_{ij} is the probability of the transition from state i to state j. Thus the ratio $\sum_{t=1}^{T-1} \gamma_t(i,j) \; / \; \sum_{t=1}^{T-1} \gamma_t(i)$ is an estimate of the probability

a_{ij}. This ratio may be taken as a new estimate, $\bar{a}_{ij}$ of a_{ij}. That is

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \gamma_t(i,j)}{\sum_{t=1}^{T-1} \sum_{j} \gamma_t(i,j)} = \frac{\sum_{t=1}^{T-1} \gamma_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \tag{4.3.13}$$

Similarly, the physical meaning of $b_j(k)$ is the probability of observation symbol v_k occurring in state j. This can be computed as the frequency of occurrence of observation symbol v_k relative to the frequency of occurrence of any observation symbol in state j. Summation of $\gamma_t(i)$ over the time index t is the expected number of times that state i is visited. (Note that summation over time index t excluding the last moment T is the expected number of transitions out of state i.) Thus $b_j(k)$ can be reestimated as:

$$\bar{b}_j(k) = \frac{\sum_{t \in O_t = v_k} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \tag{4.3.14}$$

Finally, new estimates of the initial state probabilities may be obtained from:

$$\bar{\pi}_i = \gamma_1(i) \tag{4.3.15}$$

It turns out that Eq. (4.3.12) to (4.3.14) is well-founded, and it can be proven that either:

1. The initial model λ defines a critical point of the likelihood function, where new estimates equals old ones, or
2. Model $\bar{\lambda}$ is more likely in the sense that $Pr(\mathbf{O}|\bar{\lambda}) \geq Pr(\mathbf{O}|\lambda)$, i.e. new model estimates will produce the given observation sequence $\mathbf{O}$ more likely.

Thus if $\bar{\lambda}$ is iteratively used to replace λ and repeat the above reestimation calculation, it can be guaranteed that $Pr(\mathbf{O}|\lambda)$ can be improved until some limiting

point is reached. Eq. (4.3.12) to (4.3.14) are instances of the Baum-Welch reestimation algorithm [19], which has a similar form to the EM algorithm discussed in Chapter 3. As a side note, if any set of probabilities in an HMM is fixed, maximum likelihood estimates of the remaining parameters can still be estimated the same way as before; and the likelihood should also be improved iteratively. This is often useful for debugging as a *divide−conquer* strategy. Another useful debugging method is that $Pr(\mathbf{O}|\lambda)$ should be the same no matter whether it is calculated from forward or backward probabilities, i.e.

$$\begin{aligned} Pr(\mathbf{O}|\lambda) &= \sum_{i \in S_F} \alpha_T(i) \\ &= \sum_{i \in S_I} \pi_i b_i(O_1)\beta_1(i) \\ &= \sum_i \alpha_t(i)\beta_t(i). \end{aligned}$$

It is implied by the content of Baum and Eagon [18] and Petrie [130] that the true model can be recovered from a sufficiently long observation sequence except some caveats, such as symmetries associated with the naming of states and ambiguities caused by a true model with identical colour balls in each urn. Therefore, the veil between observations and HMM can be successfully *lifted*.

Note that a single observation sequence is not enough for reestimation of the HMM parameters for practical speech recognition. The appropriate training data should be a set of independent observation sequences from the same source. For example, if each HMM represents a word in the word-based speech recognition system, several utterances for the same word are generally required to reestimate the HMM parameters. The reestimation formulas of Eq. (4.3.13) to (4.3.15) can be easily extended to such multiple observation sequences. Let $\mathbf{O}^M \equiv [\mathbf{O}^1, \mathbf{O}^2, \ldots \mathbf{O}^m]$ denote the set of m observation sequences, where $\mathbf{O}^n = O_1^n, O_2^n, \ldots O_{T_n}^n$ is the nth training sequence with T_n observations. Assuming that observation sequences are independent of each other, the parameter estimation of HMM is then based on the maximisation of

$$\log Pr(\mathbf{O}^M|\lambda) = \sum_{n=1}^{m} \log Pr(\mathbf{O}^n|\lambda) \tag{4.3.16}$$

Let $\sum_t \gamma_t^n(i,j)$ denote the expected number of transitions from state i to state j

estimated from $\mathbf{O}^n$. The average expected number of transitions, $\sum_t \gamma_t(i,j)$ is the summation of $\gamma_t^n(i,j)$ with respect to n. Thus, the reestimation equation for the transition probability, a_{ij}, can be computed:

$$\bar{a}_{ij} = \frac{\sum_n \sum_{t=1}^{T_n-1} \gamma_t^n(i,j)}{\sum_n \sum_{t=1}^{T_n-1} \sum_j \gamma_t^n(i,j)} \tag{4.3.17}$$

Let $\sum_t \gamma_t^n(i)$ denote expected number of being in state i at time t estimated from $\mathbf{O}^n$. Similarly, Eq. (4.3.14) can be extended to multiple observation sequences as:

$$\bar{b}_j(k) = \frac{\sum_n \sum_{t \in O_t^n = v_k} \gamma_t^n(j)}{\sum_n \sum_{t=1}^{T_n} \gamma_t^n(j)} \tag{4.3.18}$$

4.4. Proof of the Reestimation Algorithm

The original proof of the Baum-Welch algorithm, which dealt specifically with a finite alphabet and general output distributions, appeared in [18]. A generalised proof was then based on constructing an information-theoretic Q-function, i.e. Kullback-Leibler number [19,93]. This is actually the same as the Q-function of the EM algorithm for mixture densities discussed in Chapter 3. For models λ and $\bar{\lambda}$, the Q-function can be defined as:

$$Q(\lambda,\bar{\lambda}) = \frac{1}{Pr(\mathbf{O}|\lambda)} \sum_{all\ S} Pr(\mathbf{O},S|\lambda) \log Pr(\mathbf{O},S|\bar{\lambda}). \tag{4.4.1}$$

Here, $Q(\lambda,\bar{\lambda})$ is considered as a function of $\bar{\lambda}$ in the maximisation procedure. Therefore $1/Pr(\mathbf{O}|\lambda)$ can be considered as a constant if there is only one $\mathbf{O}$. With such an auxiliary function, it can be shown that

Theorem 4.4.1.

If $Q(\lambda,\bar{\lambda}) \geq Q(\lambda,\lambda)$, $\Rightarrow$ $Pr(\mathbf{O}|\bar{\lambda}) \geq Pr(\mathbf{O}|\lambda)$. The inequality is strict unless $Pr(\mathbf{O}|\bar{\lambda}) = Pr(\mathbf{O}|\lambda)$.

Proof: From the concavity of the log function it follows that

$$\begin{aligned}\log\frac{Pr(\mathbf{O}|\bar{\lambda})}{Pr(\mathbf{O}|\lambda)} &= \log\Big(\sum_{all\ S}\frac{Pr(\mathbf{O},S|\lambda)}{Pr(\mathbf{O}|\lambda)} * \frac{Pr(\mathbf{O},S|\bar{\lambda})}{Pr(\mathbf{O},S|\lambda)}\Big) \\ &\geq \sum_{all\ S}\frac{Pr(\mathbf{O},S|\lambda)}{Pr(\mathbf{O}|\lambda)}\log\Big(\frac{Pr(\mathbf{O},S|\bar{\lambda})}{Pr(\mathbf{O},S|\lambda)}\Big) \\ &= Q(\lambda,\bar{\lambda}) - Q(\lambda,\lambda)\end{aligned}$$

It can also be seen that λ is a critical point of $Pr(\mathbf{O}|\lambda)$ if and only if it is a critical point of Q as a function of $\bar{\lambda}$. For a broad class of models, Q, as a function of $\bar{\lambda}$, has a single critical point and this point is its unique global maximum (see Chapter 5). From Theorem 4.4.1, we have

$$\log\frac{Pr(\mathbf{O}|\bar{\lambda})}{Pr(\mathbf{O}|\lambda)} \geq Q(\lambda,\bar{\lambda}) - Q(\lambda,\lambda). \tag{4.4.2}$$

If a new model $\bar{\lambda}$ that makes the right-hand side of Eq. (4.4.2) positive can be found, it means that the model reestimation algorithm can be guaranteed to improve the $Pr(\mathbf{O}|\lambda)$. Clearly, the guaranteed improvement by this method results for $\bar{\lambda}$, which maximises $Q(\lambda,\bar{\lambda})$ unless a critical point is reached.

The remarkable fact of the Baum-Welch algorithm is that $Q(\lambda,\bar{\lambda})$ attains its maximum when $\bar{\lambda}$ is related to λ by Eq. (4.3.13) to (4.3.15). To show this let the state sequence be $S = s_1, s_2, \cdots s_T$. Then

$$\log Pr(\mathbf{O},S|\bar{\lambda}) = \log\bar{\pi}_{s_1} + \sum_{t=1}^{T-1}\log\bar{a}_{s_t s_{t+1}} + \sum_{t=1}^{T}\log\bar{b}_{s_t}(O_t) \tag{4.4.3}$$

Substituting Eq. (4.4.3) in (4.4.1) and regrouping terms in the summations according to state transitions and observed symbols, it can be seen that

$$Q(\lambda,\bar{\lambda}) = \sum_i\sum_j c_{ij}\log\bar{a}_{ij} + \sum_j\sum_{k=1}^{L} d_{jk}\log\bar{b}_j(k) + \sum_i e_i\log\bar{\pi}_i \tag{4.4.4}$$

Here

$$c_{ij} = \frac{\sum_{t=1}^{T-1} Pr(s_t=i, s_{t+1}=j, \mathbf{O}|\lambda)}{Pr(\mathbf{O}|\lambda)} \tag{4.4.5}$$

$$= \sum_{t=1}^{T-1} \gamma_t(i,j)$$

$$d_{jk} = \frac{\sum_{t \in O_t = v_k} Pr(s_t=j, \mathbf{O}|\lambda)}{Pr(\mathbf{O}|\lambda)} \tag{4.4.6}$$

$$= \sum_{t \in O_t = v_k} \gamma_t(j)$$

$$e_i = \frac{Pr(s_1=i, \mathbf{O}|\lambda)}{Pr(\mathbf{O}|\lambda)} \tag{4.4.7}$$

$$= \gamma_1(i)$$

Thus, according to Theorem 3.2.1, $Q(\lambda,\bar{\lambda})$ can be maximised if

$$\bar{a}_{ij} = \frac{c_{ij}}{\sum_j c_{ij}}$$

$$= \frac{\sum_{t=1}^{T-1} \gamma_t(i,j)}{\sum_{t=1}^{T-1} \sum_j \gamma_t(i,j)} \tag{4.4.8}$$

$$\bar{b}_j(k) = \frac{d_{jk}}{\sum_k d_{jk}}$$

$$= \frac{\sum_{t \in O_t = v_k} \gamma_t(j)}{\sum_t \gamma_t(j)} \tag{4.4.9}$$

$$\bar{\pi}_i = \frac{e_i}{\sum_i e_i} \tag{4.4.10}$$

$$= \gamma_1(i)$$

These are recognised as the Baum-Welch reestimation formulas.

The Baum-Welch reestimation algorithm can also be proved from several different points of view. Note that the reestimation formulas update the model in such a way that the constraints

$$\sum_i \pi_i = 1, \tag{4.4.11}$$

$$\sum_j a_{ij} = 1, \tag{4.4.12}$$

and

$$\sum_{k=1}^{L} b_j(k) = 1 \tag{4.4.13}$$

are automatically satisfied at each iteration. The constraints are required to make the HMM well defined. It is thus natural to look at the training problem as a problem of constrained optimisation of log $Pr(\mathbf{O}|\lambda)$, since it is usually numerically better to maximise log $Pr(\mathbf{O}|\lambda)$ instead of $Pr(\mathbf{O}|\lambda)$ [20,102].

For multiple observation sequences, the Q-function can be defined as the summation of each Q-function corresponding to the nth observation sequence $\mathbf{O}^n$ (this Q-function is in fact the case of the EM algorithm for mixture densities as discussed in Chapter 3). Following Theorem 4.4.1, it can be seen that maximisation of the Q-function will lead to maximisation of $Pr(\mathbf{O}^M|\lambda)$. From Eq. (4.4.4), it can be easily verified that reestimation formulas Eq. (4.3.17) and (4.3.18) stand.

4.5. Time Duration Modelling

One of the major weakness of conventional HMMs is related to the modelling of state duration. The HMM does not provide an adequate representation of the temporal structure of speech where the probability of state occupancy decreases exponentially with time. The probability of t consecutive observations in state i can be written as

$$d_i(t) = a_{ii}^t(1-a_{ii}) \tag{4.5.1}$$

i.e., $d_i(t)$ is the probability of taking the self-loop at state i for t times. An improvement to the standard HMM results from the use of HMMs with time duration [100,144],

which model not only the output and transition probabilities, but also a set of state duration probabilities explicitly.

To explain the principle of time duration modelling, a conventional HMM with exponential state duration density and a time duration HMM with specified state duration densities (which can be either a discrete distribution or a continuous density) are illustrated in Figure 4.5.1. In part (a), the state duration probability has exponential form as Eq. (4.5.1). In part (b), the self-transition probabilities are replaced with an explicit duration probability distribution. At time t, the process enters state i

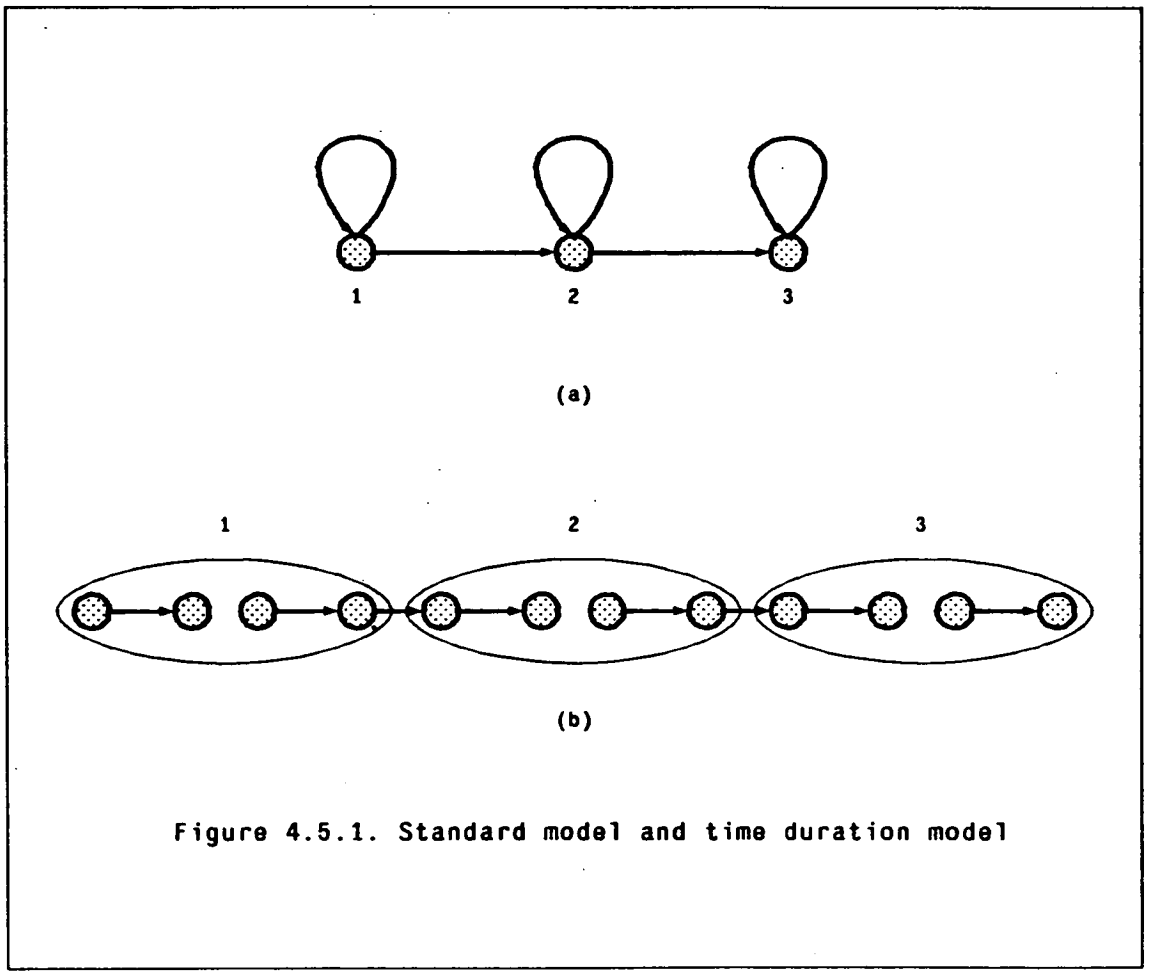


Figure 4.5.1. Standard model and time duration model

for duration τ with probability density $d_i(\tau)$ during which the observations $O_{t+1},O_{t+2},\ldots,O_{t+\tau}$ are generated. It then transfers to state j with transition probability a_{ij} only after the appropriate τ observations have occurred in the state i. Thus, by setting the time duration probability density to be the exponential density of Eq. (4.5.1) the time duration HMM can be made equivalent to the standard HMM.

The parameters $d_i(\tau)$ can be estimated from observations along with the other parameters of the HMM. For expedience the duration density is usually truncated at a maximum duration value D. To reestimate the parameters of the HMM with time duration modelling, the forward recursion must be modified as follows:

$$\alpha_t(j) = \sum_{\tau} \sum_{all\ i \atop i \neq j} \alpha_{t-\tau}(i) a_{ij} d_j(\tau) \prod_{l=1}^{\tau} b_j(O_{t-\tau+l}) \tag{4.5.2}$$

where the transition from state i to state j depends not only upon the transition probability a_{ij} but also upon all the possible time durations τ that may occur in state i. Intuitively, Eq. (4.5.2.) illustrates that when state j is reached from previous states i, the observations may stay in state j for a period of τ with duration density $d_j(\tau)$, and each observation emits its own output probability. All possible durations must be considered, which leads to summation with respect to τ. The independence assumption of observations results in the $\prod$ term of output probabilities. Similarly, the backward recursion can be written as:

$$\beta_t(i) = \sum_{\tau} \sum_{all\ j \atop j \neq i} a_{ij} d_j(\tau) \prod_{l=1}^{\tau} b_j(O_{t+l}) \beta_{t+\tau}(j) \tag{4.5.3}$$

The modified Baum-Welch algorithm can then be used based on Eq. (4.5.2) and (4.5.3). The proof of the reestimation algorithm can be based on the modified Q-function as Eq. (4.4.1) except that $Pr(\mathbf{O},S|\lambda)$ should be replaced $Pr(\mathbf{O},S,\tau|\lambda)$, which denotes the conditional probability of state sequence, S, given observation, $\mathbf{O}$, and duration in each state, τ.

$$Q(\lambda,\bar{\lambda}) = \frac{1}{Pr(\mathbf{O}|\lambda)} \sum_{\tau} \sum_{all\ S} Pr(\mathbf{O},S,\tau|\lambda) \log Pr(\mathbf{O},S,\tau|\bar{\lambda}) \tag{4.5.4}$$

In a manner similar to the standard HMM, $\gamma_{t,\tau}(i,j)$ can be defined as the transition probability from state i at time t to state j with time duration τ in state j.

$\gamma_{t,\tau}(i,j)$ can be written as:

$$\gamma_{t,\tau}(i,j) = \alpha_t(i)a_{ij}d_j(\tau)\prod_{l=1}^{\tau} b_j(O_{t+l})\beta_{t+\tau}(j) \tag{4.5.5}$$

Similarly, the probability of being in state i at time t with duration τ can be computed as:

$$\gamma_{t,\tau}(i) = \sum_j \gamma_{t,\tau}(j,i) \tag{4.5.6}$$

In a manner similar to Eq. (4.4.4) and after some algebraic operations, the reestimation algorithm can be written as follows

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1}\sum_{\tau}\gamma_{t,\tau}(i,j)}{\sum_{t=1}^{T-1}\sum_{\tau}\sum_{j}\gamma_{t,\tau}(i,j)} \tag{4.5.7}$$

$$\bar{d}_j(\tau) = \frac{\sum_{t=1}^{T}\gamma_{t,\tau}(j)}{\sum_{t=1}^{T}\sum_{\tau}\gamma_{t,\tau}(j)} \tag{4.5.8}$$

$$\bar{b}_j(k) = \frac{\sum_{t=1}^{T}\sum_{\tau}\gamma_{t,\tau}(j)*c_{t,\tau}(k)}{\sum_{t=1}^{T}\sum_{\tau}\gamma_{t,\tau}(j)\tau} \tag{4.5.9}$$

where $c_{t,\tau}(k)$ denotes the number of codewords v_k occurring during state occupancy τ for $O_{t+1},\ldots O_{t+\tau}$.

The Viterbi decoding algorithm can be used for the time duration model, and the optimal path can be determined according to:

$$\delta_t(j) = \underset{i}{Max}\ \underset{\tau}{Max}[\delta_{t-\tau}(i)a_{ij}d_j(\tau)\prod_{k=1}^{\tau} b_j(O_{t-\tau+k})] \tag{4.5.10}$$

The importance of incorporating time duration modelling is reflected in the observation that, for some speaker-dependent speech recognition systems, the recognition accuracy can be significantly improved when time duration modelling is used. However, there are drawbacks to the use of the time duration modelling discussed here. One is the greatly increased computational complexity in the order of $O(D^2)$,

where D is the truncated time duration length. Another problem is the large number of additional parameters (D), associated with each state, that must be estimated. Usually, there are much less training data to estimate $d_j(t)$ than would be used in a standard HMM. This limited training data may cause the time duration HMM to show poorer recognition accuracy than a standard HMM [145]. One proposal to alleviate some of these problems is to use a continuous density function instead of the discrete distribution $d_j(t)$ [100,144]. A more interesting alternative is to smooth the discrete time duration distribution by the Gaussian probability density function [3,96], which can significantly improve the performance when limited training data are used.

4.6. Isolated vs. Continuous Speech Recognition

For isolated word recognition, the training and recognition can be implemented directly using the basic algorithms introduced in this Chapter.

To estimate model parameters, examples of each word in the vocabulary can be collected. The model parameters can be estimated from all these examples using the Baum-Welch algorithm. It is not necessary to clip the speech from the silence at the beginning and ending because these will be absorbed in the states of the word models. If subword units, such as phone models, are used, these subword units can be first concatenated into a word model, possibly adding silence models at the beginning and end. These concatenated word models can then be treated in the same manner as word models.

For recognition, either the forward algorithm or the Viterbi algorithm can be used to score the input word against each of the word models. If no language model is used, the word model with the highest probability can be chosen as the recognised word. If a language model is used, the decoder based on a maximum a-posteriori probability can be used.

Training HMMs on continuous speech is very similar to training on isolated words. One of great advantages for hidden Markov modelling is that it can absorb a range of

boundary information of models *automatically* for continuous speech recognition. Other techniques such as DTW, or neural networks face serious problems in training models for continuous speech, because word boundaries are not automatically detectable. Tedious hand-marking is often needed.

To train the parameters of the HMM, each word can be instantiated with its model (which may be a concatenation of subword models). The words in the sentence can be concatenated with optional silence models between them. This large concatenated sentence HMM can then be trained using the corresponding sentence. Since the entire sentence HMM is trained on the entire observation sequence for the corresponding sentence, all possible word boundaries are inherently considered. Parameters of each model will be based on those good state-to-speech alignments. This is because the state sequence is hidden in HMMs, and it does not matter where the word boundaries are, since these will be automatically determined by the reestimation algorithm. Such a training method allows complete freedom to align the sentence model against the observation, and no effort is needed to find word boundaries.

In continuous speech recognition, a word may begin and end anywhere in a given observation signal. As the word boundaries cannot be detected accurately, all possible beginning and end points have to be accounted for. This converts a linear search (as for isolated word recognition) to a tree search, and a polynomial recognition algorithm to an exponential one. As an optimal full search is infeasible for large-vocabulary continuous speech recognition, several suboptimal search algorithms are used instead [10,106,122,140,163].

The Viterbi search [163] can be extended to continuous speech recognition by enumerating all the states of all the words in the grammar and using the Viterbi algorithm inside the words with between-word transitions specified by the grammar. This is efficient for moderate vocabulary recognition systems [15]. However, for large-vocabulary speech recognition, it is still very time consuming. Instead of retaining all candidates at every time frame, a threshold can be used to consider only a group of likely candidates. The state with the highest probability can be first found, and each state with smaller probability in comparison to the highest one can then be discarded

from further consideration. This is the *beam search* algorithm, which alleviates the necessity of enumerating all the states, and can lead to substantial savings in computation with little loss of accuracy.

Despite the efficiency of the Viterbi algorithm, it is a suboptimal search since it finds only the optimal state sequence instead of the optimal word sequence and the probability obtained from the Viterbi algorithm is an approximation of $Pr(\mathbf{O}|\lambda)$. In view of this problem, improved search algorithms, such as stack decoding algorithm [10], can be used, with however considerable increase in computational complexity.

4.7. Summary

An HMM is a collection of states connected by transitions with output probabilities associated with states and transition probabilities associated with arcs. HMMs have a rich representation in these two sets of parameters such that variabilities in time and acoustic space can be effectively modelled. The forward-backward algorithm and Baum-Welch algorithm can be used to automatically optimise model parameters with high efficiency in the sense of maximum likelihood estimation. A simplified technique, the Viterbi algorithm, is often used for decoding because of its simplicity. The proof of reestimation algorithm is based on constructing an information-theoretic Q-function, which is actually identical to the EM algorithm discussed in Chapter 3 except here the Markov properties are imposed. Although discussions in this Chapter are all based on the discrete HMM, most of them can be extended to the continuous HMM.

CHAPTER 5

CONTINUOUS HIDDEN MARKOV MODELS

This Chapter will discuss theories related to continuous mixture HMMs. If the observation does not come from a finite set, but from some continuous points in Euclidean d-space, the discrete output distribution $b_j(k)$ can then be extended to the continuous output probability density function. For speech recognition, this implies that the vector quantisation procedure is unnecessary. Thus the inherent quantisation error can be eliminated.

The Baum-Welch reestimation algorithm discussed in Section 4.3.3. can be extended to estimate continuous probability density functions with the help of the Kullback-Leibler statistic, the Q-function [19,20]. As discussed in Chapter 4, the Q-function is one of the main mathematical foundations of the theory, in that the parameter reestimates can be characterised as the critical point of it. Baum *et al.* [19,20] generalised the method to continuous output density functions, which require that the probability density functions be strictly log concave. This method is applicable to the Gaussian, Poisson, and Gamma distributions but not to the Cauchy distribution. Liporace [102] redefined the Q-function, and successfully relaxed the requirement so as to accommodate a broad class of elliptically symmetric density functions. Juang [86] further expanded the estimation algorithm to cope with finite mixtures of strictly log concave and/or elliptically symmetric density functions. This Chapter will first review the Liporace proof to the general continuous parameter reestimation formulas, and then discuss continuous mixture models, which are necessary to understand the unified modelling theory presented in the following Chapter.

5.1. Continuous Parameter Reestimation

This Section will first discuss general reestimation formulas for the single mixture continuous HMM that is applicable to a broad class of elliptically symmetric density functions. As direct applications, examples of the Gaussian density function are then included.

