X.D. HUANG, Y. ARIKI, M.A. JACK

HIDDEN MARKOV MODELS FOR SPEECH RECOGNITION

EDINBURGH UNIVERSITY PRESS

Edinburgh University Press
22 George Square, Edinburgh

Set by the University of Edinburgh
and printed in Great Britain by
Redwood Press Limited
Melksham, Wilts

British Library Cataloguing
in Publication Data

Huang, X. D.
Hidden Markov models for
speech recognition.—(Edinburgh
information technology series; 7)
1. Speech. Recognition by
computer systems
I. Title II. Ariki, Y.
III. Jack, Mervyn A. IV. Series
006.454

ISBN 0 7486 0162 7

CONTENTS

PREFACE

The theory of hidden Markov models was first developed in the mid 1960s by Baum and Welch. Applications of hidden Markov models to automatic speech recognition became a research topic in the 1970s in the pioneering work of Baker, Jelinek and others. The theory has subsequently been successfully applied in many state-of-the-art speech recognition systems. In the 1980s, there has been a dramatic increase in the application of hidden Markov models, not only for speech recognition, but also for many other areas. We have been involved for several years in the development of hidden Markov models for speech recognition and we believe that an appropriate textbook on hidden Markov modelling will greatly help postgraduate students in Electrical Engineering or Computer Science.

This book is primarily concerned with basic theories in hidden Markov modelling. However, some essential results of general pattern recognition and speech processing are included to help readers understand and apply the hidden Markov model for speech recognition. The main body of this book is devoted to the unified treatment of conventional vector quantisation, discrete hidden Markov models, and continuous hidden Markov models. We have presented an extensive discussion of Q-functions that are crucial in using and understanding the theory. We have also devoted many pages to practical issues in hidden Markov modelling. Finally, experimental examples are included to demonstrate how the theory is applied in practice. We hope such a treatment will be useful to both beginners as an introductory book and experts as a reference book.

In writing this book, the authors have had the benefit of advice from many people. We gratefully acknowledge the help of Ditang Fang, Hsiao-Wuen Hon, John Laver, Kai-Fu Lee, Fergus McInnes and our wives and families.

Xuedong D. Huang
Yasuo Ariki
Mervyn A. Jack

CHAPTER ONE

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 attention of scientists over the past 50 years. 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 demands integration of both speech processing and natural language processing. One of the key problems is 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 that 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 the following.

(1) Modelling of the speech signal and its variabilities 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 [10] 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 [5,15]. It is known that human experts can be trained to *read* speech spectra, which supports the proposition that distinct features exist in the speech spectrum [15]. 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 formalise completely their knowledge. Totally reliable features are required to represent speech signals, before acoustic segmentation, phonetic labelling 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 [2,7].

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 book 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) [1,4,9,11]. 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 achieved 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 [3,6].

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 [11].

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 the continuous HMM are widely used in state-of-the-art speech recognition systems [1,6,11,12,14]. For the discrete HMM, vector quantisation (VQ) makes it possible to use a non-parametric, 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 [13] uses continuous mixture probability density functions to model speech parameters directly without using VQ, and usually needs extensive training data and computation times.

On the other hand, the semi-continuous HMM [8], which is a very general model including both discrete and continuous mixture HMMs as its special forms, unifies VQ, the discrete HMM, and the continuous mixture HMM. 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 with the conventional continuous mixture HMM, the semi-continuous HMM can offer 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 with 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. Unified modelling can substantially minimise the information lost in conventional VQ and therefore leads to better performance than both the discrete HMM and the continuous mixture HMM.

This book will introduce the necessary mathematical background to understand the theory of HMMs; present a complete theory of hidden Markov modelling in depth and scope; and offer practical guidance for the use of both fundamental and advanced HMM technologies in speech recognition problems; in particular, acoustic modelling problems.

1.1. Book Organisation

Throughout the book, unless explicitly noted otherwise, the discrete probability of finite symbols O will be denoted by $Pr(O)$; and the continuous probability density function for the continuous observations $\mathbf{x}$ will be denoted by $f(\mathbf{x})$. Fundamentals of probability and pattern recognition theories involved in speech recognition will be reviewed and discussed in Chapters 2 and 3.

Chapter 4 describes VQ as a special pattern recognition technique that has 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 5. This chapter is the foundation of the statistical modelling tool, the HMM, which will be discussed throughout the book. Chapter 6 describes continuous HMMs, which parallel discrete HMMs. The continuous mixture HMM is discussed in detail, since it is strongly related to the semi-continuous HMM. Chapters 2−6 represent the theoretic foundation to the semi-continuous HMM.

The semi-continuous HMM is presented in Chapter 7. It 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 with the discrete HMM. From the discrete HMM point of view, the semi-continuous HMM can minimise the information lost in VQ. From the continuous mixture HMM point of view, the semi-continuous HMM can reduce the number of free parameters and computational complexity by tying continuous density functions. The unified theory of VQ and hidden Markov modelling, which are heavily relevant to the discussion in Chapters 5 and 6, are highlighted.

Chapter 8 discusses issues for designing a speech recognition system using HMMs. Topics such as choice of modelling unit, use of smoothing techniques, re-estimation criteria, and multiple features are included.

Chapter 9 presents experimental examples in several typical speech recognition systems. Implementational issues are discussed. C programs are included as examples. Relationships among continuous HMMs, discrete HMMs, and semi-continuous HMMs are highlighted.

References

1. A. Averbuch *et al.*, "Experiments with the Tangora 20,000 word speech recognizer," *Proc. ICASSP-87*, pp. 701-704, Dallas, USA, 1987.

2. V.W. Zue *et al.*, "Acoustic segmentation and phonetic classification in the SUMMIT system," *Proc. ICASSP-89*, pp. 389-392, Glasgow, Scotland, 1989 .

3. 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*, New York, USA, 1988.

4. J. Baker, "Stochastic modeling as a means of automatic speech recognition," Ph.D. thesis, Department of Computer Science, Carnegie-Mellon University, 1975.

5. R. Cole, M. Phillips, B. Brennan, and B. Chigier, "The CMU phonetic classification system ," *Proc. ICASSP-86*, pp. 2255-2258, Tokyo, Japan , 1986.

6. G.R. Doddington, "Phonetically sensitive discriminants for improved speech recognition," *Proc. ICASSP-89*, pp. 556-559, Glasgow, Scotland, 1989.

7. 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*, pp. 868-871, Dallas, USA , 1987.

8. X.D. Huang, "Semi-continuous hidden Markov models for speech recognition," Ph.D. thesis, Department of Electrical Engineering, University of Edinburgh, 1989.

9. F. Jelinek, "Continuous speech recognition by statistical methods," *Proc. IEEE*, vol. 64, pp. 532-556, 1976.

10. D. Klatt, "Review of the ARPA speech understanding project," *J. Acoustic Soc. America*, vol. 62, pp. 1345-1366, 1977.

11. K.F. Lee, "Large-vocabulary speaker-independent continuous speech recognition: The SPHINX system," Ph.D. thesis, Department of Computer Science, Carnegie-Mellon University, 1988; also Automatic Speech Recognition: The Development of the SPHINX System, Kluwer Academic Publishers, 1989.

12. D.B. Paul, "The Lincoln robust continuous speech recognizer," *Proc. ICASSP-89*, pp. 449-452, Glasgow, Scotland, 1989.

13. 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.

14. L.R. Rabiner, J.G. Wilpon, and F.K. Soong, "High performance connected digit recognition using hidden Markov models," *Proc. ICASSP-88*, New York, USA, 1988.

15. V.W. Zue, "The use of speech knowledge in automatic speech recognition," *Proc. IEEE*, vol. 73, pp. 1602-1615 , 1985 .

CHAPTER TWO

FUNDAMENTALS OF PATTERN RECOGNITION

Pattern recognition, which represents one of the seemingly easiest problems for human beings, has proven to be one of the most difficult problems for machines. This gap between human and machine capability has attracted the interest of many scientific researchers exploring the processes of pattern recognition. However, machine pattern recognition performance is still limited by the fact that the process of human pattern recognition is not well understood. Therefore, a mathematical approach is usually adopted to make decisions (in a probabilistic sense) about ~~which~~ category observed ~~data (pattern) belongs to~~. Both *a posteriori* knowledge obtained from specific observation data, and *a priori* knowledge of the categories play key roles in such probabilistic decision making, and automatic learning methods, which extract *a posteriori* knowledge from observation data, currently represent one of the most important issues in pattern recognition. Such learning methods can be classified into two types. The first type is supervised learning in which category information is provided for the data and only the probabilistic structure is learned. The second type is unsupervised learning in which category information is unavailable and, in this case, the category must be automatically formed together with learning of probabilistic structure. In this chapter, probability and decision theory are first introduced; then supervised and unsupervised learning methods are described in detail. The purpose of this chapter is to provide the reader with the basic knowledge required to understand the theory of hidden Markov modelling introduced later.

2.1. Probability Theory [10]

Probability can be used to advantage in representing the degree of confidence in the outcomes of some actions (observations), which are not definite. In probability theory, the term *sample space*, S, is used to refer to the totality of all possible outcomes. *Event* refers to a subset of the sample space, or a collection of outcomes. The *probability of event A* can be denoted as $Pr(A)$, which may be computed by counting the total number, N, of all observations and the number of observations N_A whose outcome belongs to the event A. Then $Pr(A)$ can be defined as the relative frequency

$$Pr(A)=\frac{N_A}{N} \qquad (2.1.1)$$

大理研[illegible]

2.1.1. Conditional probabilities

Let us consider two experiments E1 and E2, whose events are $\{A_1, A_2, \ldots, A_n\}$ and $\{B_1, B_2, \ldots, B_m\}$ respectively.

(1) Joint probability is the probability of occurrence of the composite event A_iB_j, where the event A_i from E1 and B_j from E2 occur concurrently, and is denoted as $Pr(A_iB_j)$.

(2) Marginal probability is the probability of occurrence of an event whose probability is computed from the joint probability as follows:

$$Pr(B_j)=\sum_{i=1}^{n}Pr(A_iB_j) \;\; or \;\; Pr(A_i)=\sum_{j=1}^{m}Pr(A_iB_j) \qquad (2.1.2)$$

(3) Conditional probability is the probability of occurrence of event B_j given that event A_i has occurred. It can be denoted as $Pr(B_j|A_i)$, and reads 'probability of B_j given A_i'.

An expression for the conditional probability $Pr(B|A)$ in terms of the joint probability $Pr(AB)$ and the marginal probabilities $Pr(A)$ and $Pr(B)$ can be obtained as follows. Let N_A, N_B, and N_{AB} be the number of observations whose outcomes belong to events A, B and AB, respectively, and let N be the total overall number of observations. Then,

$$Pr(AB)=\frac{N_{AB}}{N}$$

$$Pr(A)=\frac{N_A}{N} \tag{2.1.3}$$

Given that the event A has occurred, we know that the outcome is in A. There are N_A outcomes in A. Now, for B to occur given that A has occurred, the outcome should belong to A and B. There are N_{AB} outcomes in AB. Thus, the probability of occurrence of B given A is:

$$\begin{aligned} Pr(B|A) &= \frac{N_{AB}}{N_A} \\ &= \frac{N_{AB}/N}{N_A/N} \\ &= \frac{Pr(AB)}{Pr(A)} \end{aligned} \tag{2.1.4}$$

2.1.2. Useful probability expressions

The following useful expressions can be easily derived from the previous section.

$$Pr(AB)=Pr(A|B)Pr(B)=Pr(B|A)Pr(A) \tag{2.1.5}$$

$$Pr(ABC)=Pr(A)Pr(B|A)Pr(C|AB) \tag{2.1.6}$$

$$Pr(A)=\sum_{j=1}^{m}Pr(A|B_j)Pr(B_j) \tag{2.1.7}$$

$$Pr(B_j|A)=\frac{Pr(A|B_j)Pr(B_j)}{\sum_{j=1}^{m}Pr(A|B_j)Pr(B_j)} \tag{2.1.8}$$

Eq. (2.1.6) is called the *chain rule* and Eq. (2.1.8) is called the *Bayes rule* which can be obtained from Eq. (2.1.5) and Eq. (2.1.7).

If the occurrence of event A_i does not influence the probability of occurrence of event B_j and vice versa, then the events are said to be statistically independent, and can be represented as:

$$Pr(A_iB_j)=Pr(A_i)Pr(B_j) \tag{2.1.9a}$$

or

$$Pr(A_i|B_j)=Pr(A_i) \quad , \quad Pr(B_j|A_i)=Pr(B_j) \tag{2.1.9b}$$

2.1.3. Random variables

Elements in a sample space may be numbered and referred to by that number. A variable X which specifies the numerical quantity in a sample space is called a random variable. Therefore, a random variable X is a function which maps an element in the sample space S onto a set of real numbers $x \epsilon R_1$. Since each event is a subset of the sample space, an event is represented as a set of $\{\alpha\}$ which satisfies $\{\alpha|X(\alpha)\le x\}$. We shall use capital letters to denote random variables and lower-case letters to denote fixed values of the random variable. Thus, the probability that $X=x$ is denoted as:

$$Pr(X=x) \;=\; Pr(\alpha|X(\alpha)=x) \tag{2.1.10}$$

When a random variable X is discrete, the allowable values may be $x_1,x_2,\ldots,x_n$. The probability that the discrete random variable takes the value x_i is denoted as $Pr(X=x_i)$ and is called a probability mass function. The sum of probability mass functions over all values of the random variable is equal to unity ($\sum_{i=1}^{n} Pr(X=x_i) = 1$). The following frequently used expressions of two random variables X and Y can be easily derived using Eq. (2.1.5) to (2.1.8):

$$\begin{aligned} Pr(X=x_i) &= \sum_{j=1}^{m} Pr(X=x_i, Y=y_j) \\ &= \sum_{j=1}^{m} Pr(X=x_i | Y=y_j) Pr(Y=y_j) \end{aligned} \tag{2.1.11}$$

$$\begin{aligned} Pr(X=x_i | Y=y_j) &= \frac{Pr(X=x_i, Y=y_j)}{Pr(Y=y_j)} \\ &= \frac{Pr(Y=y_j | X=x_i) Pr(X=x_i)}{\sum_{i=1}^{m} Pr(Y=y_j | X=x_i) Pr(X=x_i)} \end{aligned} \tag{2.1.12}$$

In a similar manner, if the random variables X and Y are statistically independent, they can be represented as:

$$\begin{aligned} Pr(X=x_i, Y=y_j) &= Pr(X=x_i) Pr(Y=y_j) \\ &\quad i=1,2,\dots,n;\ j=1,2,\dots,m \end{aligned} \tag{2.1.13}$$

When multiple random variables are dealt with as one vector, it is called a random vector.

2.1.4. Probability density functions

When a random variable X is continuous, the probability that the continuous random variable takes the value x is denoted as $f_X(x)$ and is called a probability density function. The integral of the probability density function over all values of the random variable is equal to unity ($\int_{-\infty}^{\infty} f_X(x) = 1$). The joint, marginal and conditional probability density functions can be defined in a similar way. The following expressions of two random variables X and Y are also frequently used.

$$\begin{aligned} f_X(x) &= \int_{-\infty}^{\infty} f_{X,Y}(x,y)\, dy \\ &= \int_{-\infty}^{\infty} f_{X|Y}(x|y)\, f_Y(y)\, dy \end{aligned} \tag{2.1.14}$$

$$\begin{aligned} f_{Y|X}(y|x) &= \frac{f_{X,Y}(x,y)}{f_X(x)} \\ &= \frac{f_{X|Y}(x|y)\, f_Y(y)}{\int_{-\infty}^{\infty} f_{X|Y}(x|y)\, f_Y(y)\, dy} \end{aligned} \tag{2.1.15}$$

If the random variables X and Y are statistically independent, they are represented as:

$$f_{X,Y}(x,y) = f_X(x) f_Y(y) \tag{2.1.16}$$

2.2. Bayes Decision Theory [5]

Probability expressions mentioned in the previous section can be used to make decisions with minimum error rate. In such decision making, *a priori* knowledge about the probability of the events and observed data about present status are both employed under the decision frame.

2.2.1. A posteriori probability

Let us consider the problem of weather prediction, where we have to decide tomorrow's weather in one of the three categories (events): Sunny, Cloudy, or Rainy. Available information is the probability mass function $Pr(\omega)$ of the three categories. The variable ω is a discrete random variable with the values $\omega = \omega_j$, $j=1,2,3$. We call the probability $Pr(\omega_j)$ *a priori* probability since it reflects our *a priori* knowledge of how likely tomorrow's weather is. If we have to make decision based only on the *a priori* probability, the most plausible decision may be made by selecting the category ω_j with the highest *a priori* probability $Pr(\omega_j)$. This decision is obviously unreasonable, because every day must be predicted to be one of three categories, Sunny, Cloudy and Rainy.

If we are given further observable data, such as air pressure or temperature, we can make a more precise decision. Let x be a continuous random variable whose value is the air pressure, and $f_{X|\omega}(x|\omega)$ be a category-conditional probability density function (pdf). For simplicity, we denote the pdf $f_{X|\omega}(x|\omega)$ as $f(x|\omega_j)$, where $j=1,2,3$ unless there is ambiguity. Since we know the *a priori* probability $Pr(\omega_j)$ and category-conditional pdf $f(x|\omega_j)$, we can compute the conditional probability $Pr(\omega_j|x)$ using the Bayes rule:

$$Pr(\omega_j|x)=\frac{f(x|\omega_j)Pr(\omega_j)}{f(x)} \qquad (2.2.1)$$

where

$$f(x)=\sum_{j=1}^{3}f(x|\omega_j)Pr(\omega_j)$$

The above probability is called the *a posteriori* probability as it is the probability of category ω_j after observing the air pressure x. Bayes rule shows how the observed data x changes the decision based on the *a priori* probability $Pr(\omega_j)$ using the *a posteriori* probability $Pr(\omega_j|x)$. Decision making based on the *a posteriori* probability is more reliable, because it employs both *a priori* knowledge together with present observed data. In fact, if *a priori* knowledge is ambiguous ($Pr(\omega_1)=Pr(\omega_2)=Pr(\omega_3)$), then present observed data controls the decision. If, on the other hand, present observed data is ambiguous, then *a priori* knowledge controls the decision. There will be many kinds of decision rule based on *a posteriori* probability. Our interest is to find the decision rule which leads to minimum overall risk, or minimum error rate in decision.

2.2.2. Bayes decision rule

Bayes decision rule is designed to minimise the overall risk involved in making a decision. Let $h(a_i|\omega_j)$ be the loss function incurred for making decision a_i when the true category is ω_j. Here, the decision set is $A=\{a_1,\ldots,a_r\}$ and the category set (sample space) is $S=\{\omega_1,\ldots,\omega_s\}$. Using the *a priori* probability $Pr(\omega_j)$ and category-conditional pdf $f(x|\omega_j)$, the *a posteriori* probability $Pr(\omega_j|x)$ is computed by the Bayes rule as shown in Eq. (2.2.1). If we make decision a_i even when the true category is ω_j, we shall incur a loss $h(a_i|\omega_j)$. Since the *a posteriori* probability $Pr(\omega_j|x)$ is the probability that the true category is ω_j after observing the data x, the expected loss associated with making decision a_i is:

$$R(a_i|x)=\sum_{j=1}^{s}h(a_i|\omega_j)Pr(\omega_j|x) \qquad (2.2.2)$$

The above expression is called the conditional risk.

The overall risk R is the expected loss associated with a given decision rule. Here we employ an arbitrary decision rule $a(x)$ which maps the data x to one of decisions $A=\{a_1,\ldots,a_r\}$. Since $R(a_i|x)$ is the conditional risk associated with decision a_i, the overall risk is given by:

$$R=\int_{-\infty}^{\infty}R(a(x)|x)f(x)dx \qquad (2.2.3)$$

If the decision rule $a(x)$ is chosen so that the conditional risk $R(a(x)|x)$ is as small as possible for every $\mathbf{x}$, then the overall risk will be minimised. This leads to the *Bayes decision rule:* In order to minimise the overall risk, compute the conditional risk shown in Eq. (2.2.2) for $i=1,\ldots,r$ and select the decision a_i for which the conditional risk $R(a_i|x)$ is minimum. Bayes decision rule is also applicable to multi-variate elements $\mathbf{x}$ without loss of generality.

2.2.3. Minimum-error-rate decision rule

The loss function $h(a_i|\omega_j)$ in the Bayes decision rule can be defined as:

$$h(a_i|\omega_j)=\begin{cases}0 & i=j \\ 1 & i\neq j\end{cases} \qquad i,\ j=1,\ldots,s \tag{2.2.4}$$

This loss function assigns no loss to a correct decision where the true category is ω_i and the decision is a_i which implies that the true category must be ω_i. It assigns a unit loss to any error where $i\neq j$. The risk corresponding to this loss function equals the average probability of error, since the conditional risk is given as

$$\begin{aligned}R(a_i|x)&=\sum_{j=1}^{s}h(a_i|\omega_j)Pr(\omega_j|x)\\ &=\sum_{j=1}^{s}Pr(\omega_j|x)-Pr(\omega_i|x) \qquad (2.2.5)\\ &=1-Pr(\omega_i|x)\end{aligned}$$

and $Pr(\omega_i|x)$ is the conditional probability that decision a_i is correct after observing the data x. Therefore, in order to minimise the average probability of error, or to achieve the minimum error rate, we have to select the decision that the category ω_i is correct, if the *a posteriori* probability $Pr(\omega_i|x)$ is a maximum. This decision rule based on the maximum of the *a posteriori* probability $Pr(\omega_i|x)$ is called the minimum-error-rate decision rule, since it can minimise the error rate.

2.2.4. Classifier and decision boundary

In decision problems, observed data x are used to make a precise decision of which category is plausible. This can also be viewed as a classification problem where unknown

data x are classified into known categories, such as the classification of short interval sounds into phonemes using spectral data x. A classifier is designed to classify data x into s categories by using s discriminant functions $g_j(x)$, computing the similarities between the unknown data x and each category ω_j and selecting the category ω_i corresponding to

$$g_i(x)>g_j(x) \qquad \text{for } all\ j\neq i \tag{2.2.6}$$

Figure 2.2.1. shows the relation between a classifier and discriminant functions. In a Bayes classifier, unknown data x are classified on the basis of Bayes decision rule which minimises the conditional risk $R(a_i|x)$. Then the discriminant function is:

$$g_i(x)=-R(a_i|x) \tag{2.2.7}$$

In the minimum-error-rate classifier, the decision rule is to maximise the *a posteriori* probability $Pr(\omega_i)$. Then

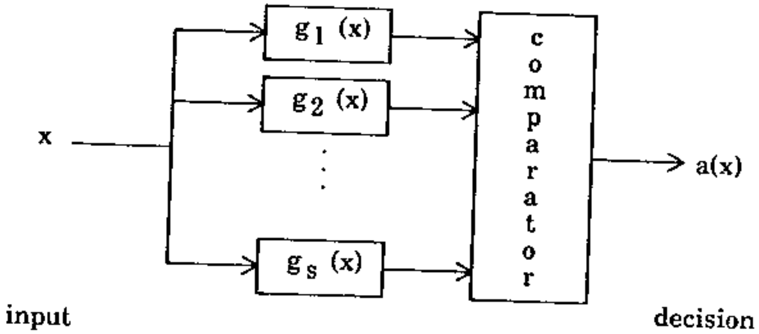

Figure 2.2.1. Block diagram of a classifier based on discriminant functions.

$$g_i(x)=Pr(\omega_i|x)$$

$$=\frac{f(x|\omega_i)Pr(\omega_i)}{\sum_{j=1}^{s}f(x|\omega_j)Pr(\omega_j)} \qquad (2.2.8)$$

As the classifier assigns data x to category ω_i, the data space is divided into s regions, $R_i,...,R_s$ which are called decision regions. The boundaries between decision regions are called decision boundaries and are represented as follows (if they are contiguous):

$$g_i(x)=g_j(x) \qquad (2.2.9)$$

2.3. Parametric Supervised Learning [5]

In the Bayes classifier, or the minimum-error-rate classifier, *a priori* probability $Pr(\omega_i)$ and a category-conditional pdf $f(x|\omega_i)$ are assumed to be known. However, the category-conditional pdf is not known in advance, and must be estimated or learned from the available training data.

Learning methods can be based on parametric or non-parametric approaches in estimating a category-conditional pdf. In parametric learning, the pdf is assumed to have certain probabilistic structure, such as the Gaussian pdf. In such cases, only the parameters of the pdf need to be estimated. On the other hand, in non-parametric learning, no model structure is assumed and the pdf is directly estimated from the training data. When large amounts of sample data are available, non-parametric learning can accurately reflect the underlying probabilistic structure of the training data. However, available sample data are normally limited in practice and parametric learning can achieve good estimates if valid model assumptions are made. It is also important to distinguish between supervised learning and unsupervised learning. In supervised learning, information about the category of the sample data x are given. Such sample data are usually called labelled. In contrast, category label information is unavailable in unsupervised learning.

In this section, we will describe parametric supervised learning, with particular emphasis on the Gaussian pdf based on maximum likelihood estimation. Parametric unsupervised learning will be described in Section 2.4.

2.3.1. Maximum likelihood estimation

Maximum likelihood estimation is one of the most widely used parametric learning methods. As the category-conditional pdf is parameterised, let φ_i be a parameter vector which characterises category ω_i. Then the category-conditional pdf is represented as a function of φ_i as $f(x|\omega_i\,,\varphi_i)$. Here, φ_i is not a random vector, but some unknown fixed value (parameter). In supervised learning, we are given the category name ω_i for a set of sample data $X_i=\{\ x_1,\ .\ .\ .\ ,x_n\ \}$. If the set of sample data X_i gives no information about parameters φ_j of the other category ω_j, then we can deal with each category independently. Therefore, the category-conditional pdf is written as $f(x|\varphi)$.

Suppose that $X=\{\ x_1,\ .\ .\ .\ ,x_n\ \}$ is a set of sample data drawn independently according to the pdf $f(x|\varphi)$. The likelihood of φ with respect to the set of sample data vectors X is defined as follows:

$$f(X|\varphi)=\prod_{k=1}^{n}f(x_k|\varphi) \qquad (2.3.1)$$

The likelihood $f(X|\varphi)$ is the probability that the set of sample data vectors X is drawn based on the values φ. The maximum likelihood estimate of φ is the value ϕ which maximises the likelihood $f(X|\varphi)$. This estimation method is called the maximum likelihood estimation method.

Since the logarithm function increases monotonically, the value $\hat{\phi}$ that maximises the log-likelihood also maximises the likelihood. If $f(X|\varphi)$ is differentiable by φ, $\hat{\phi}$ can be found by classical min-max theory (see Section 2.5); computing its partial derivative and setting it to zero. Therefore, the log-likelihood is:

$$l(\varphi)=\log f(X|\varphi)$$

$$=\sum_{k=1}^{n}\log f(x_k|\varphi) \tag{2.3.2}$$

and its partial derivative is:

$$\frac{\partial l(\varphi)}{\partial\varphi_m}=\sum_{k=1}^{n}\frac{\partial}{\partial\varphi_m}\log f(x_k|\varphi)$$

$$=0 \tag{2.3.3}$$

where φ_m is the mth component of the parameter φ.

Example 2.3.1

The Gaussian pdf, as will be discussed throughout this book, is one of the most popularly used pdfs. The univariate Gaussian pdf is given as follows:

$$f(x)=\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right] \tag{2.3.4}$$

where μ is the mean and σ^2 is the variance. The multivariate Gaussian pdf is:

$$f(\mathbf{x})=\frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}}\exp\left[-\tfrac{1}{2}(\mathbf{x}-\mu)^t\Sigma^{-1}(\mathbf{x}-\mu)\right] \tag{2.3.5}$$

where $\mathbf{x}$ is a d-component column 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.

We show here the maximum likelihood estimates for the univariate Gaussian pdf Eq. (2.3.4). The parameter φ is $(\mu,\ \sigma^2)$. The log-likelihood of Eq. (2.3.4) for one data value x_k is:

$$\log f(x_k|\varphi)=-\tfrac{1}{2}\log 2\pi\sigma^2-\frac{1}{2\sigma^2}(x_k-\mu)^2 \tag{2.3.6}$$

and the partial derivative of the above expression is:

$$\frac{\partial}{\partial\mu}\log f(x_k|\varphi)=\frac{1}{\sigma^2}(x_k-\mu) \tag{2.3.7}$$

$$\frac{\partial}{\partial\sigma^2}\log f(x_k|\varphi)=-\frac{1}{2\sigma^2}+\frac{(x_k-\mu)^2}{2\sigma^4}$$

By summing over all sample data values x_k and equating to zero, the following expressions are obtained.

$$\sum_{k=1}^{n}\frac{1}{\sigma^2}(x_k-\mu)=0 \tag{2.3.8}$$

$$-\sum_{k=1}^{n}\frac{1}{\sigma^2}+\sum_{k=1}^{n}\frac{(x_k-\mu)^2}{\sigma^4}=0$$

Finally, we obtain the following maximum likelihood estimates for μ and σ^2:

$$\hat{\mu}=\frac{1}{n}\sum_{k=1}^{n}x_k \tag{2.3.9}$$

$$\hat{\sigma}^2=\frac{1}{n}\sum_{k=1}^{n}(x_k-\hat{\mu})^2$$

In the multivariate Gaussian pdf, the same discussion is applied and the following maximum likelihood estimates are obtained (see Example 2.5.1):

$$\hat{\mu}=\frac{1}{n}\sum_{k=1}^{n}\mathbf{x}_k \tag{2.3.10}$$

$$\hat{\Sigma}=\frac{1}{n}\sum_{k=1}^{n}(\mathbf{x}_k-\hat{\mu})(\mathbf{x}_k-\hat{\mu})^t$$

The above Eq. (2.3.9) and (2.3.10) indicate that maximum likelihood estimation provides the usual definition of mean and variance in Gaussian pdf.

2.4. Parametric Unsupervised Learning [5]

In Section 2.3, we discussed a method for estimating the probability parameters of a category-conditional pdf, using làbelled sample data. In practice, the information about the category is usually unknown. We will here investigate, with unlabelled data, unsupervised learning methods. In particular, one of the most important maximum likelihood estimation techniques, the EM algorithm, will be introduced.

2.4.1. Mixture density estimation

On the assumption that the data can be modelled as a mixture of pdf, the maximum likelihood estimation can also be used to learn parameters of the mixture pdf. As it is not known from which category a data sample x is drawn, it is possible to first select a category ω_j with *a priori* probability $Pr(\omega_j)$, and then select the data x according to the category-conditional pdf $f(x|\omega_j,\varphi_j)$. Thus, the probability density function of the sample data x given the mixture pdf can be written as follows:

$$f(x_k|\varphi)=\sum_{j=1}^{s}f(x_k|\omega_j,\ \varphi_j)Pr(\omega_j) \tag{2.4.1}$$

where s is a known number of categories, $\varphi=(\varphi_1,\ldots,\varphi_s)$. The conditional pdfs $f(x|\omega_j,\varphi_j)$ are called the component densities, and the *a priori* probabilities $Pr(\omega_j)$ are called the mixing parameters. Here, we assume that the only unknown parameters are φ and the *a priori* probabilities are known. For the case where *a priori* probability is unknown, see Appendix 1.

In maximum likelihood estimation, to estimate parameters φ, the likelihood (or the log-likelihood) of the mixture pdf of Eq. (2.4.1) as well as its derivative must be computed. The likelihood of the mixture pdf Eq. (2.4.1) is:

$$f(X|\varphi)=\prod_{k=1}^{n}f(x_k|\varphi) \tag{2.4.2}$$

where X is a collection of the sample data $\{x_k\}$, $k=1,\ldots,n$. The log-likelihood is:

$$l(\varphi)=\log f(X|\varphi)=\sum_{k=1}^{n}\log f(x_k|\varphi) \tag{2.4.3}$$

The derivative of Eq. (2.4.3) with respect to φ_i is:

$$\begin{aligned}\nabla_{\varphi_i}\ l(\varphi)&=\sum_{k=1}^{n}\frac{1}{f(x_k|\varphi)}\nabla_{\varphi_i}\sum_{j=1}^{s}f(x_k|\omega_j,\ \varphi_j)Pr(\omega_j)\\&=\sum_{k=1}^{n}\frac{1}{f(x_k|\varphi)}\nabla_{\varphi_i}\,(f(x_k|\omega_i,\ \varphi_i)Pr(\omega_i))\end{aligned} \tag{2.4.4}$$

where ∇_{φ_i} is a gradient operator which is applied to the scalar and produces a vector of partial derivatives. The gradient vector can be defined as:

$$\nabla_{\varphi_i}\ l(\varphi)=\begin{bmatrix}\dfrac{\partial l(\varphi)}{\partial\varphi_{ij}}\\ \cdot\\ \cdot\\ \cdot\\ \dfrac{\partial l(\varphi)}{\partial\varphi_{ij}}\end{bmatrix} \tag{2.4.5}$$

where φ_{ij} is a jth component of the parameter vector φ_i. Since the *a posteriori* probability is defined as:

$$Pr(\omega_i|x_k, \varphi_i) = \frac{f(x_k|\omega_i, \varphi_i)Pr(\omega_i)}{\sum_{i=1}^{s} f(x_k|\omega_i, \varphi_i)Pr(\omega_i)} = \frac{f(x_k|\omega_i, \varphi_i)Pr(\omega_i)}{f(x_k|\varphi)} \tag{2.4.6}$$

the first term of the multiplication in the summand of Eq. (2.4.4) is:

$$\frac{1}{f(x_k|\varphi)} = \frac{Pr(\omega_i|x_k, \varphi_i)}{f(x_k|\omega_i, \varphi_i)Pr(\omega_i)} \tag{2.4.7}$$

Therefore, Eq. (2.4.4) can be rewritten as follows:

$$\nabla_{\varphi_i} l(\varphi) = \sum_{k=1}^{n} Pr(\omega_i|x_k, \varphi_i) \frac{\nabla_{\varphi_i}(f(x_k|\omega_i, \varphi_i)Pr(\omega_i))}{f(x_k|\omega_i, \varphi_i)Pr(\omega_i)} = \sum_{k=1}^{n} Pr(\omega_i|x_k, \varphi_i)\nabla_{\varphi_i} \log(f(x_k|\omega_i, \varphi_i)Pr(\omega_i)) \tag{2.4.8}$$

As the *a priori* probability $Pr(\omega_i)$ is assumed to be known, the final constraint on partial derivatives can be computed from:

$$\sum_{k=1}^{n} Pr(\omega_i|x_k, \varphi_i)\nabla_{\varphi_i} \log f(x_k|\omega_i, \varphi_i) = 0 \tag{2.4.9}$$

In comparison with Eq. (2.3.3) (with supervised learning), Eq. (2.4.9) is the multiplication of the *a posteriori* probability $Pr(\omega_i|x_k,\varphi_i)$ and the partial derivative of log conditional pdf. This is natural, as we definitely know from which category the data sample x_k is drawn in supervised learning where the *a posteriori* probability $Pr(\omega_i|x_k,\varphi_i)=1$. This implies that maximum likelihood estimation in supervised learning is a special case of unsupervised learning. On the other hand, in unsupervised learning, the information from which category the data sample x_k is drawn must be inferred from the *a posteriori* probability using the present values of parameter φ_i; and new values of the parameter φ_i can be estimated in the same way as for supervised learning. This suggests that, in unsupervised learning, an iterative method can be adopted. In fact, a general analytic solution to Eq. (2.4.9) does not exist.

When the *a priori* probability is unknown, it can be estimated as above, and the following estimation can be obtained (see Appendix 1):

$$Pr(\omega_i) = \frac{1}{n}\sum_{k=1}^{n} Pr(\omega_i|x_k, \varphi_i) \tag{2.4.10}$$

In summary, an iterative estimation method based on maximum likelihood in unsupervised learning can be expressed as follows:

(1) Initial estimates of the parameters φ_i of all category-conditional pdfs and their *a priori* probability $Pr(\omega_i)$ are given. Using these estimates, compute the *a posteriori* probability $Pr(\omega_i|x_k,\varphi_i)$ based on Bayes rule Eq. (2.4.6).

(2) The *a priori* probability $Pr(\omega_i)$ can be re-estimated using the *a posteriori* probability based on Eq. (2.4.10). The parameter values of category-conditional pdfs can be re-estimated using the *a posteriori* probability based on Eq. (2.4.9).

(3) Repeat steps (1) and (2), until the overall change in *a posteriori* probability between iteration steps reaches some threshold value.

Example 2.4.1. Application to mixture Gaussian pdf

Let us consider a simple application of maximum likelihood estimation in unsupervised learning to the two category case of univariate Gaussian pdf. Each mixing parameter is $Pr(\omega_i)$ and each component density is represented as:

$$f(x_k|\omega_i,\varphi_i) = \frac{1}{\sqrt{2\pi\sigma_i^2}}\exp\left[-\frac{(x-\mu_i)^2}{2\sigma_i^2}\right] \tag{2.4.11}$$

where μ_i and σ_i^2 are the mean and variance of category ω_i. The log-likelihood of Eq. (2.4.11) for one data sample x_k is:

$$\log f(x_k|\omega_i,\varphi_i) = -\tfrac{1}{2}\log 2\pi\sigma_i^2 - \frac{1}{2\sigma_i^2}(x_k-\mu_i)^2 \qquad (2.4.12)$$

and the partial derivative of the above expression is:

$$\frac{\partial}{\partial\mu_i}\log f(x_k|\omega_i,\varphi_i) = \frac{1}{\sigma_i^2}(x_k-\mu_i) \qquad (2.4.13)$$

$$\frac{\partial}{\partial\sigma_i^2}\log f(x_k|\omega_i,\varphi_i) = -\frac{1}{2\sigma_i^2} + \frac{(x_k-\mu_i)^2}{2\sigma_i^4}$$

Multiplication of the equations (2.4.13) by the *a posteriori* probability $Pr(\omega_i|x_k,\varphi_i)$ according to Eq. (2.4.9), summing over all sample data x_k and then equating to zero, yields the following expressions.

$$\sum_{k=1}^{n} Pr(\omega_i|x_k,\varphi_i)\frac{1}{\sigma_i^2}(x_k-\mu_i) = 0 \qquad (2.4.14)$$

$$\sum_{k=1}^{n} Pr(\omega_i|x_k,\varphi_i)\left\{-\frac{1}{\sigma_i^2} + \frac{(x_k-\mu_i)^2}{\sigma_i^4}\right\} = 0$$

Finally, we obtain the following maximum likelihood estimates for μ_i and σ_i^2:

$$\hat{\mu}_i = \frac{\sum_{k=1}^{n} Pr(\omega_i|x_k,\varphi_i)x_k}{\sum_{k=1}^{n} Pr(\omega_i|x_k,\varphi_i)} \qquad (2.4.15)$$

$$\hat{\sigma}_i^2 = \frac{\sum_{k=1}^{n} Pr(\omega_i|x_k,\ \varphi_i)(x_k-\hat{\mu}_i)^2}{\sum_{k=1}^{n} Pr(\omega_i|x_k,\ \varphi_i)} \qquad (2.4.16)$$

For *a priori* probability $Pr(\omega_i)$, the same expression as Eq. (2.4.10) is used.

$$P\hat{r}(\omega_i) = \frac{1}{n}\sum_{k=1}^{n} Pr(\omega_i|x_k,\ \varphi_i) \qquad (2.4.17)$$

Iterative estimation follows the steps (1) to (3) in the previous section. In the case where the Gaussian pdf is multivariate, the same discussion is applied and the following maximum likelihood estimates are obtained (see Example 2.5.1):

$$\hat{\mu}_i = \frac{\sum_{k=1}^{n} Pr(\omega_i|\mathbf{x}_k,\varphi_i)\mathbf{x}_k}{\sum_{k=1}^{n} Pr(\omega_i|\mathbf{x}_k,\varphi_i)}$$

$$\hat{\Sigma}_i = \frac{\sum_{k=1}^{n} Pr(\omega_i|\mathbf{x}_k,\ \varphi_i)(\mathbf{x}_k-\hat{\mu}_i)(\mathbf{x}_k-\hat{\mu}_i)^t}{\sum_{k=1}^{n} Pr(\omega_i|\mathbf{x}_k,\ \varphi_i)} \qquad (2.4.18)$$

2.4.2. The EM algorithm [2-4,9]

In unsupervised learning, information such as from which category (class, state) a data sample $\mathbf{x}$ comes is unobservable and only the data sample $\mathbf{x}$ is observed. Observable data are called incomplete data because they are missing the unobservable data, and data composed of observable and unobservable data are called complete data. The purpose of the EM algorithm is to maximise the log-likelihood from incomplete data, by iteratively maximising the expectation of log-likelihood from complete data. The name of the EM algorithm comes from *E* for *expectation* and *M* for *maximisation*. It can be said that the EM algorithm is a maximum likelihood estimation method, but its computation is less complex than the conventional maximum likelihood estimation method described in the previous section.