5.1.1. General case

In a manner similar to the discrete HMM, for a given continuous observation sequence $\mathbf{X} \in R^d$ and a particular choice of HMM λ, the objective in maximum likelihood estimation is to maximise the probability density function, $f(\mathbf{X}|\lambda)$, over all parameters in λ. The global density of $\mathbf{X}$ can be written as

$$\begin{aligned} f(\mathbf{X}|\lambda) &= \sum_{all\ S} f(\mathbf{X},S|\lambda) \\ &= \sum_{S} \prod_{t=1}^{T} a_{s_{t-1}s_t} b_{s_t}(\mathbf{x}_t) \end{aligned} \tag{5.1.1}$$

where $a_{s_0 s_1} = \pi_{s_1}$ for simplicity. As a general case, all of the output density functions can be assumed to have ellipsoidal symmetry, i.e. each $b_i(\mathbf{x})$ has the form

$$|\Sigma_i|^{-1/2} f_i(q_i(\mathbf{x})) \tag{5.1.3}$$

where $q_i(\mathbf{x})$ is a positive definite quadratic form,

$$q_i(\mathbf{x}) = (\mathbf{x}-\mu_i)^t \Sigma_i^{-1} (\mathbf{x}-\mu_i). \tag{5.1.3}$$

The d-by-d scaling matrices Σ_i, (for all states i), are positive-definite and symmetric, and location vectors μ_i are arbitrary points in Euclidean d-space. Following Liporace's results [102], one extra assumption for elliptically symmetric densities is necessary: if the elliptically symmetric density $b(\mathbf{x})$ satisfies the consistency conditions of Kolmogorov [44], it can be represented as

$$b(\mathbf{x}) = \int_0^\infty N(\mathbf{x}, \mu, u^2\Sigma) dG(u) \tag{5.1.4}$$

for some probability distribution G on $[0, \infty)$, where $N()$ is the multivariate Gaussian density with mean vector μ and covariance matrix $u^2\Sigma$. Thus an elliptically symmetric density function can be expressed as a continuous convex combination of related Gaussian densities. It is interesting that the distribution G need not be known. By

means of Eq. (5.1.4), $f(\mathbf{X},S|\lambda)$ can be expressed as an integral over the T-fold product space $\ddot{U} = [0, \infty)^T$

$$\begin{aligned} f(\mathbf{X},S|\lambda) &= \int_{\ddot{U}} \prod_{t=1}^{T} a_{s_{t-1}s_t} N(\mathbf{x}_t,\mu_{s_t},u_t^2\Sigma_{s_t}) dG(u_1)...dG(u_T) \\ &= E_U f(\mathbf{X},S,U|\lambda), \end{aligned} \tag{5.1.5}$$

where $U=(u_1,...,u_T)^t$, E_U denotes the average with respect to T-fold distribution $G(u_1)...G(u_T)$ and

$$f(\mathbf{X}, S, U|\lambda) = \prod_{t=1}^{T} a_{s_{t-1}s_t} N(\mathbf{x}_t,\mu_{s_t},u_t^2\Sigma_{s_t})$$

In a similar manner to the discrete HMM, the reestimation transformation is based on an auxiliary function $Q(\lambda,\bar{\lambda})$ of current parameters λ and new parameters $\bar{\lambda}$ defined by

$$Q(\lambda,\bar{\lambda}) = \frac{1}{f(\mathbf{X}|\lambda)} \sum_S [f(\mathbf{X},S|\lambda)\log f(\mathbf{X},S|\bar{\lambda})] \tag{5.1.6a}$$

$Q(\lambda,\bar{\lambda})$ can be considered as a function of $\bar{\lambda}$ in the optimisation procedure. Therefore, $1/f(\mathbf{X}|\lambda)$ can be treated as a constant if there is only one observation sequence $\mathbf{X}$ to be considered. The proof based on Eq. (5.1.6a) is valid only if the output probability density is strictly log concave. For a broad class of elliptically symmetric density functions, the Q-function can be generalised as

$$Q(\lambda,\bar{\lambda}) = \frac{1}{f(\mathbf{X}|\lambda)} \sum_S E_U [f(\mathbf{X},S,U|\lambda)\log f(\mathbf{X},S,U|\bar{\lambda})] \tag{5.1.6b}$$

The following discussion will be based on Eq. (5.1.6b); it can be simplified to Eq. (5.1.6a) which is only a special case of Eq. (5.1.6b).

The utility of $Q(\lambda,\bar{\lambda})$ is similar to that discussed in Theorem 4.4.1, i.e.

$$Q(\lambda,\bar{\lambda}) \geq Q(\lambda,\lambda) \Rightarrow f(\mathbf{X}|\bar{\lambda}) \geq f(\mathbf{X}|\lambda). \tag{5.1.7}$$

Under mild orthodoxy conditions, $Q(\lambda,\bar{\lambda})$ has additional useful properties summarised in the following theorem.

Theorem 5.1.1.

If for each state j,

(i) $b_j(\mathbf{x})$ has the representation as Eq. (5.1.4); and

(ii) there are among $\mathbf{x}_1,\ldots,\mathbf{x}_T$, $d+1$ observations, any d of which are linearly independent;

then $Q(\lambda,\bar{\lambda})$ has a unique global maximum as a function of $\bar{\lambda}$, and this maximum is the one and only critical point.

Proof: The proof involves the following three arguments.

(1) The second derivative of Q along any direction in the parameter space is strictly negative at a critical point. This implies that any critical point is a relative maximum and that if there are more than one they are isolated.

(2) $Q(\lambda,\bar{\lambda})\rightarrow-\infty$ as $\bar{\lambda}$ approaches the finite boundary of the parameter space or the point at ∞. The property implies that the global maximum is a critical point.

(3) The critical point is unique.

Detailed mathematical arguments can be found in [102].

Theorem 5.1.2:

A parameter λ is a critical point of the likelihood $f(\mathbf{X}|\lambda)$ if and only if it is a critical point of the Q-function.

Proof: Let ∇_λ be the gradient vector. A critical point of $f(\mathbf{X}|\lambda)$ is characterised by $\nabla f(\mathbf{X}|\lambda) = 0$.

$$\begin{aligned}
&\nabla_\lambda f(\mathbf{X}|\lambda)|_\lambda \\
&= \nabla_\lambda \sum_S E_U f(\mathbf{X},S,U|\lambda) \\
&= \sum_S E_U \nabla_\lambda f(\mathbf{X},S,U|\lambda) \\
&= \sum_S E_U f(\mathbf{X},S,U|\lambda)[\nabla_\lambda \log f(\mathbf{X},S,U|\lambda)] \\
&= C\ \nabla_{\bar{\lambda}} Q(\lambda,\bar{\lambda})|_{\bar{\lambda}=\lambda}
\end{aligned}$$

Thus $\nabla_\lambda f(\mathbf{X}|\lambda)|_{\lambda=\bar{\lambda}}=0$ if and only if $\nabla_{\bar{\lambda}} Q(\lambda,\bar{\lambda})|_{\bar{\lambda}=\lambda}=0$ at $\lambda=\bar{\lambda}$.

Based on Theorem 5.1.1. and Theorem 5.1.2, the reestimation transformation can be explicitly derived under the constraints of

(1) $\sum_j \bar{a}_{ij}=1$, for each state i; and

(2) The scaling matrices $\bar{\Sigma}_i$ are positive definite for each state i.

Since

$$Q(\lambda,\bar{\lambda}) = \frac{1}{f(\mathbf{X}|\lambda)}\sum_S E_U\, f(\mathbf{X},S,U|\lambda)\sum_{t=1}^{T}[\log\bar{a}_{s_{t-1}s_t} + \\ + (1/2)\log|\bar{\Sigma}_{s_t}^{-1}| - d\log u_t - (d/2)\log(2\pi) - (1/2u_t^2)(\mathbf{x}_t-\bar{\mu}_{s_t})\bar{\Sigma}_{s_t}^{-1}(\mathbf{x}_t-\bar{\mu}_{s_t})], \tag{5.1.8}$$

maximisation of a_{ij} is similar to the discrete HMM as the individual auxiliary function for a_{ij} has the same form $\sum_j w_j \log y_j$, which as a function of $\{y_i\}$, subject to the constraints $\sum_j y_j = 1$ and $y_j \geq 0$, attains a global maximum at the single point $y_j = w_j / \sum_i w_i$.

The key problem here is to maximise the Q-function with respect to μ and Σ. Let $\partial Q/\partial\bar{\mu}_j$ denote the d-dimensional vector of derivatives of $Q(\lambda,\bar{\lambda})$ with respect to the components of $\bar{\mu}_j$. Here μ_j can be obtained as the solution of

$$0 = \frac{\partial Q(\lambda,\bar{\lambda})}{\partial\bar{\mu}_j} \\ = \sum_S E_U f(\mathbf{X},S,U|\lambda)\sum_{t\in T_j(s)}\bar{\Sigma}_j^{-1}(\mathbf{x}_t-\bar{\mu}_j)/u_t^2$$

where $T_j(s) = \{t|s_t=j\}$. Interchanging the order of summation and premultiplying by $\bar{\Sigma}_j$, we have

$$0 = \sum_{t=1}^{T}\sum_{S\in S_j(t)} E_U(f(\mathbf{X},S,U|\lambda)/u_t^2)(\mathbf{x}_t-\bar{\mu}_j) \tag{5.1.9}$$

where $S_j(t) = \{s|s_t=j\}$. Observe that

$$\sum_{S\in S_j(t)} E_U(f(\mathbf{X},S,U|\lambda)/u_t^2)$$

differs from $f(\mathbf{X},s_t=j|\lambda)$ only in that $b_j(\mathbf{x}_t)$ is replaced by

$$\int_0^\infty u_t^{-2}N(\mathbf{x}_t,\mu_j,u_t^2\Sigma_j)dG(u_t) \tag{5.1.10}$$

According to Eq. (5.1.4), Eq. (5.1.10) is equal to $-2\partial b_j(\mathbf{x})/\partial q_j(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}_t}$. Therefore

$$\sum_{s \in S_j(t)} E_U(f(\mathbf{X},S,U|\lambda)/u_t^2) = \rho_t(j)\beta_t(j)$$

where

$$\rho_t(j) = \sum_i \alpha_{t-1}(i)a_{ij}[-2\frac{\partial b_j(\mathbf{x})}{\partial q_j(\mathbf{x})}|_{\mathbf{x}=\mathbf{x}_t}] \tag{5.1.11}$$

Thus

$$\bar{\mu}_j = \frac{\sum_{t=1}^{T} \rho_t(j)\beta_t(j)\mathbf{x}_t}{\sum_{t=1}^{T} \rho_t(j)\beta_t(j)} \tag{5.1.12}$$

Similarly, $\bar{\Sigma}_j$ can be obtained as

$$\bar{\Sigma}_j = \frac{\sum_{t=1}^{T} \rho_t(j)\beta_t(j)(\mathbf{x}_t-\bar{\mu}_j)(\mathbf{x}_t-\bar{\mu}_j)^t}{\sum_{t=1}^{T} \rho_t(j)\beta_t(j)} \tag{5.1.13}$$

It can be seen that each $\bar{\Sigma}_j$ is positive definite. If $\mathbf{y}$ is any d-dimensional vector,

$$\mathbf{y}^t\bar{\Sigma}_j\mathbf{y} = \sum_{t=1}^{T} c_j(t)[\mathbf{y}^t(\mathbf{x}_t-\bar{\mu}_j)]^2 \geq 0$$

where $c_j(t) > 0$. The inequality is strict provided that for any $\bar{\mu}_j$ the vectors $\{\mathbf{x}_t-\bar{\mu}_j\}$ span the d-dimensional observation space, i.e., if observation $\mathbf{x}$ satisfies the orthodoxy condition mentioned in Theorem 5.1.1.

5.1.2. Applications to single Gaussian density function

When the above reestimation formulas are applied to the multivariate Gaussian densities, $\rho_t(j) = \alpha_t(j)$. Therefore, Eq. (5.1.12) and (5.1.13) become

$$\bar{\mu}_j = \frac{\sum_{t=1}^{T} \gamma_t(j)\mathbf{x}_t}{\sum_{t=1}^{T} \gamma_t(j)} \tag{5.1.14}$$

and

$$\bar{\Sigma}_j = \frac{\sum_{t=1}^{T} \gamma_t(j)(\mathbf{x}_t - \bar{\mu}_j)(\mathbf{x}_t - \bar{\mu}_j)^t}{\sum_{t=1}^{T} \gamma_t(j)} \tag{5.1.15}$$

where $\gamma_t(j)$ is the posterior probability which has the same meaning as used for the discrete HMM. The interpretation of Eq. (5.1.14) and Eq. (5.1.15) is relatively simple. In the extreme case where $\gamma_t(j)$ equals 1 when $\mathbf{x}_t$ is from the state j and 0 otherwise, $\bar{\mu}_j$ and $\bar{\Sigma}_j$ are the mean and corresponding sample covariance matrix of those samples respectively. More generally, $\gamma_t(j)$ is between 0 and 1, and all of the samples therefore play some role in the estimates. Basically, the estimates can still be regarded as *weighted* sample means and *weighted* sample covariance matrices.

In practice, multiple independent observations have to be used for parameter estimation. In a manner similar to Eq. (4.3.17) and Eq. (4.3.18), summation with respect to variables of multiple independent observations can be applied to the denominator and numerators respectively. Notice that Eq. (5.1.15) relies on reestimated mean vector $\bar{\mu}_j$, which is inconvenient to implement, Eq. (5.1.15) can be expressed as

$$\bar{\Sigma}_j = \frac{\sum_{t=1}^{T} \gamma_t(j)\mathbf{x}_t\mathbf{x}_t^t}{\sum_{t=1}^{T} \gamma_t(j)} - \bar{\mu}_j\bar{\mu}_j^t \tag{5.1.16}$$

In the above equation, the first term depends on γ_t and $\mathbf{x}_t$, which can be computed for each observation $\mathbf{x}_t$ along with other parameters. The second term can be computed after all the observations are computed. Alternatively, a heuristic reestimation equation can be written as [86]

$$\bar{\Sigma}_j = \frac{\sum_{t=1}^{T} \gamma_t(j)(\mathbf{x}_t - \mu_j)(\mathbf{x}_t - \mu_j)^t}{\sum_{t=1}^{T} \gamma_t(j)} \tag{5.1.17}$$

This is because μ are approximately equal to $\bar{\mu}$ in contiguous iterations.

Referring to Eq. (3.2.10) and (3.2.11), the mixture densities problem discussed in Chapter 3 can be considered as a special case of this model in which the state transitions are Markov of order zero with L states ($N = L$),

$$a_{ij} = p_j, \quad j = 1, ..., L \text{ for } all\ i$$

i.e., no Markov properties are imposed on the densities.

5.2. Mixture Density Functions

The extent to which use of unimodal output densities in a speech recognition system is adequate depends on both the signal processing in the system, and on the shapes of the word models used by the system. In a speaker-independent speech recognition system, for example, because the vocal tracts of women are normally shorter than those of men, the formant frequencies for a given sound will tend to be higher in a female voice than a male voice. It will then be important to use at least bimodal output densities to model the male and female voices for a given word. On the other hand, if the signal processing algorithm can successfully perform some speaker normalisation, this difference between male and female may be removed and unimodal densities can be used. This is only one aspect of the complicated speech modelling problem. In practice, multimodal densities will definitely be superior to unimodal densities if there is sufficient training data, and this is generally required for speaker-independent speech recognition. One way to model multimodal signals is to use mixtures of unimodal distributions, such as Gaussian densities. Output distributions in the continuous mixture HMM are then a mixture of unimodal densities. Another approach is to use a large number of states based on unimodal densities [43]. In comparison to the discrete HMM, it makes no difference if a mixture of discrete distributions is used since the discrete output distributions can model any multimodal densities.

In use of continuous probability density functions, a first candidate for a family of output distributions is the family of multivariate Gaussians, since

(1) Gaussian mixture densities (with an appropriate chosen mixture) can be used to approximate any continuous probability density function in the sense of minimising the error between two density functions;

(2) by the central limit theorem, the distribution of the sum of a large number of independent random variables tends towards a Gaussian distribution; and

(3) the Gaussian distribution has the greatest entropy of any distribution with a given variance.

Continuous HMMs based on other probability density functions, such as Laplacian, K_o-type [48], as well as the Parzen estimation of the probability density functions [159] have also been reported in the literature. These are not considered further here because of the advantages of Gaussian density mentioned above.

The number of free parameters is very important in a statistical speech recognition system. One way to drastically reduce the number of free parameters in the Gaussian density based continuous HMM is to assume that the off-diagonal terms in the covariance matrices are zero. This is a reduction from $O(d^2)$ to $O(d)$ in terms of both the amount of computation and the number of free parameters to be estimated. This means that less training data and time will be required. The disadvantage is that the assumption that different elements of the observation vector are uncorrelated may be so inaccurate as to significantly degrade recognition accuracy. The extent to which this is the case depends on the signal processing and the models used in a particular system. It is often an empirical issue as to whether the computational expense of a full-covariance Gaussian is worth the performance improvement, if any. Even though that diagonal-covariance assumptions are almost surely incorrect, the diagonal-covariance approach often provides better performance than the full-covariance approach if the training data are insufficient.

Formally, the probability density function $B=\{b_j(\mathbf{x})\}$ attached to state j, $1 \leq j \leq N$, if chosen as for Gaussian mixture densities, can be written as:

$$\begin{aligned} b_j(\mathbf{x}) &= \sum_{k=1}^{M} c_{jk} b_{jk}(\mathbf{x}) \\ &= \sum_{k=1}^{M} c_{jk} N(\mathbf{x}, \boldsymbol{\mu}_{jk}, \boldsymbol{\Sigma}_{jk}) \end{aligned} \tag{5.2.1}$$

where $N(\mathbf{x},\boldsymbol{\mu},\boldsymbol{\Sigma})$ denotes a multi-dimensional Gaussian density function of mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$; M denotes the number of mixture-components; and c_{jk} is the weight for the kth mixture component satisfying

$$\sum_{k=1}^{M} c_{jk} = 1 \tag{5.2.2}$$

so that

$$\int_{-\infty}^{\infty} b_j(\mathbf{x})d\mathbf{x} = 1. \tag{5.2.3}$$

5.3. Continuous Mixture Hidden Markov Model

When $b_j(\mathbf{x})$ is represented by mixture of densities as Eq. (5.2.1), the summand in Eq. (5.1.1) over all S is, in fact, the joint density $f(\mathbf{X},S|\lambda)$, which can be expressed as

$$\begin{aligned} f(\mathbf{X},S|\lambda) &= \prod_{t=1}^{T} a_{s_{t-1}s_t} b_{s_t}(\mathbf{x}_t) \\ &= \sum_{k_1=1}^{M}\sum_{k_2=1}^{M}\cdots\sum_{k_T=1}^{M}[\prod_{t=1}^{T} a_{s_{t-1}s_t} b_{s_tk_t}(\mathbf{x}_t)]^* c_{s_1k_1}c_{s_2k_2}\cdots c_{s_Tk_T} \end{aligned} \tag{5.3.1}$$

Let Ω^T be Tth Cartesian product of $\Omega=\{1,2,...M\}$, which is defined as the branch alphabet. In the summand of Eq. (5.3.1), it can be considered as the summation of:

$$f(\mathbf{X},S,K|\lambda)=\prod_{t=1}^{T} a_{s_{t-1}s_t} b_{s_tk_t}(\mathbf{x}_t)c_{s_tk_t} \tag{5.3.2}$$

Therefore, the joint probability density of the truncated stochastic process $\mathbf{X}$ is

$$f(\mathbf{X}|\lambda)=\sum_{S}\sum_{K\in\Omega^T} f(\mathbf{X},S,K|\lambda) \tag{5.3.3}$$

Eq. (5.3.3) can be interpreted such that there are N^T possible stochastic state sequences that may lead to the observation $\mathbf{X}$, with each possible state sequence being a superposition of M^T branch layers.

Similar to Eq. (5.1.6), an auxiliary function $Q(\lambda,\bar{\lambda})$ of two model points, λ and $\bar{\lambda}$, given an observation $\mathbf{X}$ can be written as:

$$Q(\lambda,\bar{\lambda})=\sum_{S}\sum_{K\in\Omega^T}\frac{f(\mathbf{X},S,K|\lambda)}{f(\mathbf{X}|\lambda)}\log f(\mathbf{X},S,K|\bar{\lambda}) \tag{5.3.4}$$

here, only strict log-concave densities will be considered for simplicity. Similarly the Q-function can be redefined extending to the case of elliptically symmetric density functions.

From Eq.(5.3.2), the following decomposition can be shown:

$$\begin{aligned}\log f(\mathbf{X},S,K|\bar{\lambda}) &= \sum_{t=1}^{T}\log\bar{a}_{s_{t-1}s_t}+\sum_{t=1}^{T}\log\bar{b}_{s_tk_t}(\mathbf{x}_t)+\sum_{t=1}^{T}\log\bar{c}_{s_tk_t} \\ &= \log\bar{\pi}_{s_1}+\sum_{t=1}^{T-1}\log\bar{a}_{s_ts_{t+1}}+\sum_{t=1}^{T}\log\bar{b}_{s_tk_t}(\mathbf{x}_t)+\sum_{t=1}^{T}\log\bar{c}_{s_tk_t}\end{aligned} \tag{5.3.5}$$

Maximisation of the likelihood by way of reestimation can be accomplished on individual parameter sets due to the separability shown in Eq. (5.3.5). The separation of Eq. (5.3.5) is the key to the increased versatility of a reestimation algorithm in accommodating mixture observation densities. The auxiliary function can be rewritten in a separated form:

$$\begin{aligned}Q(\lambda,\bar{\lambda}) &= \sum_S\sum_K\frac{f(\mathbf{X},S,K|\lambda)}{f(\mathbf{X}|\lambda)}\log f(\mathbf{X},S,K|\bar{\lambda}) \\ &= \sum_S\sum_K\frac{f(\mathbf{X},S,K|\lambda)}{f(\mathbf{X}|\lambda)}[\log\bar{\pi}_{s_1}+\sum_{t=1}^{T-1}\log\bar{a}_{s_ts_{t+1}}+ \\ &\qquad +\sum_{t=1}^{T}\log\bar{b}_{s_tk_t}(\mathbf{x}_t)+\sum_{t=1}^{T}\log\bar{c}_{s_tk_t}] \\ &= Q_\pi(\lambda,\bar{\pi})+\sum_i Q_{a_i}(\lambda,\bar{a}_{ij})+\sum_j\sum_{k=1}^{M}Q_b(\lambda,\bar{\mathbf{b}}_{jk})+ \\ &\qquad +\sum_j Q_{c_j}(\lambda,\bar{c}_{jk}),\end{aligned} \tag{5.3.6}$$

where

$$\begin{aligned}Q_\pi(\lambda,\bar{\pi}) &= \sum_S\sum_K f(S,K|\mathbf{X},\lambda)\log\bar{\pi}_{s_1} \\ &= \sum_i\sum_K f(s_1=i,\,K|\mathbf{X},\lambda)\log\bar{\pi}_i\end{aligned} \tag{5.3.7}$$

$$Q_{a_i}(\lambda,\bar{a}_{ij}) = \sum_j\sum_{t=1}^{T-1}\sum_K f(s_t=i,s_{t+1}=j,K|\mathbf{X},\lambda)\log\bar{a}_{ij} \tag{5.3.8}$$

$$Q_b(\lambda,\bar{\mathbf{b}}_{jk}) = \sum_{t=1}^{T} f(s_t=j,k_t=k|\mathbf{X},\lambda)\log\bar{b}_{jk}(\mathbf{x}_t), \tag{5.3.9}$$

and

$$Q_{c_j}(\lambda,\bar{c}_{jk}) = \sum_{k=1}^{M}\sum_{t=1}^{T} f(s_t=j,k_t=k|\mathbf{X},\lambda)\log\bar{c}_{jk} \tag{5.3.10}$$

Formally, individual maximisation of Q_π, Q_{a_i} (for all i) Q_{c_j} (for all j) subject to the stochastic constraints respectively is the maximisation of individual functions which have the same form as the discussed in the previous Sections. Thus, a global maximum at the single point can be obtained in a similar manner. From discussion in Section 5.1,

when $b_{jk}(\mathbf{x}_t)$ is strictly log concave or elliptically symmetric, Q_{b_j} has a unique global maximum that is a critical point of Q_{b_j} (for all j).

The reestimates that for fixed λ maximise Q_π, Q_{a_i}, and Q_{c_j}, as a function of $\bar{\pi}_i$, $\bar{a}_{ij}$, and $\bar{c}_{jk}$, respectively can be calculated as:

$$\bar{\pi}_i = \frac{\sum_K \frac{f(\mathbf{X}, s_1 = i, K|\lambda)}{f(\mathbf{X}|\lambda)}}{\sum_K \frac{f(\mathbf{X}, K|\lambda)}{f(\mathbf{X}|\lambda)}} = \frac{f(\mathbf{X}, s_1 = i, K|\lambda)}{f(\mathbf{X}|\lambda)} \tag{5.3.11}$$

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \sum_K \frac{f(\mathbf{X}, s_t = i, s_{t+1} = j, K|\lambda)}{f(\mathbf{X}|\lambda)}}{\sum_{t=1}^{T-1} \sum_K \frac{f(\mathbf{X}, s_t = i, K|\lambda)}{f(\mathbf{X}|\lambda)}} = \frac{\sum_{t=1}^{T-1} \frac{f(\mathbf{X}, s_t = i, s_{t+1} = j|\lambda)}{f(\mathbf{X}|\lambda)}}{\sum_{t=1}^{T-1} \frac{f(\mathbf{X}, s_t = i|\lambda)}{f(\mathbf{X}|\lambda)}} \tag{5.3.12}$$

$$c_{jk} = \frac{\sum_{t=1}^{T} \frac{f(\mathbf{X}, s_t = j, k_t = k|\lambda)}{f(\mathbf{X}|\lambda)}}{\sum_{t=1}^{T} \frac{f(\mathbf{X}, s_t = j|\lambda)}{f(\mathbf{X}|\lambda)}} \tag{5.3.13}$$

The intermediate probability density function $\gamma_t(i,j)$, $\gamma_t(i)$ and $\zeta_t(i,k)$ can be defined as:

$$\begin{aligned} \gamma_t(i,j) &= f(s_t = i, s_{t+1} = j|\mathbf{X}, \lambda) \\ &= \frac{f(\mathbf{X}, s_t = i, s_{t+1} = j|\lambda)}{f(\mathbf{X}|\lambda)} \\ &= \frac{\alpha_t(i) a_{ij} [\sum_{k=1}^{M} c_{jk} b_{jk}(\mathbf{x}_{t+1})] \beta_{t+1}(j)}{\sum_{k \in S_F} \alpha_T(k)} \quad \text{for } 1 \le t \le T-1 \end{aligned} \tag{5.3.14}$$

$$\begin{aligned}\gamma_t(i) &= f(s_t=i|\mathbf{X},\lambda)\\ &= \frac{f(\mathbf{X},s_t=i|\lambda)}{f(\mathbf{X}|\lambda)}\\ &= \frac{\alpha_t(i)\beta_t(i)}{\sum_{k\in S_F}\alpha_T(k)} \quad \text{for } 1\le t\le T\end{aligned} \tag{5.3.15}$$

and

$$\begin{aligned}\zeta_t(j,k) &= f(s_t=j,k_t=k|\mathbf{X},\lambda)\\ &= \frac{f(\mathbf{X},s_t=j,k_t=k|\lambda)}{f(\mathbf{X}|\lambda)}\\ &= \frac{\sum_i \alpha_{t-1}(i)a_{ij}c_{jk}b_{jk}(\mathbf{x}_t)\beta_t(j)}{\sum_{i\in S_F}\alpha_T(i)} \quad \text{for } 1\le t\le T\end{aligned} \tag{5.3.16}$$

As a result, the new estimate of initial probabilities, π_i, and transition probabilities, $\bar{a}_{ij}$, can be expressed in a similar way to Eq. (4.3.15) and (4.3.13). The only difference to Eq.(4.3.15) and (4.3.13) is that $b_j(O_{t+1})$ in Eq. (4.3.15) and (4.3.13) are replaced by the continuous mixture probability density function of observation vector $\mathbf{x}$. Similarly, for the weighting coefficients, c_{ik} can be written as:

$$\bar{c}_{jk}=\frac{\sum_{t=1}^{T}\zeta_t(j,k)}{\sum_{t=1}^{T}\gamma_t(j)} \tag{5.3.17}$$

Maximisation of $Q_b(\lambda,\bar{\mathbf{b}}_{jk})$ with respect to $\bar{\mathbf{b}}_{jk}$ is a well known method for many familiar density functions. The solution to the maximisation problem is, in general, obtained through differentiation; i.e., find $\{\bar{\mathbf{b}}_{jk}\}$ that satisfies:

$$\begin{aligned}&\nabla_{\bar{\mathbf{b}}_{jk}} Q_b(\lambda,\bar{\mathbf{b}}_{jk})\\ &=\sum_{t=1}^{T} f(s_t=j,k_t=k|\mathbf{X},\lambda)\frac{\nabla_{\bar{\mathbf{b}}_{jk}}\bar{b}_{jk}(\mathbf{x}_t)}{\bar{b}_{jk}(\mathbf{x}_t)}=0\end{aligned} \tag{5.3.18}$$

where $\nabla_{\bar{\mathbf{b}}_{jk}}\bar{b}_{jk}(\mathbf{x}_t)$ denotes derivatives with respect to parameters of $\bar{b}_{jk}$. Thus, for the Gaussian mixture density functions represented in Eq. (5.2.1), the partial derivatives are with respect to $\bar{\mu}_{jk}$ and $\bar{\Sigma}_{jk}^{-1}$, and they are:

$$\begin{aligned}\frac{\partial \bar{b}_{jk}(\mathbf{x})}{\partial \bar{\mu}_{jk}} &= N(\mathbf{x},\bar{\mu}_{jk},\bar{\Sigma}_{jk})\bar{\Sigma}_{jk}^{-1}(\mathbf{x}-\bar{\mu}_{jk}) \\ \frac{\partial \bar{b}_{jk}(\mathbf{x})}{\partial \bar{\Sigma}_{jk}^{-1}} &= \frac{1}{2}N(\mathbf{x},\bar{\mu}_{jk},\bar{\Sigma}_{jk})[\bar{\Sigma}_{jk}-(\mathbf{x}-\bar{\mu}_{jk})(\mathbf{x}-\bar{\mu}_{jk})^t]\end{aligned} \tag{5.3.19}$$

Substituting Eq. (5.3.19) in Eq. (5.3.18), it can be seen that the solution to Eq. (5.3.19), i.e., reestimates $\bar{\mu}_{jk}$ and $\bar{\Sigma}_{jk}$, can be given by

$$\begin{aligned}\bar{\mu}_{jk} &= \frac{\sum_{t=1}^{T}\frac{f(\mathbf{X},s_t=j,k_t=k|\lambda)}{f(\mathbf{X}|\lambda)}\mathbf{x}_t}{\sum_{t=1}^{T}\frac{f(\mathbf{X},s_t=j,k_t=k|\lambda)}{f(\mathbf{X}|\lambda)}} \\ &= \frac{\sum_{t=1}^{T}\zeta_t(j,k)\mathbf{x}_t}{\sum_{t=1}^{T}\zeta_t(j,k)}\end{aligned} \tag{5.3.20}$$

$$\begin{aligned}\bar{\Sigma}_{jk} &= \frac{\sum_{t=1}^{T}\frac{f(\mathbf{X},s_t=j,k_t=k|\lambda)}{f(\mathbf{X}|\lambda)}(\mathbf{x}_t-\bar{\mu}_{jk})(\mathbf{x}_t-\bar{\mu}_{jk})^t}{\sum_{t=1}^{T}\frac{f(\mathbf{X},s_t=j,k_t=k|\lambda)}{f(\mathbf{X}|\lambda)}} \\ &= \frac{\sum_{t=1}^{T}\zeta_t(j,k)(\mathbf{x}_t-\mu_{jk})(\mathbf{x}_t-\mu_{jk})^t}{\sum_{t=1}^{T}\zeta_t(j,k)}\end{aligned} \tag{5.3.21}$$

The interpretation of Eq. (5.3.17), (5.3.20) and (5.3.21) is similar to that of the discrete HMM. The reestimation of c_{jk} is the ratio between the expected number of times that the kth density in the mixture at state j is used, and the expected number of times of being in state j. The reestimation of $\bar{\mu}_{jk}$ is the ratio between the expected mean of the kth density in state j and the expected number of times of being in state j. Interpretation of Σ can be given in a similar manner. Notice that Eq. (5.3.20) and Eq. (5.3.21) have identical forms to Eq. (3.2.10) and Eq. (3.2.11) respectively except that the Markov property is imposed on the posterior probability here.