In general, suppose that a measure space $\mathbf{Y}$ of *unobservable data* exists corresponding to a measure space $\mathbf{X}$ of *incomplete data*. Here, $\mathbf{X}$ is easy to observe and measure, while **Y** contains some hidden information that is unobservable. For example, **y** may be a hidden number which refers to component densities of observable data **x**; and (**x**,**y**) becomes *complete data*. Let $f(\mathbf{y}|\Phi)$ and $f(\mathbf{x}|\Phi)$ be members of a parametric family of pdf defined on **Y** and **X** respectively for parameter Φ. For a given $\mathbf{x} \epsilon \mathbf{X}$, the EM algorithm is to maximise the log-likelihood of the observable, real data **x**, $L(\mathbf{x},\Phi) = \log f(\mathbf{x}|\Phi)$ over Φ by exploiting the relationship between $f(\mathbf{x},\mathbf{y}|\Phi)$ and $f(\mathbf{y}|\mathbf{x},\Phi)$.

The joint pdf $f(\mathbf{x},\mathbf{y}|\Phi)$ is given by

$$f(\mathbf{x},\mathbf{y}|\Phi)=f(\mathbf{y}|\mathbf{x},\Phi)f(\mathbf{x}|\Phi) \tag{2.4.19}$$

and the following log-likelihood can be obtained from the above expression:

$$\log f(\mathbf{x}|\Phi)=\log f(\mathbf{x},\mathbf{y}|\Phi)-\log f(\mathbf{y}|\mathbf{x},\Phi) \tag{2.4.20}$$

For two parameter sets $\overline{\Phi}$ and Φ, the expectation of incomplete log-likelihood $L(\mathbf{x},\overline{\Phi})$ over complete data (**x**,**y**) conditioned by **x** and Φ is:

$$\begin{aligned}E[L(\mathbf{x},\overline{\Phi})|\mathbf{x},\Phi]&=E[\log f(\mathbf{x}|\overline{\Phi})|\mathbf{x},\Phi]\\&=\int \log f(\mathbf{x}|\overline{\Phi})f(\mathbf{y}|\mathbf{x},\Phi)dy\\&=\log f(\mathbf{x}|\overline{\Phi})\\&=L(\mathbf{x},\overline{\Phi})\end{aligned} \tag{2.4.21}$$

where $E[.|\mathbf{x},\Phi]$ is the expectation conditioned by **x** and Φ over complete data (**x**,**y**). Then, using Eq. (2.4.20), the following expression is obtained:

$$L(\mathbf{x},\overline{\Phi}) = Q(\Phi,\overline{\Phi}) - H(\Phi,\overline{\Phi}) \tag{2.4.22}$$

where

$$\begin{aligned}Q(\Phi,\overline{\Phi}) &= E[\log f(\mathbf{x},\mathbf{y}|\overline{\Phi})|\mathbf{x},\Phi)\\&= \int \log f(\mathbf{x},\mathbf{y}|\overline{\Phi})f(\mathbf{y}|\mathbf{x},\Phi)dy\end{aligned} \tag{2.4.23}$$

and

$$\begin{aligned}H(\Phi,\overline{\Phi}) &= E[\log f(\mathbf{y}|\mathbf{x},\overline{\Phi})|\mathbf{x},\Phi]\\&= \int \log f(\mathbf{y}|\mathbf{x},\overline{\Phi})f(\mathbf{y}|\mathbf{x},\Phi)dy\end{aligned} \tag{2.4.24}$$

The basis of the EM algorithm lies in the fact that if $Q(\Phi,\overline{\Phi}) \geq Q(\Phi,\Phi)$, then $L(\mathbf{x},\overline{\Phi}) \geq L(\mathbf{x},\Phi)$, since it follows from Jensen's inequality that $H(\Phi,\overline{\Phi}) \leq H(\Phi,\Phi)$ [4] (see Appendix 2). This fact implies that the incomplete log-likelihood $L(\mathbf{x},\Phi)$ increases monotonically on any iteration of parameter update from Φ to $\overline{\Phi}$, via maximisation of the Q-function which is the expectation of log-likelihood from complete data. When the random vector **y** (which is unobserved data) is a discrete random vector, the Q-function is represented as:

$$Q(\Phi,\overline{\Phi})=\sum_{\mathbf{y}\epsilon\mathbf{Y}}\frac{f(\mathbf{x},\mathbf{y}|\Phi)}{f(\mathbf{x}|\Phi)}\log f(\mathbf{x},\mathbf{y}|\overline{\Phi}) \tag{2.4.25}$$

This expression is used later, by specifying unobserved data **y** in vector quantisation and hidden Markov modelling respectively.

The general EM algorithm can be described in the following way. Given a current Φ that is a maximiser of $L(\mathbf{x},\Phi)$, obtain a next approximation $\overline{\Phi}$ as follows:

1. Choose an initial estimate Φ.
2. E-step. Compute $Q(\Phi,\overline{\Phi})$ based on the given Φ.
3. M-step. Choose $\overline{\Phi} \epsilon \underset{\Phi}{argmax}\, Q(\Phi,\overline{\Phi})$. Here, $\underset{\Phi}{argmax}\, Q(\Phi,\overline{\Phi})$ denotes the set of values $\overline{\Phi}$ which maximise $Q(\Phi,\overline{\Phi})$.
4. Set $\Phi = \overline{\Phi}$, repeat from step 2 until convergence.

The EM algorithm is used in applications which permit easy maximisation of the Q-function instead of maximising $L(\mathbf{x},\Phi)$ directly. In such applications, the M-step maximisation of $Q(\Phi,\overline{\Phi})$ is easily carried out.

2.4.3. The EM algorithm in multiple data

The discussion in the previous section was concerned with a single observed incomplete data point $\mathbf{x}_k$. To apply it to multiple observed data $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, the log-likelihood of the observable data $L(\mathbf{x},\Phi)$ is extended to:

$$L(\mathbf{X},\Phi) = \sum_{k=1}^{n} \log f(\mathbf{x}_k|\Phi) \tag{2.4.26}$$

The same extension is applicable to Eq. (2.4.20), then

$$\sum_{k=1}^{n} \log f(\mathbf{x}_k|\Phi) = \sum_{k=1}^{n} \log f(\mathbf{x}_k,\mathbf{y}_k|\Phi) - \sum_{k=1}^{n} \log f(\mathbf{y}_k|\mathbf{x}_k,\Phi) \tag{2.4.27}$$

The expectation of the log-likelihood of observed data, Eq. (2.4.21), is:

$$\begin{aligned}
&E[L(\mathbf{X},\bar{\Phi})|\mathbf{X},\Phi] \\
&\quad = E[\sum_{k=1}^{n} \log f(\mathbf{x}_k|\bar{\Phi})|\mathbf{X},\Phi] \\
&\quad = \sum_{k=1}^{n} \int \log f(\mathbf{x}_k|\bar{\Phi}) \prod_{l=1}^{n} f(\mathbf{y}_l|\mathbf{x}_l,\Phi) d\mathbf{y}_1 \cdots \mathbf{y}_n \\
&\quad = \sum_{k=1}^{n} \log f(\mathbf{x}_k|\bar{\Phi}) \int \prod_{l=1}^{n} f(\mathbf{y}_l|\mathbf{x}_l,\Phi) d\mathbf{y}_1 \cdots d\mathbf{y}_n \qquad (2.4.28) \\
&\quad = \sum_{k=1}^{n} \log f(\mathbf{x}_k|\bar{\Phi}) \\
&\quad = L(\mathbf{X},\bar{\Phi})
\end{aligned}$$

By applying expectation to Eq. (2.4.27) and using Eq. (2.4.28), the same expression as Eq. (2.4.22) is obtained. Here, the Q-function of multiple observed data is:

$$\begin{aligned}
&Q(\Phi,\bar{\Phi}) \\
&\quad = E[\sum_{k=1}^{n} \log f(\mathbf{x}_k,\mathbf{y}_k|\bar{\Phi})|\mathbf{X},\Phi] \\
&\quad = \sum_{k=1}^{n} \int \log f(\mathbf{x}_k,\mathbf{y}_k|\bar{\Phi}) \prod_{l=1}^{n} f(\mathbf{y}_l|\mathbf{x}_l,\Phi) d\mathbf{y}_1 \cdots d\mathbf{y}_n \qquad (2.4.29) \\
&\quad = \sum_{k=1}^{n} \left\{ \prod_{l \neq k}^{n} \int f(\mathbf{y}_l|\mathbf{x}_l,\Phi) d\mathbf{y}_l \int \log f(\mathbf{x}_k,\mathbf{y}_k|\bar{\Phi}) f(\mathbf{y}_k|\mathbf{x}_k,\Phi) d\mathbf{y}_k \right\} \\
&\quad = \sum_{k=1}^{n} \int \log f(\mathbf{x}_k,\mathbf{y}_k|\bar{\Phi}) f(\mathbf{y}_k|\mathbf{x}_k,\Phi) d\mathbf{y}_k \\
&\quad = \sum_{k=1}^{n} \mathbf{Q}_k(\Phi,\bar{\Phi})
\end{aligned}$$

In the same way, $H(\Phi,\bar{\Phi})$ is obtained:

$$H(\Phi,\bar{\Phi}) = \sum_{k=1}^{n} \mathbf{H}_k(\Phi,\bar{\Phi}) \tag{2.4.30}$$

The above two expressions mean that the Q-function of multiple observed data is easily obtained by summing the Q-functions of individual observed data. When the random vector $\mathbf{y}$ which is unobserved data is a discrete random vector, the Q-function of multiple observed data is represented as:

$$\begin{aligned}
Q(\Phi,\bar{\Phi}) &= \sum_{k=1}^{n} Q_k(\Phi,\bar{\Phi}) \\
&= \sum_{k=1}^{n} \sum_{y_k} \frac{f(\mathbf{x}_k,\mathbf{y}_k|\Phi)}{f(\mathbf{x}_k|\Phi)} \log f(\mathbf{x}_k,\mathbf{y}_k|\bar{\Phi}) \qquad (2.4.31)
\end{aligned}$$

Example 2.4.2

Let a data sample $\mathbf{x}_k$ be the *observed incomplete data* and $(\mathbf{x}_k y_k)$ be the *complete data*, where y_k is an unobservable integer between 1 and s, indicating the number of component density $f(\mathbf{x}|\omega_{y_k}\varphi_{y_k})$ and mixing parameter $Pr(\omega_{y_k})$ of the mixture pdf Eq. (2.4.1). Let us compute the Q-function given in Eq. (2.4.31) for multiple observed data $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ and multiple unobserved data $\mathbf{Y} = \{i_1, \ldots, i_n\}$. We assume that a parametric family of mixture probability density functions is given and that a particular Φ is the parameter value to be estimated. A log-likelihood of one complete data point $(\mathbf{x}_k, y_k)$ is obtained as:

$$f(\mathbf{x}_k, y_k|\Phi) = Pr(\omega_{y_k}) f(\mathbf{x}_k|\omega_{y_k}, \varphi_{y_k}) \qquad (2.4.32)$$

A log-likelihood of one incomplete data point $\mathbf{x}_k$ is:

$$f(\mathbf{x}_k|\Phi) = \sum_{y_k} f(\mathbf{x}_k, y_k|\Phi) = \sum_{y_k} Pr(\omega_{y_k}) f(\mathbf{x}_k|\omega_{y_k}, \varphi_{y_k}) \qquad (2.4.33)$$

Therefore, *a posteriori* probability $Pr(y_k|\mathbf{x}_k, \Phi)$ is

$$\begin{aligned} Pr(y_k|\mathbf{x}_k, \Phi) &= \frac{f(\mathbf{x}_k, y_k|\Phi)}{f(\mathbf{x}_k|\Phi)} \\ &= \frac{Pr(\omega_{y_k}) f(\mathbf{x}_k|\omega_{y_k}, \varphi_{y_k})}{\sum_{y_k} Pr(\omega_{y_k}) f(\mathbf{x}_k|\omega_{y_k}, \varphi_{y_k})} \qquad (2.4.34) \\ &= Pr(\omega_{y_k}|\mathbf{x}_k, \varphi_{y_k}) \end{aligned}$$

By inserting Eq. (2.4.32) and Eq. (2.4.34) into Eq. (2.4.31), the Q-function is obtained as:

$$\begin{aligned} Q(\Phi, \overline{\Phi}) &= \sum_{k=1}^{n} Q_k(\Phi, \overline{\Phi}) \\ &= \sum_{k=1}^{n} \sum_{y_k} \frac{f(\mathbf{x}_k, y_k|\Phi)}{f(\mathbf{x}_k|\Phi)} \log f(\mathbf{x}_k, y_k|\overline{\Phi}) \qquad (2.4.35) \\ &= \sum_{k=1}^{n} \left\{ \sum_{y_k} Pr(\omega_{y_k}|\mathbf{x}_k, \varphi_{y_k}) \log \overline{Pr}(\omega_{y_k}) f(\mathbf{x}_k|\omega_{y_k}, \overline{\varphi}_{y_k}) \right\} \end{aligned}$$

Since the summand is summed over all y_k $(1 \le y_k \le s)$, y_k does not depend on k. Therefore we denote y_k by i. Dividing the log term into two log terms, the following expression is obtained:

$$\begin{aligned} Q(\Phi, \overline{\Phi}) = &\sum_{i} \left\{ \sum_{k=1}^{n} Pr(\omega_i|\mathbf{x}_k, \varphi_i) \right\} \log \overline{Pr}(\omega_i) \qquad (2.4.36) \\ &+ \sum_{i} \left\{ \sum_{k=1}^{n} Pr(\omega_i|\mathbf{x}_k, \varphi_i) \log f(\mathbf{x}_k|\omega_i, \overline{\varphi}_i) \right\} \end{aligned}$$

Maximisation of the Q-function is obtained by maximising each term in Eq. (2.4.36) for each i with respect to $\overline{Pr}(\omega_i)$ and $\overline{\varphi}_i$. From the second term of Eq. (2.4.36), maximisation is obtained by setting the partial derivative to zero:

$$\sum_{k=1}^{n} Pr(\omega_i|\mathbf{x}_k, \varphi_i) \nabla_{\overline{\varphi}_i} \log f(\mathbf{x}_k|\omega_i, \overline{\varphi}_i) = 0 \qquad (2.4.37)$$

This is the same as Eq. (2.4.9) which is the maximisation condition of log-likelihood of incomplete data, shown in Section 2.4.1. From the first term of Eq. (2.4.36), maximisation is obtained as (see Section 2.5):

$$\overline{Pr}(\omega_i) = \frac{1}{n} \sum_{k=1}^{n} Pr(\omega_i|\mathbf{x}_k, \varphi_i) \qquad (2.4.38)$$

We can arrive at the same maximisation solution by maximum likelihood estimation and by the EM algorithm. It is easily understood that the EM algorithm is more general than maximum likelihood estimation directly applied to the log-likelihood of incomplete data, since the EM algorithm can provide the Q-function which implies the log-likelihood of complete data.

2.5. Min-max Theory [6-8]

In both maximum likelihood estimation and in the EM algorithm, maximisation operations are inevitable. Min-max theory is a traditional extremisation method widely used in many application fields. In this section, min-max theory for unconstrained optimisation and equally constrained optimisation are described.

2.5.1. Optimisation - univariate case

Consider the maximisation of the univariate objective function $f(x)$. This problem is solved by searching the local maximum of $f(x)$ and its associated point x_m. The definition of local maximum is given as:

$$f(x_m)-f(x)>0 \tag{2.5.1}$$

where x is the neighbourhood of a local maximum point x_m within a small distance Δx. In order to search the local maximum, a stationary (or critical) point x_s is searched at first, since local maximum points, local minimum points and saddle points correspond to the stationary points. The stationary point is defined as the point where function $f(x)$ does not change within some range. Therefore the total derivative $df(x)$ is zero at the stationary point. Then,

$$\frac{df(x)}{dx}=0 \qquad \text{at} \quad x=x_s \tag{2.5.2}$$

To discriminate between local maximum point, local minimum point and saddle point, the second order and the third order derivatives of $f(x)$ must be considered. If the second order derivative $f^{(2)}(x)$ satisfies the following condition:

$$f^{(2)}(x_s)>0 \tag{2.5.3}$$

it is said to be a local minimum. On the other hand, if

$$f^{(2)}(x_s)<0 \tag{2.5.4}$$

then it is a local maximum. This becomes clear when the Taylor series expansion of $f(x)$ is considered:

$$f(x_s+\Delta x)=f(x_s)+\frac{\Delta x f^{(1)}(x_s)}{1!} \tag{2.5.5}$$
$$+\frac{(\Delta x)^2 f^{(2)}(x_s)}{2!}+\frac{(\Delta x)^3 f^{(3)}(x_s)}{3!}+\cdots$$

Since x_s is a stationary point, $f^{(1)}(x_s)$ equals zero, and from Eq. (2.5.5) it follows that

$$f(x_s+\Delta x)-f(x_s)\approx\frac{(\Delta x)^2 f^{(2)}(x_s)}{2} \tag{2.5.6}$$

In the case of Eq. (2.5.4), the right-hand side of Eq. (2.5.6) is always negative independent of the sign of Δx, then

$$f(x_s)-f(x_s+\Delta x)>0 \tag{2.5.7}$$

It follows directly from the definitions given in Eq. (2.5.1) that $f(x_s)$ is a local maximum. The condition of local minimum is shown in the same way. For the saddle point, $f^{(2)}(x_s)=0$, then from the Taylor series expansion,

$$f(x_s+\Delta x)-f(x_s)\approx\frac{(\Delta x)^3 f^{(3)}(x_s)}{3!} \tag{2.5.8}$$

If $f^{(3)}(x_s)\neq 0$ then the stationary point x_s is neither a local maximum point nor a local minimum point.

In summary, the local maximum or local minimum is computed by searching for a stationary point by equating the first order derivative to zero, and then deciding its type by examining the second order or the third order derivative.

2.5.2. Optimisation - multivariate case

In the same way as for the univariate case, the local minimum, local maximum and saddle point in the multivariate case can be defined. The Taylor series expansion of $f(\mathbf{x})$ for vector $\mathbf{x}$ is given as follows:

$$f(\mathbf{x}_s+\Delta\mathbf{x})\approx f(\mathbf{x}_s)+\frac{\Delta\mathbf{x}^t\nabla f(\mathbf{x}_s)}{1!}+\frac{\Delta\mathbf{x}^t\nabla^2 f(\mathbf{x}_s)\Delta\mathbf{x}}{2!} \tag{2.5.9}$$

where $\nabla f(\mathbf{x})$ is a gradient vector whose elements are partial derivatives of $f(\mathbf{x})$ with respect to the elements of the vector $\mathbf{x}=(x_1,x_2,\ldots,x_d)^t$. Then

$$\nabla f(\mathbf{x})=(\partial f/\partial x_1,\ldots,\partial f/\partial x_d)^t \tag{2.5.10}$$

The $\nabla^2 f(\mathbf{x})$ is the Hessian matrix whose elements are the second partial derivatives of $f(\mathbf{x})$ with respect to the elements of the vector $\mathbf{x}$.

The stationary point is defined as the point where the function $f(\mathbf{x})$ does not change within some range. Therefore the total derivative $df(\mathbf{x})$ is zero at the stationary point. We know the following relationship between total derivative and partial derivative.

$$df(\mathbf{x})=\sum_{i=1}^{d}\frac{\partial f(\mathbf{x})}{\partial x_i}dx_i \tag{2.5.11}$$

Therefore, if the total derivative $df(\mathbf{x})$ equals zero, all the partial derivatives must be zero. This leads to the condition that, if $\mathbf{x}_s$ is a stationary point, $\nabla f(\mathbf{x}_s)=0$. Local minimum points and maximum points are defined by using the sign of $\Delta\mathbf{x}^t\nabla^2 f(\mathbf{x}_s)\Delta\mathbf{x}$ as in the univariate case.

As an example, let us consider maximisation of a multivariate Gaussian pdf with respect to its mean vector μ and covariance matrix Σ.

Example 2.5.1

The multivariate Gaussian pdf of a continuous random vector $\mathbf{x}$ is given:

$$N(\mathbf{x},\mu,\Sigma)=\frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}}\exp\left[-\tfrac{1}{2}(\mathbf{x}-\mu)^t\Sigma^{-1}(\mathbf{x}-\mu)\right] \tag{2.5.12}$$

The maximisation starts with finding the stationary points; then the gradient vector is set equal to zero.

$$\nabla_\mu N(\mathbf{x},\mu,\Sigma)=N(\mathbf{x},\mu,\Sigma)\Sigma^{-1}(\mathbf{x}-\mu)=0 \tag{2.5.13}$$

$$\nabla_{\Sigma^{-1}}N(\mathbf{x},\mu,\Sigma)=\tfrac{1}{2}N(\mathbf{x},\mu,\Sigma)(\Sigma-(\mathbf{x}-\mu)(\mathbf{x}-\mu)^t)=0 \tag{2.5.14}$$

These expressions are used in Sections 2.3 and 2.4 to derive the estimated parameters μ and Σ.

To prove Eq. (2.5.13) and (2.5.14), the following three expressions are useful.

(1) $\nabla_{\mathbf{b}}(\mathbf{b}^tA\mathbf{b})=A\mathbf{b}+A^t\mathbf{b}$ (2.5.15)

where A and $\mathbf{b}$ are a d-by-d matrix and d-dimensional column vector. Eq. (2.5.15) is easily shown by taking partial derivatives with respect to element b_k of the vector $\mathbf{b}$.

$$\begin{aligned}\frac{\partial}{\partial b_k}\mathbf{b}^tA\mathbf{b}&=\frac{\partial}{\partial b_k}\sum_i\sum_j b_ia_{ij}b_j\\&=\sum_j a_{kj}b_j+\sum_i b_ia_{ik}=[A\mathbf{b}+A^t\mathbf{b}]_k\end{aligned} \tag{2.5.16a}$$

where $[.]_i$ denotes the ith element of the vector $A\mathbf{b}+A^t\mathbf{b}$. When the matrix A is symmetric, Eq. (2.5.15) is:

$$\nabla_{\mathbf{b}}(\mathbf{b}^tA\mathbf{b})=2A\mathbf{b} \tag{2.5.16b}$$

(2) $\nabla_A\mathbf{b}^tA\mathbf{b}=\mathbf{b}\mathbf{b}^t$ (2.5.17)

This is shown by taking derivatives with respect the element a_{ij} of the matrix A.

$$\begin{aligned}\frac{\partial}{\partial a_{ij}}\mathbf{b}^tA\mathbf{b}&=\frac{\partial}{\partial a_{ij}}\sum_i\sum_j b_ia_{ij}b_j\\&=b_ib_j=[\mathbf{b}\mathbf{b}^t]_{i,j}\end{aligned} \tag{2.5.18}$$

where $[.]_{i,j}$ denotes the matrix element of the ith row and jth column.

(3) $\nabla_A|A|=A^{-1}|A|$ (2.5.19)

To show the above expression, the following definition and rule are used[11].

$$|A|=a_{i1}A_{i1}+\cdots+a_{ij}A_{ij}+\cdots+a_{id}A_{id} \tag{2.5.20}$$

$$|A|^{-1}=|A^{-1}| \tag{2.5.21}$$

where $|A|$ is the determinant of matrix A, and A_{ij} is the (i,j) *cofactor*. From the definition of $|A|$, it can be said that the *cofactor* A_{ik} does not include the element a_{ij} for all k. Therefore

$$\frac{\partial}{\partial a_{ij}}|A|=A_{ij}=[A_{\mathrm{cof}}]_{ij} \tag{2.5.22}$$

where A_{cof} is a cofactor matrix whose (i, j) element is the *cofactor* A_{ij}. The following relation between determinant and cofactor matrix exists:

$$A\ A_{cof}^t = |A| I \tag{2.5.23}$$

where I is a unit matrix whose diagonal elements are all unity and whose off-diagonal elements are all zero. Inserting A_{cof} of Eq. (2.5.23) into Eq. (2.5.22) and assuming symmetry of the matrix A, Eq. (2.5.19) is obtained.

Eq. (2.5.13) is derived using Eq. (2.5.15), giving

$$\begin{aligned}\nabla_\mu N(\mathbf{x},\mu,\Sigma) &= N(\mathbf{x},\mu,\Sigma)\nabla_\mu(-\tfrac{1}{2}(\mathbf{x}-\mu)^t\Sigma^{-1}(\mathbf{x}-\mu)) \\ &= N(\mathbf{x},\mu,\Sigma)\Sigma^{-1}(\mathbf{x}-\mu)\end{aligned} \tag{2.5.24}$$

and Eq. (2.5.14) is derived using Eq. (2.5.17), (2.5.19) and (2.5.21), giving

$$\begin{aligned}&\nabla_{\Sigma^{-1}} N(\mathbf{x},\mu,\Sigma) \\ &= N(\mathbf{x},\mu,\Sigma)\nabla_{\Sigma^{-1}}(-\tfrac{1}{2}(\mathbf{x}-\mu)^t\Sigma^{-1}(\mathbf{x}-\mu)) \\ &\qquad + N(\mathbf{x},\mu,\Sigma)|\Sigma|^{\frac{1}{2}}\nabla_{\Sigma^{-1}}(|\Sigma|^{-\frac{1}{2}}) \\ &= -\tfrac{1}{2}N(\mathbf{x},\mu,\Sigma)(\mathbf{x}-\mu)(\mathbf{x}-\mu)^t + \tfrac{1}{2}N(\mathbf{x},\mu,\Sigma)|\Sigma|\nabla_{\Sigma^{-1}}|\Sigma^{-1}| \\ &= \tfrac{1}{2}N(\mathbf{x},\mu,\Sigma)(\Sigma-(\mathbf{x}-\mu)(\mathbf{x}-\mu)^t)\end{aligned} \tag{2.5.25}$$

Let us consider, as another example, the problem of minimising mean square error. This problem is to minimise the expected square error between estimated value and the realised value.

Example 2.5.2

Suppose that the time sequence of data $\{x_t\}$ is given, the linearly estimated value at time t from the past p data is given as:

$$\hat{x}_t = \alpha_1 x_{t-1} + \alpha_2 x_{t-2} + \cdots + \alpha_j x_{t-j} + \cdots + \alpha_p x_{t-p} \tag{2.5.26}$$

The error between estimated value $\hat{x}_t$ and the realised value x_t is as follows:

$$e_t = x_t - \hat{x}_t = x_t - \sum_{i=1}^{p}\alpha_i x_{t-i} \tag{2.5.27}$$

Therefore the mean square error is:

$$E[e_t^2] = E[(x_t - \sum_{i=1}^{p}\alpha_i x_{t-i})^2] \tag{2.5.28}$$

where E[.] is the expectation over time t. To obtain optimal parameters $\{\alpha_i\}$ which minimise the mean square error, the gradient vector of Eq. (2.5.28) with respect to j $(1 \le j \le p)$ is set equal to zero, then

$$\begin{aligned}&\frac{\partial}{\partial\alpha_j}E[(x_t - \sum_{i=1}^{p}\alpha_i x_{t-i})^2] \\ &= E[-2(x_t - \sum_{i=1}^{p}\alpha_i x_{t-i})x_{t-j}] \\ &= -2E[x_t x_{t-j}] + 2\sum_{i=1}^{p}\alpha_i E[x_{t-i}x_{t-j}] = 0\end{aligned} \tag{2.5.29}$$

The expectation is computed over t, giving

$$E[x_{t-i}x_{t-j}] = \frac{1}{N}\sum_{t=0}^{N-1}x_{t-i}x_{t-j} \tag{2.5.30}$$

The expectation is called the auto-correlation as a function of the distance $|i-j|$ between two data points. We denote the auto-correlation here by r_{ij} and, in the special case, by r_j when $i=0$. From the definition, $r_{ij}=r_{ji}$. Then, Eq. (2.5.29) is represented as:

$$\sum_{i=1}^{p}\alpha_i r_{ij} = r_j \tag{2.5.31}$$

The above expression is known as the Yule-Waker equation and the parameter α_i can be found, by using the auto-correlation r_{ij} and r_j which can be computed from data $\{x_t\}$.

In speech signal processing, the optimised parameters $(-\alpha_i)$ are called linear predictive coefficients, and are used as feature vectors (input data) in speech recognition (see Section 3.1.3).

2.5.3. Equality constrained optimisation

In this section, an objective function $f(\mathbf{x})$ with the following equality constraint is considered.

$$f_1(\mathbf{x})=c \qquad (2.5.32)$$

where c is a constant. The most straightforward solution for this problem might be to express the element x_i in terms of other elements and substitute this into the objective function $f(\mathbf{x})$, and then unconstrained optimisation might be applied. However, in general, it is difficult to express explicitly the element x_i by other elements. Instead, the Lagrange method can be used to solve this problem efficiently.

For $\mathbf{x}$ to be a stationary point, the total derivative must be zero.

$$df(\mathbf{x})=\sum_{i=1}^{d}\frac{\partial f(\mathbf{x})}{\partial x_i}dx_i=0 \qquad (2.5.33)$$

The $\{dx_i\}$ are not independent, but they are related through the total derivative of constraint Eq. (2.5.32).

$$df_1(\mathbf{x})=\sum_{i=1}^{d}\frac{\partial f_1(\mathbf{x})}{\partial x_i}dx_i=0 \qquad (2.5.34)$$

Here multiplying Eq. (2.5.34) by a parameter λ and adding to Eq. (2.5.33), gives

$$d(f(\mathbf{x})+\lambda f_1(\mathbf{x}))=\sum_{i=1}^{d}(\frac{\partial f(\mathbf{x})}{\partial x_i}+\lambda\frac{\partial f_1(\mathbf{x})}{\partial x_i})dx_i=0 \qquad (2.5.35)$$

The above expression leads to the new constrained condition for a stationary point as follows:

$$(\frac{\partial f(\mathbf{x})}{\partial x_i}+\lambda\frac{\partial f_1(\mathbf{x})}{\partial x_i})dx_i=0 \qquad (2.5.36)$$

The parameter λ is called the Lagrange multiplier, and a stationary point is obtained by extremising the following augmented objective function f_a, under the equality constraint of Eq. (2.5.32):

$$f_a(\mathbf{x})=f(\mathbf{x})+\lambda f_1(\mathbf{x}) \qquad (2.5.37)$$

For multiple constraints $f_j(\mathbf{x})=c_j$ $(1\le j\le n)$, Eq. (2.5.35) is extended using multiple Lagrange multipliers as follows:

$$d(f(\mathbf{x})+\sum_{j=1}^{n}\lambda_j f_j(\mathbf{x})) \qquad (2.5.38)$$

$$=\sum_{i=1}^{d}(\frac{\partial f(\mathbf{x})}{\partial x_i}+\sum_{j=1}^{n}\lambda_j\frac{\partial f_j(\mathbf{x})}{\partial x_i})dx_i=0$$

Then, the following augmented objective function

$$f_a(\mathbf{x})=f(\mathbf{x})+\sum_{j=1}^{n}\lambda_j f_j(\mathbf{x}) \qquad (2.5.39)$$

is to be extremised. Let us consider an example of a case with a single constraint.

Example 2.5.3

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

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

attains its stationary point when

$$x_i = \frac{c_i}{\sum_{i=1}^{C} c_i} \tag{2.5.40}$$

The proof for the above comes from extremising the following augmented objective function

$$f_a = f(x) + \lambda \sum_{i}^{C} x_i \tag{2.5.41}$$

Equating the partial derivative of f_a to zero with respect to x_i

$$\frac{c_i}{x_i} - \lambda = 0 \tag{2.5.42}$$

Multiplying by x_i and summing over i gives $\lambda = \sum_{i}^{C} c_i$, hence the result is obtained.

Let us further consider an example of a case with multiple constraints. The following example is called Fuzzy vector quantisation and is explained further in Chapter 4.

Example 2.5.4

Consider the following objective function

$$f(m_{ij}) = \sum_{i=1}^{T}\sum_{j=1}^{L} m_{ij}^{F} d(\mathbf{x}_i, \mathbf{z}_j) \tag{2.5.43}$$

which is extremised under the constraints

$$f_i(m_{ij}) = \sum_{j=1}^{L} m_{ij} = 1 \tag{2.5.44}$$

Then the augmented objective junction is

$$f_a(m_{ij}) = \sum_{i=1}^{T}\sum_{j=1}^{L} m_{ij}^{F} d(\mathbf{x}_i, \mathbf{z}_j) + \sum_{i=1}^{T} \lambda_i \sum_{j=1}^{L} m_{ij} \tag{2.5.45}$$

Equating an element of the gradient vector of f_a to zero:

$$\frac{\partial f_a(m_{ij})}{\partial m_{ij}} = F m_{ij}^{(F-1)} d(\mathbf{x}_i, \mathbf{z}_j) + \lambda_i = 0 \tag{2.5.46}$$

We obtain

$$m_{ij} = (-\lambda_i / F d(\mathbf{x}_i, \mathbf{z}_j))^{\frac{1}{F-1}} \tag{2.5.47}$$

From the constraint Eq. (2.5.44) of m_{ij}

$$\sum_{j=1}^{L} m_{ij} = (-\lambda_i / F)^{\frac{1}{F-1}} \sum_{j=1}^{L} (1/d(\mathbf{x}_i, \mathbf{z}_j))^{\frac{1}{F-1}} = 1 \tag{2.5.48}$$

Then parameters λ_i are:

$$\begin{aligned} -\lambda_i &= F/(\sum_{j=1}^{L} (1/d(\mathbf{x}_i, \mathbf{z}_j))^{\frac{1}{F-1}})^{(F-1)} \\ &= F(\sum_{k=1}^{L} (1/d(\mathbf{x}_i, \mathbf{z}_k))^{\frac{1}{F-1}})^{-(F-1)} \end{aligned} \tag{2.5.49}$$

Substituting the above expression into Eq. (2.5.46), we have

$$m_{ij}^{(F-1)} d(\mathbf{x}_i, \mathbf{z}_j) = (\sum_{k=1}^{L} (1/d(\mathbf{x}_i, \mathbf{z}_k))^{\frac{1}{F-1}})^{-(F-1)} \tag{2.5.50}$$

Then m_{ij} which is an element of a stationary point is:

$$m_{ij} = (\sum_{k=1}^{L} (d(\mathbf{x}_i, \mathbf{z}_j)/d(\mathbf{x}_i, \mathbf{z}_k))^{\frac{1}{F-1}})^{-1} \tag{2.5.51}$$

The m_{ij} can be considered as the contribution of data $\mathbf{x}_i$ to the reference point $\mathbf{z}_j$, and form the basis of a Fuzzy clustering technique.

2.6. Information Theory [1]

In pattern recognition, parameters of probability structure are estimated by maximising the log-likelihood, to increase the probability of correct classification of the input data. Another criterion is mutual information, which can be maximised, instead of log-likelihood, in order to improve channel quality between input and output. In this section, the concept of entropy and mutual information in information channel is briefly described.

2.6.1. Entropy

When we observe an outcome a_i, the information derivable from the outcome will depend on its probability. If the probability $Pr(a_i)$ is small, we can derive a large degree of information because the outcome which has occurred is very rare. On the other hand, if the probability is large, the information derived may be small because the outcome is well expected. The amount of information is defined as follows:

$$I(a_i)=\log\frac{1}{Pr(a_i)} \tag{2.6.1}$$

For logarithms to base 2, the unit of information is called the *bit*. This means that one bit of information is required to specify what kind of outcome has occurred. In this sense, the amount of information represents some ambiguity.

In information theory, an outcome a_i is called a *symbol*, and the sample space S is called an *alphabet*. In this section, we use these terms according to normal convention. The symbol a_i is produced from an information source with alphabet S, according to the probability of the symbol. The important property of an information source is the entropy $H(S)$ which is defined as the average amount of information as follows:

$$\begin{aligned} H(S) &= \sum_S Pr(a_i)I(a_i) \\ &= \sum_S Pr(a_i)\log\frac{1}{Pr(a_i)} \end{aligned} \tag{2.6.2}$$

Where, $\sum_S$ indicates summation over all symbols on alphabet S. This entropy $H(S)$ is the amount of information required in specifying what kind of symbol has occurred on average. It is also the averaged ambiguity for the symbol. In coding the symbols, entropy $H(S)$ is the lower limit of averaged code length necessary to transmit the symbols. This is called Shannon's first theorem. If the entropy increases, then ambiguity increases, therefore a large amount of information is required to specify and transmit the symbol.

2.6.2. Mutual information

Here, we consider transmission of symbols through an information channel. Suppose the input alphabet is $A=\{a_i\}$, $i=1,2,..,r$ and the output alphabet is $B=\{b_j\}$, $j=1,2,...,s$, then the information channel is defined by the channel matrix $M_{ij}=Pr(b_j|a_i)$, where $Pr(b_j|a_i)$ is the conditional probability that output symbol b_j is received when input symbol a_i is sent. Figure 2.6.1 shows an example of an information channel.

Before transmission of a symbol a_i, the average amount of information, or the ambiguity of the input alphabet A is the *a priori* entropy $H(A)$.

$$H(A)=\sum_A Pr(a_i)\log\frac{1}{Pr(a_i)} \tag{2.6.3}$$

where $Pr(a_i)$ is the *a priori* probability.

After transmission, for the observed symbol b_j, the average amount of information, or the ambiguity of the input alphabet A, is reduced to the following *a posteriori* entropy.

$$H(A|b_j)=\sum_A Pr(a_i|b_j)\log\frac{1}{Pr(a_i|b_j)} \tag{2.6.4}$$

where the $Pr(a_i|b_j)$ are the *a posteriori* probabilities.

		Output alphabet										
		b_1	b_2	b_3	.	.	.	.	.	.	.	b_s
Input alphabet	a_1	1/5	1/5	1/5	0	0	0	0	0	1/5	0	1/5
	a_2	0	0	1/2	0	1/8	0	0	1/4	0	0	1/8
	a_3	1/6	0	0	0	0	0	0	2/3	0	1/6	0
	.											
	.											
	.											
	.											
	.											
	a_r	0	0	0	0	0	0	0	0	0	1/2	1/2

Figure 2.6.1. Example of information channel.

Averaging the *a posteriori* entropy $H(A|b_j)$ over all output symbols b_j leads to the following equation:

$$H(A|B)=\sum_B Pr(b_j)H(A|b_j)$$

$$=\sum_B Pr(b_j)\sum_A Pr(a_i|b_j)\log\frac{1}{Pr(a_i|b_j)} \qquad (2.6.5)$$

$$=\sum_A\sum_B Pr(a_i,b_j)log\frac{1}{Pr(a_i|b_j)}$$

This conditional entropy is the average amount of information, or the ambiguity of the input alphabet A required in specifying the input symbol, after observing an output symbol.

Mutual information is defined as the reduction in ambiguity, in other words, information obtained through a channel by observing an output symbol.

$$I(A;B)=H(A)-H(A|B)$$

$$=\sum_A Pr(a_i)\log\frac{1}{Pr(a_i)}-\sum_A\sum_B Pr(a_i,b_j)\log\frac{1}{Pr(a_i|b_j)}$$

$$=\sum_A\sum_B Pr(a_i,b_j)\log\frac{1}{Pr(a_i)}$$

$$-\sum_A\sum_B Pr(a_i,b_j)\log\frac{1}{Pr(a_i|b_j)} \qquad (2.6.6)$$

$$=\sum_A\sum_B Pr(a_i,b_j)\log\frac{Pr(a_i|b_j)}{Pr(a_i)}$$

$$=\sum_A\sum_B Pr(a_i,b_j)\log\frac{Pr(a_i,b_j)}{Pr(a_i)Pr(b_j)}$$

If the information channel is noiseless, the input symbol can be specified definitely by observing an output symbol. In this case, the conditional entropy $H(A|B)$ equals zero. Therefore, we can obtain the maximum mutual information $I(A;B)=H(A)$. In the general case, the information channel is noisy so that the conditional entropy $H(A|B)$ is not zero. Then, maximising the mutual information means (if the channel matrix can be tuned) obtaining a low noise information channel, offering a close relationship between input and output alphabet.