5.4. Summary

The original Q-function of Baum and *et al.* is enough to give a proof to the continuous HMM with strict log-concave density functions. The proof can be extended to accommodate a broad class of elliptically symmetric density functions by Liporace's redefined Q-function, although, in practice, strict log-concave density functions have already covered most widely used density functions such as Gaussian. Without loss of generality, finite mixture continuous HMMs have been discussed with the Gaussian density function; it can also be applied to elliptically symmetric density functions.

Although it is possible to quantise any continuous observations via codebook, etc., there might be serious degradation associated with such quantisation. The rationale of the continuous HMM is that the continuous observations can be modelled directly without quantisation. However, the choice of different density functions to model a given observation largely depends on the characteristics of observations. In addition, a single continuous probability density function associated with each state is usually not enough to model complicated observations; and finite mixture models are required. Furthermore, simple-minded implementation of the continuous mixture HMM may not give any improvement at all in comparison to the discrete HMM. A better usage of the continuous mixture HMM will be discussed in the following Chapters.

CHAPTER 6

UNIFIED THEORY WITH SEMI-CONTINUOUS MODELS

The discrete HMM and the continuous mixture HMM have been introduced in previous Chapters. Both the discrete HMM and the continuous mixture HMM have their own advantages and disadvantages. Traditionally, these two models have been treated separately. A general model of these two distinctive models is the semi-continuous HMM, in which VQ, the discrete HMM, and the continuous mixture HMM can be unified. In comparison to the continuous mixture HMM, the semi-continuous HMM can maintain the modelling ability of large-mixture probability density functions. In addition, the number of free parameters and the computational complexity can be reduced because all of the probability density functions are tied together in the codebook. The semi-continuous HMM thus provides a good solution to the conflict between detailed acoustic modelling and insufficient training data. In comparison to the discrete HMM, robustness can be enhanced by using multiple codewords in deriving the semi-continuous output probability; and the VQ codebook itself can be optimised together with the HMM parameters in terms of the maximum likelihood criterion. Such a unified modelling can substantially minimise the information lost by conventional VQ. In practice the speech recognition accuracy of the semi-continuous HMM can be improved in comparison to both continuous mixture HMMs and discrete HMMs. This Chapter will highlight the unified modelling theory concerning VQ, the discrete HMM, and the continuous mixture HMM.

6.1. Discrete HMM vs. Continuous HMM

In the discrete HMM, VQ produces the closest codeword from the codebook for each acoustic observation. This mapping from continuous acoustic space to quantised discrete space may cause serious quantisation errors for subsequent hidden Markov modelling. To reduce VQ errors, various smoothing techniques have been proposed [72,80,96,123,150,162]. A distinctive technique is hidden Markov modelling based on multiple VQ codebooks, which has been shown to offer improved speech recognition accuracy [63,96]. In the multiple VQ codebook approach, VQ distortion can be significantly minimised by partitioning the parameters into separate codebooks. Another disadvantage of the discrete HMM is that the VQ codebook and the discrete HMM are separately modelled, which may not be an optimal combination for pattern classification [73]. The discrete HMM which uses discrete output probability distributions to model various acoustic events is inherently superior to the continuous mixture HMM with a mixture of a small number of probability density functions since the discrete distributions could model events with any distribution provided enough training data exist.

On the other hand, the continuous mixture HMM models the acoustic observation directly using estimated continuous probability density functions without VQ, and has been shown to improve the recognition accuracy in comparison to the discrete HMM [26,132,138]. For speaker-independent speech recognition, a mixture of a large number of probability density functions [128,140] or a large number of states in the single-mixture case [43] are generally required to model the characteristics of different speakers. In addition, it has also been observed that some difficulties due to modelling assumptions that introduce singularities may arise in the continuous single mixture HMM [116], and a mixture of densities are generally needed. However, mixtures of a large number of probability density functions will considerably increase not only the computational complexity, but also the number of free parameters that require to be reliably estimated. In addition, the continuous mixture HMM has to be used with care as continuous probability density functions make more assumptions than the discrete HMM, especially when the diagonal covariance Gaussian probability density is used for

simplicity [26,138]. To obtain better recognition accuracy, acoustic parameters must be well chosen according to the assumption of the continuous probability density functions used.

The fact that maximum mutual information parameter estimation [26] has been shown to significantly improve the performance of the maximum likelihood parameter estimation for the continuous HMM but not for the discrete HMM, can be considered as an indication that

(1) the effect of the VQ errors is an important factor in the discrete HMM; and

(2) selection of appropriate probability density function is an important factor for the continuous HMM.

As far as the computational complexity is concerned, in the discrete HMM, the VQ computation depends on the codebook size and distortion measure; computing the discrete output probability of an observation is then a table-lookup. On the other hand, in the continuous model, many multiplication operations are required even when using the simplest single-mixture, multivariate Gaussian density with a diagonal covariance matrix because the total number of density functions being matched is usually quite large.

6.2. Semi-Continuous Hidden Markov Model

In the discrete HMM, the discrete probability distributions are sufficiently powerful to model any random events with a reasonable number of parameters. The major problem with the discrete output probability is that the VQ operation partitions the acoustic space into separate regions according to some distortion measure. This introduces errors since the partition operations may destroy the original signal structure. To overcome this limitation, the VQ codebook can be modelled as a family of finite mixture probability density functions such that the distributions are overlapped, rather than partitioned. Each codeword of the codebook can then be represented by one of the probability density functions (say, Gaussian) and may be used together with

others to model the acoustic event. The use of a parametric family of finite mixture densities (a mixture density VQ) can then be closely combined with the HMM methodology. From the continuous mixture HMM point of view, the continuous probability density functions in the continuous mixture HMM are tied into the VQ codebook, where each codeword is represented as a continuous probability density function. Such a tying can reduce the number of free parameters to be estimated as well as the computational complexity. From the discrete HMM point of view, the partition of the VQ is unnecessary, and is replaced by the mixture density modelling with overlap, which can effectively minimise the VQ errors. Thus, the VQ problems and HMM modelling problems can be unified under the same probabilistic framework to obtain an optimised VQ/HMM combination, which forms the foundation of the semi-continuous HMM.

Provided that each codeword of the VQ codebook is represented by a continuous probability density function, for a given state s_t of the HMM, the probability density function that produces a vector $\mathbf{x}$ can then be written as:

$$\begin{aligned} b_{s_t}(\mathbf{x}) &= f(\mathbf{x}|s_t) \\ &= \sum_{j=1}^{L} f(\mathbf{x}|v_j,s_t)Pr(v_j|s_t) \end{aligned} \tag{6.2.1}$$

where L denotes the VQ codebook level. For the sake of simplicity, the probability density function, $f(\mathbf{x}|v_j,s_t)$, can be assumed to be independent of the Markov states s_t. Thus, for a given state i, Eq. (6.2.1) can be written as:

$$\begin{aligned} b_i(\mathbf{x}) &= \sum_{j=1}^{L} f(\mathbf{x}|v_j)Pr(v_j|s_t=i) \\ &= \sum_{j=1}^{L} f(\mathbf{x}|v_j)b_i(j) \end{aligned} \tag{6.2.2}$$

This equation is the key to semi-continuous hidden Markov modelling. The estimation of $f(\mathbf{x}|v_j)$ is crucial in the system design. In fact, Eq. (6.2.2) works in a similar way to other non-parametric or heuristic methods, such as the HMM based on the Parzen estimator [159], the fuzzy VQ [162], and the multi-labeling VQ [123]. However, the representation using a continuous probability density function can be more conveniently extended into the unified modelling framework than other heuristic techniques.

The central concept in semi-continuous hidden Markov modelling can be depicted in Figure 6.2.1. The VQ codebook consists of a mixture of continuous probability density functions (for example, each codeword may be represented by a mean vector and a covariance matrix). Conventional VQ operation produces a codeword index which has minimum distortion to the given observation **x**. In the semi-continuous HMM, VQ operation produces values of continuous probability density functions $f(\mathbf{x}|v_j)$ for all the codewords v_j $(1 \leq j \leq L)$. These codebook density functions are subsequently used by the semi-continuous output probability (Eq. (6.2.2)). The structure of the semi-continuous HMM can be exactly the same as the discrete HMM. However, the output probabilities (as for the discrete HMM) in the semi-continuous HMM are not used directly as in the discrete HMM. In contrast, the VQ codebook density functions, $f(\mathbf{x}|v_j)$, are combined with the discrete output probability as Eq. (6.2.2) to form a *semi-continuous* output density function dynamically. The semi-continuous output probability is thus a combination of *discrete* model-dependent weighting coefficients with these *continuous* VQ codebook probability density functions. Such a representation can be used either as a feedback to the VQ codebook to reestimate the original VQ codebook together with the HMM parameters, or for semi-continuous decoding. Note that each discrete output probability is weighted by the continuous conditional probability density function derived from VQ. If these continuous VQ density functions are considered as the continuous probability density functions in the continuous mixture HMM, this also resembles the L-mixture HMM with all the continuous output probability density functions shared with each other in the VQ codebook, and the discrete output probabilities in state i, $b_i(j)$, become the weighting coefficients for the mixture components. In comparison to the continuous mixture HMM, the semi-continuous HMM can maintain the modelling ability of large-mixture probability density functions. In addition, the number of free parameters and the computational complexity can be reduced because all of the probability density functions are tied together in the codebook. The semi-continuous HMM thus provides a good solution to the conflict between detailed acoustic modelling and insufficient training data. In comparison to the standard discrete HMM, robustness can be enhanced by using multiple codewords in deriving the semi-continuous output probability in a similar manner to fuzzy VQ and multi-labeling VQ. However, the VQ codebook itself can be optimised here together with

the HMM parameters in terms of the maximum likelihood criterion. Such a unified modelling provides an elegant way to minimise the information lost by conventional VQ, which cannot be obtained from fuzzy VQ or multi-labeling VQ.

In practice, Eq. (6.2.2) can be simplified with M most significant values of $f(\mathbf{x}|v_j)$ for each $\mathbf{x}$ without affecting the performance. Experience has shown that values in the range of 2-8 are adequate. This can be conveniently obtained during the VQ operations by sorting the VQ output and keep the M most significant values. Let $\eta(\mathbf{x})$ denote the set of VQ codewords, v_j, for those most significant values of $f(\mathbf{x}|v_j)$ of $\mathbf{x}$. Then Eq. (6.2.2) can be re-written as

$$b_i(\mathbf{x}) = \sum_{v_j \in \eta(\mathbf{x})} f(\mathbf{x}|v_j) b_i(j) \tag{6.2.3}$$

Since the number of VQ codewords in $\eta(\mathbf{x})$ is of lower order than the VQ level, Eq. (6.2.3) can significantly reduce the amount of computational load for subsequent modelling in comparison to Eq. (6.2.2). In the semi-continuous HMM, most of the computational load lies in the calculation of the continuous VQ density function. The computational complexity of the semi-continuous HMM mainly depends on the VQ level and the size of $\eta(\mathbf{x})$.

The semi-continuous output probability represented in Eq. (6.2.3) also bridges the gap between the continuous mixture HMM and the discrete HMM. If $\eta(\mathbf{x})$ contains only the most significant $f(\mathbf{x}|v_j)$ (i.e, only the closest codeword to $\mathbf{x}$), the semi-continuous HMM degenerates to the discrete HMM with a probability density codebook. On the other hand, a large VQ codebook can be used such that each state of the HMM contains a codeword (a probability density function). The discrete output probability $b_i(j)$ in state i can be defined as

$$b_i(j) = \begin{cases} 1 & \text{if } codeword\ j \in state\ i \\ 0 & otherwise \end{cases} \tag{6.2.4}$$

Therefore, each state has its own codeword, i.e., a single probability density function. The only term contributing to the semi-continuous output probability at state i will be the codeword from the state itself. This is the case for single-mixture continuous hidden Markov modelling. Of course, Eq. (6.2.4) can be modified to let each state contain several disjoint codewords extending to the case of the continuous mixture HMM. From

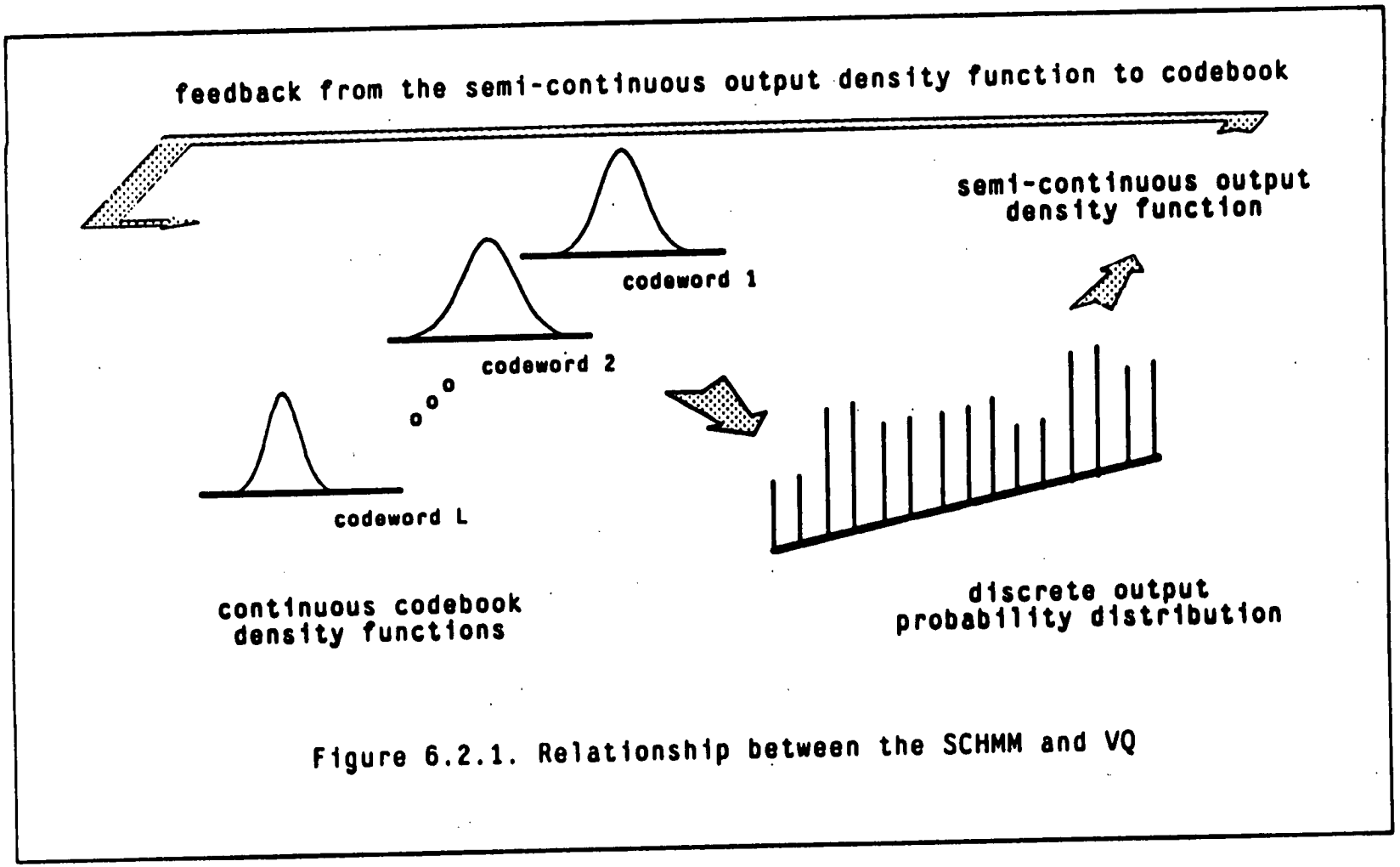


Figure 6.2.1. Relationship between the SCHMM and VQ

above discussion, it can be seen that the semi-continuous HMM is more flexible and more general than either the discrete HMM or the continuous mixture HMM. The conventional HMM can be considered as a special case of the semi-continuous HMM.

If the Gaussian density function is considered here (other probability density functions can be applied here in a similar manner as those described in Chapter 5), given the VQ codebook v_j, the probability density function $f(\mathbf{x}|v_j)$ can be estimated with one of the following:

(1) the EM algorithm as described in Chapter 3;

(2) Sample estimates of the covariance matrices based on the conventional codebook as will be shown in Chapter 8 [72];

(3) Gaussian clustering techniques as will be shown in Chapter 8 [70];

(4) Feedback from semi-continuous hidden Markov modelling [71,75].

Detailed feedback from semi-continuous hidden Markov modelling, namely unified modelling of VQ and HMM, will be discussed in the following Sections.

6.3. Reestimation Formulas

Feedback from hidden Markov modelling to the VQ codebook is a reestimation problem in a manner similar to the Baum-Welch algorithm, as used for both the discrete HMM and the continuous mixture HMM. Here, only the Gaussian density function is considered due to its advantages. Other probability density functions can be applied in principle in a similar way to those described in Chapter 5.

In the semi-continuous HMM, if the $b_i(k)$ are considered as the weighting coefficients (c_{jk}) to mixture probability density functions in the continuous mixture HMM, the reestimation algorithm for the weighting coefficients (as discussed in Chapter 5) can be applied to reestimate $b_i(k)$. In a manner similar to the conventional HMM, reestimation formulations can be more readily computed by defining a forward probability, $\alpha_t(i)$, and a backward probability, $\beta_t(i)$ for any time t and state i except

that Eq. (6.2.2) or Eq. (6.2.3) is used as output probabilities. Further to the forward and backward probabilities, the intermediate probabilities, $\chi_t(i,j,k)$, $\gamma_t(i,j)$, $\gamma_t(i)$, $\zeta_t(i,j)$, and $\zeta_t(j)$ can be defined as follows for efficient reestimation of the model parameters:

$$\begin{aligned}\chi_t(i,j,k) &= f(s_t=i,\ s_{t+1}=j,\ \mathbf{x}_{t+1}\sim v_k|\mathbf{X},\ \lambda) \\ &= \frac{\alpha_t(i)a_{ij}b_j(k)f(\mathbf{x}_{t+1}|v_k)\beta_{t+1}(j)}{f(\mathbf{X}|\lambda)}, \quad 1\le t\le T-1\end{aligned} \tag{6.3.1}$$

$$\gamma_t(i,j) = f(s_t=i,\ s_{t+1}=j|\mathbf{X},\ \lambda), \quad 1\le t\le T-1 \tag{6.3.2}$$

$$\begin{aligned}\gamma_t(i) &= f(s_t=i|\mathbf{X},\ \lambda) \\ &= \frac{\alpha_t(i)\beta_t(i)}{\sum_{k\in S_F}\alpha_T(k)}, \quad 1\le t\le T\end{aligned} \tag{6.3.3}$$

$$\zeta_t(i,k) = f(s_t=i,\mathbf{x}_t\sim v_k|\mathbf{X},\ \lambda), \quad 1\le t\le T \tag{6.3.4}$$

$$\zeta_t(k) = f(\mathbf{x}_t\sim v_k|\mathbf{X},\ \lambda), \quad 1\le t\le T. \tag{6.3.5}$$

Here, $\mathbf{x}_t\sim v_k$ means that $\mathbf{x}_t$ is quantised to v_k. All these intermediate probabilities can be represented by $\chi_t()$ as

$$\gamma_t(i,j) = \sum_{k=1}^{L}\chi_t(i,j,k), \quad 1\le t\le T-1 \tag{6.3.6}$$

$$\gamma_t(i) = \sum_j\gamma_t(i,j), \quad 1\le t\le T-1 \tag{6.3.7}$$

In Eq. (6.3.7), $\gamma_T(i)$ must be computed from Eq. (6.3.3).

$$\zeta_t(i,k) = \begin{cases} \sum_j \chi_{t-1}(j,i,k) & \text{if } 1<t\le T \\ \dfrac{\pi_i b_i(k)f(\mathbf{x}_1|v_k)\beta_1(i)}{f(\mathbf{X}|\lambda)} & t=1 \end{cases} \tag{6.3.8}$$

$$\zeta_t(k) = \sum_i\zeta_t(i,k), \quad 1\le t\le T \tag{6.3.9}$$

Using Eq. (6.3.6) to (6.3.9), reestimation equations for $\bar{\pi}_i$, $\bar{a}_{ij}$, and $\bar{b}_i(k)$, can be derived (detailed proof will be presented in Section 6.5.). In fact, the reestimation formulas have the same form as the continuous mixture HMM as:

$$\bar{\pi}_i = \gamma_1(i); \tag{6.3.10}$$

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \gamma_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}; \tag{6.3.11}$$

$$\bar{b}_j(k) = \frac{\sum_{t=1}^{T} \zeta_t(j,k)}{\sum_{t=1}^{T} \gamma_t(j)}. \tag{6.3.12}$$

The feedback from the HMM estimation results to the VQ codebook implies that the VQ codebook is optimised based on the HMM likelihood maximisation rather than minimising the total distortion errors from the set of training data. To reestimate the parameters of the VQ codebook, i.e. the mean vectors, μ_j, and covariance matrices, Σ_j, of the codeword v_j, it can be written as:

$$\bar{\mu}_j = \frac{\sum_{t=1}^{T} \zeta_t(j)\mathbf{x}_t}{\sum_{t=1}^{T} \zeta_t(j)}, \quad 1 \le j \le L; \tag{6.3.13}$$

$$\bar{\Sigma}_j = \frac{\sum_{t=1}^{T} \zeta_t(j)(\mathbf{x}_t - \bar{\mu}_j)(\mathbf{x}_t - \bar{\mu}_j)^t}{\sum_{t=1}^{T} \zeta_t(j)}, \quad 1 \le j \le L. \tag{6.3.14}$$

As discussed in Chapter 4, Eq. (6.3.10) to (6.3.14) can be extended to multiple independent observation sequences by summing numerators and denominators respectively corresponding to each observation sequence. Reestimation formulas for transition and output probabilities are, here, formally, the same as for conventional ones. However, to reestimate parameters of the VQ codebook, observations for different models will be used together since different models use the same VQ codebook. Thus, reestimation of mean vectors and covariance matrices of different models will involve inter-dependencies. According to Eq. (6.3.13) and (6.3.14), if any observation $\mathbf{x}_t$, (no matter what model it is designated for), has a large value of posterior probability $\zeta_t(j)$, it will have a large contribution on reestimation of parameters of codeword v_j. Different VQ density functions to be reestimated in such a manner will be strongly correlated due to large values of $\zeta_t(j)$. Let F denote the set of models to be reestimated. There are

multiple independent observation sequences for each model. In general, reestimation formulas for the VQ codebook can be written as:

$$\bar{\mu}_j = \frac{\sum_{m\in F}[\sum_{n}\sum_{t=1}^{T}\zeta_t^{m,n}(j)\mathbf{x}_t^{m,n}]}{\sum_{m\in F}[\sum_{n}\sum_{t=1}^{T}\zeta_t^{m,n}(j)]}, \quad 1\le j\le L; \tag{6.3.15}$$

and

$$\bar{\Sigma}_j = \frac{\sum_{m\in F}[\sum_{n}\sum_{t=1}^{T}\zeta_t^{m,n}(j)(\mathbf{x}_t^{m,n}-\bar{\mu}_j)(\mathbf{x}_t^{m,n}-\bar{\mu}_j)^t]}{\sum_{m\in F}[\sum_{n}\sum_{t=1}^{T}\zeta_t^{m,n}(j)]}, \quad 1\le j\le L. \tag{6.3.16}$$

where variables in [ ] are those designated for model m, and multiple independent observation sequences for model m are denoted by summation with respect to n. Here $\gamma_t^{m,n}(j)$ is computed from observation sequence n of model m.

In Eq. (6.3.15) and (6.3.16), reestimation formulas for the mean vectors and covariance matrices of the VQ codebook can be regarded as reestimation formulas for the output densities of the continuous mixture HMM in which all the output densities are tied up with different models across the inventory of modelling units. In comparison to Eq. (3.2.10) to (3.2.11), it can be observed that the EM algorithm for the mixture density VQ codebook design is merely a special form of these reestimation formulas. Here, a unified modelling approach to VQ and hidden Markov modelling of speech signals is established. In the EM algorithm, the posterior probability density is used as the weight in the reestimation formulas of the VQ codebook. In the unified modelling approach, the weight is also the posterior probability density, but the Markov properties associated with each model are imposed. Therefore, optimisation of hidden Markov modelling directly leads to optimisation of the the VQ codebook. This is quite different from conventional VQ, which minimises the overall average distortion without consideration of subsequent modelling at all.

If Eq. (6.2.3) is used as the semi-continuous output probability densities, the reestimation computational complexity can also be significantly reduced. With such a simplification, Eq. (6.3.1) can be written as:

$$\chi_t(i,j,k) = \begin{cases} \dfrac{\alpha_t(i)a_{ij}b_j(k)f(\mathbf{x}_{t+1}|v_k)\beta_{t+1}(j)}{f(\mathbf{X}|\lambda)} & \text{if } v_k \in \eta(\mathbf{x}_{t+1}) \\ 0 & \textit{otherwise} \end{cases} \tag{6.3.17}$$

Similarly, Eq. (6.3.2) to (6.3.5) can be treated in the same manner by using Eq. (6.3.6) to (6.3.9).

6.4. Semi-Continuous Decoder

No matter if the models are reestimated based on the discrete HMM or the semi-continuous HMM, the forward-backward algorithm or the Viterbi algorithm can be modified by replacing the discrete output probability distribution with the the semi-continuous output probability density function. For example, the Viterbi algorithm can be modified as follows:

Modified Viterbi Algorithm

Step1: Initialisation, for all states i,

$$\delta_1(i) = \pi_i \sum_{v_l \in \eta(\mathbf{x}_1)} [f(\mathbf{x}_1|v_l)b_i(v_l)]$$

$$\Psi_1(i) = 0.$$

Step 2: Recursion, from time at $t=2$ to T, for all states j,

$$\delta_t(j) = \underset{i}{Max}[\delta_{t-1}(i)a_{ij}] \sum_{v_l \in \eta(\mathbf{x}_t)} [f(\mathbf{x}_t|v_l)b_j(v_l)]$$

$$\Psi_t(j) = \underset{i}{argmax}[\delta_{t-1}(i)a_{ij}]$$

Step 3: Termination

$$P^* = \underset{s \in S_F}{Max}[\delta_T(s)]$$

$$s_T^* = \underset{s \in S_F}{argmax}[\delta_T(s)]$$

Step4: Path (state sequence) backtracking, from time at T-1 to 1

$$s_t^* = \Psi_{t+1}(s_{t+1}^*)$$

If the discrete HMM parameters are used by the semi-continuous decoder, the continuous density function of the VQ codebook, $f(\mathbf{x}|v_j)$, are empirically required to be normalised within [0, 1] as to retain consistency with discrete probabilities. The normalisation of $f(\mathbf{x}|v_j)$ can be done with

$$f(\mathbf{x}|v_j) \leftarrow \frac{f(\mathbf{x}|v_j)}{\sum_{k=1}^{L} f(\mathbf{x}|v_k)} \tag{6.4.1}$$

or

$$f(\mathbf{x}|v_j) \leftarrow \frac{f(\mathbf{x}|v_j)}{\sum_{v_k \in \eta(\mathbf{x})} f(\mathbf{x}|v_k)} \tag{6.4.2}$$

Experiments show that both Eq. (6.4.1) and Eq. (6.4.2) work well if discrete HMM parameters are used. However, if the model is reestimated based on the semi-continuous HMM, such a normalisation is unnecessary since training and decoding can be done in a consistent way.

6.5. Proof of the Reestimation Formulas

To reestimate the parameters of the VQ codebook, i.e. the mean vectors, μ_j, and covariance matrices, Σ_j, of the codebook v_j, all the training data for the different HMMs should initially be collected as for conventional VQ. As different HMMs for different speech units can be assumed to be independent, the likelihood to be maximised in the unified modelling approach should then be the summation of each individual likelihood of the HMM as:

$$\sum_{m \in F} \sum_{n} \log f(\mathbf{X}^{m,n}|\lambda^m) \tag{6.5.1}$$

where $\mathbf{X}^{m,n}$ denotes the observation sequence n for model m; λ^m denotes the parameters of the mth HMM. Let V now denote the VQ codeword set and assume that all the HMMs have the same structure. Following the concept of the Kullback-Leibler

statistics, the Q-function can be modified as:

$$Q(\Lambda,\bar{\Lambda}) = \sum_{m\in F}\sum_{n}\sum_{S}\sum_{V}\frac{f(\mathbf{X}^{m,n},S,V|\lambda^m)}{f(\mathbf{X}^{m,n}|\lambda^m)}\log f(\mathbf{X}^{m,n},S,V|\bar{\lambda}^m) \tag{6.5.2}$$

where Λ and $\bar{\Lambda}$ denote $\{\lambda^m\}$ and $\{\bar{\lambda}^m\}$ respectively. The Q-function defined in Eq. (6.5.2) can be well separated into Q_π^m, Q_a^m, Q_b^m, and Q_v in a manner similar to Eq. (5.3.6). Q_π^m, Q_a^m, and Q_b^m have the same form as Q_π, Q_a, and Q_c for the mth continuous mixture HMM as discussed in Chapter 5. They can be maximised independently for each m. However, for reestimation of mean vectors and covariance matrices, the Q-function Q_v involves the summation of different models. If the intermediate probability density function, $\zeta_t^{m,n}(j)$, is defined as following:

$$\zeta_t^{m,n}(j) = f(\mathbf{x}_t^{m,n}\sim v_j|\mathbf{X}^{m,n},\lambda^m) \tag{6.5.3}$$

Q_v can be written as:

$$Q_v(\Lambda,\bar{v}_j) = \sum_{m\in F}\sum_{n}\sum_{t=1}^{T} f(\mathbf{x}_t^{m,n}\sim v_j|\mathbf{X}^{m,n},\lambda^m)\log f(\mathbf{x}_t^{m,n}|\bar{v}_j) \tag{6.5.4}$$

Notice the similarity of Eq. (6.5.4) and the Q-function defined for the continuous mixture HMM (Eq. (5.3.9)). If the VQ codeword density is assumed to be Gaussian, it is not difficult to extend the reestimation formulas of Eq. (5.3.20) and (5.3.21) to Eq. (6.3.15) and (6.3.16). In general, a broad class of elliptically symmetric probability density functions can also be accommodated in a similar manner to those discussed in Chapter 5.

6.6. Summary

In comparison to the discrete HMM, robustness of the semi-continuous HMM can be enhanced by using multiple codewords in deriving the semi-continuous output probability in a similar manner to fuzzy VQ and multi-labeling VQ. Unlike fuzzy VQ or multi-labeling, with the semi-continuous HMM, the VQ codebook itself can be adjusted together with the HMM parameters in order to obtain an optimal combination. The unified modelling approach can therefore achieve an optimal combination of HMM and VQ codebook parameters.

The semi-continuous HMM takes the advantages of both the discrete HMM and the continuous mixture HMM, by which it is possible to model a mixture of a large number of probability density functions with a limited amount of training data and computational complexity. Robustness is also enhanced by using multiple codewords in the semi-continuous output probability. From the continuous HMM point of view, the semi-continuous HMM can be considered as a special form of continuous mixture HMM with tied mixture continuous density functions. Because of the binding of the continuous density functions, in the semi-continuous HMM, the number of free parameters and computational complexity are reduced in comparison to the continuous mixture HMM while retaining the modelling power of the continuous HMM with a mixture of a large number of probability density functions. However, it should be pointed out here that the applicability of the continuous mixture HMM or the semi-continuous HMM relies on appropriately chosen acoustic parameters and assumption of the continuous probability density function. Acoustic features must be well represented for the chosen probability density function.

CHAPTER 7

USING HIDDEN MARKOV MODELS FOR SPEECH RECOGNITION

In the previous Chapters, the basic theory of hidden Markov models has been introduced. A successful HMM-based speech recognition relies on availability of a large training database, powerful learning algorithms, and detailed speech models. Access to a large database implies that the given parameters of a speech recognition system can be well-trained, which is essential for statistical modelling. Powerful learning algorithms can extract available information for the purposes of pattern classification based on HMM assumptions. Finally, detailed speech models are essential to model the various uncertainties present in speech. Of course, these factors are inter-dependent. For example, detailed models usually require more parameters, which result in a requirement for more training data to reliably estimate these parameters.

This Chapter will discuss practical issues and various improved modelling techniques related to these problems.

7.1. Representation of speech units

In hidden Markov modelling, one of the most important issues is how to represent speech units. In fact, it is possible to use HMMs to represent any unit of speech. Even if speech units are poorly selected, the HMM has an ability to absorb the suboptimal characteristics within the model parameters. A central philosophy in hidden Markov modelling is that knowledge sources, such as phonetic or syntactic knowledge sources, that can be represented as an HMM should be represented as an HMM. This is because such an HMM representation can be automatically trained from training data, and improved recognition accuracy will normally result if training and decoding are treated under the same framework.

For example, if subword models with a grammar are used, the word and sentence knowledge can be incorporated into the recognition systems by representing each word as a network of subword models which encodes every way in which the word could be pronounced. The grammar can be represented as a network whose transitions are words, and the network can encode all legal sentences. If the subword model is represented as an HMM, a large HMM that encodes all the legal sentences can thus be obtained.

In above described systems, the modelling unit can be phrases, words, syllables, phonemes, and other subword units. The choice of speech units largely relies on the specific task to be carried out by the speech recognition system. For example, if the task is digit recognition, whole-word models will be a natural choice. On the other hand, if the task is large vocabulary speech recognition, subword unit models will have distinctive advantages.

7.1.1. Whole-word models

The most natural unit of speech is the word, and it has been widely used for many speech recognition systems [43,63,104,123,140]. One of the most distinctive advantages to use whole-word models is that these are able to capture within-word phoneme coarticulation effects. When using the whole-word models, many phonological variations can be automatically accommodated and when whole-word models are adequately

trained, they will usually yield the best recognition performance. Therefore, for small vocabulary recognition, whole-word models are widely used.

While words are suitable units for recognition, they are not a practical choice for large vocabulary recognition. Since each word has to be treated individually and data cannot be shared between word models, this implies a prohibitively large amount of training data and storage. In addition, for some task configurations, the recognition vocabulary may consist of words which have not appeared in the training procedure. As a result, some form of word model composition technique is required to generate word models, which do not appear during training.

7.1.2. Subword models

Instead of using whole-word models, various subword models can be used such that data can be shared across words. Such an approach relies on the assumption that a word model can be constructed based on existing linguistic knowledge. Each word can be represented as a lexical item by a concatenation of subword models. Typical subword models include

(1) linguistically defined subword units, such as phone models, diphone models [90,149], word-dependent phone models [30,96,113], and triphone models [9,31];

(2) acoustically defined subword models, such as fenone models [13] and segment models [94]; and

(3) hybrid models such as generalised triphone models [77,96], and allophonic models [8].

Subword modelling can be considered as partitioning of the parameter space into smaller sets that can be adequately trained. Linguistically defined subword models use human specific knowledge for partitioning; acoustically defined subword models use automatic algorithms to explore the acoustic similarities; and hybrid models are a combination using both linguistic and acoustic knowledge.

Phones are the most well-understood subword unit. Since there are only about 50 phones in English, HMMs based on phone models can be adequately trained. These models are also task-independent, and can be trained on one task and tested on another, although performance may possibly deteriorate. As the realisation of a phone is context-sensitive, the phone models are generally inadequate to model coarticulation effects in a given word. This causes the phone-based HMMs to yield lower recognition accuracy than the whole-word-based HMMs [13,95,127].

To model coarticulation effects, the basic requirement is to model phones according to their context as in the case of triphone models. Here, context refers to the immediate left and/or right neighbouring phones. Triphone models take into consideration the left and the right neighbouring phones; if two phones have the same identity but different left or right context, they are considered different triphones. In triphone modelling, both within-word and between-word coarticulation can be taken into account [77]. While triphone models are good for modelling coarticulation effects, there are a very large number of them, which can only be sparsely trained.

Some phones have the same effect on neighbouring phones, but triphone modelling assumes that every triphone context is different. For example, /b/ and /p/ are both labial stops, and have similar effects on the following vowel. If these similar contexts can be identified and merged, the number of models can be reduced, leading to fewer free parameters and models. One approach is to merge perceptually similar contexts using human knowledge [39,40]. A more interesting approach is to automatically identify and merge triphones in similar acoustic realisations, with clustering procedures being used to produce generalised triphone models from triphone models [96]. The clustering distance measures the ratio between the probability that the individual distributions generated the training data and the probability that the combined distribution generated the training data. This is consistent with the maximum-likelihood criterion used in hidden Markov model parameter reestimation.

The same clustering procedure can also be applied to *allophone* models, where various sources of variability, such as articulation variabilities, linguistic variabilities, or speaker variabilities, of phone models can be taken into consideration in the

allophone models [97]. The bottom-up subword clustering process finds a good mapping for each of the allophones to *generalised allophones*. Alternative procedures based on the use of decision trees have also been proposed to generate allophonic models [8].

7.2. Insufficient Training Data Problems and Smoothing

The maximum likelihood estimates of the parameters of a hidden Markov process have been shown to be consistent (converge to the true values as the number of training data $\rightarrow\infty$) [17]. The practical implication of this theorem is that, in training, as many observations as possible are required. However, in reality, only finite training data are available. If the training data are limited, this will result in some of parameters being inadequately trained and the value of these parameters tends to be very small. If these small parameter values characterise the true nature of the speech signal, no problem exists. However, if such small parameter values result from problems with insufficient training data, then classification based on poorly trained models will result in fatal errors. One solution to the problem of insufficient training data is to increase the size of the training data. Often, this has its limitations. A second solution is to reduce the number of free parameters to be reestimated. This also has its limitations because a number of significant parameters are always needed to model physical events. A third possible solution is to interpolate one set of parameter estimates with another set of parameter estimates for which an adequate amount of training data exists. The following Sections will introduce two successful solutions to insufficient training data problems, namely, parameter tying and deleted interpolation.

7.2.1. Tied transition and output probabilities

Parameter tying can reduce the number of free parameters to be estimated thereby partially solving problems arising due to insufficient training data. Note that the concept of tying is directly accommodated in the semi-continuous HMM, where the VQ codebook can be considered as tying of the continuous output densities of the continuous mixture HMM. The fenone models [13] can also be viewed as a kind of tying output

probabilities in word models, where each fenone model represents one VQ codeword. Basically the idea is to set up an equivalence relation between parameters in different models [10]. In this manner the number of free parameters can be reduced, and parameters can be well estimated. This is especially suitable to reestimation of covariance matrices in the continuous mixture HMM [63,126].

Let us take the discrete HMM as example. Both the transition and output probabilities with different states can share the same transition or output probabilities. For transitions, let $\tau(i,j)$ be the set of transitions to which the transition from state i to j is tied; and $\tau(i)$ be the set of states to which the output probabilities are tied. Then $\bar{a}_{ij}$ and $\bar{b}_j(\mathrm{k})$ can be reestimated as follows:

$$\bar{a}_{ij} = \frac{\sum_{i',j' \in \tau(i,j)} \sum_{t=1}^{T-1} \gamma_t(i',j')}{\sum_{i' \in \tau(i,j)} \sum_{t=1}^{T-1} \sum_{k} \gamma_t(i',k)} \tag{7.2.1}$$

$$\begin{aligned} \bar{b}_j(k) &= \frac{\sum_{j' \in \tau(j)} \sum_{t \in O_t = v_k} \gamma_t(j')}{\sum_{j' \in \tau(j)} \sum_{t=1}^{T} \gamma_t(j')} \\ &= \frac{\sum_{j' \in \tau(j)} \sum_{t \in O_t = v_k} \sum_{i} \gamma_t(j',i)}{\sum_{j' \in \tau(j)} \sum_{t=1}^{T} \sum_{i} \gamma_t(j',i)} \end{aligned} \tag{7.2.2}$$

The maximum likelihood property of the Baum-Welch algorithm still holds with these tied probabilities, and this can be proved by modifying the Q-function with the introduction of an equivalence relation $\tau(i,j)$ or $\tau(j)$ in a manner similar to the semi-continuous HMM for the VQ density function. Note here that output probabilities associated with the state (state-dependent HMM) can also be considered as a tied model in which output probabilities are associated with the transition (transition-dependent HMM). This type of HMM has been used extensively [10,96]. In the transition-dependent HMM, if output probabilities associated with each transition to a specific state are tied together, it will be the same as the state-dependent HMM. Eq. (7.2.2) can thus be regarded as a tied transition-dependent HMM, where output probabilities associated with transitions to state j are tied together.

7.2.2. Deleted interpolation

Since a major cause of inaccuracy in HMM is often the extremely small probability values derived as a result of insufficient training data, it is reasonable to impose some constraints on the parameters. One solution is to set a lower limit or *floor* to them. This can be done by setting these parameters that are less than some values to be equal to the floor and re-normalising them in order to meet the stochastic constraints. Such a smoothing technique may not be sufficiently sensitive to distinguish the unlikely output symbols from the impossible ones, and this creates a problem when the models are not well trained and many codewords are not observed. An alternative technique is *deleted interpolation*, which has been used with good results [81,96].

The basic idea of interpolation is to design two models; one of which is the desired model for which estimates may not be robust; the other is a *smaller* model for which the amount of training data is adequate to give relatively robust (but less accurate) estimates; the parameters from these two models can then be interpolated. A smaller model may be chosen by tying one or more sets of parameters from the desired model, or simply from the uniform distribution. For instance, if two output distribution estimates, b_j^1 and b_j^2, are derived from two different estimates, the first one may not be well trained because of insufficient data problems; the second one may have the reduced number of parameters which are relatively well trained or may be a uniform distribution. We would like to use the parameters of the second model to smooth the first one. These two models can be combined into:

$$b_j = \kappa^1 b_j^1 + \kappa^2 b_j^2, \qquad \kappa^1 + \kappa^2 = 1 \tag{7.2.3}$$

where κ^1 represents the weighting of the first model, and κ^2 represents the weighting of the second model. A key issue is the determination of the optimal value of κ, which should be a function of the amount of training data.

Recalling the mixture density problem and Example 3.2.1. discussed in Chapter 3, reestimation of κ has indeed the same form as the mixture density estimation. Thus, it can be done via the EM algorithm. In Eq. (7.2.3), b_j can be regarded as density functions. As b_j^1 has been estimated with the maximum likelihood criterion, if the same training data are used to determine the weighting of the desired model, κ^1, it will be 1,

which is consistent with the same criterion to estimate b_j. Therefore, one solution is to divide the training data into two disjoint parts. Then b_j^1 and b_j^2 can be estimated from one part and κ^1 from the other, while b_j^1 and b_j^2 are held fixed. Such a *deleted* interpolation has more general implication, namely, it weights each distribution according to its ability to predict *unseen* data. In most real situations, it is hoped that the estimated model parameters can be used to predict some as yet unseen observations such that the purpose of recognition can be successfully solved. Intuitively, when b_j^1 is well-trained, it will predict unseen data well, and κ^1 will be generally large.

In general, the training data can be divided into K blocks, and all the blocks except a *deleted* block can be used to estimate κ (model parameters (b_j) are estimated from the deleted block). The values κ are estimated after all possible deletions. According to the discussion in Chapter 3, the maximum likelihood estimates for κ^1 can be written as:

$$\bar{\kappa}^1 = \frac{1}{K}\sum_{i=1}^{K}\frac{\kappa^1 Pr^1(\mathbf{y}_i)}{\kappa^1 Pr^1(\mathbf{y}_i) + (1-\kappa^1)Pr^2(\mathbf{y}_i)} \tag{7.2.4}$$

where $Pr^1(\mathbf{y}_i)$ is the probability of producing all the data in block i using distribution 1, which was trained from all K blocks except block i, i.e., if data are used to estimate any model i, they will be deleted in the κ computation in order to better predict unseen events. The above formulation assumes that the same κ is used for distribution 1 everywhere; it is a form of tied estimate. In practice, it may be desirable to use a different κ for each phone, or a different κ for each distribution. Furthermore, with the above reestimation formula, each iteration of deleted interpolation is as expensive as an iteration of the normal forward-backward algorithm. To reduce computation load, separate counts (expected numbers before Baum-Welch reestimation) for each block during the final iteration of the forward-backward procedure can be kept, and deleted interpolation can be carried out on the *counts* heuristically [96], with $\bar{\kappa}$ being reestimated from one block based on κ parameters estimated from the other block.

The idea of deleted interpolation can be generalised to interpolate more than two distributions, which is the same as the estimation problem of mixture densities. For example, it is often necessary to interpolate discrete output probability distribution with a uniform distribution as Eq. (7.2.3). To improve the performance of a speaker-

independent speech recognition system, we can divide training data into male and female groups, and build models for male and female respectively. However, such a division may result in insufficient training data for each group. Thus, one solution is to smooth the desired model (i.e. male or female model) with an averaging model (i.e. trained from both female and male data), and uniform distribution.

$$b_j = \kappa^1 b_j^1 + \kappa^2 b_j^2 + \kappa^3 b_j^3, \qquad \kappa^1+\kappa^2+\kappa^3 = 1 \tag{7.2.5}$$

Weighting of each model κ^1, κ^2, and κ^3 can then be determined from deleted interpolation according to how well-trained each model is.

7.3. Implementational Issues

This Section will briefly introduce several implementation issues, such as scaling, logarithmic representation, thresholding, initial estimates, and multiple features.

7.3.1. Scaling

In hidden Markov modelling, when the observation sequence length, T, becomes large, both the forward and the backward variables, $\alpha_t()$ and $\beta_t()$, will approach zero in exponential fashion. For sufficiently large T, the dynamic range of the α and β computation will exceed the precision range of essentially any machine. Thus in practice, the number of observations necessary to adequately train a model and/or compute its probability will result in underflow on computer if probabilities are represented directly.

The scaling principle is to multiply $\alpha_t(i)$ and $\beta_t(i)$ by some scaling coefficient so that it remains within the dynamic range of the computer for $1 \le t \le T$. All of these scaling coefficients should be removed at the end of the computation in order to guarantee the accuracy of the Baum-Welch algorithm.

Let $\alpha_t(i)$ be calculated according to Eq. (4.3.6) and then be multiplied by a scaling coefficient, c_t

$$c_t = [\sum_i \alpha_t(i)]^{-1} \tag{7.3.1}$$

so that $\sum_i c_t \alpha_t(i) = 1$ for $1 \leq t \leq T$. $\beta_t(i)$ can also be multiplied by c_t for $1 \leq t \leq T$ and $1 \leq i \leq N$. The recursion involved in computing the forward and backward variables can be scaled at each stage of time t by c_t. Notice that α and β are computed recursively in exponential fashion; therefore, at time t, the total scaled factor applied to the forward variable, $\alpha_t()$ is

$$C_t = \prod_{i=1}^{t} c_i \tag{7.3.2}$$

and the total scaled factor applied to the backward variable, $\beta_t()$ is

$$D_t = \prod_{i=t}^{T} c_i \tag{7.3.3}$$

This is because the individual scaling factors are multiplied together in the forward and backward recursion. Let $\alpha_t'()$ and $\beta_t'()$, and $\gamma_t'()$ denote scaled $\alpha_t()$, $\beta_t()$, and $\gamma_t()$ respectively. Note that

$$\begin{aligned} \sum_{i \in S_F} \alpha_T'(i) &= C_T \sum_{i \in S_F} \alpha_T(i) \\ &= C_T Pr(\mathbf{O}|\lambda) \end{aligned} \tag{7.3.4}$$

The scaled intermediate probability, $\gamma_t'(i,j)$ can then be written as:

$$\begin{aligned} \gamma_t'(i,j) &= \frac{C_t \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j+1) D_{t+1}}{C_T \sum_{i \in S_F} \alpha_T(i)} \\ &= \gamma_t(i,j) \end{aligned} \tag{7.3.5}$$

Thus, the intermediate probabilities can be used in the same way as un-scaled probabilities because the scaling factor C_T is cancelled out in Eq. (7.3.5). Therefore, reestimation formulas can be kept exactly the same as discussed in Chapter 3 except that $Pr(\mathbf{O}|\lambda)$ should be computed according to

$$Pr(\mathbf{O}|\lambda) = \frac{\sum_{i \in S_F} \alpha_T'(i)}{C_T} \tag{7.3.6}$$

In practice, the scaling operation need not be performed at every observation time. It can be used at any scaling interval for which the underflow is likely to occur. In the un-scaled interval, c_t can be kept as unity. In explicit time duration modelling, the

scaling operation must be involved for each output probability computation in a manner similar to those described here.

7.3.2. Logarithmic computation

An alternative way to avoid underflow is to use a logarithmic representation for all the probabilities. This not only ensures that underflow cannot happen, but also offers the benefit that integers can be used to represent the logarithmic values, thereby changing floating point operations to fixed point ones. If we represent probability P by, $\log_b P$, more precision can be obtained by setting b closer to one. To multiply two numbers, we simply add their logarithms. Adding two numbers is more complicated. Let us assume that we want to add P_1 and P_2 and that $P_1 \geq P_2$:

$$\begin{aligned} &\log_b(P_1+P_2) \\ &= \log_b(b^{\log_b P_1} + b^{\log_b P_2}) \\ &= \log_b P_1 + \log_b(1+b^{\log_b P_2 - \log_b P_1}) \end{aligned} \tag{7.3.6}$$

Since integers are used to represent logarithms, if $\log_b(1+b^{\log_b P_2 - \log_b P_1})$ is less than 0.5, the sum will simply be $\log_b P_1$. In other words, if P_2 is so many orders of magnitude smaller than P_1, adding the two numbers will just result in P_1. Moreover, if we could store all possible values of $\log_b P_2 - \log_b P_1$, the quantity $\log_b(1+b^{\log_b P_2 - \log_b P_1})$ could be stored as a table, $T(n)$, where

$$T(n) = \begin{cases} \log_b(1+b^n) & \text{if } T(n) \geq 0.5 \\ 0 & \textit{otherwise} \end{cases} \tag{7.3.7}$$

The number of possible values depends on the value of b. In practice, the size of $T(n)$ can be determined by decreasing n from 0 till $\log_b(1+b^n)$ approaches zero. Varying b from 1.0001 to 1.00001, the size of $T(n)$ increases from 99,041 to 1,220,614 when 32-bit integers are used for logarithms [26,96].

With the aid of this table,

$$\log_b(P_1+P_2) = \begin{cases} \log_b P_1 + T(\log_b P_2 - \log_b P_1) & \text{if } P_1 > P_2 \\ \log_b P_2 + T(\log_b P_1 - \log_b P_2) & \textit{otherwise} \end{cases} \tag{7.3.8}$$

This implements the addition of two probabilities as one integer add, one subtract, two compares, and one table lookup. Although using logarithms undoubtedly introduces

errors, in practice, the errors can be of the same order as when the scaling procedure is used for float representation.

7.3.3. Thresholding the forward-backward computation

The amount of computation in the forward-backward computation can be reduced by thresholding the forward and backward variables. Recalling that in the Baum-Welch reestimation formulas, if certain γ become very small relative to other γ, it is observed that these small γ have little effect on the final reestimates. Since the forward variable $\alpha_t(i)$ appears as a factor in γ, if during the course of the forward computation certain α become very small relative to other α at time t, these small α can be assumed to be zero without significantly affecting the performance.

Let $\hat{\alpha}_t$ denote the maximum $\alpha_t(i)$ at time t with respect to different state i,

$$\hat{\alpha}_t = \underset{i}{Max}\ \alpha_t(i) \tag{7.3.9}$$

Then, given a threshold c, for each state i such that $\alpha_t(i) < c\,\hat{\alpha}_t$, set $\alpha_t(i)$ equal to zero before moving on to compute α at time $t+1$. Thus, at time $t+1$, only those α from time t which are greater than zero will be included. The backward pass can be thresholded in the same manner. In addition, if $\alpha_t(i)$ is zero, $\beta_t(i)$ can be set to zero too. In large-vocabulary speech recognition systems, this thresholding is very important in reducing the computation to a manageable size.

The appropriate value of the threshold c must be determined empirically. If c is too large, more computation than necessary will be performed. If c is too small, the forward-backward algorithm will deteriorate to Viterbi training, i.e., only the best path is used for estimation of model parameters, which will usually deteriorate final recognition accuracy.

7.3.4. Initial estimates

In theory, the reestimation algorithm of the HMM should give a local maximum of the likelihood function. A key question is how to choose initial estimates of the HMM

parameters so that the local maximum is the global maximum. Experience has shown that, for the discrete HMM, uniform initial estimates work well, though good initial estimates may be helpful to output probabilities; for the continuous or semi-continuous HMM, however, good initial estimates are essential. Such initial estimates can be obtained from the discrete HMM parameters as discussed in following Chapters.