2.7. Summary

This chapter has introduced some of the fundamentals of statistical pattern recognition. The basic concept in statistical pattern recognition involves probability theory and Bayes decision theory as these are essentially based on probabilistic decision making with *a priori* and *a posteriori* knowledge of the patterns. The most important problem in pattern recognition is how to obtain the *a posteriori* knowledge from given examples. Supervised and unsupervised methods to estimate the category-conditional probability density function of the patterns have been described. In particular, the EM algorithm, owing to its elegance in formalisation of iterative learning, has been described in depth comparing it with the traditional maximum likelihood estimation. Finally, a brief review of information theory has been included, since such concepts are essential to the acoustic-phonetic and acoustic-language modelling presented later in this text.

References

1. N. Abramson, *Information Theory and Coding*, McGraw Hill, 1963.
2. L.E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains," *Ann. Math. Stat.*, vol. 41, pp. 164-171 , 1970.
3. 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.
4. 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.
5. R.O. Duda and R.E. Hart, *Pattern Classification and Scene Analysis*, John Wiley, 1973.
6. G.Stephenson, *Mathematical Methods for Science Students*, Longman, 1973.
7. P.E. Gill, W. Murray, and M.H. Wright, *Practical Optimization*, Academic Press, 1981.
8. D.A. Pierre, *Optimization Theory with Applications*, Dover Edition, 1986.
9. R.A. Redner and H.F. Walker, "Mixture densities, maximum likelihood and the EM algorithm," *SIAM review*, vol. 26, pp. 195-239, 1984.
10. K.S. Shanmugan and A.M.Breipohl, *Random Signals: Detection, Estimation and Data Analysis*, John Wiley, 1988.

11. G. Strang, *Linear Algebra and its Applications*, Harcourt Brace Jovanovich, 1988.

CHAPTER THREE

BASIC TECHNIQUES FOR SPEECH PROCESSING

The task of speech recognition is to recover a linguistic message which is encoded as 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 databases, and the creation of powerful new statistical algorithms have 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 units, usually words. The language modelling stage determines valid linguistic words or sentences. 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 presents some basic knowledge required to understand speech recognition methodology. The description does not cover all topics, but is restricted to those which are essential for understanding the following chapters.

3.1. Speech Signal Processing

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 [34,84]. 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 involve characteristics that are not directly evident in the time domain. These are the most widely used signal analysis techniques in speech recognition.

Associated with a given 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 representations of speech signal spectra, notably those based on linear predictive coding (LPC) [4,49,69,84] analysis and those based on short-time Fourier analysis [25,84].

3.1.1. Short-time Fourier analysis

Short-time Fourier (spectral) analysis is a method for analysing time-varying waveforms in the frequency domain. Components of the speech signal are time-varying at the *articulator rate*, so the speech signal is suited to short-time analysis. A number of fundamental concepts and definitions of short-time Fourier analysis can be found in [34,84], and details of a typical speech recognition system based on short-time Fourier analysis can be found in [53].

Short-time analysis depends on *windowing* of the speech signal to isolate a short-time interval for spectral analysis. The short-time analysis interval is called a *frame*, and the length of the frame is called the *frame length*. The windowing proceeds along the time axis by shifting an appropriate interval to represent the temporal dynamic feature. The shifting interval is called the *frame interval*. The role of windowing is to prevent an abrupt change at the end points of the frame by attenuating the amplitude of the speech signal, as well as to represent temporal dynamic features. The windowing is carried out by multiplying the speech signal by an appropriate window function, and the results depend on the properties of the specific window function employed.

Let a continuous speech signal be denoted $s(t)$ and the window function by $w(t-\tau)$, then the signal after windowing is given as:

$$x(t,\tau)=s(t)w(t-\tau) \qquad (3.1.1)$$

where τ is a time when the window is applied. $x(t,\tau)$ is a signal as a function of time t with the window position τ.

Short-time Fourier analysis is carried out on the signal $x(t,\tau)$ by the Fourier transform. The Fourier transform is a mapping function from time domain to spectral domain, and its formula in continuous frequency is defined as:

$$X(j\omega,\tau)=\int_{-\infty}^{\infty} x(t,\tau)e^{-j\omega t}dt \qquad (3.1.2)$$

where $\omega=2\pi f$ is radian frequency. The corresponding inverse Fourier transform, which is a mapping function from spectral domain to time domain, is defined as:

$$x(t,\tau)=\frac{1}{2\pi}\int_{-\infty}^{\infty} X(j\omega,\tau)e^{j\omega t}d\omega \qquad (3.1.3)$$

Since, in practice, the continuous time signal $x(t,\tau)$ and its spectrum $X(j\omega,\tau)$ are quantised by sampling and digitising for computer processing, techniques of digital

signal processing are mainly employed. Suppose that the continuous signal $x(t,\tau)$ is sampled at a sampling period of T seconds, the discrete sampled data results in $x_{k,i}$, where k and i indicate discrete time and i corresponds to the time when a window is applied, like τ in the continuous case. The corresponding discrete Fourier transform of a discrete sample sequence $\{x_{k,i}\}$ is defined as:

$$X_{n,i}=\sum_{k=0}^{N-1} x_{k,i}(e^{j2\pi/N})^{-kn} \qquad (3.1.4)$$

where N is the number of sampled data to be analysed (frame length). The inverse discrete Fourier transform is defined as:

$$x_{k,i}=\frac{1}{N}\sum_{n=0}^{N-1} X_{n,i}(e^{j2\pi/N})^{kn} \qquad (3.1.5)$$

A fast algorithm for computation of the discrete Fourier transform is called an FFT (Fast Fourier Transform) and is normally applicable where N is a power of 2.

An interesting question involves the relation between frequency resolution and time resolution. The time resolution indicates the number of data samples to be analysed (frame length). Since frames with short frame length can represent rapidly changing dynamic features, high time resolution is obtained by shortening the frame length. On the other hand, frequency resolution limited by the frequency step Δf is defined as:

$$\Delta f=\frac{1}{NT} \qquad (3.1.6)$$

where N is the frame length and T is the sampling period. The above expression is easily obtained by comparing Eq. (3.1.2) and (3.1.4), setting $kT \rightarrow t$ and $n/NT \rightarrow n\Delta f \rightarrow f$. From Eq. (3.1.6), it is clear that long frame length N increases frequency resolution Δf and decreases time resolution. Short frame length increases time resolution and decreases frequency resolution.

Example 3.1.1

The functions sinc and rect are defined by

$$\mathrm{sinc}(x)=\begin{cases}1, & \text{if } x=0\\ \sin(x)/x, & \text{otherwise}\end{cases} \tag{3.1.7}$$

$$\mathrm{rect}(x)=\begin{cases}1, & \text{if } |x|\le 1\\ 0, & \text{otherwise}\end{cases} \tag{3.1.8}$$

The inverse Fourier transform of $\mathrm{rect}(2\omega/\omega_0)$ is shown to be $\frac{1}{T}\mathrm{sinc}(\frac{\omega_0}{2}t)$. Starting from the inverse Fourier transform of $\mathrm{rect}(2\omega/\omega_0)$, the following expression is obtained:

$$\frac{1}{2\pi}\int_{-\infty}^{\infty}\mathrm{rect}(2\omega/\omega_0)e^{j\omega t}d\omega \tag{3.1.9}$$

$$=\frac{1}{2\pi}\int_{-\omega_0/2}^{\omega_0/2}e^{j\omega t}d\omega=\frac{1}{2\pi jt}(e^{j\omega_0 t/2}-e^{-j\omega_0 t/2})$$

$$=\frac{1}{T}\frac{\sin(\frac{\omega_0}{2}t)}{\frac{\omega_0}{2}t}=\frac{1}{T}\mathrm{sinc}(\frac{\omega_0}{2}t)$$

Bandpass filter analysis methods can be considered as a special form of short-time Fourier analysis [84], and such bandpass filter methods have been widely used in speech recognition [18,25]. The bandpass filter system may use a variety of different frequency spacings [25,26,107], 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.

3.1.2. The z-transform [84,89]

The z-transform plays an important role in the analysis and representation of discrete-time systems. Many properties of a discrete sample sequence can be well observed after a z-transform representation. Consider the discrete sample sequence denoted $\{x_k\}$, then its z-transform is defined as:

$$X(z)=\sum_{k=-\infty}^{\infty}x_k z^{-k} \tag{3.1.10}$$

where z is a complex variable. The window function is not considered here for simplicity. The continuous spectrum may be obtained by letting $z=e^{j\omega T}$. This can be roughly explained by modifying Eq. (3.1.4) as follows:

$$\begin{aligned}X_n&=\sum_{k=0}^{N-1}x_k(e^{j2\pi nT/NT})^{-k}\\ &=\sum_{k=0}^{N-1}x_k(e^{j\omega T})^{-k}\end{aligned} \tag{3.1.11}$$

where $\omega=2\pi n/NT=2\pi n\Delta f$. If the frame length N becomes large (to infinity), $\Delta f=1/NT$ will be unlimitedly small as discussed in the previous section, and $f=n\Delta f$ becomes continuous. Therefore, it can be said that the z-transform Eq. (3.1.10) is the mapping function from the discrete sample sequence $\{x_k\}$ to the continuous spectrum.

The corresponding inverse z-transform is defined as:

$$x_k=\frac{1}{2\pi j}\int_c X(z)z^{k-1}dz \tag{3.1.12}$$

where c is a circular contour centered at the origin and lying in the region of convergence on the z-plane.

Example 3.1.2

Here we will show why the z-transform can convert the discrete sample sequence to the continuous spectrum. The sampling theorem tells us that the continuous signal $x(t)$ is completely represented by the discrete sample sequence $\{x_k\}$ under the condition that the sampling frequency $1/T$ is twice the highest frequency contained in the continuous signal $x(t)$. The formula of the sampling theorem is:

$$x(t)=\sum_{k=-\infty}^{\infty} x_k \mathrm{sinc}(\frac{\omega_0}{2}(t-kT)) \qquad (3.1.13)$$

where $\omega_0=2\pi/T$ and $\mathrm{sinc}(t)=\sin(t)/t$ (see Example 3.1.1). By applying the Fourier transform to both sides of Eq. (3.1.13), the left side is converted to $X(j\omega)$ and the function $\mathrm{sinc}(\frac{\omega_0}{2}(t-kT))$ of the right side is converted to:

$$\int_{-\infty}^{\infty}\mathrm{sinc}(\frac{\omega_0}{2}(t-kT))e^{-j\omega t}dt=\int_{-\infty}^{\infty}\mathrm{sinc}(\frac{\omega_0}{2}x)e^{-j\omega x}dx\; e^{-j\omega Tk} \qquad (3.1.14)$$

In the above expression, we set $t-kT=x$. Using the result of Example 3.1.1, the above expression is further converted to:

$$T\ \mathrm{rect}(2\omega/\omega_0)e^{-j\omega Tk}=Te^{-j\omega Tk} \qquad (|\omega|\le\omega_0/2) \qquad (3.1.15)$$

Therefore, the Fourier transform of Eq. (3.1.13) is:

$$X(j\omega)=T\sum_{k=-\infty}^{\infty} x_k e^{-j\omega Tk} \qquad (|\omega|\le\omega_0/2) \qquad (3.1.16)$$

Here let z denote $e^{j\omega T}$, then Eq. (3.1.16) is:

$$X(j\omega)=T\sum_{k=-\infty}^{\infty} x_k z^{-k} \qquad (3.1.17)$$

By the definition of the z-transform Eq. (3.1.10), the above expression is represented as:

$$X(j\omega)=TX(z) \qquad z=e^{j\omega T} \qquad (3.1.18)$$

Eq. (3.1.18) means that the z-transform of the discrete sample sequence $\{x_k\}$ can represent the spectrum of the continuous time signal $x(t)$ which is completely recovered by the discrete sample sequence $\{x_k\}$, by letting $z=e^{j\omega T}$. This advantage of the z-transform enables us to analyse, in the frequency domain, the behaviour of a speech processing (filter) which works in the time domain.

Fourier transforms can replace differentiation with an algebraic operation. In the same way, z-transforms can replace differencing with an algebraic operation. Let us investigate this z-transform property.

Example 3.1.3

Let us consider the z-transform of a linear combination of discrete signal x_t. The problem is to find the z-transform of the following expression:

$$e_t=\sum_{i=0}^{p}\alpha_i x_{t-i} \qquad (3.1.19)$$

where t shows discrete time. By applying Eq. (3.1.10) to (3.1.19)

$$E(z)=\sum_{t=-\infty}^{\infty}\ \sum_{i=0}^{p}\alpha_i x_{t-i}z^{-t} \qquad (3.1.20)$$

Here, setting $t-i=k$,

$$\begin{aligned}E(z)&=\sum_{i=0}^{p}\alpha_i(\sum_{k=-\infty}^{\infty}x_k z^{-k})z^{-i}\\&=\sum_{i=0}^{p}\alpha_i z^{-i}X(z)\end{aligned} \qquad (3.1.21)$$

The above expression indicates that the discrete sample sequence $\{x_{t-i}\}$ is converted to the z-plane in the form of $X(z)z^{-i}$ by the z-transform. Therefore the time delay iT is represented as z^{-i} in the z-plane.

Convolution is one of the most commonly used operations in filtering theory. Convolution in the time domain is simply represented as an algebraic multiplication in the frequency domain by the z-transform.

Example 3.1.4

The convolution of two discrete sample sequences $\{x_k\}$ and $\{c_k\}$ is defined as follows:

$$y_n = \sum_{k=-\infty}^{\infty} x_k c_{n-k} \tag{3.1.22}$$

Let us derive the z-transform of the convolution. By directly applying the z-transform to Eq. (3.1.22),

$$Y(z) = \sum_{n=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} x_k c_{n-k} z^{-n} \tag{3.1.23}$$

By replacing $n-k$ with i, Eq. (3.1.21) is

$$\begin{aligned} Y(z) &= \sum_{i=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} x_k c_i z^{-k} z^{-i} \\ &= X(z)C(z) \end{aligned} \tag{3.1.24}$$

Then, the z-transform of the convolution becomes the product of two z-transforms of the respective discrete sample sequences.

3.1.3. LPC analysis [84]

Linear predictive coding (LPC) can provide a complete description for a speech production model. The basic idea underlying LPC is that each discrete speech sample, x_t, 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

$$e_t = x_t + \sum_{i=1}^{p} \alpha_i x_{t-i}. \tag{3.1.25}$$

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 as is seen in Example 2.5.2. The linear equation (2.5.31) in Example 2.5.2 is efficiently solved by Levinson's recursive solution methods[69].

For the purpose of understanding the behaviour in the frequency domain of LPC processing which deals with discrete speech samples, let us apply the z-transform to Eq. (3.1.25). From Example 3.1.3, the following expression is obtained:

$$E(z) = \sum_{i=0}^{p} \alpha_i z^{-i} X(z) \tag{3.1.26}$$

Let us denote $H(z)$ as follows:

$$H(z) = 1/\sum_{i=0}^{p} \alpha_i z^{-i} \tag{3.1.27}$$

then Eq. (3.1.26) is expressed as:

$$X(z) = H(z)E(z) \tag{3.1.28}$$

The spectra of e_t and x_t are obtained by setting $z = e^{j\omega T}$. Since the denominator of $H(z)$ has p complex roots, the $H(e^{j\omega T})$ has $p/2$ resonant frequencies, which correspond to formant frequencies. This implies that the LPC technique can model the spectrum of the vocal tract as a spectrum of an order-p all-pole model $H(z)$. The expression Eq. (3.1.28) indicates that the spectrum $X(z)$ of discrete speech samples is produced as the product of the spectrum $H(z)$ of the vocal tract and the spectrum $E(z)$, which is the spectrum of the unpredictable signal formed from the past p speech samples. Therefore the $E(z)$ corresponds to the spectrum of voice excitation.

Suppose that the voice excitation is white noise in the case of unvoiced speech, and an impulse in the case of voiced speech, so that the spectrum of voice excitation $E(z)$ becomes spectrally flat. Under such conditions, the speech spectrum $X(z)$ equals the spectrum at the vocal tract, so that

$$X(z) = GH(z) \tag{3.1.29}$$

where G is the gain constant.

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 $n+4$ [69]. Detailed description of such LPC details can be found in [34,69,84].

From the stochastic process point of view, a speech sample, x_t, can be considered as a time series of a stationary Gaussian process. An N-sample segment of a speech sample, $\mathbf{x} = (x_{c+1}, x_{c+2}, \ldots, x_{c+N})^t$ (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, $\{\alpha_i\}$, of the process density [49], 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 (see Section 2.3.1). The purpose is to use the information provided by known samples to obtain good estimates for unknown parameters. Intuitively, good estimates should correspond to the value that in some sense best agrees with the actual observed samples. The *likelihood* can then be defined as a function of parameters, $\{\alpha_i\}$, with respect to the set of samples, namely, $Pr(\mathbf{x}|\alpha)$. The maximum likelihood estimate of $\{\alpha_i\}$ is then that value which maximises the likelihood function. From such a statistical point of view, LPC analysis can be closely welded into hidden Markov modelling [56,82] to provide a computationally efficient model for speech recognition.

3.1.4. Cepstral analysis [92]

The basic model of speech production can be considered as a vocal tract filter $H(z)$ excited by a periodic excitation function $E(z)$ for voiced speech or white noise $E(z)$ in the case of unvoiced speech. Therefore short-time spectra comprise a slowly varying spectral 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 [84]. The observed speech sample sequence results in a convolution of the excitation and the vocal tract impulse response in the time domain, because its spectrum is the product of the excitation and the filter spectra in the frequency domain as shown in the previous section (see Example 3.1.4).

If, in the frequency domain, the product of the excitation and filter spectrum is transformed to the summation of these two spectra (logarithm operation), the transformation from the frequency domain back to the time domain by Fourier transform results in the *cepstrum*, which can represent the excitation and vocal tract separately. The parameter for cepstrum is called *quefrency* and is effectively a (pseudo) time domain parameter. The excitation locates at high *quefrency* owing to its periodic high frequency, and the vocal tract locates at low *quefrency* owing to its smoothed spectral envelope. This separable representation is very suitable to the deconvolution of speech and this analysis is called cepstral analysis [84,95]. In general, this kind of analysis which can separate two convolutionally related properties into a summation by some transformation is called homomorphic analysis [84].

There are two types of cepstral analysis: FFT cepstral and LPC cepstral analysis [6,79]. In the FFT cepstral analysis, a fast Fourier transform is directly applied to the speech signal. On the other hand, in LPC cepstral analysis, the z-transform is applied to the speech signal modelled by LPC analysis. Here, we will investigate LPC cepstral analysis, especially in deriving the LPC cepstral coefficients from LPC coefficients.

To investigate properties of the *LPC cesptrum*, the excitation $E(z)$ and vocal tract filter $H(z)$, in the speech spectrum $X(z)$, are linearly separated by a complex logarithm operation applied to Eq. (3.1.28). Then

$$\log X(z)=\log H(z)+\log E(z) \tag{3.1.30}$$

The LPC cepstral coefficients c_n are defined as the inverse z-transform of the above log-spectrum $\log X(z)$. This indicates that the characteristics of vocal tract and excitation are well represented separately in the cepstral coefficients. The higher order coefficients take the excitation property and the lower order coefficients take the vocal tract property.

Cepstral coefficients, which can also be obtained from LPC analysis [84], 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.

$$c_n = -\alpha_n - \sum_{i=1}^{n-1} \frac{n-i}{n} \alpha_i c_{n-i}, \quad n \geq 1 \tag{3.1.31}$$

where $\alpha_i=0$ when $i>p$ (p is the order of LPC analysis).

A variety of speech recognition systems using cepstral analysis have been reported [22,26,62,87]. A distinctive advantage of the cepstral analysis is that correlation between coefficients is extremely small so that simplified modelling assumptions can be applied.