7.3.5. Multiple features

It is helpful to use multiple features in a practical speech recognition system. For example, LPC cepstral coefficients can be used together with energy and other dynamic information [26,55,96,126,140,158].

One way to incorporate different features into a speech recognition system is to model these multiple features as one vector. Continuous or semi-continuous hidden Markov modelling will be appropriate to such a representation as either diagonal covariance or full covariance can be well used to accommodate different feature representations [26,128,140]. Since different features may have different physical meanings, or even be strongly correlated, it is often required to use appropriate probability density functions. In addition, dimension reduction approaches based on principal component or discriminant analysis projection is required [26,76]. Alternatively, if the semi-continuous HMM or the discrete HMM is used, each feature representation can be quantised by its own VQ codebook [63,74,96].

When multiple codebooks are used, each codebook represents a set of different speech parameters. One way to combine these multiple output observations is to assume that they are independent, with the output probability computed as the product of the output probability of each codebook. It has been shown that performance using multiple codebooks can be substantially improved [98]. In the semi-continuous HMM, the semi-continuous output probability of multiple codebooks can also be computed as the product of the semi-continuous output probability for each codebook as Eq. (6.2.2), which consists of L-mixture continuous density functions. In other words, the semi-continuous output probability could be modified as:

$$b_i(\mathbf{x}) = \prod_c \sum_{j=1}^{L} f^c(\mathbf{x}|v_j^c) b_i^c(v_j^c) \tag{7.3.10}$$

where c denotes the codebook used. The reestimation algorithm for the multiple codebook based HMM could be extended if Eq. (6.3.1) is computed for each codeword of each codebook c with the rest of the codebook probabilities. Since multiplication of the semi-continuous output probability density of each codebook leads to several independent items in the Q-function as shown in Chapter 6, for codebook c_l, $\chi_t(i,j,k^{c_l})$ could be extended as:

$$\chi_t(i,j,k^{c_l}) = \frac{\alpha_t(i)a_{ij}b_j^{c_l}(k)f^{c_l}(\mathbf{x}_{t+1}|v_k^{c_l})\prod_{c\neq c_l}[\sum_{m=1}^{L} f^c(\mathbf{x}_{t+1}|v_m^c)b_j^c(v_m^c)]\beta_{t+1}(j)}{f(\mathbf{X}|\lambda)} \tag{7.3.11}$$

Other intermediate probabilities can also be computed in a manner similar to Eq. (7.3.11), and it can be easily proved that this is consistent with the maximum likelihood criterion.

7.4. Summary

This Chapter has discussed several practical issues in using HMMs for speech recognition. Modelling units play an important role in speech recognition, and have been an active research area. Detailed modelling requires enough training data, which imposes a problem because of limited training data. Tying and interpolation are two important techniques in solving this problem. The use of these techniques can be found in many practical applications. Finally, implementational issues such as scaling, logarithmic representation, thresholding, and the use of multiple features are summarised. The next Chapter will begin the description of experimental evaluations of the many theories introduced thus far.

CHAPTER 8

EXPERIMENTAL RESULTS WITH DIGIT RECOGNITION

Isolated digit recognition, because of its simplicity, has been widely used to evaluate fundamental algorithms. Such systems include studies of the DTW algorithm [136], VQ with simple or no time-alignment [29,156], the discrete HMM [137], the continuous HMM [48,132,138]. For the semi-continuous decoder and other techniques, isolated digits will be used here for evaluation of different algorithms. Interest will concentrate on the comparison of those different algorithms and models used for speech recognition rather than absolute recognition accuracy.

The experiments reported in this Chapter include:

(1) Speaker-dependent and speaker-independent isolated digit recognition using the discrete HMM;

(2) Speaker-dependent and speaker-independent isolated digit recognition using the semi-continuous decoder based on the discrete HMM parameters;

(3) The Gaussian clustering method used for VQ in speaker-independent isolated digit recognition mode.

Experimental results with the Gaussian clustering method strongly suggest the need for the unified modelling theory. The semi-continuous decoder experimental results in this Chapter provide the basis for the unified modelling theory which will be evaluated experimentally in the following Chapters.

8.1. Databases and Analysis Conditions

For speaker-dependent isolated digit recognition experiments, the database used here is four speakers (two males and two females) with 40 repetitions of each word (total 1600 words), from which 5 to 20 repetitions of each word are used separately as the training set. The others are used for evaluating the performance of the semi-continuous decoder. All the speakers represented a variety of British accents (two being Scottish and the other two English).

The speech signal was collected through a Sennheiser HME1019 headset microphone in close proximity to the mouth, but not directly in front of it. The input is passed through a lowpass filter with a cutoff frequency of 8kHz, and digitised at 20 kHz, with 12-bit resolution. The LPC analysis is then performed by the autocorrelation method, with a 25.6ms (512-point) Hamming window every 10ms. A pre-emphasis (factor 0.98) is applied to the speech prior to this analysis. The first 12 cepstral coefficients are derived from the 24th-order LPC coefficients.

For speaker-independent isolated digit recognition experiments, the database consists of 99 speakers (74 males and 25 females) with 1 repetition of each digit. The set of 100 speakers consisted of 30 from Edinburgh (15 male and 15 female), and 70 (59 male and 11 female) from industrial and commercial sites in Maidenhead, Portsmouth and London. One of the Edinburgh female speakers was subsequently excluded from the database because of an error during the processing of the data.

The data were collected through a table-mounted Sennheiser MKH406 microphone, passed through a PCM digital coder, and recorded onto video cassette. The level of background noise varied among sessions, although efforts were made to reduce it by removing noise sources (when possible) and using a sound-absorbing screen around the speaker. As with the database of speaker-dependent isolated digits, the sampling rate was 20kHz, with 12-bit resolution. A 12th-order LPC analysis was applied, in a 25.6ms Hamming-windowed frame every 10 ms, to derive 12th-order cepstral coefficients.

8.2. Experiments with the Discrete HMM

8.2.1. Speaker-dependent recognition results

In the discrete HMM, the prerequisite is to design a VQ codebook. Techniques described in Chapter 3, such as the k-means clustering algorithm, or the LBG algorithm, can be used here to generate the VQ codebook. Experimental results show that both the k-means clustering and the LBG algorithm work reasonably well for the discrete HMM. The k-means clustering algorithm is used here for generation of the VQ codebook in the isolated digit recognition experiments.

As the characteristic parameters used in the speech database are the cepstral coefficients derived from the LPC analysis, the distortion measure used in VQ is based on the Euclidean distortion. Each speaker in the database has his own codebook which was generated independently. The whole of the available training data used for parameter estimation of the HMM are used for generating the VQ codebook. The initial parameters for the k-means clustering are randomly chosen from the speech database. Every isolated word is guaranteed to contribute an equal proportion to the initial parameters used for clustering. The k-means clustering algorithm is iterated until the decrease of the overall average distortion of the VQ codebook falls below a given threshold, and several randomly selected initial parameters for the vector quantisation codebook are used, but only the codebook corresponding to the least overall average distortion is used for final evaluation. As will be seen later, the codebook with different levels is used for the evaluation experiments.

The left-to-right HMM with five states shown in Figure 8.2.1. is used for acoustic pattern matching. In the HMM used here, the initial parameters for the discrete output probabilities are evenly distributed, i.e., each probability of the output probability distributions equals to $\frac{1}{L}$, where L is the VQ level, and they can be also based on the histogram of the training data of the codebook, i.e., each probability in the output probability distribution is equal to the proportion of the corresponding codeword in the training data. The same probability distribution, either an even distribution or the

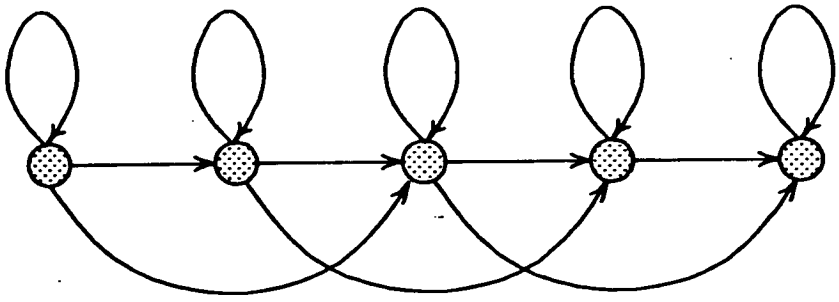


Figure 8.2.1. HMM used in digit recognition

histogram of the VQ labels, is used for all the initial output probability distributions. Experiments show that initial parameters for transition probabilities are not as sensitive as the output probability distributions, although use of initial parameters based on the VQ histogram does perform slightly better than use of the uniform distribution. With these initial parameters, the forward and backward variables are computed recursively as described in Chapter 4. The scaling procedure introduced in Chapter 7 is used in the calculation of the forward and backward variables in order to avoid underflow. The Baum-Welch algorithm is used iteratively for estimation of parameters based on the forward and backward variables with multiple independent observations of each isolated digit. The convergence criterion of the Baum-Welch algorithm is that the increase of the log-likelihood of total multiple observations conditioned on the given model is below the given threshold, or the number of iterations is above 7. After each iteration, the output probability distributions are smoothed using a floor between 0.0001 and 0.00001, and renormalised to meet stochastic constraints. The Viterbi algorithm with logarithmic representation of the transition probabilities and output probabilities is adopted here for decoding.

Experiments using different VQ levels varying from 20 to 60 with 10 training tokens for each word are conducted. The average recognition accuracy of four different speakers is shown in Table 8.1, where it can be seen that increasing the VQ level can generally improve the performance in terms of recognition accuracy. This is because the larger the codebook, the more powerfully it can represent the acoustic events. While the increase in the codebook size can improve the recognition performance, however, the larger VQ codebook will need more computation, and more training data to reliably estimate the model parameters.

Table 8.1
Results of the discrete HMM with different VQ levels

Average Recognition Rate of 4 Speakers

training tokens=10, vocabulary=digits

VQ level	speaker	discrete HMM
20	speaker 1	98.75%
	speaker 2	97.00%
	speaker 3	98.00%
	speaker 4	98.50%
20	Average	98.06%
30	speaker 1	98.85%
	speaker 2	98.00%
	speaker 3	97.71%
	speaker 4	100.00%
30	Average	98.64%
60	speaker 1	99.42%
	speaker 2	98.28%
	speaker 3	98.57%
	speaker 4	98.86%
60	Average	98.78%

The available training data and the number of free parameters to be estimated from the training data are often crucial for parametric statistical modelling techniques. With limited training data, the statistical model should have as few parameters as possible to model the speech signals, however, on the other hand, for accurate discrimination of different speech signals, the statistical model needs as many as possible. The compromise between limited training data and numbers of free parameters has usually to be made with care in speech recognition system design. In the experiments conducted below, with the VQ level fixed at 60 and the number of training

tokens for each word varying from 5 to 20, the experimental results are listed in Table 8.2. It can be seen that an increase in training data can generally improve the performance in terms of recognition accuracy. The problem is that in practice the available training data is often too limited.

Table 8.2.
Results of the discrete HMM with different training tokens.

Average Recognition Rate of 4 Speakers

VQ level=60, vocabulary=digits

training tokens	speaker	discrete HMM
5	speaker 1	98.66%
	speaker 2	98.33%
	speaker 3	96.33%
	speaker 4	99.66%
	Average	98.24%
10	speaker 1	99.42%
	speaker 2	98.28%
	speaker 3	98.57%
	speaker 4	98.86%
	Average	98.78%
15	speaker 1	100.00%
	speaker 2	99.14%
	speaker 3	98.00%
	speaker 4	98.57%
	Average	98.92%
20	speaker 1	100.00%
	speaker 2	98.28%
	speaker 3	99.14%
	speaker 4	98.57%
	Average	98.99%

8.2.2. Speaker-independent recognition results

For speaker-independent isolated digit recognition, the HMM can conveniently put all speakers together to train the parameters of the model [96,138]. In speaker-independent isolated digit recognition, the VQ level should usually be higher than that used in speaker-dependent isolated digit recognition in order to model the characteristics of different speakers. As the speakers available are limited (only some 100 speakers), only one HMM for each digit is constructed. Approximately half of the database (12 females and 38 males) was used as a training set, and the remainder of the speakers

used for evaluation.

With 50 different speakers, the VQ codebook of level 80 was generated in a manner similar to the speaker-dependent digit recognition experiments. The training procedure used here is also the same as the procedure used in the speaker-dependent isolated digit recognition experiments except that different speakers' observations are simply treated as the same speaker. The Viterbi algorithm is used for decoding the 49 new speakers.

Table 8.3
Performance of speaker-independent digit recognition

	Average Recognition Accuracy of 49 Speakers
discrete HMM	88.9%

The experimental results of speaker-independent isolated digit recognition are listed in Table 8.3. From the experiments conducted here, it can be seen that the recognition accuracy of speaker-independent recognition is significantly lower than that of speaker-dependent recognition. This may be because either the available training data are not enough, or the vector quantisation lost too much information. There are many ways to improve speaker-independent speech recognition performance such as use of differential cepstrum and multiple codebooks as discussed in Chapter 10.

8.3. Experiments with the Semi-Continuous Decoder

8.3.1. Speaker-dependent recognition results

As the semi-continuous decoder assumes that the VQ codebook is a parametric family of mixture probability densities, the Gaussian probability density function of each codeword is calculated from the cell of each codeword after the generation of the codebook. The mean and covariance matrices are the maximum likelihood sample mean and covariance matrices as:

$$\mu_k = \frac{1}{K_k} \sum_{\mathbf{x}_i \in C_k} \mathbf{x}_i$$
$$\Sigma_k = \frac{1}{K_k} \sum_{x_i \in C_k} (\mathbf{x}_i - \mu_k)(\mathbf{x}_i - \mu_k)^t \qquad (8.3.1)$$

where C_k denotes the cell of the codeword k, and K_k denotes the number of observations in C_k. These equations are the well-known maximum likelihood estimates of the mean and covariance matrix of the training data, $\mathbf{x}_i$, in the corresponding cell (cluster), C_k.

With the semi-continuous output probability, the summation over all the VQ codewords is unnecessary, and one option is to use only several of the most significant values $f(\mathbf{x}|v_j)$ obtained from VQ. This can significantly reduce the amount of computational load if the VQ level is relatively large. In the isolated digit recognition experiments conducted here, as the VQ level is relatively small (20 to 60), the semi-continuous output probability is used directly.

One important issue in the semi-continuous decoder is how to smooth the continuous probability density functions derived from the VQ codebook. The covariance matrices may become singular when the number of observations in the cell is too small. One way to obtain reliable covariances is to pool several cells C_k of the nearest codewords together to estimate the average covariance matrix. The newly obtained covariance matrix can then be used by these codewords. After estimation of covariance matrices, the semi-continuous output probability density function is then normalised to be within [0, 1] as discussed in Chapter 6.

In the first tuning experiments, the Gaussian probability density function of the VQ is assumed to be diagonal. The level of the VQ codebook is set to be 20, which is obviously too small. However, with such a VQ codebook, the semi-continuous decoder performs better than both the discrete HMM and the DTW-based recogniser [110]. Here, the discrete HMM parameters are used for the semi-continuous decoder. The training data for each HMM include 10 repetitions of each digit, and each digit is represented by one HMM as described in Section 8.2.1. The average recognition accuracy of four speakers (each speaker trained and tested independently) are listed in Table 8.4. In the DTW-template-based recogniser, the templates are composed by averaging all the training templates into one template using DTW adaptation techniques [110].

Table 8.4. Performance comparison of semi-continuous decoder, discrete HMM, and template DTW		
Average Recognition Rate of 4 Speakers		
VQ level=20, training tokens=10, vocabulary=digits		
diagonal SCHMM	discrete HMM	DTW after 10 adaptations
98.68%	98.06%	98.48%

From Table 8.4, it can be seen that the recognition error rate of the semi-continuous decoder is reduced by more than 30% in comparison to the corresponding discrete HMM. Even though the discrete HMM performs worse than the DTW algorithm, the semi-continuous decoder can reduce the error rate of the DTW algorithm with non-vector-quantised templates by 13%. From this experiment, it can be seen that the semi-continuous decoder can effectively reduce the errors of using the VQ with only modest computational complexity increases. The experimental results reported here may be subject to the relatively small VQ codebook, which has considerable distortion. To investigate the performance of the semi-continuous decoder with less VQ errors, different VQ levels were used to test the discrete HMM and the semi-continuous decoder.

Table 8.5.
Comparison of semi-continuous decoder with diagonal and full covariance with different VQ level.

Average Recognition Rate of 4 Speakers

VQ level=from 20 to 60, training tokens=10, vocabulary=digits

VQ level	speaker	diagonal SCHMM	full SCHMM
20	speaker 1	99.25%	99.71%
	speaker 2	97.50%	97.72%
	speaker 3	98.50%	98.20%
	speaker 4	99.50%	99.20%
20	Average	98.68%	98.71%
30	speaker 1	99.43%	99.71%
	speaker 2	98.57%	97.71%
	speaker 3	98.57%	99.14%
	speaker 4	99.71%	99.14%
30	Average	99.32%	98.92%
60	speaker 1	99.42%	99.71%
	speaker 2	98.57%	97.71%
	speaker 3	98.86%	99.14%
	speaker 4	99.42%	98.86%
60	Average	99.06%	98.85%

In a manner similar to the experiments described in Section 8.2.1, experiments using different VQ levels varying from 20 to 60 with 10 training tokens for each word were conducted, and the average recognition accuracy of four different speakers is shown in Table 8.5. From Table 8.5, it can be seen that increasing the VQ level can generally improve the performance provided that enough training data is available for estimation of the model parameters. When the VQ level increases to 60, the performance then deteriorates, and the reason for this deterioration may be due to the limited training data used here, i.e., there are too many free parameters in the discrete HMM when the VQ level increases. In cases where training data is limited, decreasing the VQ codebook level from 60 to 30 can increase the performance of the semi-continuous decoder with both the diagonal and full covariance matrix. The free parameters to be estimated in the diagonal covariance assumption are much less than for the full covariance assumption. Assuming the vector dimension is d, each full covariance matrix has $O(d^2)$ free parameters while the diagonal covariance has only $O(d)$ free parameters. As the full covariance matrices need more training data to estimate the off-diagonal data for the semi-continuous decoder, the performance of the semi-continuous decoder with the full covariance is generally worse than that of the semi-continuous decoder with the

diagonal covariance. Another reason is that the cepstral coefficients are almost uncorrelated. Therefore, the diagonal Gaussian density is enough to model given observations.

Since the number of training tokens is relatively small in the above experiments, the performance of the semi-continuous decoder using a relatively large amount of training data has been studied. Using different numbers of tokens in the training set varying from 5 to 20 with the VQ level of 60 fixed, a series of experiments with full Gaussian covariance and diagonal covariance semi-continuous decoder were conducted, and experimental results are listed in Table 8.6.

Table 8.6.
Comparison of semi-continuous decoder with diagonal and full covariance with different training tokens.

Average Recognition Rate of 4 Speakers

VQ level=60, vocabulary=digits

training tokens	speaker	diagonal SCHMM	full SCHMM
5	speaker 1	99.66%	99.00%
	speaker 2	99.00%	98.33%
	speaker 3	98.00%	97.33%
	speaker 4	99.34%	96.34%
	Average	99.00%	97.75%
10	speaker 1	99.42%	99.71%
	speaker 2	98.57%	97.71%
	speaker 3	98.86%	99.14%
	speaker 4	99.42%	98.86%
	Average	99.06%	98.85%
15	speaker 1	100.00%	100.00%
	speaker 2	98.86%	99.14%
	speaker 3	97.71%	98.57%
	speaker 4	100.00%	99.71%
	Average	99.14%	99.35%
20	speaker 1	100.00%	100.00%
	speaker 2	99.14%	99.14%
	speaker 3	98.85%	99.14%
	speaker 4	100.00%	99.71%
	Average	99.49%	99.49%

From Table 8.6, it is observed that in cases where the training data is limited (5 to 10 training tokens), the diagonal covariance semi-continuous decoder is generally better than the full covariance semi-continuous decoder. Only when the training tokens for each word increase above 10, does the full covariance semi-continuous decoder

performance exceed or equal that of the diagonal covariance semi-continuous decoder. The diagonal covariance semi-continuous decoder consistently performs better than the discrete HMM under all conditions. In fact, with only five training tokens for each word, the diagonal semi-continuous decoder performs better than the discrete HMM trained with 20 tokens for each word.

The effects of increasing the quantity of training data and the VQ codebook level for the discrete HMM are not as significant as for the semi-continuous decoder. Table 8.1. to Table 8.6. show that for all VQ levels and training sets, the performance of the diagonal covariance semi-continuous decoder and full covariance semi-continuous decoder (except with five training tokens) are always superior to the discrete HMM. The error reduction of the semi-continuous decoder is about 49% (98.99% vs. 99.49%) in comparison to the error rate of the discrete HMM. The most remarkable observation is that the performance of the semi-continuous decoder may work well even if less-well-trained models are used. To further investigate the performance of the discrete HMM, and the semi-continuous decoder based on the discrete HMM parameters, a database of multi-speakers is used in the following Sections.

8.3.2. Speaker-independent recognition results

Speaker-independent recognition conditions are similar to those described in Section 8.2.2. The recognition accuracy in speaker-independent isolated digit recognition using the semi-continuous decoder based on the discrete HMM parameters, and the discrete HMM is listed in Table 8.7.

Table 8.7
Performance comparison of speaker-independent recognition

Average Recognition Accuracy of 49 Speakers	
the discrete HMM	88.9%
the diagonal covariance decoder	90.4%
the full covariance decoder	91.2%

From the experiments conducted here, it can be seen that with the diagonal semi-continuous decoder, the error reduction relative to the discrete HMM is about 13%, which is much less impressive than that of the speaker-dependent recognition

experiments. This suggests that the semi-continuous decoder is very suitable to speaker-dependent speech recognition. In the speaker-independent experiments, the semi-continuous decoder using full covariance matrices can effectively reduce error rates by 20% in comparison to the discrete HMM. The fact that the full semi-continuous decoder works better than the diagonal semi-continuous decoder indicates that the model can be relatively well-trained, and more complicated techniques should be applied. More detailed speaker-independent speech recognition experiments will be discussed in Chapter 10.

8.4. Experiments with Gaussian Clustering Results

In the conventional k-means clustering algorithm, as used in VQ codebook design, the Euclidean distortion measure is generally used, with each component of the parameter vector having uniform weight. Due to the inherent variability of speech signals and the existence of coordinate-dependent noise levels, parameter vectors for different speech sounds require different coordinate weighting to achieve optimum identification. Furthermore, in the semi-continuous decoder, where it is required to estimate parameters of a Gaussian distribution for each codeword, estimation of the probability density function after k-means clustering may be inferior to estimation during the clustering process itself.

If the parameter vectors are considered to be random vectors whose probability distribution is modeled by a multivariate Gaussian density function then the clustering problem involves estimation of the parameters of the Gaussian density function. The EM algorithm can then be used to estimate the parameters of the mixture densities. Because of the computational complexity of the EM algorithm, a simple k-means clustering technique is used here to investigate the effects of the VQ codebook on the semi-continuous decoder.

The clustering problem involves partitioning the training data into different Gaussian densities, say L-mixture; and estimating the unknown parameters of each partition, i.e. the codebook cell. Suppose that $C_1, C_2, \cdots C_L$ are L partitions and that

the observations, $\mathbf{x}_i$, are independent samples on the population with probability density function $f(\mathbf{x}|\varphi_k)$ in the kth partition. The criterion used here for clustering can be written as:

$$E(\mathbf{x}|\varphi) \equiv \sum_{k=1}^{L} \sum_{x_i \in C_k} [-\log(|\Sigma_k|) - (\mathbf{x}_i - \mu_k)^t \Sigma_k^{-1} (\mathbf{x}_i - \mu_k)] \tag{8.4.1}$$

where $\varphi = (\varphi_1, \varphi_2, \cdots \varphi_L)$ denotes the parameters to be estimated, which is composed of the mean vector, μ_k, and covariance matrix, Σ_k, of the kth Gaussian components.

The clustering procedure is then the maximisation of Eq. (8.4.1) with respect to φ by partitioning of the observation data $\mathbf{x}$ into L cells. Given a partition, the optimal parameter to maximise the above likelihood function can be obtained by taking partial derivatives of Eq. (8.4.1) with respect to μ_k and Σ_k. The optimal value of μ_k to maximise Eq. (8.4.1) can be obtained as:

$$\mu_k = \frac{1}{K_k} \sum_{\mathbf{x}_i \in C_k} \mathbf{x}_i \tag{8.4.2}$$

The partial derivative with respect to Σ_k^{-1} is:

$$\frac{\partial E}{\partial \Sigma_k^{-1}} = K_k \frac{|\Sigma_k^{-1}|}{|\Sigma_k^{-1}|} \Sigma_k - \sum_{x_i \in C_k} (\mathbf{x}_i - \mu_k)(\mathbf{x}_i - \mu_k)^t \tag{8.4.3}$$

Since Eq. (8.4.1) represents a negative concave function, by equating to zero, the maximum of E can be obtained where

$$\Sigma_k = \frac{1}{K_k} \sum_{x_i \in C_k} (\mathbf{x}_i - \mu_k)(\mathbf{x}_i - \mu_k)^t \tag{8.4.4}$$

From Eq. (8.4.2) and Eq. (8.4.4), it is observed that this is the well-known maximum likelihood estimate of the mean and covariance matrix of the training data in the corresponding cluster. A modified k-means clustering algorithm thus can be achieved by using the log Gaussian probability density function as a *distortion measure* and the likelihood function of Eq. (8.4.1) as total distortion measure. Given the partition, Eq. (8.4.1) is used to classify all the vectors which are closest (maximum) over L different Gaussian distributions into their corresponding distributions. After obtaining the partition, Eq. (8.4.2) and (8.4.4) are used to reestimate the new mean vector and covariance matrix for each component. The iteration of these preceding two steps can be guaranteed to increase the likelihood function $E()$; and a local maximum can be obtained after several iterations.

In practice, the criterion used in Eq. (8.4.1) may not be bounded. If one of the covariance matrices tends to a singular matrix, then the log-likelihood function increases without bound. This difficulty can be solved by collecting a number of observations which is greater than the observation dimension. If the total number in each cell is less than the vector dimension, the observation vectors in the nearest cell can be pooled together to estimate the covariance matrices.

To evaluate the performance of the algorithm, the speaker-independent experiments were repeated. The conditions used here are the same as for the experiments described in Section 8.3. In the Gaussian clustering, Σ_k is based on the diagonal covariance Gaussian assumption, and the initial parameters are based on the conventional k-means clustering results. The distortion measure used for the parameter estimation in the discrete HMM can be based either on the Euclidean distortion or the Gaussian density function as a distortion measure. Experiments show that the differences between these two distortion measures are negligible for the semi-continuous decoder. The experimental results of the Gaussian clustering and posterior estimation after conventional k-means clustering are listed in Table 8.8.

Table 8.8
Performance with Gaussian VQ and k-Means VQ

Average Recognition Rate of 49 Speakers with VQ level of 80		
	Gaussian VQ	*k-Means VQ*
discrete HMM	90.8%	88.9%
diagonal SCHMM	90.8%	90.4%
full SCHMM	91.0%	91.2%

From Table 8.8, it can be observed that the recognition accuracy of the semi-continuous decoder with the Gaussian clustering technique is about the same as that with the conventional k-means clustering algorithm. Although the Gaussian density function can be closely combined into the clustering procedures, the performance is about the same. However, the recognition accuracy of the discrete HMM can be improved marginally. The improvement of the discrete HMM may come from the consistent use of Gaussian density as distortion measure in the training and decoding procedure. This suggests that the discrete HMM parameters should be also reestimated in a consistent way as for the semi-continuous decoder.

8.5. Summary

This Chapter has described isolated digit recognition experimental results using

(1) The discrete HMM for speaker-dependent and speaker-independent digit recognition;

(2) The semi-continuous decoder based on the discrete HMM for speaker-dependent and speaker-independent digit recognition;

(3) Vector quantisation codebook generation with the Gaussian clustering technique and the conventional k-means clustering technique.

The semi-continuous decoder has been shown to offer improved performance in terms of recognition accuracy in comparison to the discrete HMM. The average recognition error rate of the semi-continuous decoder can be reduced by 10% to 50% in comparison to the discrete HMM.

Unfortunately, the experiments with Gaussian clustering techniques do not perform as well as expected. This indicates that the performance of VQ using either the conventional k-means clustering techniques or other improved techniques which aim to minimise the distortion measures can offer only limited success. The improvement of the VQ codebook probably relies on the mutual optimisation of the HMM and the codebook itself, which leads to the unified modelling theory as introduced in Chapter 6.

CHAPTER 9

EXPERIMENTAL RESULTS WITH PHONEME RECOGNITION

The unified modelling theory using semi-continuous HMM discussed in Chapter 6 will be explored in the experiments conducted here and in Chapter 10. This Chapter will include experiments which have been carried out to compare the performance of

(1) the discrete HMM;

(2) the continuous HMM;

(3) the semi-continuous decoder based on the discrete HMM;

(4) the semi-continuous reestimation of model parameters (with fixed VQ codebook);

(5) the unified modelling technique with two simplified variants

in a phoneme classification task.

The phoneme recognition task can be used as a more difficult domain to compare semi-continuous HMMs, continuous HMMs, and discrete HMMs. Other phoneme classification problems, such as coarticulation effects, speech modelling units, and model structure, are not studied in this work.

9.1. Database and Analysis Conditions

Table 9.1
List of the set of phonemes used in the experiments

Monophthongal Vowels			
phoneme	example	phoneme	example
/i/	b*i*d	/ii/	b*ea*d
/e/	b*e*d	/a/	b*a*d
/aa/	b*ar*d	/uh/	b*u*d
/@@/	b*ir*d	/@/	*a*bout
/o/	p*o*t	/oo/	p*or*t
/u/	p*u*t	/uu/	b*oo*t
Diphthongal Vowels			
phoneme	example	phoneme	example
/ei/	d*ay*	/ou/	g*o*
/au/	c*ow*	/ai/	*eye*
/oi/	b*oy*	/i@/	b*eer*
/e@/	b*are*	/u@/	t*our*
Consonants			
phoneme	example	phoneme	example
/p/	*p*ea	/t/	*t*ea
/k/	*k*ey	/b/	*b*ee
/d/	*d*ye	/g/	*g*uy
/m/	*m*e	/n/	*kn*ee
/ng/	si*ng*	/th/	*th*in
/dh/	*th*en	/f/	*f*an
/v/	*v*an	/s/	*s*ea
/z/	*z*ee	/sh/	*sh*e
/zh/	bei*ge*	/ch/	ea*ch*
/jh/	e*dge*	/h/	*h*at
/y/	*y*es	/w/	*w*ay
/r/	*r*ay	/l/	*l*ay
Syllabic Consonants			
phoneme	example	phoneme	example
/r=/	mirro*r*	/l=/	animal
/n=/	a*n*d	-	-

The database consists of two repetitions of the same continuous speech sentences from a male speaker. Each set has 98 sentences with 579 words composed of English phonemes and variations (total 47). The length of the first set of speech signals is 179 seconds, and the length of the second set is 188 seconds. The sentences have been hand-

labelled to the level of individual phonemes. Each of the 47 individual HMMs are trained and decoded using hand-labelled phonemes, with the (unbalanced) number of phonemes used to derive individual HMMs varying from 2 to 191. It should be stressed that this is very little training data. The phonemes used to construct the HMM are listed in Table 9.1.

The speech is sampled at 16 kHz, and preemphasised with a filter whose transform function is $1\text{-}0.97z^{-1}$. Two different analysis conditions are applied to the speech database.

(A1) The waveform is blocked into frames every 10ms by a Hamming window with length of 20ms. Then 10th-order cepstral coefficients derived from the 12th-order autocorrelation LPC method are used as final parameter representation. The 47 phonemes as listed in Table 9.1. are used in the experiments.

(A2) The waveform is blocked into frames every 5ms by a Hamming window with length of 20ms. Then 14th-order cepstral coefficients derived from the 14th-order autocorrelation LPC method are used as final parameter representation. As the syllabic consonants /r=/, /l=/, and /n=/ have only limited number of training data, they are merged into /r/, /l/, and /n/ respectively. This leads to a total of 44 phonemes.

9.2. Experimental Results of the Discrete HMM

For the discrete HMM used for phoneme recognition, a model for each phoneme with structure shown in Figure 9.2.1. was constructed, and each phoneme is represented by one of these HMMs. The model itself is ready to be extended to the HMM-based continuous speech recognition system. However, only the acoustic modelling will be considered here.

The VQ codebook is generated using all of the training data by employing the LBG algorithm [101] as described in Chapter 3. In a manner similar to experiments described in the previous Chapter, HMM parameters are estimated using the first set of sentences

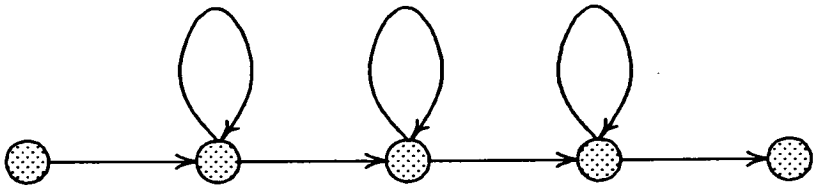


Figure 9.2.1. HMM used in phoneme recognition

in the database iteratively until the increase of total observation probability is below given threshold or the number of iterations is above 7. The floor, 0.000001, is applied to the final output probability. The second set of sentences in the database is used in decoding by the Viterbi algorithm.

When analysis condition A1 is used, i.e. the 10th-order cepstral coefficients with window shift of 10ms, with the VQ level varying from 128 to 256, the average phoneme recognition accuracy of the discrete HMM is 50.0% and 50.1% respectively for 47 phonemes. The fact that increasing the VQ level from 128 to 256 does not result in improved phoneme recognition accuracy is an indication that for the limited training data used here, a VQ level of 128 is adequate. The recognition results for 47 different phonemes are listed in Table 9.2. The first candidate to the fifth candidate are shown in the table. These results will be used as a benchmark afterwards.

Table 9.2. Phoneme recognition results using the discrete HMM					
recognition accuracy (correct recognised tests)					
VQ level of 128 (number of tests = 1748)					
-	*First*	*Second*	*Third*	*Fourth*	*Fifth*
A1 analysis condition	50.0% (874)	68.5% (1197)	78.0% (1364)	83.2% (1454)	87.9% (1537)
A2 analysis condition	53.2% (930)	71.2% (1245)	81.2% (1419)	86.2% (1507)	89.5% (1564)

As analysis condition A1 may be too simple, analysis condition A2 is conducted in parallel with A1. The discrete HMM with the VQ level of 128 is conducted for A2. From Table 9.2, it can be observed that the average recognition accuracy can be improved to 53.2%. (If 47 phonemes are used with 14 order cepstral coefficients, the average recognition accuracy is 52.3%). This indicates that the analysis order of 10 is not large enough when the sampling rate is 16kHz. As higher dimension analysis requires more computation and memory, albeit 18-20 order LPC analysis should be ideal for the experiments, only A1 and A2 analysis conditions are used here to investigate various algorithms.

From the confusion matrix obtained in the experiments, it is observed that the recognition accuracy of vowels is measurably lower than that of consonants. An interesting fact is that confusion between vowels and consonants is very small.

9.3. Experimental Results of the Continuous HMM

The same analysis conditions and speech database were used for experiments in continuous hidden Markov modelling. Due to the limited training data available and un-correlated property of the LPC cepstrum, in the continuous HMM used here, the continuous output probability density function is represented by a single Gaussian density function with diagonal covariance matrix.

The initial parameters for the mean vectors in the continuous HMM are crucially related to the recognition performance. With random initial mean vectors, the continuous HMM actually performs only slightly better than the discrete HMM. One reliable way to determine the initial mean values for the continuous HMM is to use the discrete HMM parameters as initial guesses. For each discrete output probability distribution in the discrete HMM, the codeword corresponding to the maximum probability in the discrete output probability distribution is used as the initial mean vector for the corresponding continuous output probability density function in the continuous HMM. The posterior estimates of the covariance matrix for the corresponding codeword can then be used in the same way as the mean vector for the initial covariance parameters. Experimental results show that using the initial parameters of the discrete HMM can measurably improve the performance of the continuous HMM. When the continuous HMM based on the discrete HMM is used, the measured accuracy of A1 improves to 55.3%, which is higher than both of the discrete HMM (50.0%) and the continuous HMM whose parameters are estimated with random initial parameters (50.6%).

The use of diagonal Gaussian probability density functions can modestly reduce the free parameters to be estimated in comparison to the discrete HMM. That is one reason why the continuous HMM offers higher recognition accuracy in comparison to the discrete HMM. Although it has been shown that the continuous HMM should use mixture Gaussian densities [138], in the experiments conducted here, single mixture Gaussian continuous HMM works reasonably well because of limited training data and speaker-dependent task. The detailed experimental results with the continuous HMM are listed in Table 9.3.

Table 9.3. Phoneme recognition results using the continuous HMM					
recognition accuracy (correct recognised tests)					
Diagonal Covariance (number of tests = 1748)					
	First	*Second*	*Third*	*Fourth*	*Fifth*
A1 analysis condition	55.3% (966)	73.7% (1289)	83.0% (1451)	88.7% (1551)	91.7% (1602)
A2 analysis condition	58.5% (1023)	77.0% (1347)	84.1% (1470)	88.8% (1552)	91.5% (1600)

For the analysis condition A2, the recognition accuracy of the continuous HMM is 58.5% for 44 phonemes. This is also significantly better than the discrete HMM (53.2%).

9.4. Experimental Results of the Semi-Continuous Decoder

The HMM parameters can be reestimated based on the semi-continuous output probabilities without reestimating the VQ codebook. Under analysis condition A1, using discrete HMM parameters and reestimated semi-continuous parameters, the recognition accuracy of the semi-continuous decoder was tested. Experimental results show that nearly the same recognition accuracy is achieved. Here, the covariance of the corresponding codeword is obtained from the nearest neighbour estimates of the original codebook; and the VQ codebook is not reestimated.

In the semi-continuous decoder, varying the range of the most significant $f(\mathbf{x}|v_j)$ in the semi-continuous output probability density function from one to five, it was observed that the choice of the top 3 to 4 values of $f(\mathbf{x}|v_j)$ from the VQ procedure is appropriate. In a manner similar to the semi-continuous decoder, only several of the most significant codewords need be used for training. The experimental results show that the top one codeword representation gives performance similar to the multi-codeword representation for training. However, when the number of top-codewords in the semi-continuous decoder increases, the overall recognition accuracy can be improved, but saturates after 5. The best recognition accuracy of the semi-continuous decoder is

54.9%, which is better than the discrete HMM (50.0%) but worse than the continuous HMM (55.3%). Results under A1 analysis conditions are shown in Table 9.4, where *training mixture* and *decoder mixture* denote the number of top-codewords used in Eq. (6.2.2) for training and decoding respectively.

Table 9.4.
Results of the semi-continuous decoder

recognition accuracy (correct recognised tests)					
Training Mixture=4, Decoder Mixture=4, (number of tests = 1748)					
	First	*Second*	*Third*	*Fourth*	*Fifth*
A1 Analysis Condition	54.9% (961)	72.0% (1259)	83.9% (1647)	87.9% (1537)	91.5% (1600)

If the model parameters are reestimated based on the semi-continuous output probability, recognition accuracy is about the same as the semi-continuous decoder based on the discrete HMM. This shows that the training procedure is not as sensitive to VQ errors as the decoding procedure. Although the VQ errors can be minimised by using the semi-continuous decoder, the recognition accuracy is still not as good as that of the continuous HMM. The VQ codebook is still an important factor affecting the overall performance. This further indicates that VQ codebook should be reestimated according to the HMM parameters to achieve an optimal codeword/model combination. The unified modelling theory will be investigated in the following Sections.

9.5. Experimental Results of the Unified Modelling Theory

This Section will report experimental results of two variants of the unified modelling theory using the phoneme classification task.

9.5.1. Simplified codebook reestimation

For unified modelling of VQ and hidden Markov model, due to the extensive computation load of the unified reestimation equations given in Chapter 6, a simplified

method is investigated here. The complete semi-continuous modelling will be discussed in the next Chapter.

The simplified VQ (SVQ) codebook reestimation procedure can be divided into two stages. In the first stage, the discrete HMM reestimation is run, and discrete HMM parameters are then used as initial parameters for unified modelling. Only the top one codeword is used to represent the semi-continuous output probability density during parameter reestimation, which is determined according to the most significant codeword in the corresponding discrete output probability in the state. The reestimates (means and covariances) can be used either to replace the original means and covariances in the VQ codebook or to form an average with other reestimates according to the pre-selected codeword in a similar manner to those described in Chapter 6. This can be considered as a special technique of unified modelling; within individual model v where the reestimation of the mean and covariance is constrained only to the most significant $b_i(O_j)$. These means and covariances are used as feedback to the VQ codebook for reestimation. In the second stage, after reestimation of the mean vectors and the covariance matrices, the reestimation algorithm for the weighting coefficients (i.e. the discrete output probability distributions) can be used again together with the transition probabilities on the reestimated codebook to obtain the final discrete output probability distributions. With the simplified approach used here, the computational complexity of the parameter reestimation can be significantly reduced in comparison to standard unified reestimation equations.

For the A1 analysis condition, the SVQ is used for evaluation. The experimental results show that use of either of the replacement or averaging techniques during reestimation produces similar recognition accuracy. This suggests strong correlation between reestimated codewords. Using the reestimated VQ codebook, it is interesting to note that the discrete HMM accuracy (55.3%) (using Gaussian density functions as a VQ distortion measure) is comparable to the continuous HMM accuracy (55.3%). This indicates that the quality of VQ can be much improved with the unified modelling approach.

With reestimated model parameters and VQ codebook, when the number of top-codewords of the decoder (decoder mixture) is 3, the recognition accuracy of the semi-continuous decoder is 58.3%, which is measurably better than both the discrete HMM and the continuous HMM. For the A2 analysis condition, similar results to the A1 analysis condition are achieved. The semi-continuous decoder with the top three codewords performs (60.0%) measurably better than the corresponding discrete HMM (53.2%) and continuous HMM (58.5%). Detailed experimental results are listed in Table 9.5. The error reduction of the simplified unified modelling is about 14% to 16% in comparison to the discrete HMM. The reason for this modest improvement may be that the modelling unit is too blurred. If the recognition accuracy of the top five candidates is considered, it can be seen that the error reduction of the simplified unified modelling is 33% and 17% in comparison to the discrete HMM and the continuous HMM respectively.

Table 9.5.
Results of the SVQ (number of tests = 1748)

recognition accuracy (correct recognised tests)					
Training Mixture = 128, Decoder Mixture = 3, VQ level = 128					
	First	*Second*	*Third*	*Fourth*	*Fifth*
A1 Analysis Condition	58.3% (1019)	74.8% (1308)	83.6% (1461)	89.4% (1563)	92.7% (1620)
A2 Analysis Condition	60.0% (1049)	77.0% (1345)	86.2% (1507)	90.2% (1576)	93.0% (1627)

Using the SVQ, a series of experiments were conducted with different numbers of the top-codewords in both the training and decoding procedures under the A2 analysis condition to investigate the performance of the semi-continuous decoder and reestimator. Here, it was observed that in the training procedure, the increase of the number of top-codewords does not contribute any significant improvement. On the contrary, the decoding procedure needs more top-codewords to accommodate the VQ errors and can modestly improve the recognition accuracy. An interesting observation is that if the number of top-codewords in the decoding procedure is less than the number of top-codewords in the training procedure, the recognition accuracy is usually not satisfactory. When the number of top-codewords in the training procedure equals the VQ level, the recognition accuracy is actually worse than that for a limited number of top-codewords.

One possible reason for a reduction in recognition accuracy with an increase in the number of top-codewords may be that the floor for the weighting coefficients, $b_j(k)$, is not accurate so that some codewords are falsely represented resulting in damage to the smoothing by several codeword density functions.

9.5.2. Forming VQ codebook from the continuous HMM

Another simplified unified modelling technique used in the experiments directly connects the continuous HMM to VQ, i.e. each individual continuous output density function is considered as one of the codewords in the VQ codebook which is modelled as a parametric family of mixture probability density functions. This approach involves collecting the means and covariances of the continuous HMM output probability density functions to form a VQ codebook, and will be referred to as FVQ here.

In the A1 analysis condition, there are a total of 47 different phoneme models with 3 output probability density functions in each model. Therefore, a VQ codebook of 141 (47*3) levels is constructed. Here again, Eq. (6.3.12) is used to reestimate the discrete output probability (the training mixture used here is 141). The recognition accuracy of the semi-continuous decoder with the top three codewords using the FVQ is 58.5% which is the same as for the SVQ counterpart (58.3%). If the decoder is used with the top one codeword (i.e. a discrete model using Gaussian density functions as a distortion measure), the recognition accuracy is 53.8% which is slightly lower than the SVQ counterpart (55.3%). For the A2 analysis condition, once again, similar results can be obtained as shown in Table 9.6. A VQ codebook of 132 (44*3) levels is constructed. From experiments conducted here, it can be observed that the unified modelling indeed offers better performance. Even when simplified approaches are used, the recognition accuracy can be improved consistently in comparison to both the discrete HMM and the continuous HMM. It should be pointed out that the idea of forming a new VQ codebook from the continuous HMM reestimates can also be applied to large vocabulary speech recognition systems. The information-theoretic clustering of probability density functions of the HMM can be used to obtain a VQ codebook of any level.

Table 9.6. Results of the FVQ					
recognition accuracy (correct recognised tests)					
Training Mixture = L Decoder Mixture=3; (number of tests = 1748)					
	First	*Second*	*Third*	*Fourth*	*Fifth*
A1 Analysis Condition (L=141)	58.5% (1023)	76.4% (1336)	84.8% (1483)	89.8% (1570)	92.7% (1621)
A2 Analysis Condition (L=132)	61.8% (1081)	78.8% (1378)	85.9% (1502)	90.1% (1575)	93.0% (1627)

9.6. Results of Time Duration Modelling

Time duration modelling with Gaussian smoothing has been shown to offer improved recognition accuracy in isolated phoneme recognition experiments [3]. For the A2 analysis condition, the performance of the discrete HMM with time duration can be improved to 60.5% as shown in Table 9.7. Here, an error reduction of 14% is achieved. One of the reasons for such an improvement is that no grammatical constraints are used; and the time duration modelling can effectively eliminate those phonemes with different time durations.

Table 9.7. Results of time duration modelling (A2 analysis condition)				
recognition accuracy (correct recognised tests)				
Discrete HMM, VQ level =128 (number of tests = 1748)				
First	*Second*	*Third*	*Fourth*	*Fifth*
60.5% (1058)	77.4% (1353)	86.0% (1503)	90.7% (1583)	92.5% (1616)

Since time duration modelling introduces substantial increases in computation, the number of top-codeword density functions used in the training procedure is limited to one. Using the semi-continuous decoder on the discrete parameters (where the distance is based on the Gaussian density because the codebook is represented by the parametric family of mixture Gaussian densities), the average recognition accuracy of different

decoder mixtures is listed in Table 9.8. It can be seen that the recognition accuracy can reach 68.2% when the number of decoder mixture equals 4. For the new vector quantisation codebook obtained from the continuous HMM output probability density functions (FVQ), the recognition accuracy of the semi-continuous decoder with different decoder mixtures is listed in Table 9.9. The recognition accuracy is once again similar to the reestimated VQ codebook (68.6%).

Table 9.8.
Results of time duration modelling (A2 analysis condition)
recognition accuracy (correct recognised tests)
SVQ, Training Mixture=1, VQ level=128 (number of tests = 1748)

decoder mixture	*First*	*Second*	*Third*	*Fourth*	*Fifth*
1	65.3% (1142)	81.8% (1430)	88.2% (1541)	91.7% (1602)	94.3% (1648)
2	67.2% (1174)	84.1% (1470)	90.1% (1574)	93.8% (1640)	95.4% (1667)
3	67.8% (1185)	85.0% (1485)	90.5% (1582)	93.7% (1638)	95.4% (1668)
4	68.2% (1192)	84.7% (1481)	90.9% (1589)	93.9% (1641)	95.6% (1671)
5	68.0% (1189)	84.4% (1475)	91.0% (1590)	94.1% (1644)	95.8% (1674)

Table 9.9. Results of time duration modelling (A2 analysis condition) recognition accuracy (correct recognised tests)					
FVQ, Training Mixture=1, VQ level=132 (number of tests = 1748)					
decoder mixture	*First*	*Second*	*Third*	*Fourth*	*Fifth*
1	65.1% (1138)	80.8% (1412)	87.2% (1525)	90.5% (1581)	92.7% (1620)
2	67.5% (1179)	83.0% (1451)	89.4% (1563)	92.6% (1618)	94.5% (1651)
3	68.3% (1194)	83.8% (1465)	90.2% (1576)	92.9% (1624)	94.4% (1650)
4	68.4% (1196)	84.2% (1472)	89.9% (1571)	93.1% (1627)	94.9% (1658)
5	68.6% (1199)	84.4% (1476)	90.6% (1584)	93.3% (1630)	95.4% (1668)

The error reduction of the simplified unified modelling with time duration is more than 20% in comparison to the discrete version. In comparison to the benchmark work of the discrete HMM and the continuous HMM, the error reduction is now more than 33% and 24% respectively.

9.7. Summary

The major experimental results are summarised in Table 9.10. The unified theory of VQ and hidden Markov modelling of speech signals is shown to offer significant improvement in terms of recognition accuracy in comparison to both the conventional discrete HMM and the conventional continuous HMM (single mixture). In the database used here, the reduction in average recognition error rate of the semi-continuous decoder using the discrete HMM parameters is not significant, (for isolated digit recognition 20%-50% error reduction has been achieved). This may be because the overall recognition accuracy in the phoneme recognition task is not optimised, and the recognition accuracy is relatively low. Nevertheless, it can be seen that the improvement of the unified modelling technique is significant although a simplified

approach is adopted. This strongly suggests the powerful ability of the unified modelling theory. A more complete evaluation of the unified modelling approach will be further explored in the next Chapter.

Table 9.10
Comparison of discrete HMM, continuous HMM, and semi-continuous HMM

A2, Average Recognition (number of tests = 1748)		
model	*VQ level*	*accuracy (correct recognised tests)*
discrete HMM	128	53.2% (930)
+ time duration	128	60.5%(1058)
continuous HMM	-	58.5% (1023)
SCHMM with SVQ	128	60.0% (1049)
+ time duration	128	68.2% (1192)
SCHMM with FVQ	132	61.8% (1081)
+ time duration	132	68.6% (1199)

CHAPTER 10

EXPERIMENTAL RESULTS WITH SPHINX

SPHINX is a state-of-the-art large vocabulary speaker-independent continuous speech recognition system developed at Carnegie Mellon University [96]. The discrete HMM approach based on multiple VQ codebooks has been used in SPHINX. In this Chapter, experimental results for the unified modelling theory will be highlighted in experiments applied to SPHINX. The semi-continuous HMM with multiple VQ codebooks will be evaluated for *large vocabulary speaker-independent continuous* speech recognition in comparison with the discrete HMM and the continuous mixture HMM.

This chapter will report experimental results of

(1) the semi-continuous HMM using cepstrum and bilinear transformed cepstrum;

(2) the continuous mixture HMM using bilinear transformed cepstrum; and

(3) the discrete HMM using cepstrum and bilinear transformed cepstrum

in large vocabulary speaker-independent continuous speech recognition based on SPHINX.

10.1. Database and Analysis Conditions

The task, *resource management*, which is designed for inquiry of naval resources, will be used to evaluate the unified modelling theory. The resource management task was created to evaluate the recognition systems of the recent DARPA speech recognition projects [31,96,128].

At the lexical level, the 997-word resource management task is very complex. There are many confusing pairs, such as *what* and *what's*; *the* and *a*; *four* and *fourth*; *any* and *many*; and many others. Most of the proper nouns can appear in singular, plural, and possessive forms. On the other hand, at the grammatic level, the task is not a very difficult one because the sentences are generated from a set of 900 sentence templates which resemble realistic questions in a database query system.

The most obvious and correct way to model the resource management task language is to use a finite state language that generates the same set of sentences as those 900 templates. As the perplexity of such a grammar is too low (about 9), a grammar that generates all sentences including the 900 sentence templates and some illegal sentences is proposed, i.e., the *word pair grammar*.

The word pair grammar specifies only the list of words that can legally follow any given words, which can be extracted from the 900 sentence templates. Each template is a network of *tags*, or categories of words. Given these templates, what tags can follow any given tags can be easily determined. From this information and the list of words in each tag, what words can follow any given word can then be chosen. Of the 994,009 word pairs, only 57,878 are legal word pairs. This grammar has a test-set perplexity of about 60. To use this grammar for recognition, each word HMM can only follow those word HMMs in the legal word pair set for the given model. The transition probability between the given HMM to the following word HMM is $\frac{1}{K}$, where K is the number of words that can follow the given word.

The complete database of speech consists of 4358 training sentences from 105 speakers and 300 test sentences from 12 speakers. For the tuning experiments conducted here, the training data consist of 2880 sentences from 72 speakers, and the tuning test data consist of 45 sentences from 12 speakers, which are extracted randomly from the 300 test sentences.

For both training and evaluation, the standard SPHINX analysis conditions consist of the following:

sampling rate: 16 kHz
analysis method: bilinear transformed LPC cepstrum
LPC analysis order: 14
cepstrum order: 12
bilinear transformation constant: 0.6
window type: Hamming window
window length and shift: 20 ms and 10 ms
pre-emphasis: $1\text{-}0.97z^{-1}$

10.2. Experimental Results Using Bilinear Transformed Cepstrum

In SPHINX, the signal processing stage is based on bilinear transformed LPC cepstrum, which converts the linear frequency axis into a form of mel-scale [96,154]. The recognition accuracy can be substantially improved because of such a mel-scale representation. This Section will report experimental results based on the bilinear transformed LPC cepstrum.

10.2.1. Results of the discrete HMM

The discrete HMM used here is the same as SPHINX except that only 200 generalised triphones are used instead of 1000 [96]. Three VQ codebooks are used for 12 LPC bilinear transformed LPC cepstral coefficients; 12 differenced bilinear transformed LPC cepstral coefficients; and energy and differenced energy. The generalised triphones are obtained through a greedy context merging procedure from triphone models based on the discrete output probabilities. Function word and phrase modelling are not used in

this experiment [96]. In each HMM, there are 7 states and 12 transitions as shown in Figure 10.2.1. with transition-dependent output probabilities. Three groups of transition-dependent output probabilities are tied as represented by three groups (B, M, and E) in the figure.

The phone model is first reestimated from the training data, and these models are then used as initial parameters for 200 generalised triphones. The discrete output probabilities are finally smoothed by employing deleted interpolation with the phone models and uniform distribution. Viterbi beam search is used for decoding. The percent correct (correct word percentage) and word accuracy (percent correct − percent insertion) results of the discrete HMM are 89.5% and 88.0% respectively. The word accuracy here (88.0%) is significantly lower than SPHINX (91.0%). This is because only

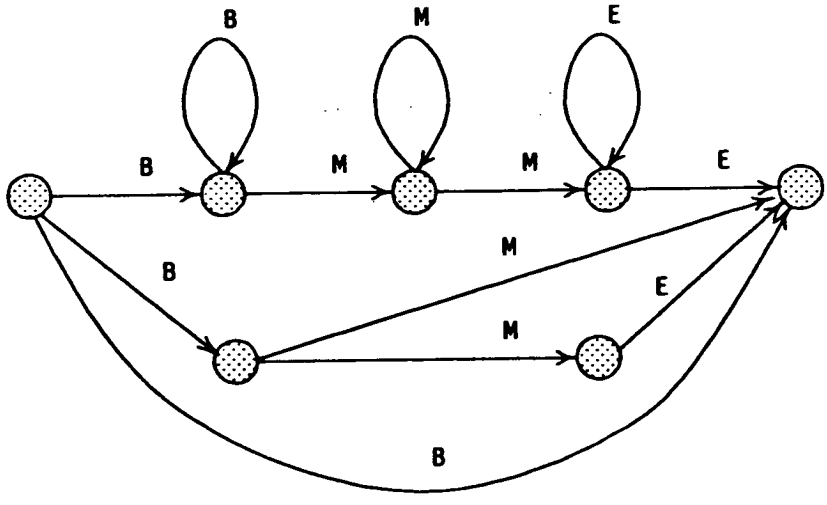


Figure 10.2.1. HMM used in SPHINX

200 generalised triphones are used here. Although use of less detailed modelling units has led to poor performance, the interest here is to compare the performance of different modelling techniques, and 200 generalised triphones should be adequate to see relative differences.