Example 3.1.5

Let us derive Eq. (3.1.31). Since the z-transform of the cepstral coefficients is equal to the log-spectrum $X(z)$,

$$\log X(z)=C(z) \tag{3.1.32}$$

where $C(z)$ is the following z-transform of cepstral coefficients c_n

$$C(z)=\sum_{n=-\infty}^{\infty} c_n z^{-n} \tag{3.1.33}$$

To take out the logarithm operation, the derivative of Eq. (3.1.32) is computed with respect to z^{-1}:

$$\frac{\partial X(z)}{\partial z^{-1}}=X(z)\frac{\partial C(z)}{\partial z^{-1}} \tag{3.1.34}$$

Here, the speech spectrum $X(z)$ is modelled by the all-pole model with LPC coefficients α_i, giving

$$X(z)=\frac{G}{\sum_{i=-\infty}^{\infty} \alpha_i z^{-i}}=\frac{G}{Z_T(\alpha_i)} \tag{3.1.35}$$

where Z_T denotes the z-transform operation on the sequence inside the parentheses, and $\alpha_i=0$ when i is negative or greater than the order p of LPC analysis. The derivatives of $X(z)$ and $C(z)$ are:

$$\frac{\partial X(z)}{\partial z^{-1}}=\frac{-Gz\sum_{i=-\infty}^{\infty} i\alpha_i z^{-i}}{(\sum_{i=-\infty}^{\infty} \alpha_i z^{-i})^2}=\frac{-GZ_T(i\alpha_i)z}{Z_T(\alpha_i)^2} \tag{3.1.36}$$

$$\frac{\partial C(z)}{\partial z^{-1}}=z\sum_{n=-\infty}^{\infty} nc_n z^{-n}=Z_T(nc_n)z$$

By substituting Eq. (3.1.35) and (3.1.36) into (3.1.34), the following z-transform equation is obtained:

$$Z_T(nc_n)Z_T(\alpha_i)+Z_T(i\alpha_i)=0 \tag{3.1.37}$$

By applying the inverse z-transform to Eq. (3.1.37), the relational equation between LPC coefficient α_i and LPC cepstral coefficient c_n is obtained. The first term of the above equation is the product of two z-transforms, so it is represented in the convolution form by applying the inverse z-transform.

$$\sum_{i=0}^{n-1} \alpha_i(n-i)c_{n-i}+n\alpha_n \tag{3.1.38}$$

$$=\sum_{i=1}^{n-1}(n-i)\alpha_i c_{n-i}+n\alpha_n+nc_n=0$$

Finally, c_n is represented as:

$$c_n = -\alpha_n - \sum_{i=1}^{n-1} \frac{n-i}{n} \alpha_i c_{n-i}, \quad n \ge 1 \qquad (3.1.39)$$

where $\alpha_i = 0$ when $i > p$.

3.1.5. The distance measure [92]

The distance measure, also known as the distortion or dissimilarity measure, between two speech frames has played a key role in speech coding, analysis, and recognition. In order to discriminate spoken phonemes or words, the distance between templates or prototypes and unknown input frames must be defined, and we would like the defined distance to reflect the physical spectral distance. Several studies have been conducted to investigate the properties of distance measures from both theoretical and practical points of view [40,41,78]. The most widely used distance measure in speech recognition is the LPC cepstral distance, owing to its direct correspondence to spectral distance and the computational simplicity of LPC cepstral coefficients given by Eq. (3.1.31).

Example 3.1.6

The LPC cepstral distance can be shown to correspond to the distance in the spectrum.

Power spectrum is defined as the power of the spectrum $X(z)$ at $z = e^{j\omega T}$

$$\textit{Power spectrum} = |X(z)|^2 \qquad (z = e^{j\omega T}) \qquad (3.1.40)$$

Log power spectrum is defined as the log of the power spectrum. The difference between two log power spectra of template frame **y** and input frame **x** is:

$$d = \log|X(z)|^2 - \log|Y(z)|^2 \qquad (3.1.41)$$

On the basis of the above definition, the spectral distance is defined as:

$$L = \frac{T}{2\pi} \int_{-\pi/T}^{\pi/T} d^2 d\omega \qquad (3.1.42)$$

where the lower and upper limits of the integral are decided according to the sampling theorem. The sampling frequency $1/T$ must be twice the highest frequency contained in the speech ($2f \le 1/T$). Then

$$-\frac{\pi}{T} \le \omega = 2\pi f \le \frac{\pi}{T} \qquad (3.1.43)$$

Our purpose is to show that the spectral distance L corresponds to the LPC cepstral distance (Euclid distance in LPC cepstrum).

$$L = \sum_{n=-\infty}^{\infty} (c_n^x - c_n^y)^2 \qquad (3.1.44)$$

where c_n^x and c_n^y are the nth cepstral coefficients of the input frame **x** and the template frame **y**.

To prove Eq. (3.1.44), the following expression is used on the basis of complex logarithm.

$$\log|X(z)| = \mathrm{Re}(\log X(z)) \qquad (3.1.45)$$

where Re(.) indicates the real part of the complex variable. The above expression is shown by substituting the complex variable $z = e^{\alpha + j\beta}$. From Eq. (3.1.32), the z-transform $C(z)$ of the cepstral coefficients c_n is equal to the log-spectrum $X(z)$,

$$\log X(z) = C(z) \qquad (3.1.46)$$

The difference d between log power spectra is:

$$\begin{aligned} d &= \log|X(z)|^2 - \log|Y(z)|^2 \qquad (3.1.47) \\ &= 2\mathrm{Re}(\log X(z) - \log Y(z)) \\ &= 2\mathrm{Re}\Big(\sum_{n=-\infty}^{\infty} (c_n^x - c_n^y) z^{-n}\Big) \qquad (c_n^x = c_n^y = 0, \text{ if } n < 0) \end{aligned}$$

$$= \sum_{n=-\infty}^{\infty} (c_n^x - c_n^y) z^{-n} \qquad (c_n^x = c_{-n}^x,\ c_n^y = c_{-n}^y)$$

The spectral distance L (Eq. (3.1.42)) is given by:

$$L = \frac{T}{2\pi} \int_{-\frac{\pi}{T}}^{\frac{\pi}{T}} d^2 d\omega \tag{3.1.48}$$

$$= \frac{T}{2\pi} \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} (c_n^x - c_n^y)(c_m^x - c_m^y) \int_{-\frac{\pi}{T}}^{\frac{\pi}{T}} e^{-j\omega T(n-m)} d\omega$$

$$= \sum_{n=-\infty}^{\infty} (c_n^x - c_n^y)^2$$

Other distance measures may also be defined by exploiting the ratio of power spectra, such as

$$d_1 = \frac{|Y(z)|^2}{|X(z)|^2} = e^{-d} \tag{3.1.49}$$

where $d = \log|X(z)|^2 - \log|Y(z)|^2$. The maximum likelihood distance (Itakura-Saito distance) is defined as:

$$E = \frac{T}{2\pi} \int_{-\frac{\pi}{T}}^{\frac{\pi}{T}} (d + e^{-d} - 1) d\omega \tag{3.1.50}$$

The maximum likelihood distance weights linearly, when $d >> 0$. When $d << 0$, the distance weights exponentially. The cosh measure is defined as:

$$D = \int_{-\frac{\pi}{T}}^{\frac{\pi}{T}} ((d + e^{-d} - 1) + (-d + e^{d} - 1)) d\omega \tag{3.1.51}$$

$$= \int_{-\frac{\pi}{T}}^{\frac{\pi}{T}} (e^{d} + e^{-d} - 2) d\omega$$

This distance measure is devised to guarantee symmetry with respect to d.

The distance measure can generally be replaced by employing a continuous probability density function, in which the larger the density, the smaller the distance, because of the uncertainty and randomness of speech signals. 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 with conventional distance measures.

Example 3.1.7

Let us consider the case of the minimum-error-rate classifier described in Section 2.2.3. In the classifier, input speech data $\mathbf{x}$ (such as LPC cepstral coefficients) for each frame are classified into one of the categories φ_k $(1 \le k \le S)$, based on the minimum value of the discriminant functions. The discriminant function is the *a posteriori* probability $Pr(\varphi_k|\mathbf{x})$ which can be computed by Bayes rule using *a priori* probability $Pr(\varphi_k)$ and category conditional probability density function (pdf) $f(\mathbf{x}|\varphi_k)$. Then, the log discriminant function $g_k(\mathbf{x})$ of the category k is given as:

$$g_k(\mathbf{x}) = \log f(\mathbf{x}|\varphi_k) + \log Pr(\varphi_k) \tag{3.1.52}$$

Here we employ the most widely used Gaussian density function as the category conditional pdf. The above log discriminant function becomes:

$$g_k(\mathbf{x}) = -\tfrac{1}{2}(\mathbf{x} - \mu_k)^t \Sigma_k^{-1} (\mathbf{x} - \mu_k) \tag{3.1.53}$$

$$- \frac{d}{2}\log 2\pi - \tfrac{1}{2}\log|\Sigma_k| + \log Pr(\varphi_k)$$

The above discriminant function $g_k(\mathbf{x})$ can be simplified in various ways. In the case where the covariance matrix Σ_k of each category is pooled irrespective of the category, and the *a priori* probability is the same for all categories, the discriminant function deteriorates to the following function:

$$g_k(\mathbf{x})=(\mathbf{x}-\mu_k)^t\Sigma^{-1}(\mathbf{x}-\mu_k) \qquad (3.1.54)$$

where Σ is the pooled covariance matrix. The discriminant function is the *Mahalanobis distance* between the input speech data $\mathbf{x}$ and the template μ_k of category k. The classification is carried out according to the minimum value of the *Mahalanobis distance*. In the case where the element of d-dimensional vector $\mathbf{x}$ is uncorrelated (statistically independent) and the variances σ^2 are the same, the discriminant function deteriorates to the following function:

$$g_k(\mathbf{x})=(\mathbf{x}-\mu_k)(\mathbf{x}-\mu_k)^t \qquad (3.1.55)$$

The discriminant function is the Euclidean distance between the input speech and the templates. This kind of classifier is called a minimum distance classifier.

The Euclidean-like distance measure can be viewed as a special form of Gaussian density, and parameters suited to a Euclidean-like distance measure, such as LPC cepstral coefficients, can be modelled by a Gaussian density function.

3.2. 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 issue in speech recognition research. The most important progress has been achieved using techniques based on the dynamic time warping (DTW) algorithm, hidden Markov models (HMM), and neural networks.

3.2.1. Dynamic time warping (DTW)

Dynamic time warping (DTW) [50,85,93], also known as dynamic programming (DP) matching, was introduced for non-linear 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, non-linearly *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 [37,103].

Suppose we are given two acoustic patterns, **X**, **Y**, which consist of a time sequence of short time representations or 'frames' of the speech signal.

$$\mathbf{X} = \mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_{T_x}$$
$$\mathbf{Y} = \mathbf{y}_1,\mathbf{y}_2,\ldots,\mathbf{y}_{T_y} \qquad (3.2.1)$$

where T_x and T_y are the total number of 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 LPC cepstral coefficients. Figure 3.2.1(a) and (b) show the two acoustic patterns **X** and **Y** in one dimension as a function of time.

In comparing the two patterns **X** and **Y**, three methods may be considered. The first is to compare them directly along with the corresponding time, i.e. frame distances $d(\mathbf{x}_i,\mathbf{y}_i)$ are computed from $i=0$ to T_y and are accumulated. For the interval $T_x<i\le T_y$, the vector $\mathbf{x}_i$ is regarded as zero. Overall distance $D(\mathbf{X},\mathbf{Y})$ results in the hatched area shown in Figure 3.2.1(c). The second method is called linear matching which compresses the pattern **Y** linearly to match **X** and compares them along with the corresponding time as in the first method. The third is called non-linear matching which compresses and stretches the patterns **X** and **Y** non-linearly and compares them along with the corresponding time. The overall distance $D(\mathbf{X},\mathbf{Y})$ of the linear matching and

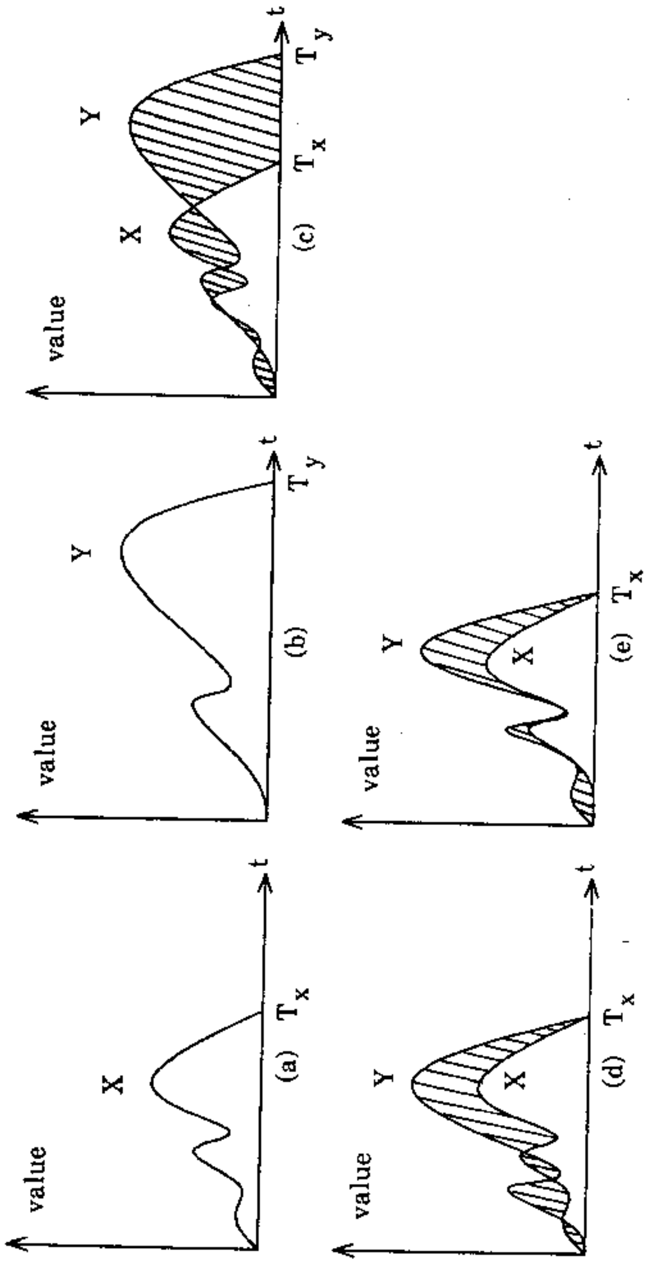

Figure 3.2.1. Acoustic patterns and their matching measure:
(a)(b)acoustic pattern x and y, (c)matching without alignment,
(d)linear matching and (e)non-linear matching.

non-linear matching are shown as the hatched areas in Figure 3.2.1(d) and (e). Comparing Figure 3.2.1(c), (d) and (e), we see that the overall distance is being reduced each time.

In general, even for the same 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. When speech is spoken quickly, stationary sections may shorten and non-stationary sections may remain of almost the same length. In comparing the two acoustic patterns **X** and **Y**, we want to absorb this kind of unessential distance caused by the time difference. Dynamic time warping is a matching method for time sequence patterns to absorb the unessential differences by non-linearly aligning the corresponding times as shown in Figure 3.2.1(e).

Intuitively, the matching between two acoustic patterns **X** and **Y** will be regarded as a temporal alignment in a two-dimensional plane as shown in Figure 3.2.2. The sequence of matched pairs $c(k)=(i(k), j(k))$ of **X** and **Y** form a time registration path or a time warping function F shown as:

$$F=c(1),c(2),...,c(k),...,c(K) \qquad (3.2.2)$$

The analysis vectors (frames) of the two patterns **X** and **Y** 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 directions respectively. The slope of 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 of **Y** 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 distance between two patterns X and Y over the time registration path F is a weighted sum of the individual frame distance $d(c(k))=d(\mathbf{x}_{i(k)}, \mathbf{y}_{j(k)})$ for the pairs of frames over the path, and is expressed as follows:

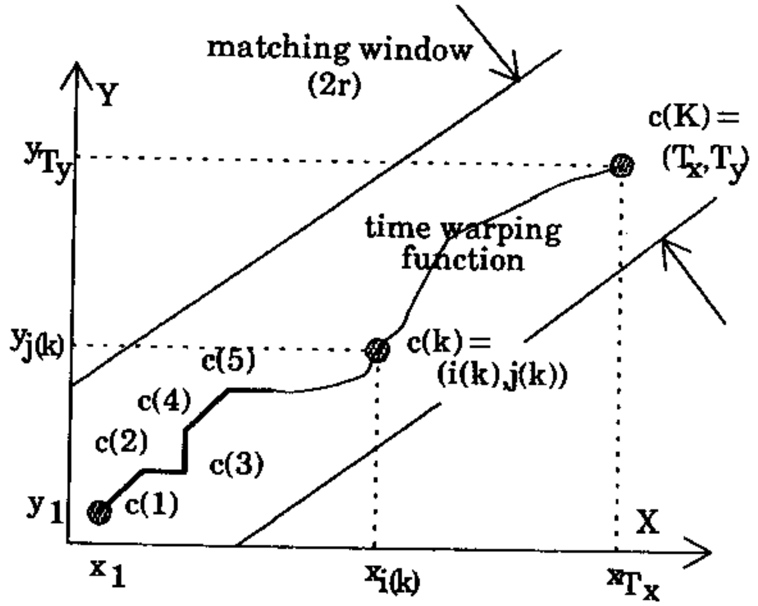

Figure 3.2.2. Non-linear matching in the DTW algorithm

$$D(\mathbf{X},\mathbf{Y},F)=\frac{\sum_{k=1}^{K} d(c(k))w(k)}{\sum_{k=1}^{K} w(k)} \qquad (3.2.3)$$

The weight $w(k)$ given to each frame distance depends 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 F which minimises the overall distance $D(\mathbf{X},\mathbf{Y},F)$ subject to appropriate constraints, such as end point, continuity, and monotonicity constraints [85]. Then, the minimised overall distance is:

$$D(\mathbf{X},\mathbf{Y})=\min_{F}\frac{\sum_{k=1}^{K} d(c(k))w(k)}{\sum_{k=1}^{K} w(k)} \qquad (3.2.4)$$

Now let us consider the constraints in minimising the overall distance.

(1) *End point constraint*

From starting point constraint,

$$i(1)=j(1)=1 \qquad (3.2.5)$$

From ending point constraint,

$$i(K)=T_x, \quad j(K)=T_y \qquad (3.2.6)$$

(2) *Continuity* and *monotonicity constraints*

From increasing monotonicity between consecutive matched pairs $c(k)$ and $c(k-1)$,

$$0\le i(k)-i(k-1), \quad 0\le j(k)-j(k-1) \qquad (3.2.7)$$

and, from the continuity,

$$i(k)-i(k-1)=1 \quad \text{and} \quad j(k)-j(k-1)\le 2$$

hence $c(k-1)$ is expressed as follows:

$$c(k-1) = \begin{cases} (i(k)-1, j(k)) \\ (i(k)-1, j(k)-1) \\ (i(k)-1, j(k)-2) \end{cases} \qquad (3.2.8)$$

The above relation between $c(k)$ and $c(k-1)$ is called the matching path. If we select different constraints on continuity, other types of matching path are available.

(3) *Matching window*

In order to inhibit an unreasonable registration path, the tolerant range is restricted to within a width $2r$ of the matching window as shown in Figure 3.2.2.

The denominator of Eq. (3.2.4) is the normalising factor of overall distance, and depends on the registration path F. To simplify Eq. (3.2.4), we select the denominator to be independent of the registration path F by the following two methods.

(1) *Symmetry*

$$w(k)=(i(k)-i(k-1))+(j(k)-j(k-1)) \qquad (3.2.9)$$

In this case, the denominator of Eq. (3.2.4) is $N=T_x+T_y$

(2) *Asymmetry*

$$w(k)=i(k)-i(k-1) \qquad (3.2.10)$$

In this case, the denominator equals $N=T_x$

Consequently, the minimised overall distance is expressed as:

$$D(\mathbf{X},\mathbf{Y})=\frac{1}{N}\min_{F}\sum_{k=1}^{K}d(c(k))w(k) \qquad (3.2.11)$$

Since Eq. (3.2.11) is minimised by selecting the best registration path $F=c(1),c(2),...,c(k),...,c(K)$, the solution of this problem is regarded as a multistage (K) decision process. Dynamic programming can decompose this multistage decision process into a sequence of K one-stage decision processes, by seeking the recurrent relation of the process. For this decomposition, dynamic programming can effectively reduce the computation time required for the search of the best registration path.