10.2.2. Results of the continuous mixture HMM

In the continuous mixture HMM implemented here, the independence assumption is made for different feature coefficients. The Gaussian density with diagonal covariance can thus be used. The cepstrum, difference cepstrum, normalised energy, and difference energy are packed into one vector. This is similar to the one codebook implementation of the discrete HMM [96]. Unlike the discrete HMM, here different features have different covariance matrices, and such a packing is consistent with the independence assumption. Each continuous output probability consists of 4 diagonal Gaussian probability density functions. To obtain reliable initial models for the continuous mixture HMM, the Viterbi alignment with the discrete HMM is used to phonetically segment and label training speech. These labeled segments are then clustered by using the LBG clustering algorithm to obtain initial means and diagonal covariances. The forward-backward algorithm is used iteratively for the monophone models, which are then used as initial models for the generalised triphone models. The continuous mixture Viterbi beam search is used for decoding.

The percent correct and word accuracy results of the continuous mixture HMM are 84.25% and 81.3% respectively. The word accuracy here (81.30%) is significantly lower than for the corresponding discrete HMM (88.0%). Although the performance of the continuous mixture HMM has been variously reported as significantly better than the performance of the discrete HMM [138], for the experiments conducted here, it is *significantly* worse than the discrete HMM. Explanations for this paradox can be:

(1) Multiple codebooks are used in the discrete HMM, therefore the VQ errors for the discrete HMM are not so serious here. For the discrete HMM based on a single VQ codebook, the performance of the continuous HMM is comparable to that of the discrete HMM.

(2) The number of mixture-components may be too small for speaker-independent large-vocabulary speech recognition, but increase in the number of mixture-components will lead to unaffordable computational complexity.

(3) The diagonal covariance assumption is not appropriate for the bilinear transformed LPC cepstrum since many coefficients are strongly correlated after the transformation. Indeed, investigation of the average covariance matrix for the bilinear transformed LPC cepstrum shows that values of off-diagonal components are generally quite large.

(4) The 200 generalised triphones were obtained based on the discrete output probabilities, which is necessarily sub-optimal for the continuous mixture HMM.

From this experiment, it can be observed that the continuous probability density function must be appropriately chosen according to feature representations. In continuous mixture hidden Markov modelling, the feature representation, the probability density, and the number of mixture-components will be important related factors. In general, the probability density function can be well chosen according to feature representations, but the increase of the number of mixture-components will be restricted by the available training data and computing resources. On the other hand, semi-continuous hidden Markov modelling has distinctive advantages since it is possible to model a mixture of a large number of densities with a limited amount of training data and computational complexity.

10.2.3. Results of unified modelling

For the semi-continuous HMM, multiple codebooks are used instead of packing different feature parameters into one vector as with the continuous mixture HMM. The initial model for the semi-continuous HMM comes directly from the discrete HMM for all the generalised triphones. The initial VQ covariance matrices are obtained from the k-means clustering algorithm based on the VQ codebook. The forward-backward and Baum-Welch algorithm are iteratively used to simultaneously reestimate the model parameters and three VQ codebooks using the standard semi-continuous HMM. Deleted interpolation is finally employed to smooth the discrete output probabilities of the

generalised triphone models with corresponding phone models as well as uniform distributions. The semi-continuous Viterbi beam search is used again for decoding. In computing the semi-continuous output probability density function, only M most significant codewords are used for subsequent processing. Experiments with top one and top four codewords were conducted.

Under the same analysis conditions as previously, the percent correct and word accuracy results of the semi-continuous HMM are shown in Table 10.1. The results of the discrete HMM and the continuous mixture HMM are also listed for comparison.

Table 10.1.
Average recognition accuracy based on 200 generalised triphones 4358 training sentences; 300 test sentences

types	*percent correct (word accuracy)*
Discrete HMM	89.5% (88.0%)
Continuous Mixture HMM	84.2% (81.3%)
Semi-continuous HMM + top1	87.2% (84.0%)
Semi-continuous HMM + top4	90.6% (89.1%)

From Table 10.1, it can be observed that the semi-continuous HMM with top-one codeword has poorer performance than the discrete HMM, but substantially higher than the continuous mixture HMM. This indicates that a mixture of a large number of densities is very helpful. The poor performance of the continuous mixture HMM and the semi-continuous HMM with the top codeword indicates that bilinear transformed cepstral coefficients cannot be well modelled by the diagonal Gaussian assumption. However, the semi-continuous HMM with the top four codewords works modestly better than the discrete HMM although the assumption is inappropriate. In fact, the semi-continuous HMM with the top four codewords works better than both the discrete and continuous mixture HMM. Detailed observations suggest that the semi-continuous HMM can significantly improve the performance of some speakers, but not others. Overall, it is only slightly better than the discrete HMM. The improvement may primarily come from the smoothing effect of the semi-continuous HMM, i.e. the robustness of multiple codewords and multiple codebooks in the semi-continuous output

probability representation. It should be pointed out here that even though 200 generalised triphone models are relatively well trained in comparison to the standard SPHINX version [96], smoothing by multiple codewords can still play an important role. As the diagonal Gaussian assumption may be inappropriate, the covariance matrices need not be reestimated. Indeed, fixed covariance matrices are marginally better than reestimated ones due to inappropriate assumptions.

10.3. Experimental Results Using Less Correlated Data

If the diagonal Gaussian covariance is used, each dimension in the speech vector should not be correlated. In practice, this can be partially satisfied by using less correlated features as acoustic observation representations, or by using principal component projection to reduce correlation. To see the importance of feature representation, experiments with less correlated data were conducted for the discrete HMM and the semi-continuous HMM.

Principal component projection was first used to reduce the correlation of the bilinear transformed LPC cepstrum. In the implementation here, the projection matrix is computed by pooling together the bilinear transformed cepstrum of the whole training sentences, and then computing the eigenvector of that pooled covariance matrix. For the tuning database, result comparison between projected data and un-projected data is shown in Table 10.2. Only insignificant improvements are obtained based on such a projection. This may be because the covariance for each codeword is quite different, and such a projection only makes *average* covariance diagonal, which is inadequate.

Table 10.2.
Average accuracy of projected and un-projected data
2880 training sentences; 45 tuning test sentences

types	*percent correct (word accuracy)*
semi-continuous HMM + top1	87.8% (85.3%)
semi-continuous HMM + top1 + projection	88.3% (85.8%)

As bilinear transformed cepstral coefficients cannot be modelled well by diagonal Gaussian probability density functions, experiments without bilinear transformation were conducted. Here, 18th order cepstral coefficients derived from 18th order LPC analysis are first compared with 12th order cepstral coefficients derived from 14th order LPC analysis. Results for the discrete HMM are shown in Table 10.3. The results using bilinear transformed cepstrum are also included for reference. It is interesting that the recognition accuracy of the 18th order bilinear transformed cepstrum is about the same as that of the 12th order bilinear transformed cepstrum. The mel-scale representation actually has the effect of smoothing high-frequency spectra, which leads to similar performance of cepstrum using different analysis order. In contrast, the recognition accuracy of the 18th order cepstrum is better than that of the 12th order cepstrum, but worse than that of the bilinear transformed cepstrum. This indicates that mel-scale representation is indeed suitable for speaker-independent speech recognition [154]. It should be pointed out here that the generalised triphones are produced from the bilinear transformed LPC cepstrum, which may not be an optimal configuration for other analysis methods.

Table 10.3.
Average accuracy of discrete HMMs (200 generalised triphones) 2880 training sentences; 45 tuning test sentences

types	*percent correct (word accuracy)*
12th cepstrum	84.4% (81.6%)
18th cepstrum	86.1% (82.9%)
12th bilinear transformed cepstrum	88.2% (86.2%)
18th bilinear transformed cepstrum	88.3% (86.2%)

The 18th order cepstrum is used here for the semi-continuous HMM because of less correlated characteristics of the cepstrum. With 4358 training sentences, test results of 300 sentences are listed in Table 10.4.

Table 10.4. Average accuracy of 18th order cepstrum (200 generalised triphones) 4358 training sentences; 300 test sentences	
types	*percent correct (word accuracy)*
Discrete HMM	86.3% (83.8%)
Semi-continuous HMM + top1	86.6% (85.5%)
Semi-continuous HMM + top2	88.8% (87.6%)
Semi-continuous HMM + top4	89.3% (88.5%)
Semi-continuous HMM + top6	89.6% (88.6%)
Semi-continuous HMM + top8	89.3% (88.2%)

Here, the recognition accuracy of the semi-continuous HMM is significantly improved in comparison with the discrete HMM, and error reduction is over 29%. Even the semi-continuous HMM with the top one codeword used is still better than the discrete HMM (85.5% vs. 83.8%). Use of multiple codewords (top4 and top6) in the semi-continuous output probability density function greatly improves the word accuracy (from 85.5% to 88.6%). Further increase of codewords used in the semi-continuous output probability density functions shows no improvement on word accuracy, but substantial growth of computational complexity. From Table 10.4, it can be seen that the semi-continuous HMM with the top four codewords is adequate (88.5%). In contrast, when bilinear transformed data was used (Table 10.1), the error reduction is less than 10% in comparison to the discrete HMM, and the semi-continuous HMM with the top one codeword is actually slightly worse than the discrete HMM. This strongly indicates that appropriate features are very important if continuous probability density function, especially with the diagonal covariance assumption, is used. If this assumption is inappropriate, maximum likelihood estimation will only maximise the *wrong* assumption.

Although more than 29% error reduction has been achieved for 18th order LPC analysis using diagonal covariance assumption, the last results with the discrete HMM (bilinear transformed cepstrum, word accuracy 88.3%) and the semi-continuous HMM (18th order cepstrum, word accuracy 88.6%) are about the same. This suggests that

bilinear transformation is helpful for recognition, but produces correlated coefficients, which is inappropriate to the diagonal Gaussian assumption. Removal of the diagonal covariance assumption by use of full covariance can be expected to further improve recognition accuracy [43]. Regarding use of full covariance, the semi-continuous HMM has a distinctive advantage. Since Gaussian probability density functions are tied to the VQ codebook, by chosing M most significant codewords, computational complexity can be several orders lower than the conventional continuous mixture HMM while maintaining the modelling power of many mixture-components. In addition, all experiments conducted here were based on only 200 generalised triphones; as smoothing can play a more important role in those less-well-trained models, more improvement can be expected for 1000 generalised triphones (where the word accuracy for the discrete HMM is 91% with bilinear transformed data).

10.4. Summary

The applicability of the continuous mixture HMM or the semi-continuous HMM relies on appropriately chosen acoustic parameters and assumption of the continuous probability density function. Acoustic features must be well represented if diagonal covariance is applied to the Gaussian probability density function. This is strongly indicated by the experimental results based on the bilinear transformed cepstrum and cepstrum. The discrete HMM can be substantially improved by bilinear transformation. However, bilinear transformation introduces strong correlations, which is inappropriate for diagonal Gaussian modelling. Using the cepstrum without bilinear transformation, the diagonal semi-continuous HMM can be significantly improved in comparison to the discrete HMM. In contrast, if the bilinear transformed cepstrum is used, the recognition accuracy of the diagonal semi-continuous HMM is only slightly higher than that of the discrete HMM.

This Chapter has demonstrated that the unified modelling approach based on the semi-continuous HMM can offer improved recognition accuracy in comparison to both the discrete HMM and the continuous mixture HMM in speaker-independent continuous speech recognition. Using SPHINX, the semi-continuous HMM reduced error rate of the

discrete HMM and the continuous mixture HMM by 29% (for LPC cepstrum) and 41% (for bilinear transformed cepstrum) respectively. One of the most important factors in unified modelling, as the continuous mixture HMM, is to appropriately choose signal processing methods and probability density functions to achieve optimal performance.

CHAPTER 11

SUMMARY AND CONCLUSIONS

11.1. Summary of Results

This thesis has been concerned with hidden Markov modelling of speech signals. The unified modelling approach to vector quantisation, the discrete HMM, and the continuous HMM is proposed. The semi-continuous HMM incorporates the advantages of both the discrete HMM and the continuous mixture HMM, and results in a powerful tool for modelling speech signals in comparison to other established modelling techniques. The speech recognition accuracy of the unified modelling approach has been shown to be better than both the continuous mixture HMM and the discrete HMM.

The unified modelling theory was developed based on discussions from Chapter 3 to Chapter 6; Chapter 7 then discussed several practical issues that are important for the experimental implementation reported in this thesis.

In Chapter 8, the semi-continuous decoder was first tested in speaker-dependent and speaker-independent mode on a vocabulary of ten digits and compared with conventional discrete HMM and multi-template-based DTW speech recognition systems. The semi-continuous decoder has been shown to offer improved performance (an error reduction of 10%-50%) in comparison to the discrete HMM.

Chapter 9 started the experiments for the unified modelling theory based on semi-continuous HMMs. Speaker-dependent phoneme recognition experiments were conducted; two simplified unified modelling approaches were tested and have been shown to offer significant improvement in terms of recognition accuracy in comparison to both the conventional discrete HMM and the continuous HMM (single mixture). Different numbers of mixture densities were also tested in both the semi-continuous

decoder and reestimation procedure. The optimal number of mixture densities ranges from three to four in these experiments. The use of limited number of mixture densities can not only improve the performance but also significantly reduce the amount of computation. The average recognition error rate of the semi-continuous decoder using the discrete HMM parameters can be reduced in comparison to the discrete mode, but not as significantly as by the unified modelling techniques. The improvements of the unified modelling theory are best for the phoneme classification task, though simplified approaches are used. The HMM with time duration has also been evaluated. Experimental results again show that the unified modelling theory proposed in this thesis works significantly better than standard HMMs. These experimental results strongly suggest the powerful ability of the semi-continuous HMM.

Finally, the unified modelling theory was highlighted in Chapter 10. Experiments for large-vocabulary speaker-independent continuous speech recognition were conducted. The semi-continuous HMM was compared with the discrete HMM and the continuous mixture HMM in the DARPA resource management task. Experimental results have clearly demonstrated that the semi-continuous HMM offers improved recognition accuracy in comparison to both the discrete HMM (an error reduction of 29%) and the continuous mixture HMM (an error reduction of 41%). However, the applicability of either the continuous mixture HMM or the semi-continuous HMM relies on appropriately chosen acoustic parameters and assumption of the continuous probability density function.

11.2. Conclusions

As introduced previously, sufficient training data, automatic learning algorithms, and detailed modelling are three very important factors to a successful speech recognition system. The significance of the unified modelling approach is that it can model a mixture of a large number of probability density functions with a limited amount of training data. In the semi-continuous output probability, robustness can be enhanced by using multiple codewords. In addition, the VQ codebook itself can be adjusted together with the HMM parameters in order to obtain the optimum maximum

likelihood of the HMM. The unified modelling approach can therefore achieve an optimal combination of HMM parameters and VQ codebook parameters. These merits of the unified modelling can be viewed as a good solution to the conflict between detailed acoustic modelling and insufficient training data. From the continuous HMM point of view, detailed acoustic modelling can be achieved by increasing mixture densities. However, the performance will suffer from insufficient training data if there are too many free parameters by increasing mixture densities. The semi-continuous HMM can be considered as a special form of continuous mixture HMM with tied mixture continuous density functions. Because of the binding of the continuous density functions, in the semi-continuous HMM, the number of free parameters and computational complexity are reduced in comparison to the continuous mixture HMM while retaining the modelling powers of continuous HMM with a mixture of a large number of probability density functions. From the discrete HMM point of view, detailed acoustic modelling can be achieved by increasing the size of VQ codebook. However, once again, the performance will suffer from insufficient training data. On the other hand, the multiple codewords representation of the semi-continuous HMM can be well immunised from such a problem. In addition, the VQ codebook itself can be globally optimised together with the HMM.

The reestimation of the VQ codebook can also be viewed as a special form of maximum likelihood VQ [70,119]. Although both the maximum likelihood VQ and the HMM forward-backward algorithms attempt to maximise the likelihood, the new unified modelling approach has some obvious advantages, namely it maximises the likelihood of the HMM. As a result, later estimates are more suitable for classification when the pre-classified data and the HMM parameters are optimised together.

For a variety of experiments conducted in different tasks, the semi-continuous HMM has clearly demonstrated its superior performance. The applicability of the continuous mixture HMM or the semi-continuous HMM relies on appropriately chosen acoustic parameters and assumption of the continuous probability density function. This is strongly indicated by the experimental results based on the bilinear transformed cepstrum and the cepstrum described in this thesis. Acoustic features must be well represented if diagonal covariance is applied to the Gaussian probability density

function since the maximum likelihood estimation criterion is used in the experiments.

11.3. Future Work

The argument for maximum likelihood estimation is based on an assumption that the true distribution of speech is a member of the family of distributions used in the estimation. However, as shown in experiments using the bilinear transformed cepstrum, this can well be challenged. Typical HMMs make many inaccurate assumptions about the speech production process. Therefore, an estimation criterion that can work well in spite of these inaccurate assumptions should offer improved recognition accuracy in comparison to the maximum likelihood criterion. Other reestimation criteria, such as the maximum mutual information criterion [26] or corrective training criterion [12,97], can be used. The maximum mutual information estimation is based on minimisation of the average uncertainty of the word sequence to be recognised, given the acoustic observations instead of finding *true* model parameters [26]. Therefore, the invalid argument for maximum likelihood estimation can be corrected. Suppose the language model is given, a possible solution is then to maximise the average mutual information between the acoustic observation sequence and the complete set of models $(\lambda_1, \lambda_2, \cdots \lambda_v)$. If all words are equiprobable, one possible criterion is

$$I = \max_{\lambda}[\sum_{v}(\log Pr(\mathbf{O}^v|\lambda_v) - \log\sum_{w} Pr(\mathbf{O}^v|\lambda_w))]$$

i.e., choose λ so as to separate the correct model λ_v from all other models on the training sequence $\mathbf{O}^v$. By summing over all training observations, one would hope to attain the most separated set of models possible. The steepest descent methods can be used to maximise the above equation [11]. Alternatively, the heuristic *corrective* reestimation procedure [12,97], attempts to maximise recognition accuracy on the training data. This algorithm tests the decoder using reestimated models according to training data in the training procedure, and subsequently improves the correct models and suppresses misrecognised or near-miss models. Another similar approach is to combine discriminant analysis into hidden Markov modelling in which the *between class* matrices can be obtained from these *misrecognised* data [43]. Such a projection can significantly reduce recognition error rate. Though the maximum likelihood

estimation criterion is used for evaluation experiments of the unified modelling theory, other techniques, which can be applied to conventional hidden Markov modelling, can be readily applied to the unified modelling approach.

Regarding unified modelling, this thesis presented a complete theory for VQ and acoustic pattern matching. For continuous speech recognition, the pronunciation dictionary, which maps words into phoneme sequences, is usually generated according to phonetic knowledge. The phoneme sequences are actually unrelated to the acoustic pattern matching. It is therefore attractive to incorporate the dictionary into the maximum likelihood estimation framework thus achieving unified modelling from VQ, acoustic pattern matching to pronunciation dictionary.

While it can be concluded that the unified modelling theory based on the semi-continuous HMM is a powerful technique for modelling non-stationary stochastic processes with multi-modal probabilistic functions of Markov chains, by itself it has not totally solved the general speech recognition problem, but has provided some insight into the acoustic-modelling problem; insight offered here for future researchers to visit, enhance and report.

APPENDIX A. PUBLICATIONS BY AUTHOR

Refereed Journal Papers

[1] X. Huang and M. Jack, Semi-continuous hidden Markov models for speech signals, *Computer Speech and Language, Vol. 3, pp. 239-251, 1989*

[2] X. Huang, G. Duncan and M. Jack, Formant estimation based on weighted least square adaptive filters, *IEE Proceedings Part F, Communications, Radar and Signal Processing, Vol. 135, pp. 539-546, 1988*

[3] X. Huang, M. Jack and Y. Ariki, Parameter re-estimation of semi-continuous hidden Markov models with feedback to vector quantization codebook, *IEE Electronics Letters, Vol. 24, No. 22, pp. 1375-1377, 1988*

[4] X. Huang and M. Jack, Performance comparison between semi-continuous and discrete hidden Markov models, *IEE Electronics Letters, Vol. 24, No. 3, pp. 149-150, 1988*

[5] X. Huang and M. Jack, Hidden Markov modelling of speech based on a semi-continuous model, *IEE Electronics Letters, Vol. 24, No. 1, pp. 6-7, 1988*

[6] X. Huang, M. Jack and G Duncan, Formant labelling using pole-focused spectra, *IEE Electronics Letters, Vol. 23, pp. 1047-1048, 1987*

[7] X. Huang, G. Duncan and M.A. Jack, Automatic formant estimation algorithm based on pointwise analysis, *IEE Electronics Letters, Vol. 23, pp. 807-809, 1987*

[8] X. Huang, L. Cai, D. Fang, B. Chi, L. Zhou, and L. Jiang, A computer system for Chinese speech input, *Journal of Computer Science and Technology, Vol. 1, pp. 75-83, 1986*

Conference Papers

[1] X. Huang, H. Hon and K. Lee, Large-vocabulary speaker independent continuous speech recognition using continuous and semi-continuous hidden Markov models, *European Conference on Speech Technology, Paris, 1989*

[2] X. Huang and M. Jack, Unified modelling of vector quantization and hidden Markov model using semi-continuous hidden Markov models, *IEEE ICASSP 89, pp. 639-642, Glasgow, 1989*

[3] X. Huang and M. Jack, Semi-continuous hidden Markov models with Maximum Likelihood VQ, *IEEE Workshop on Speech Recognition, Arden House, (Harriman, NY) 1988*

[4] X. Huang and M. Jack, On several problems of hidden Markov models, *Seventh FASE Symposium, SPEECH 88, Edinburgh, pp. 17-22, 1988*

[5] X. Huang and M. Jack, Maximum likelihood clustering applied to semi-continuous hidden Markov models for speech recognition, *IEEE International Symposium on Information Theory, Kobe, Japan, p. 71, 1988*

[6] X. Huang, G. Duncan and M. Jack, Improved formant estimation using a pointwise analysis method, *Digital Signal Processing - 87, Florence, Italy, Edited by V. Cappellini and A. Constantinides, North-Holland, pp. 814-818, 1987*

[7] X. Huang, G. Duncan and M. Jack, Formant estimation algorithm based on temporal synchronous analysis, *Proceedings of the European Conference on Speech Technology, Edinburgh, Scotland, pp. 331-334, 1987*

[8] X. Huang, L. Cai, and D. Fang, A large-vocabulary Chinese speech recognition system, *IEEE ICASSP 87, Dallas, USA, pp. 1167-1170, 1987*

[9] X. Huang, D. Fang, and L. Cai, State-guided dynamic programming (SGDP) algorithm for speech recognition, *IEEE Workshop on ASSP, Beijing, China, pp. 233-236, 1986*

References

1. A.V. Aho and J.D. Ullman, "The Theory of Parsing, Translation and Computing," *Prentice Hall*, 1972.
2. M.R. Anderberg, "Cluster analysis for applications," *Academic Press, New York*, 1973.
3. Y. Ariki and M.A. Jack, "Enhanced time duration constraints in hidden Markov modelling for phoneme recognition," *IEE Electronics Letters*, vol. 25, 1989.
4. B.S. Atal and S.L. Hanauer, "Speech Analysis and Synthesis by Linear Prediction," *J. Acoustic Soc. America*, vol. 50, pp. 537-655, 1971.
5. B.S. Atal and L.R. Rabiner, "A Pattern Recognition Approach to Voiced-Unvoiced-Silence Classification with Applications to Speech Recognition," *IEEE Trans. Acoustic, Speech, and Signal Processing*, vol. ASSP-24, pp. 201-212, 1976.
6. B.S. Atal, "Predictive coding a speech signals and subjective error criteria," *IEEE Trans. Acoustic, Speech, and Signal Processing*, vol. ASSP-27, pp. 247-254, 1979.
7. A. Averbuch and *et al.*, "Experiments with the Tangora 20,000 word speech recognizer," *Proc. ICASSP-87, Dallas, USA*, pp. 701-704, 1987.
8. L. Bahl, P. Brown, P. de Souza, P. Gopalakrishnan, J. Jelinek, and R. Mercer, "Large vocabulary natural language continuous speech recognition," *Proc. ICASSP-89, Glasgow, Scotland*, pp. 465-467, 1989.
9. L.R. Bahl and *et al.*, "Further Results on the Recognition of a Continuously Read Natural Corpus," *Proc. ICASSP-80, USA*, 1980.
10. L.R. Bahl, F.Jelinek, and R. Mercer, "A maximum likelihood approach to continuous speech recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. PAMI-5, pp. 179-190, 1983.
11. L.R. Bahl, P.F. Brown, P.V. de Souza, and R.L. Mercer, "Maximum Mutual Information Estimation of Hidden Markov Model Parameters for Speech Recognition," *Proc. ICASSP-86, Tokyo, Japan*, pp. 49-52, 1986.
12. L.R. Bahl, P.F. Brown, P.V. de Souza, and R.L. Mercer, "A new algorithm for the estimation of hidden Markov parameters," *Proc. ICASSP-88, NY, USA*, 1988.
13. L.R. Bahl, P.F. Brown, P.V. de Souza, and R.L. Mercer, "Acoustic Markov models used in the Tangora speech recognition system," *Proc. ICASSP-88, NY, USA*, 1988.
14. J. Baker, "The DRAGON system -- An overview," *IEEE Trans. Acoustic, Speech, and Signal Processing*, vol. ASSP-23, pp. 24-29, 1975.
15. J. Baker, "Stochastic Modeling as a means of automatic speech recognition," *Ph.D. Thesis, Dept. of Computer Science, Carnegie-Mellon University*, 1975.
16. J. Baker, "Trainable Grammars for Speech Recognition," *Proc. Spring Conference Acoustic Soc. America*, pp. 547-550, 1979.
17. L.E. Baum and T. Petrie, "Statistical inference for probabilistic functions of finite state Markov chains," *Ann. Math. Stat.*, vol. 37, pp. 1559-1563, 1966.
18. L.E. Baum and J.E. Eagon, "An inequality with applications to statistical estimation for probabilistic functions of a Markov process and to a models for ecology ," *Bull. AMS* , vol. 73, pp. 360-363, 1967.

19. L.E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occuring in the statistical analysis of probabilistic functions of Markov chains," *Ann. Math. Stat.*, vol. 41, pp. 164-171 , 1970.

20. L.E. Baum, "An inequality and associated maximization technique in statistical estimation of probabilistic functions of Markov processes," *Inequalities*, vol. 3, pp. 1-8, 1972.

21. H. Bourlard and C. Wellekens, "Links between Markov models and multilayer perceptrons," *Philips Manuscript no. M263*, 1988.

22. H. Bourlard and C. Wellekens, "Speech dynamics and recurrent neural networks," *Proc. ICASSP-89, Glasgow, Scotland*, pp. 33-36, 1989.

23. J.S. Bridle, M.D. Brown, and R.M. Chamberlain, "An Algorithm for Connected Word Recognition," *Proc. ICASSP-82, Paris, France*, pp. 899-902, 1982.

24. J.S. Bridle, K.M. Ponting, M.D. Brown, and A.W. Borrett, "A Noise Compensating Spectrum Distance Measure Applied to Automatic Speech Recognition," *Proc. ICASSP-84, San Diego, USA*, pp. 307-314, 1984.

25. J.S. Bridle, "Stochastic Models and Template Matching: Some Important Relationships Between Two Apparently Different Techniques for Automatic Speech Recognition," *Inst. of Acoustic Autumn Conference*, 1984.

26. P.F. Brown, "Acoustic-phonetic modeling problem in automatic speech recognition," *Ph.D. Thesis, Dept. of Computer Science, Carnegie-Mellon University*, 1987.