Let $G(c(K))$ denote the minimised overall distance $D(\mathbf{X},\mathbf{Y})$ without the denominator N in Eq. (3.2.11) to represent explicitly the accumulated frame distance from $k=1$ to $k=K$. Then $G(c(K))$ can be expressed as follows:

$$G(c(K))=G(T_x,T_y)=\min_{c(1),...,c(K-1)}\sum_{k=1}^{K}d(c(k))w(k) \qquad (3.2.12)$$

Here, the ending point $c(K)$ is fixed. The above expression is further expanded to:

$$G(c(K))=\min_{c(1),...,c(K-1)}[\sum_{k=1}^{K-1}d(c(k))w(k)+d(c(K))w(K)]$$

$$=\min_{c(K-1)}[\min_{c(1),...,c(K-2)}[\sum_{k=1}^{K-1}d(c(k))w(k)]+d(c(K))w(K)]$$

The first term in the outer [.] can be replaced by $G(c(K-1))$, so Eq. (3.2.12) is expressed in the following recurrent relation.

$$G(c(K))=\min_{c(K-1)}[G(c(K-1))+d(c(K))w(K)]$$

In general, this line of reasoning provides the following recurrence relation:

$$G(c(k))=\min_{c(k-1)}[G(c(k-1))+d(c(k))w(k)] \qquad (3.2.13)$$

The above expression indicates that a sequence of K one-stage decision processes replaces the original K-stage process. It is exactly the mathematical expression of *principle of optimality* on which the dynamic programming is based, i.e.

> *An optimal policy has the property that whatever the initial state and the initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision[81].*

Using Eq. (3.2.8) for the $c(k-1)$ constraint and Eq. (3.2.10) for $w(k)$, Eq. (3.2.13) is expressed as follows:

$$G(i,j)=\min\begin{vmatrix}G(i-1,j)\\G(i-1,j-1)\\G(i-1,j-2)\end{vmatrix}+d(\mathbf{x}_i,\mathbf{y}_j) \qquad (3.2.14)$$

Here, the indicator k to specify the position of matched pairs on the registration path is omitted for simplicity. The above expression can be implemented by proceeding along $\mathbf{X}$ one frame at a time and, for each successive frame $\mathbf{x}_i$, computing a frame distance $d(\mathbf{x}_i,\mathbf{y}_j)$ and an accumulated frame distance $G(i,j)$ for each value of j permitted by the search area constraints of the matching window. The initial value of the accumulated frame distance is:

$$G(1,1)=d(\mathbf{x}_1,\mathbf{y}_1) \tag{3.2.15}$$

The accumulated frame distance is the weighted sum of the frame distances on the optimal partial path from the initial point to (i,j) and is found by optimising over the points that may be reached from the predecessors of (i,j) on such partial paths.

Typical DTW-based speech recognition systems can be found in [16,50,71,72,76,85,93,94]. 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 as a special case of hidden Markov modelling, which is a parametric technique and offers flexibility and improved recognition accuracy [17,55].

3.2.2. Hidden Markov modelling

Statistics and probability theory have much to offer speech recognition. First, classical multivariate statistical distributions, defined over a given pattern space, provide an adequate model for the variability of pattern representations. Second, the question of whether or not a given pattern *belongs* to 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 [29,31]. Applications of basic theories of statistical pattern recognition, such as Bayesian decision [2,5,53,104], Bayesian learning [100], and feature analysis [19], can be widely found in speech recognition.

Statistical methods (which can absorb acoustic variations) can be integrated with the DTW approaches (which can absorb the time variation of acoustic speech patterns) to achieve robust recognition. The first step toward this idea might be replacement of the frame distance $d(\mathbf{x}_i, \mathbf{y}_j)$ with probability $Pr(\mathbf{x}_i| s_j)$ which is the probability that the input frame $\mathbf{x}_i$ is produced from the template state s_j. This is the information theoretic expansion of the distance to a probability, as discussed in Section 3.1.5. The second step might be replacement of the weight $w(k)$ with the transition probability from one state to another possible state in the template. Such a combination of DTW and statistics finally leads to the concept of hidden Markov modelling.

In the above discussion, the template $\mathbf{Y}$ results in a sequence of states which have some probability of emitting the input frame $\mathbf{x}_i$, instead of a sequence of real data frames. To guarantee a robust probabilistic model $\mathbf{Y}$, the number of states must be reduced in comparison with the number of frames included in the template. This is a tradeoff between obtaining the probabilistic model and losing the time information in the template. If the number of states is reduced to one, this reduces to the Bayesian classifier of the individual input frames. Therefore it can be said that hidden Markov modelling locates between DTW and frame-wise Bayesian classifier methods.

Hidden Markov modelling is a technique for the study of observed items arranged in a discrete-time series. The items in the series can be individually 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 [1,8,22,53,62,64,86], formant and pitch tracking [21,61], speech signal processing [35], and language modelling [58,73]. This book concentrates on the HMM technique, in particular, with specific emphasis on its use in acoustic modelling.

The work of Markov [70] and Shannon [96,97] 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 [10,11,13] 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 *et al.* [12] 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 [64,91], continuous mixture HMMs [57,65], autoregressive HMMs [56,82], semi-continuous HMMs [47,48], and trainable finite-state (hidden) grammars [9]. A special case of the results in [12] has been designated by Dempster, Laird and Rubin [27] for maximum likelihood estimation of mixture probability 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 probability density function (continuous HMM). The double stochastic processes enable modelling of not only acoustic phenomena, but also time scale distances. Unlike other non-parametric and ad hoc approaches, the parameters estimated from the Baum-Welch algorithm [12] 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 [17,55]. 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 distance 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.

3.2.3. 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 [66,90]. 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 [67] to discover internal abstractions for speech recognition are becoming very attractive [15,20,39,43,46,68,80,88].

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 [109] 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 which have been proposed are the single-layer perceptron [31], the multi-layer perceptron [46,90], and Kohonen's feature map algorithm [60]. 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 [90], 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 [20,39] or isolated phoneme recognition [83,105,106] because speech signals must be segmented before neural network modelling. A number of these studies have reported encouraging recognition performance for isolated speech recognition [105] and limited success in continuous speech recognition [38,43].

3.2.4. Algorithms for continuous speech

In isolated word recognition, the starting point and ending point of the word pattern is easily detected automatically because the single word is included in the input acoustic pattern. The end point detection is called segmentation. The segmented word pattern can be recognised by the methods of DTW, HMM and neural networks, described in the previous sections. However, in continuous speech recognition, segmentation itself is a difficult problem, because word boundaries are no longer evident owing coarticulation in natural speaking. Therefore, isolated word techniques which require pre-segmentation can not be applied to continuous speech. The best approach to continuous speech is to optimise simultaneously the segmentation and recognition of the words, instead of successive optimisation of recognition after segmentation, as is used in the isolated word recognition.

For the simultaneous optimisation of segmentation and recognition, every possible segmentation is hypothesised for all possible word sequences. Then the most plausible word sequence and its corresponding end points are determined as the final optimisation result. The dynamic programming technique is again applicable to solve the multi-stage decision problem: the word sequence decision and its corresponding end point decision. The word sequence should follow a set of grammatical constraints; however for simplicity we consider the case where no grammatical constraints are used, supposing spoken sentences to be composed of any concatenation of words contained in the prescribed vocabulary (for connected word recognition with grammatical constraint, see Section 3.3.2). The basic idea used in connected word recognition by dynamic programming can also be extended to hidden Markov modelling.

Let us consider segmentation of continuous speech whose duration is T, irrespective of recognition. The number of segments contained in this segmentation is K, and each segment is referred to as $1,2,\ldots,k,\ldots,K$. By denoting the end frame of the segment k as $E(k)$, the interval of the segment k is represented as $(E(k-1)+1, E(k))$. Here, $E(K)=T$, and $E(0)=0$. We call this segmentation Δ_K hereafter. Then, the segmentation Δ_K has obviously two kinds of parameters to be optimised: the number of segments K and the boundary $E(k)$ of the segment k.

Now word recognition can be regarded as a problem of computing the minimised overall distance between the template of the word w_k and the segment k. We will call hereafter the minimised overall distance the word distance. This recognition problem is solved by DTW as described in the previous section, and the word distance is represented as follows:

$$\textit{Word distance} = D((E(k-1)+1, E(k)), w_k) \qquad (3.2.16)$$

The connected word recognition problem is formalised as a minimisation problem of the following word sequence (accumulated) distance, with respect to the segment number K, the segment boundary $E(k)$ and the word sequence $\{w_k\}$:

Word sequence distance (3.2.17)

$$= \sum_{k=1}^{K} D((E(k-1)+1, E(k)), w_k)$$

Then the minimised word sequence distance $A(T)$ is given as:

$$A(T) = \min_{K} \min_{\Delta_K} \min_{w_1,\ldots,w_K} \sum_{k=1}^{K} D((E(k-1)+1, E(k)), w_k) (3.2.18)$$

This minimisation problem is solved by dynamic programming as in the previous section by dealing with the last word separately.

$$A(T) = \min_{K} \min_{\Delta_K} \min_{w_1,\ldots,w_{K-1}} \min_{w_K} [\sum_{k=1}^{K-1} D((E(k-1)+1, E(k)), w_k)$$

$$+ D((E(K-1)+1, E(K)), w_K)]$$

$$= \min_{K} \min_{\Delta_K} [\min_{w_1,\ldots,w_{K-1}} \sum_{k=1}^{K-1} D((E(k-1)+1, E(k)), w_k)$$

$$+ \min_{w_K} D((E(K-1)+1, E(K)), w_K)]$$

By separately dealing with the last segment K,

$$A(T) = \min_{E(K-1)} \min_{K-1} \min_{\Delta_{K-1}} [\min_{w_1,\ldots,w_{K-1}} \sum_{k=1}^{K-1} D((E(k-1)+1, E(k)), w_k)$$

$$+ \min_{w_K} D((E(K-1)+1, E(K)), w_K)]$$

$$= \min_{E(K-1)} [\min_{K-1} \min_{\Delta_{K-1}} \min_{w_1,\ldots,w_{K-1}} \sum_{k=1}^{K-1} D((E(k-1)+1, E(k)), w_k)$$

$$+ \min_{w_K} D((E(K-1)+1, E(K)), w_K)]$$

Here we denote the end frame of the $K-1$ segment as l, namely, $E(K-1)=l$. Then the above expression is rewritten as:

$$A(T) = \min_{l} [A(l) + \min_{w_K} D((l+1, T), w_K)] \qquad (3.2.19)$$

In general, this line of reasoning provides the following recurrence relation at an arbitrary time t:

$$A(t) = \min_{l} [A(l) + \min_{w} D((l+1, t), w)] \qquad (3.2.20)$$

where w_k is replaced by w because the segment k is not explicitly expressed in Eq. (3.2.20).

Mainly three implementation methods have been proposed according to the difference of control sequence or optimisation sequence.

(1) *Two-level DP*[94]

Eq. (3.2.20) includes two types of dynamic programming. One is used in the computation of word distance D and the other is used in the computation of the word sequence distance. Since it includes two dynamic programming operations, it is called a two-level dynamic programming method. In this method, the best matched word w is decided, by using word level dynamic programming, to have an arbitrary segment starting at $l+1$ and ending at t, and its word distance is kept. Then this information is used at the word sequence level dynamic programming. Therefore, the two-level dynamic programming method separately utilises the dynamic programming twice. This requires a large amount of computation time.

(2) *Level building method* and *One-pass DP*[16,72,76]

In order to reduce the computation time of two-level DP, Eq. (3.2.20) is modified as follows:

$$A(t)=\min_{w}[\ \min_{l}[A(l)+\ D((l+1,\ t),\ w)]] \qquad (3.2.21)$$

Our purpose is to avoid explicit searching for the best position l, which requires pre-computation of $\min_{w} D((l+1,\ t),\ w)$. The inner minimisation with respect to l in Eq. (3.2.21) is interpreted as the minimisation of the accumulated frame distance from frame 1 to t of input speech, fixing the last word template w. Let us denote it as $G^w(t,T_w)$, where T_w denotes the end time of the template word w. Then, it is represented as:

$$G^w(t,T_w)=\min_{l}[A(l)+\ D((l+1,\ t),\ w)] \qquad (3.2.22)$$

The above minimisation of the accumulated frame distance from frame 1 to t is achieved by putting $A(l)$ as the initial value to the word distance $D((l+1,\ t),\ w)$ at starting point $l+1$, and computing the word distance for the word w using the same recurrent expression as Eq. (3.2.14). Using Eq. (3.2.22), Eq. (3.2.21) is rewritten as:

$$A(t)=\min_{w}G^w(t,T_w) \qquad (3.2.23)$$

The above expression is free from explicit segmentation by l. The minimum word sequence distance $A(t)$ at time t is obtained by selecting, at time t, the word w to which the accumulated frame distance $G^w(t,T_w)$ is minimised among all words. $A(t)$ is used again as the initial value in computing every word distance starting from $t+1$.

In the level-building method, the accumulated frame distance $G^w(t,T_w)$ is computed for the specified template frame of the specified word w at all input frames. On the other hand, in one-pass DP, the accumulated frame distance $G^w(t,T_w)$ is computed for the specified input frame at all template frames of the specified word w. Since one-pass DP is time synchronous, the process proceeds in time increasing order, and is therefore suitable for real time processing.

3.3. Language Modelling

Acoustic pattern matching is only the first step in the recognition and understanding of natural continuous speech. Lexical knowledge (i.e. vocabulary definition) is required 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 (i.e. knowledge of the structure of extended discourse and knowledge of what people are likely to say in particular contexts) can be of value. In practical speech recognition, it may not be possible to separate the use of these different level of knowledge. Specifically, language modelling here refers to syntax constraints.

3.3.1. Role of language models

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 of words. These probabilities guide the recognition process and are a contributing factor in determination of the final transcription from a set of partial hypotheses [51]. Given acoustic evidence observation **O**, the operations of speech recognition are to find the most likely word string, $\hat{\mathbf{W}}$, satisfying

$$Pr(\hat{\mathbf{W}}|\mathbf{O}) = \max_{\mathbf{W}} Pr(\mathbf{W}|\mathbf{O}) \qquad (3.3.1)$$

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

$$Pr(\mathbf{W}|\mathbf{O}) = \frac{Pr(\mathbf{W})Pr(\mathbf{O}|\mathbf{W})}{Pr(\mathbf{O})} \qquad (3.3.2)$$

where $Pr(\mathbf{W})$ is the *a priori* probability that the word string **W** will be uttered. $Pr(\mathbf{O}|\mathbf{W})$ is the probability that when the speaker says word string **W** the acoustic evidence **O** will be observed (this is the output in the acoustic pattern matching as discussed in the previous sections), and $Pr(\mathbf{O})$ is the average probability that **O** will be observed. Since $Pr(\mathbf{O})$ is not related to **W**, it is irrelevant to recognition. It follows from Eq. (3.3.1) and (3.3.2) that the purpose of the recognition operation is to find the word string $\hat{\mathbf{W}}$ that maximises the product

$$Pr(\hat{\mathbf{W}})Pr(\mathbf{O}|\hat{\mathbf{W}}) = \max_{\mathbf{W}} Pr(\mathbf{W})Pr(\mathbf{O}|\mathbf{W}) \qquad (3.3.3)$$

where $Pr(\mathbf{O}|\mathbf{W})$ relies 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.

Figure 3.3.1 shows the process of continuous speech recognition, with modules of speech signal processing, and acoustic pattern matching as described in the previous

section, and language modelling. In the figure, one side corresponds to speech generation by humans and the other side corresponds to the 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.

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 [45] is widely used to specify the permissible word sequences in natural language processing. In the Chomsky hierarchy, the simplest language type is the regular grammar, which in fact can only generate the same set of sentences as those which have been predefined. A more powerful language type is the context-free grammar. For speech recognition, stochastic context-free languages [9] have also been proposed in the spirit of hidden Markov modelling. In contrast to conventional context-free parsing algorithms, such as the Earley algorithm [33] or the CYK (Cocke-Younger-Kasami) algorithm [45], Ney *et al.* [77] incorporated the DTW algorithm into the parsing algorithm. A more efficient LR parsing algorithm [101] has also been adopted based on HMMs [59].

On the other hand, stochastic grammars, such as trigram or bigram [54], 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 [22,62].

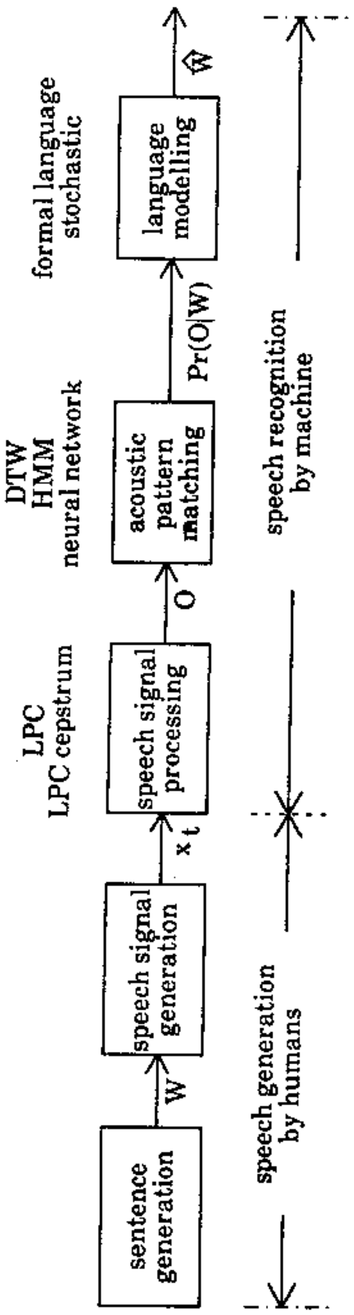

Figure 3.3.1. Process of continuous speech recognition.

3.3.2. The Chomsky language modelling [3,102]

In Chomsky formal language theory, language is modelled to be generated by a grammer $G=(V,T,P,S)$, where V and T are finite sets of *variables* and *terminals*, respectively. P is a finite set of production rules and S is a special variable called the *start symbol*. The language, string of terminal symbols, is produced by applying production rules sequentially to the *start symbol*. The production rule is of the form $\alpha \rightarrow \beta$, where α and β are arbitrary strings of grammar symbols V and T, and the α is not empty.

In formal language, four major languages and their associated grammars are hierarchically structured (*Chomsky hierarchy*). The most general grammar is phrase structure (Type 0) grammar in which there are no constraints on $\alpha \rightarrow \beta$. The next constrained grammar is context sensitive (Type 1) grammar in which the constraint is $|\alpha| \leq |\beta|$, where $|.|$ indicates the length of the string. The further constrained grammar is context free (Type 2) grammar in which the production rule is $A \rightarrow \beta$, where A is a variable. This production rule is shown to be equivalent to Chomsky normal form: $A \rightarrow w$ and $A \rightarrow BC$, where w is a terminal and B, C are variables. The most constrained grammar is regular (Type 3) grammar in which the production rule is expressed as: $A \rightarrow w$ and $A \rightarrow wB$.

It has been shown that there are four kinds of machines which can accept the languages produced by Types 3, 2, 1, and 0: finite automata, push down automata, linear bounded automata and Turing machine respectively. Since the context free grammar can represent the phrase structure of natural language, it has been applied to natural language processing. The regular grammar can only be applied to spoken language in restricted applications. More general application of the context free grammar is a topic covered in [42]. Here, we will describe the application of regular grammar to spoken language.

The regular grammar is equivalent to the finite state automat $M=(V,T,\delta,S,F)$, where V is a finite set of states, T is a finite input alphabet, S in V is the initial state, F is a set of final states, and δ is the transition function mapping (V,T) to V. $B=\delta(A,w)$ is a state reachable from the state A for the given input symbol w. This is the same as the production rule $A \rightarrow wB$ and $A \rightarrow w$ of the regular grammar, whose variables correspond to states and whose terminals correspond to input symbols. Hereafter, we use finite state automata to model a task dependent language or artificial language.

Figure 3.3.2 shows an example of a finite state automata which can accept a language for commanding three robots R1, R2 and R3 to go forward or backward, to turn right or left, and to stop. In the go command, exact numerical values of length in metres is permitted, otherwise, the robot keeps going by the specification "on". In the same way, the turn command is followed by the exact numerical value of rotation in degrees, otherwise, it keeps turning round by specifying "on" until stop command is spoken. There are 9 states and 22 symbols. The initial state is 0 and the final state is 8.

In the case of application of the automata to natural language processing, the input symbol at each state is uniquely given. On the other hand, in the case of its application to the spoken language, an input symbol is unknown at each state as well as its corresponding starting and ending time. Therefore at each state, all the symbols going out of the state are assumed to occur, and every frame is assumed to be the ending frame of the words. This means that all the symbols branching from all the states must be taken into consideration at every frame to determine which symbol occurs.