27. G. Bruno and *et al.*, "A Bayesian-Adaptive Decision Method for V/UV/S Classification of Segments of a Speech Signal ," *IEEE Trans. Acoustic, Speech, and Signal Processing*, vol. ASSP-35, pp. 556-559, 1987.

28. D.J. Burr, "Speech Recognition Experiments with Perceptrons," *AIP Conference Proceeding, Neural Information Processing System, Denver*, 1987.

29. D.K. Burton, J.E. Shore, and J.T. Buck, "Isolated-word speech recognition using multisection vector quantization codebook," *IEEE Trans. Acoustic, Speech, and Signal Processing*, vol. ASSP-33, pp. 837-848, 1985.

30. Y.L. Chow, R.M. Schwartz , S. Roucos, O.A. Kimball, P.J Price, G.F. Kubala, M.D. Dunham, M.A. Kranser, and J. Makhoul, "The role of word-dependent coarticulatory effects in a phoneme-based speech recognition system," *Proc. ICASSP-86, Tokyo, Japan.*

31. Y.L. Chow, M.D. Dunham, O.A. Kimball, M.A. Kranser, G.F. Kubala, J. Makhoul, P.J Price, S. Roucos, and R.M. Schwartz , "BYBLOS: The BBN continuous speech recognition system," *Proc. ICASSP-87, Dallas, USA*, pp. 89-92, 1987.

32. Y.-L. Chow and S. Roukos, "Speech understanding using a unification grammar," *Proc. ICASSP-89, Glasgow, Scotland*, pp. 727-730, 1989.

33. R. Cole, M. Phillips, B. Brennan, and B. Chigier, "The CMU phonetic classification system ," *Proc. ICASSP-86, Tokyo, Japan* , pp. 2255-2258, 1986.

34. T.M. Cover and R.C. King, "A Convergent Gambling of the Entropy of English," *IEEE Trans. Information Theory*, vol. IT-24, pp. 413-421, 1978.

35. B.A. Dautrich, L.R. Rabiner, and T.B. Martin, "On the Effects of Varying Filter Bank Parameters on Isolated Word Recognition," *IEEE Trans. Acoustic, Speech, and Signal Processing*, vol. ASSP-31, pp. 793-806, 1983.

36. S.B. Davis and P. Mermelstein, "Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences," *IEEE Trans. Acoustic, Speech, and Signal Processing*, vol. ASSP-28, pp. 357-366, 1980.

37. N.E. Day, "Estimating the Component of a Mixture of Normal Distributions," *Biometrika*, vol. 56, pp. 463-474, 1969.

38. A.P. Dempster, N.M. Laird, and D.B. Rubin, "Maximum-likelihood from incomplete data via the EM algorithm," *J. Royal Statist. Soc. Ser. B (methodological)*, vol. 39, pp. 1-38, 1977.

39. L. Deng, M. Lennig, V. Gupta, and P. Mermelstein, "Modeling acoustic-phonetic detail in an HMM-based large-vocabulary speech recognizer," *Proc. ICASSP-88, NY, USA*, pp. 509-512, 1988.

40. A. Derouault, "Context-Dependent Phonetic Markov Models for Large Vocabulary Speech Recognition," *Proc. ICASSP-87, Dallas, USA*, pp. 360-363, 1987.

41. A.M. Derouault and B. Merialdo, "Natural Language Modeling for Phoneme-to-Text Transcription," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. PAMI-8, pp. 742-749, 1986.

42. P.A. Devijver and J. Kittler, "Pattern Recognition: A Statistical Approach," *Prentics/Hall International*, 1982.

43. G.R. Doddington, "Phonetically sensitive discriminants for improved speech recognition," *Proc. ICASSP-89, Glasgow, Scotland*, pp. 556-559, 1989.

44. J. Doob, "Stochastic Processes," *New York: Wiley, p. 10*, 1953.

45. R.O. Duda and R.E. Hart, "Pattern classification and scene analysis," *New York, NY:Wiley*, 1973.

46. P. Dumouchel, V. Gupta, M. Lennig, and P. Mermelstein, "Three Probabilistic Language Models for a Large-Vocabulary Speech recognizer," *Proc. ICASSP-88, NY, USA*, pp. 513-516, 1988.

47. J. Earley, "An efficient context-free parsing algorithm," *Communications ACM*, vol. 13, pp. 94-102, 1970.

48. S. Euler and D. Wolf, "Speaker independent isolated word recognition based on continuous hidden Markov models using multidimensional spherically invariant functions," *Digital Signal Processing - 87, Florence, Italy, Edited by V. Cappellini and A. Constantinides, North-Holland* , pp. 539-542, 1987.

49. F. Fallside and W. Woods, "Computer speech processing," *Prentice/Hall International, UK*, 1985.

50. M. Feder, A.V. Oppenheim, and E. Weinstein, "Methods for noise cancellation based on the EM algorithm," *Proc. ICASSP-87, Dallas, USA*, pp. 201-204, 1987.

51. M. Ferretti, G. Maltese, and S. Scarci, "Language model and acoustic model information in probabilistic speech recognition," *Proc. ICASSP-89, Glasgow, Scotland*, pp. 707-710, 1989.

52. E.W. Forgy, "Cluster analysis of multivariate data: Efficiency vs. interpretability of classifications," *Biometrics*, vol. 21, p. 768, abstract, 1965.

53. G.D. Forney, "The Viterbi Algorithm," *Proc. IEEE*, vol. 61, pp. 268-278, 1973.

54. M. Franzini, M. Witbrock, and K. Lee, "A connectionist approach to continuous speech recognition," *Proc. ICASSP-89, Glasgow, Scotland*, pp.

425-428, 1989.

55. S. Furui, "Speaker-Independent Isolated Word Recognition Using Dynamic Features of Speech Spectrum," *IEEE Trans. Acoustic, Speech, and Signal Processing*, vol. ASSP-34, pp. 52-59, 1986.

56. A. Gersho, "On the structure of vector quantizers," *IEEE Trans. Information Theory*, vol. IT-28, pp. 157-166, 1982.

57. P.E. Gill, W. Murray, and M.H. Wright, "Practical Optimization," *Academic Press*, 1981.

58. B. Gold, R.P. Lippmann, and M.L. Malpass, "Some neural net recognition results on isolated words," *IEEE International Conference on Neural Networks*, 1987.

59. A.H. Gray and J.D. Markel, "Distance Measures for Speech Processing," *IEEE Trans. Acoustic, Speech, and Signal Processing*, vol. ASSP-24, pp. 380-391, 1976.

60. R. Gray, A. Buzo, A. Gray, and Y. Matusyama, "Distortion Measures for Speech Processing," *IEEE Trans. Acoustic, Speech, and Signal Processing*, vol. ASSP-28, pp. 367-376, 1980.

61. R.M Gray and E.D. Karnin, "Multiple local optima in vector quantizers," *IEEE Trans. Information Theory*, vol. IT-28, pp. 256-261, 1982.

62. R.M. Gray, "Vector quantization," *IEEE ASSP Magazine*, pp. 4-29, April, 1984.

63. V.N. Gupta, M. Lennig, and P. Mermelstein, "Integration of Acoustic Information in a Large Vocabulary Word Recognizer," *Proc. ICASSP-87, Dallas, USA*, pp. 697-700, 1987.

64. T. Harrison and F. Fallside, "A connectionist model for phoneme recognition in continuous speech," *Proc. ICASSP-89, Glasgow, Scotland*, pp. 417-420.

65. V. Hasselblad, "Estimation of Parameters for a Mixture of Normal Distributions," *Technometrics*, vol. 8, pp. 431-44, 1966.

66. J. Haton, N. Carbonnel, D. Fohr, J. Mari, and A. Kriouille, "Interaction between stochastic modeling and knowledge-based techniques in acoustic-phonetic decoding of speech," *Proc. ICASSP-87, Dallas, USA* , pp. 868-871, 1987.

67. C. Hemphill and J. Picone, "Speech recognition in a unification grammar framework," *Proc. ICASSP-89, Glasgow, Scotland*, pp. 723-726, 1989.

68. J. Hopcraft and J. Ullman, "Introduction to Automata Theory, Languages and Computation," *Addison-Wesley, Reading, MA*, 1979.

69. W.Y. Huang and R.P. Lippmann, "Comparison between neural net and conventional classifiers," *IEEE International Conference on Neural Networks*, 1987.

70. X.D. Huang and M.A. Jack, "Maximum likelihood clustering applied to semi-continuous hidden Markov models for speech recognition ," *IEEE International Symposium on Information Theory, Kobe, Japan* , p. 71, 1988.

71. X.D. Huang, M.A. Jack, and Y. Ariki, "Parameter re-estimation of semi-continuous hidden Markov models with feedback to vector quantization codebook," *IEE Electronics Letters*, vol. 24, no. 22, pp. 1375-1377, 1988.

72. X.D. Huang and M.A. Jack, "Hidden Markov modelling of speech based on a semi-continuous model," *IEE Electronics Letters*, vol. 24, no. 1, pp. 6-7, 1988.

73. X.D. Huang and M.A. Jack, "Unified modeling of vector quantization and hidden Markov model using semi-continuous hidden Markov models," *Proc. ICASSP-89, Glasgow, Scotland*, pp. 639-642, 1989.

74. X.D. Huang, H.W. Hon, and K.F. Lee, "Large-vocabulary speaker-independent continuous speech recognition with semi-continuous hidden Markov models ," *Eurospeech 89, Paris, France*, 1989.

75. X.D. Huang and M.A. Jack, "Semi-continuous hidden Markov models for speech recognition," *Computer Speech and Language*, vol. 3, pp. 239-251, 1989.

76. M.J. Hunt and C. Lefebvre, "A comparison of several acoustic representation for speech recognition with degraded and undegraded speech," *Proc. ICASSP-89, Glasgow, Scotland*, pp. 262-265, 1989.

77. M. Hwang, H. Hon, and K. Lee, "Modelling between-word coarticulation in continuous speech recognition," *Eurospeech 89, Paris, France*, 1989.

78. F. Itakura and S. Saito, "A statistical method for estimation of speech spectral density and formant frequencies," *Electron. Commun. Japan*, vol. 53-A, pp. 36-43, 1970.

79. F. Itakura, "Minimum Prediction Residual Principle Applied to Speech Recognition," *IEEE Trans. Acoustic, Speech, and Signal Processing*, vol. ASSP-23, pp. 67-72, 1975.

80. F. Jelinek, "Continuous speech recognition by statistical methods," *Proceedings of IEEE*, vol. 64, pp. 532-556, 1976.

81. F. Jelinek and R.L. Mercer, "Interpolated estimation of Markov source parameters from sparse data," *Proceedings of the workshop on pattern recognition in practice, Amsterdam, The Netherlands: North-Holland*, 1980.

82. F. Jelinek, "The development of an experimental discrete dictation recognizer," *Proceedings of IEEE*, vol. 73, pp. 1616-1624 , 1985.

83. F. Jelinek, "Self-organized language modeling for speech recognition," *IBM Europe Institute, Advances in Speech Processing, Austria*, 1986.

84. B.-H. Juang, "On the Hidden Markov Model and Dynamic Time Warping for Speech Recognition --- A Unified View," *AT&T Bell Laboratories Tech. J.*, vol. 63, pp. 1213-1243, 1984.

85. B.H. Juang and L.R. Rabiner, "Mixture autoregressive hidden Markov models for speech signals," *IEEE Trans. Acoustic, Speech, and Signal Processing*, vol. ASSP-33, pp. 1404-1413, 1985.

86. B.H. Juang, "Maximum-likelihood estimation for mixture multivariate stochastic observations of Markov chain," *AT&T Technical Journal*, vol. 64, pp. 1235-1249, 1985.

87. S. Katz, "Estimation of probabilities from sparse data for the language model component of a speech recognizer," *IEEE Trans. Acoustic, Speech, and Signal Processing*, vol. ASSP-35, pp. 400-401, 1987.

88. K. Kita, T. Kawabata, and H. Saito, "HMM continuous speech recognition using predictive LR parsing," *Proc. ICASSP-89, Glasgow, Scotland*, pp. 703-706, 1989.

89. D. Klatt, "Review of the ARPA speech understanding project," *J. Acoustic Soc. America*, vol. 62, pp. 1345-1366, 1977.

90. D. Klatt, "Problem of variability in speech recognition and in models of speech perception," *In J. Perkell and D. Klatt (editor), Variability and Invariance in Speech Processes, Lawrence Erlbaum Assoc. N.J.*, pp. 300-320,

1986.

91. T. Kohonen, "Self-Organization and Associative Memory," *Springer-Verlag, Berlin*, 1984.

92. G.E. Kopec, "Formant tracking using hidden Markov models and vector quantization," *IEEE Tran. ASSP*, vol. ASSP-34, pp. 709-729, 1986.

93. G.F. Kubala, Y. Chow, A. Derr, M. Feng, O. Kimball, and J. Makhoul, S. Kullback, and R.A. Leibler, "On information and sufficiency," *Ann. Math. Stat*, vol. 22, pp. 79-86, 1951.

94. C.H. Lee, F.K. Soong, and B.H. Juang, "A segment model based approach to speech recognition," *Proc. ICASSP-88, NY, USA*, 1988.

95. C.H. Lee, B.H. Juang, F.K. Soong, and L.R. Rabiner, "Word recognition using whole word and subword models," *Proc. ICASSP-89, Glasgow, Scotland*, pp. 683-686, 1989.

96. K.F. Lee, "Large-vocabulary speaker-independent continuous speech recognition: The SPHINX system," *Ph.D. thesis, Dept. of Computer Science, Carnegie-Mellon University (also "Automatic Speech Recognition: The Development of the SPHINX System", Kluwer Academic Publishers, 1989)*, 1988.

97. K.F. Lee, "Hidden Markov models: past, present, and future," *Eurospeech 89, Paris, France*, 1989.

98. K.F. Lee, H.W. Hon, and R. Reddy, "The SPHINX speech recognition system," *Proc. ICASSP-89, Glasgow, Scotland*, pp. 445-449, 1989.

99. S.E. Levinson, L.R. Rabiner, and M.M. Sondhi , "An introduction to the application of theory of probabilistic functions of a Markov process to automatic speech recognition," *Bell System Technical Journal*, vol. 62, pp. 1035-1074, 1983.

100. S.E. Levinson, "Continuously variable duration hidden Markov models for automatic speech recognition," *Computer Speech and Language*, vol. 1, pp. 29-45, 1986.

101. Y. Linde, A. Buzo, and R.M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Commun.*, vol. COM-28, pp. 84-95, 1980.

102. L.R. Liporace, "Maximum likelihood estimation for multivariate observations of Markov sources," *IEEE Trans. Information theory*, vol. IT-28, pp. 729-734, 1982.

103. R.P Lippmann, "Neural nets for computing," *Proc. ICASSP-88, NY, USA*, pp. 1-6 , 1988.

104. R.P. Lippmann, "An introduction to computing with neural nets," *IEEE ASSP Magazine*, pp. 4-22, 1987.

105. R.P. Lippmann and B. Gold, "Neural-net classifiers useful for speech recognition," *IEEE International Conference on Neural Networks*, 1987.

106. B.T. Lowerre and D.R. Reddy, "The Harpy Speech Understanding System," *Trends in Speech Recognition, Prentice-Hall*, 1980.

107. J. Makhoul, S. Roucos, and H. Gish, "Vector quantisation in speech coding," *Proceedings of IEEE*, vol. 73, pp. 1551-1588, 1985.

108. J.D. Markel and A.H. Gray, "Linear Prediction of Speech," *Spring-Verlag*, 1976.

109. A.A. Markov, "An Example of Statistical Investigation in the Text of *Eugen Onyegin* Illustrating Coupling of *Tests* in Chains," *Proc. Acad. Sci. St., Petersburgh VI Ser.*, vol. 7, pp. 153-162, 1913.

110. F. McInnes, M.A. Jack, and J. Laver, "Experiments with template adaptation in an isolated word recognition system," *European Conference on Speech Technology, Edinburgh, Scotland*, pp. 484-487, 1987.

111. F. McInnes, "Adaptation of Reference Patterns in Word-Based Speech Recognition," *Ph.D. Thesis, Faculty of Science, University of Edinburgh*, 1988.

112. M. Miller and D. Snyder, "The role of likelihood and entropy in incomplete-data problems: application to estimating point-process intensities and Toeplitz constrained covariances," *Proceedings of the IEEE*, vol. 75, pp. 892-907, 1987.

113. H. Murveit and M. Weintraub, "Speaker-independent connected speech recognition using hidden Markov models," *Proc. ICASSP-88, NY, USA*, 1988.

114. C.S. Myers and L.R. Rabiner, "A Level Building Dynamic Time Warping Algorithm for Connected Word Recognition," *IEEE Trans. Acoustic, Speech, and Signal Processing*, vol. ASSP-29, pp. 284-297, 1981.

115. A. Nadas, L.R. Bahl, R. Bakis, P.S. Cohen, A.G. Cole, F. Jelinek, and B.L. Lewis, "Continuous speech recognition with automatically selected acoustic prototypes obtained by either bootstrapping or clustering," *Proc. ICASSP-81, Atlanta, USA*, pp. 1153-1155, 1981.

116. A. Nadas, "A decision theoretic formulation of a training problem in speech recognition and a comparison of training by unconditional versus conditional maximum likelihood ," *IEEE Trans. Acoustic, Speech, and Signal Processing* , vol. ASSP-31, pp. 814-817, 1983.

117. A. Nadas, "Estimation of probabilities in the language model of the IBM speech recognition system ," *IEEE Trans. Acoustic, Speech, and Signal Processing* , vol. ASSP-32, pp. 859-861, 1984.

118. A. Nadas, "On Turing's formula for word probabilities," *IEEE Trans. Acoustic, Speech, and Signal Processing*, vol. ASSP-33, pp. 1432-1436, 1985.

119. A. Nadas and D. Nahamoo, "Automatic speech recognition via pseudo-independent marginal mixtures ," *Proc. ICASSP-87, Dallas, USA*, pp. 1285-1287, 1987.

120. M. Nakamura and K. Shikano, "A study of English word category prediction based on neural netowrks," *Proc. ICASSP-89, Glasgow, Scotland*, pp. 731-734, 1989.

121. H. Ney, "The Use of a One-stage Dynamic Programming Algorithm for Connected Word Recognition," *IEEE Trans. Acoustic, Speech, and Signal Processing*, vol. ASSP-32, pp. 263-271, 1984.

122. H. Ney, D. Mergel, A. Noll, and A. Paeseler, "A Data-Driven Organization of the Dynamic Programming Beam Search for Continuous Speech Recognition," *Proc. ICASSP-87, Dallas, USA*, pp. 833-836, 1987.

123. M. Nishimura and K. Toshioka, "HMM-based speech recognition using multi-dimensional multi-labeling," *Proc. ICASSP-87, Dallas, USA*, pp. 1163-1166, 1987.

124. N. Nocerino, F.K. Soong, L.R. Rabiner, and D.H. Klatt, "Comparative Study of Several Distortion Measures for Speech Recognition," *Proc. ICASSP-85, Tampa, USA*, pp. 25-28, 1985.

125. D.B. Paul, "An 800 bps adaptive vector quantization vocoder using a perceptual distance measure," *Proc. ICASSP-83, USA*, pp. 73-76, 1983.

126. D.B. Paul, R.P. Lippmann, Y. Chen, and C. Weinstein, "Robust HMM-based Techniques for Recognition of Speech Produced under Stress and in Noise," *Speech Technology*, 1986.

127. D.B. Paul and E.A. Martin, "Speaker stress-resistant continuous speech recognition," *Proc. ICASSP-88, NY, USA*, 1988.

128. D.B. Paul, "The Lincoln robust continuous speech recognizer," *Proc. ICASSP-89, Glasgow, Scotland*, pp. 449-452, 1989.

129. S.M. Peeling, R.K. Moore, and M.J. Tomlinson, "The multi-layer perceptron as a tool for speech pattern processing research," *Proceeding Institute of Acoustics Autumn Conference*, 1986.

130. T Petrie, "Probabilistic functions of finite state Markov chains," *Ann. Math. Stat.* , vol. 40, pp. 97-115, 1969.

131. A.B. Poritz, "Linear Predictive Hidden Markov Models and The Speech Signal," *Proc. ICASSP-82, Paris, France*, pp. 1291-1294, 1982.

132. A.B. Poritz and A.G. Richter, "On hidden Markov models in isolated word recognition," *Proc. ICASSP-86, Tokyo, Japan*, pp. 705-708, 1986.

133. A.B. Poritz, "Hidden Markov Models: A Guided Tour," *Proc. ICASSP-88, NY, USA*, pp. 7-13, 1988.

134. R. W. Prager, T.D. Harrison, and F. Fallside, "Boltzman machines for speech recognition," *Computer Speech & Language*, vol. 1, pp. 3-27, 1986.

135. L.R. Rabiner and R.W. Schafer, "Digital Processing of Speech Signals," *Prentice Hall*, 1978.

136. L.R. Rabiner and S. Levinson, "Isolated and Connected Word Recognition-Theory and Selected Applications," *IEEE Trans. Communication*, vol. COM-29, 1981.

137. L.R. Rabiner, S.E. Levinson, and M.M. Sondhi, "On the applications of vector quantization and hidden Markov models to speaker-independent, isolated word recognition," *Bell System Technical Journal*, vol. 62, pp. 1075-1105, 1983.

138. L.R. Rabiner, B.H. Juang, S.E. Levinson, and M.M. Sondhi , "Recognition of isolated digits using hidden Markov models with continuous mixture densities," *AT&T Technical Journal*, vol. 64, pp. 1211-1234, 1985.

139. L.R. Rabiner and B.H. Juang, "An introduction to hidden Markov models," *IEEE ASSP Magazine*, pp. 4-16, Jan. 1986.

140. L.R. Rabiner, J.G. Wilpon, and F.K. Soong, "High Performance Connected Digit Recognition Using Hidden Markov Models," *Proc. ICASSP-88, NY, USA*, 1988.

141. R.A. Redner and H.F. Walker, "Mixture densities, maximum likelihood and the EM algorithm," *SIAM review*, vol. 26, pp. 195-239, 1984.

142. S. Renals and R. Rohwer, "Learning phoneme recognition using neural networks," *Proc. ICASSP-89, Glasgow, Scotland*, pp. 413-416, 1989.

143. D.E. Rumelhart and J.L. McClelland, *Parallel Distributed Processing; Explorations in the Microstructure of Cognition. MIT press* , I and II, 1986.

144. M.J. Russell and R.K. Moore, "Explicit Modelling of State Occupancy in Hidden Markov Models for Automatic Speech Recognition," *Proc. ICASSP-85, Tampa, USA*, pp. 5-8, 1985.

145. M.J. Russell and A.E. Cook, "Experimental Evaluation of Duration Modelling Techniques for Automatic Speech Recognition," *Proc. ICASSP-87, Dallas, USA*, pp. 2376-2379, 1987.

146. H. Sakoe and S. Chiba , "A Dynamic Programming Approach to Continuous Speech Recognition ," *Proc. Int. Congress on Acoustics, Budapest, Hungary, Paper 20 C-13*, 1971.

147. H. Sakoe, "Two-Level DP-Matching – A Dynamic Programming-Based Pattern Matching Algorithm for Connected Word Recognition," *IEEE Trans. Acoustic, Speech, and Signal Processing*, vol. ASSP-27, pp. 588-595, 1979.

148. R.W. Schafer and L.R. Rabiner, "System for Automatic Formant Analysis of Voiced Speech," *J. Acoustic Soc. America*, vol. 47, pp. 634-648, 1970.

149. R. Schwartz, J. Klovstad, J. Makhoul, and J. Sorensen, "A preliminary design of a phonetic vocoder based on a diphone model," *Proc. ICASSP-80, USA*, pp. 32-35, 1980.

150. R. Schwartz, O. Kimball, F. Kubala, M. Feng, Y. Chow, C. Barry, and J. Makhoul, "Robust smoothing methods for discrete hidden Markov models," *Proc. ICASSP-89, Glasgow, Scotland*, pp. 548-551, 1989.

151. C.E. Shannon, "A Mathematical Theory of Communications," *Bell System Technical J.*, vol. 27, pp. 379-423, 623-656, 1948.

152. C.E. Shannon, "Prediction and Entropy of Printed English," *Bell System Technical J.*, vol. 30, pp. 50-64, 1951.

153. C.E. Shannon, "Coding theorems for a discrete source with a fidelity criterion," *in Information and Decision Process, R.E. Machol, Ed. McGraw-Hill,*, pp. 93-126, 1960.

154. K. Shikano, "Evaluation of LPC spectral matching measures for phonetic unit recognition," *CMU Technical Report CMU-CS-86-108, Computer Science Dept*, 1986.

155. D. Shipman and V.W. Zue, "Properties of Large Lexicons; Implication for Advanced Isolated Word Recognition System," *Proc. ICASSP-82, Paris, France*, pp. 546-549, 1982.

156. J.E. Shore and D.K. Burton, "Discrete utterance recognition without time alignment," *IEEE Trans. Information Theory*, vol. IT-29, pp. 473-491, 1983.

157. M.M. Sondhi and S.E. Levinson, "Computing Relative Redundancy to Measure Grammatical Constraint in Speech Recognition Tasks," *Proc. ICASSP-78, USA*, pp. 409-412, 1978.

158. F.K. Soong and A.E. Rosenberg, "On the use of instantaneous and transitional spectral information in speaker recognition," *Proc. ICASSP-86, Tokyo, Japan*, pp. 877-880, 1986.

159. S. Soudoplatoff, "Markov modeling of continuous parameters in speech recognition," *Proc. ICASSP-86, Tokyo, Japan*, pp. 45-48, 1986.

160. R.M. Stern and M.J. Lasry, "Dynamic Speaker Adaptation for Feature-Based Isolated Word Recognition," *IEEE Trans. Acoustic, Speech, and Signal Processing*, vol. ASSP-35, pp. 751-763, 1987.

161. M. Tomita, "Efficient parsing for natural language - a fast algorithm for practical systems," *Kluwer Academic Publishers*, 1986.

162. H.P. Tseng, M. Sabin, and E. Lee, "Fuzzy vector quantization applied to hidden Markov modeling," *Proc. ICASSP-87, Dallas, USA*, pp. 641-644, 1987.

163. A.J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. on Information Theory*, vol. IT-13, pp. 260-269, 1967.

164. A. Waibel, "Recognition of Lexical Stress in a Continuous Speech Understanding System --- A Pattern Recognition Approach," *Proc. ICASSP-86, Tokyo, Japan*, pp. 2287-2290, 1986.

165. A. Waibel, K. Lang, and G. Hinton, "Speech Recognition Using Time-Delay Neural Networks," *IEEE Workshop on Speech Recognition, Arden House, (Harriman, NY)*, 1988.

166. R.L. Watrous, "Connectionist Speech Recognition Using the Temporal Flow Model," *IEEE Workshop on Speech Recognition, Arden House, (Harriman, NY)*, 1988.

167. C. Wellekens, "Explicit time correlation in hidden Markov models for speech recognition," *Proc. ICASSP-87, Dallas, USA*, pp. 384-386, 1987.

168. G.M. White and R.B. Neely, "Speech Recognition Experiments with Linear Prediction, Bandpass Filtering, and Dynamic Programming," *IEEE Trans. Acoustic, Speech, and Signal Processing*, vol. ASSP-24, pp. 183-188, 1976.

169. J.H. Wolf, "Pattern Clustering by Multivariate Mixture Analysis," *Multivariate Behavioral Res.*, vol. 5, pp. 329-350, 1970.

170. W.A. Woods, "Language Processing for Speech Understanding ," *chapter 12 (pp.305-334) in F. Fallside and W.A. Woods (eds), Computer Speech Processing, Prentice/Hall International*, 1985.

171. J.Z. Young, "Programmes of the Brain," *Oxford University Press*, 1975.

172. V.W. Zue, "The use of speech knowledge in automatic speech recognition," *Proceedings of IEEE*, vol. 73, pp. 1602-1615 , 1985 .

173. V.W. Zue and *et al.*, "Acoustic segmentation and phonetic classification in the SUMMIT system," *Proc. ICASSP-89, Glasgow, Scotland*, pp. 389-392, 1989 .