Example 3.3.1. Connected word recognition using finite state automata [16]

Let us consider application of the automata shown in Figure 3.3.2 to the connected word recognition task described in Section 3.2.4. Each symbol w (word) in the automata has its origin state p and destination state q according to the transition function $\delta(p,w)=q$. This constraint is represented in Eq. (3.2.20) as follows:

$$A_q(t)=\min_{l}\min_{p}[A_p(l)+\min_{w_{pq}} D((l+1,\,t),\,w_{pq})] \qquad (3.3.4)$$

where $A_p(l)$ and $A_q(t)$ are the minimised word sequence distance ending at state p at frame l from the initial state, and ending at state q at frame t respectively. This expression means that the word sequence distance $A_q(t)$ from the initial state to the state q must be computed at every frame, to deal with the ambiguity of word boundary and word name itself. (The recogniser must determine what words locate where.) $\{w_{pq}\}$ is the word set whose origin and destination states are p and q respectively. Minimisation with respect to p is required to determine the best state sequence.

In the two-level DP, Eq. (3.3.4) is computed directly, and in the level building or one-pass DP, Eq. (3.3.4) is modified as in Eq. (3.2.21)

$$A_q(t)=\min_{p}\min_{w_{pq}}[\min_{l}[A_p(l)+D((l+1,\,t),\,w_{pq})]] \qquad (3.3.5)$$

In the same way as for Eq. (3.2.22), $G_q^{w_{pq}}$ is defined as the minimisation of the accumulated frame distance from frames 1 to t of input speech, fixing the last word w_{pq}, i.e.

$$G_q^{w_{pq}}(t,T_{w_{pq}})=\min_{l}[A_p(l)+D((l+1,\,t),\,w_{pq})] \qquad (3.3.6)$$

Using Eq. (3.3.6), Eq. (3.3.5) can be rewritten as:

$$A_q(t)=\min_{p}\min_{w_{pq}}G_q^{w_{pq}}(t,T_{w_{pq}}) \qquad (3.3.7)$$

The above expression is further simplified by integrating two minimisations with respect to p and w_{pq} into one with respect to w_q which is the words ending at state q.

$$A_q(t)=\min_{w_q} G_q^{w_q}(t,T_{w_q}) \tag{3.3.8}$$

The above expression means that minimisation of the word sequence distance ending at state q at frame t from the initial state is obtained by selecting, at time t, the word w_q to which the accumulated frame distance $G_q^{w_q}(t,T_{w_q})$ from the initial state is minimised among all words, which end at the state q. Finally, overall minimisation is obtained in the final states at time T.

- In the above application of the automata to connected word recognition, the finite state automata plays a role of reducing the number of word combinations at each frame. If we do not use the automata, the possible number of words following the previous word is 23 (vocabulary size is 22). The automata can reduce this number to 2.9 words on average. This number is called the static branching factor, and is computed by counting the total number of branches (23) which go out from states and dividing it by the number of states (8) except the final states. The problem of connected word recognition by finite state automata is that the number of states increases enormously when it is applied to more natural language. This means difficulty of not only computation, but also generation of such finite state automata from a large corpus, either manually or automatically.

3.3.3. Stochastic language modelling [51]

In connected word recognition by finite state automata, since the solution is globally optimised, the recognised word sequence can completely represent the input spoken sentence. This rigid parsing can be the cause of an enormous increase in the number of states. To solve this problem, stochastic language modelling is proposed, which can give the probability of the word sequence $Pr(\mathbf{W})$, instead of parsing the spoken sentence rigidly using the grammar of formal language model.

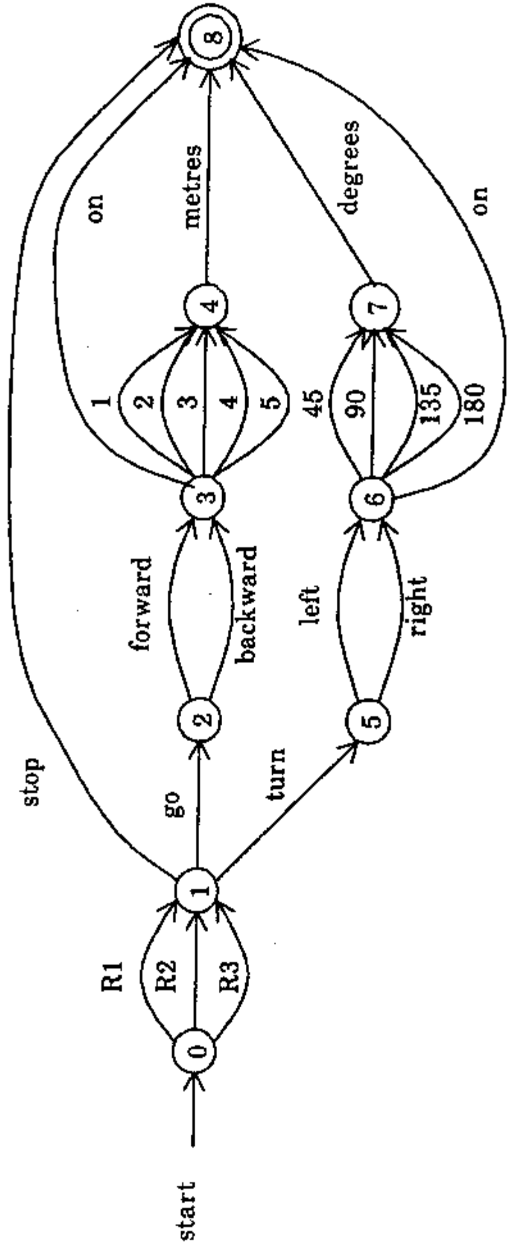

Figure 3.3.2. Example of finite state automata.

As described in Section 3.3.1, the probability $Pr(\mathbf{W})$ is learned to be maximised from a corpus in stochastic language modelling. In general, $Pr(\mathbf{W})$ can be decomposed as

$$Pr(\mathbf{W}) = Pr(w_1,w_2,...,w_n) \tag{3.3.9}$$

$$= Pr(w_n|w_1,...,w_{n-1})Pr(w_{n-1}|w_1,...,w_{n-2}) \cdots$$

$$Pr(w_i|w_1, \ldots ,w_{i-1}) \cdots Pr(w_2|w_1)Pr(w_1)$$

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

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 $w_1, \ldots ,w_{i-1}$ would be unique or would have occurred only a few times.

A practical solution to the above problems is to assume that $Pr(w_i|w_1, \ldots ,w_{i-1})$ only depends on $w_{i-M+1}, \ldots ,w_{i-1}$. This leads to an M-gram language model, such as unigram $Pr(w_i)$, bigram $Pr(w_i|w_{i-1})$, or trigram $Pr(w_i|w_{i-2},w_{i-1})$ language models [53]. 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. The trigram can be estimated by observing the frequencies of the word pair $C(w_{i-2},w_{i-1})$ and triplet $C(w_{i-2},w_{i-1},w_i)$ as follows:

$$Pr(w_i|w_{i-2},w_{i-1}) \tag{3.3.10}$$

$$= C(w_{i-2},w_{i-1},w_i)/C(w_{i-2},w_{i-1})$$

However, if the training corpus is not large enough, many actually existing word successions will not be well enough observed, leading to many extremely small probabilities. To treat the insufficient data problem, smoothing methods, such as the Turing–Good estimate [74], deleted interpolation of trigram, bigram and unigram models [28,52], as well as neural-network-based NETgram [75], can be well applied.

This stochastic language modelling can be applied to connected word recognition as described in Section 3.2.4. Since probability is used in the language model, the acoustic modelling has to be able to give the acoustic probability $Pr(\mathbf{O}|\mathbf{W})$ instead of distance, and minimisation is replaced by maximisation of the probability. In the case of trigram, the word sequence probability, which is equivalent to word sequence distance (Eq. (3.2.20) in Section 3.2.4), must be computed at every frame for all the paths whose last two words w_{i-2} and w_{i-1} are different, instead of seeking the best path. To these paths, the trigram $Pr(w_i|w_{i-2},w_{i-1})$ is applied, after computing the probability of the following word w_i. The updated word sequence probabilities are kept and used in the succeeding steps. An efficient algorithm for this purpose, called the stack decoding algorithm, is proposed[53].

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 [3,23,24,32,44,98,100,108].

3.3.4. Complexity measures of language

Language can be thought of as an information source whose outputs are words w_i. The averaged amount of information per word, i.e. *entropy* $H(L)$ as described in Section 2.6.1, for the given language is measured as:

$$H(L) = -\sum_{\mathbf{w}_1^k} \frac{1}{k} Pr(\mathbf{w}_1^k) \log Pr(\mathbf{w}_1^k) \quad (3.3.11)$$

where $\mathbf{w}_1^k = w_1, w_2, ..., w_k$ is a word sequence of length k. The entropy $H(L)$ indicates the ambiguity or required information (bits) to specify a certain word produced by the language, as described in Section 2.6.1. This means that there are on average $2^{H(L)}$ possible words which can follow a previous word. This number is called the *perplexity* [53,99] and is formally defined as:

$$PP = 2^{H(L)} \quad (3.3.12)$$

A language shows higher perplexity when the number of words branching from a previous word becomes larger on average. In this sense, the perplexity is used to measure the complexity of the language. The perplexity is also used to measure the complexity of the task itself, because a task is described by its suitable language in terms of vocabulary, a number of states or grammar rules. Let us consider perplexity in two cases: formal language modelling (finite state automata), and stochastic modelling.

(1) *Formal language modelling*

In the finite state automata, the entropy at a state j is computed as follows:

$$H(w|j) = -\sum_{w} Pr(w|j) \log Pr(w|j) \quad (3.3.13)$$

where $Pr(w|j)$ is the word occurrence probability at the state j. The expectation of the above entropy over all the states is:

$$H(L) = \sum_{j} \pi(j) H(w|j) \quad (3.3.14)$$

where $\pi(j)$ is the occurrence probability of state j. In the example of Figure 3.3.2, suppose that the state occurrence probabilities $\pi(j)$ are equal and the word occurrence probabilities $Pr(w|j)$ at the state j are equal, then the entropy $H(L)$ is:

$$H(L) = \frac{1}{8} \log(3\cdot3\cdot2\cdot6\cdot1\cdot2\cdot5\cdot1) \quad (3.3.15)$$

$$= \frac{1}{8} \log 1080 \quad (bits)$$

Then the perplexity PP is:

$$PP = 2^{H(L)} = 1080^{1/8} \approx 2.39 \quad (3.3.16)$$

The above expression indicates that about 2.4 words follow the previous word on average under the assumption that word occurrence and state occurrence probabilities are equal. In reality, since these probabilities are not equal, it is expected that the perplexity may be reduced more.

(2) *Stochastic modelling*

In the stochastic modelling, entropy for a given corpus can be approximately estimated by [54]

$$H(L) = -\frac{1}{n} \log Pr(w_1, w_2, ..., w_n) \quad (3.3.17)$$

where n is the size of the corpus [53]. To estimate $H(L)$, it is necessary to know the actual probabilities $Pr(w_1, w_2, ..., w_n)$ in the language. These are in practice ultimately incalculable, and estimates $\hat{P}r(w_1, w_2, ..., w_n)$ are used instead. They can be computed by Eq. (3.3.9) according to the employed stochastic models (bigram or trigram). The perplexity PP is:

$$PP = 2^{H(L)}$$

$$= \hat{P}r(w_1, w_2, ..., w_n)^{-1/n} \quad (3.3.18)$$

Approximately, perplexity 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 5000 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 [53]. In tasks of 1000 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 [62].

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 [36] can be used, though it is more expensive to compute than perplexity.

3.4. 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 speech signal processing and language modelling. Selection of signal processing methods largely depends on the subsequent distance 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 in HMM. 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 [7,14,30,63]. Although the discussion here was organised through speech 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 designs.

References

1. A. Averbuch *et al.*, "Experiments with the Tangora 20,000 word speech recognizer," *Proc. ICASSP-87*, pp. 701-704, Dallas, USA, 1987.

2. G. Bruno *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.

3. A.V. Aho and J.D. Ullman, *The Theory of Parsing, Translation and Computing*, Prentice Hall, 1972.

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, "Effectiveness of linear prediction characteristics of the speech wave for automatic speaker identification and verification," *J. Acoust. Soc. Ame.*, vol. 55, pp. 1304-1312, 1974.

7. 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*, New York, USA, 1988.

8. J. Baker, "The DRAGON system – An overview," *IEEE Trans. Acoustic, Speech, and Signal Processing*, vol. ASSP-23, pp. 24-29, 1975.

9. J. Baker, "Trainable grammars for speech recognition," *Proc. Spring Conference Acoustic Soc. America*, pp. 547-550, 1979.

10. 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.

11. 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.

12. L.E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains," *Ann. Math. Stat.*, vol. 41, pp. 164-171 , 1970.

13. 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.

14. H. Bourlard and C. Wellekens, "Links between Markov models and multilayer perceptrons," Philips Manuscript no. M263, 1988.

15. H. Bourlard and C. Wellekens, "Speech dynamics and recurrent neural networks," *Proc. ICASSP-89*, pp. 33-36, Glasgow, Scotland, 1989.

16. J.S. Bridle, M.D. Brown, and R.M. Chamberlain, "An algorithm for connected word recognition," *Proc. ICASSP-82*, pp. 899-902, Paris, France, 1982.

17. 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.

18. 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*, pp. 307-314, San Diego, USA, 1984.

19. P.F. Brown, "Acoustic-phonetic modeling problem in automatic speech recognition," Ph.D. thesis, Department of Computer Science, Carnegie-Mellon University, 1987.

20. D.J. Burr, "Speech recognition experiments with perceptrons," *AIP Conference Proceeding, Neural Information Processing System*, Denver, 1987.

21. X. Chen, C. Cai, P. Guo, and S. Ying, "A hidden Markov model applied to Chinese four-tone recognition," *Proc. ICASSP-87*, pp. 797-800, Dallas, USA, 1987.

22. 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*, pp. 89-92, Dallas, USA, 1987.

23. Y.L. Chow and S. Roukos, "Speech understanding using a unification grammar," *Proc. ICASSP-89*, pp. 727-730, Glasgow, Scotland, 1989.

24. 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.

25. 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.

26. 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.

27. 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.

28. 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.

29. P.A. Devijver and J. Kittler, *Pattern Recognition: A Statistical Approach*, Prentice Hall International, 1982.

30. G.R. Doddington, "Phonetically sensitive discriminants for improved speech recognition," *Proc. ICASSP-89*, pp. 556-559, Glasgow, Scotland, 1989.

31. R.O. Duda and R.E. Hart, *Pattern Classification and Scene Analysis*, John Wiley, 1973.

32. P. Dumouchel, V. Gupta, M. Lennig, and P. Mermelstein, "Three probabilistic language models for a large-vocabulary speech recognizer," *Proc. ICASSP-88*, pp. 513-516, New York, USA, 1988.

33. J. Earley, "An efficient context-free parsing algorithm," *Communications ACM*, vol. 13, pp. 94-102, 1970.

34. F. Fallside and W. Woods, *Computer Speech Processing*, Prentice Hall International, 1985.

35. M. Feder, A.V. Oppenheim, and E. Weinstein, "Methods for noise cancellation based on the EM algorithm," *Proc. ICASSP-87*, pp. 201-204, Dallas, USA, 1987.

36. M. Ferretti, G. Maltese, and S. Scarci, "Language model and acoustic model information in probabilistic speech recognition," *Proc. ICASSP-89*, pp. 707-710, Glasgow, Scotland, 1989.

37. G.D. Forney, "The viterbi algorithm," *Proc. IEEE*, vol. 61, pp. 268-278, 1973.

38. M. Franzini, M. Witbrock, and K. Lee, "A connectionist approach to continuous speech recognition," *Proc. ICASSP-89*, pp. 425-428, Glasgow, Scotland, 1989.

39. B. Gold, R.P. Lippmann, and M.L. Malpass, "Some neural net recognition results on isolated words," *IEEE International Conference on Neural Networks*, 1987.

40. 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.

41. 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.

42. H.Ney, "Dynamic programming speech recognition using a context-free grammar," *Proc. ICASSP-87*, pp. 69-72, 1987.

43. T. Harrison and F. Fallside, "A connectionist model for phoneme recognition in continuous speech," *Proc. ICASSP-89*, pp. 417-420, Glasgow, Scotland.

44. C. Hemphill and J. Picone, "Speech recognition in a unification grammar framework," *Proc. ICASSP-89*, pp. 723-726, Glasgow, Scotland, 1989.

45. J. Hopcraft and J. Ullman, *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley, 1979.

46. W.Y. Huang and R.P. Lippmann, "Comparison between neural net and conventional classifiers," *IEEE International Conference on Neural Networks*, 1987.

47. 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.

48. 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.

49. 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.

50. F. Itakura, "Minimum prediction residual principle applied to speech recognition," *IEEE Trans. Acoustic, Speech, and Signal Processing*, vol. ASSP-23, pp. 67-72, 1975.

51. F. Jelinek, "Continuous speech recognition by statistical methods," *Proc. IEEE*, vol. 64, pp. 532-556, 1976.

52. F. Jelinek and R.L. Mercer, "Interpolated estimation of Markov source parameters from sparse data," *Proc. the Workshop on Pattern Recognition in Practice*, Amsterdam, The Netherlands, North-Holland, 1980.

53. F. Jelinek, "The development of an experimental discrete dictation recognizer," *Proc. IEEE*, vol. 73, pp. 1616-1624 , 1985.

54. F. Jelinek, "Self-organized language modeling for speech recognition," *IBM Europe Institute, Advances in Speech Processing, Austria*, 1986.

55. 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.

56. 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.

57. B.H. Juang, "Maximum-likelihood estimation for mixture multivariate stochastic observations of Markov chain," *AT&T Technical Journal*, vol. 64, pp. 1235-1249, 1985.

58. 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.

59. K. Kita, T. Kawabata, and H. Saito, "HMM continuous speech recognition using predictive LR parsing," *Proc. ICASSP-89*, pp. 703-706, Glasgow, Scotland, 1989.

60. T. Kohonen, *Self-Organization and Associative Memory*, Springer-Verlag, 1984.

61. G.E. Kopec, "Formant tracking using hidden Markov models and vector quantization," *IEEE Tran. ASSP*, vol. ASSP-34, pp. 709-729, 1986.

62. K.F. Lee, "Large-vocabulary speaker-independent continuous speech recognition: The SPHINX system," Ph.D. thesis, Department of Computer Science, Carnegie-Mellon University, 1988; also Automatic Speech Recognition: The Development of the SPHINX System, Kluwer Academic Publishers, 1989.

63. K.F. Lee, "Hidden Markov models: past, present, and future," *Eurospeech 89*, Paris, France, 1989.

64. S.E. Levinson, "Continuously variable duration hidden Markov models for automatic speech recognition," *Computer Speech and Language*, vol. 1, pp. 29-45, 1986.

65. L.R. Liporace, "Maximum likelihood estimation for multivariate observations of Markov sources," *IEEE Trans. Information theory*, vol. IT-28, pp. 729-734, 1982.

66. R.P Lippmann, "Neural nets for computing," *Proc. ICASSP-88*, pp. 1-6 , New York, USA, 1988.

67. R.P. Lippmann, "An introduction to computing with neural nets," *IEEE ASSP Magazine*, pp. 4-22, 1987.

68. R.P. Lippmann and B. Gold, "Neural-net classifiers useful for speech recognition," *IEEE International Conference on Neural Networks*, 1987.

69. J.D. Markel and A.H. Gray, *Linear Prediction of Speech*, Spring-Verlag, 1976.

70. 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.

71. F. McInnes, "Adaptation of reference patterns in word-based speech recognition," Ph.D. thesis, Faculty of Science, University of Edinburgh, 1988.

72. 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.

73. 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.

74. A. Nadas, "On Turing's formula for word probabilities," *IEEE Trans. Acoustic, Speech, and Signal Processing*, vol. ASSP-33, pp. 1432-1436, 1985.

75. M. Nakamura and K. Shikano, "A study of English word category prediction based on neural networks," *Proc. ICASSP-89*, pp. 731-734, Glasgow, Scotland, 1989.

76. 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.

77. 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*, pp. 833-836, Dallas, USA, 1987.

78. N. Nocerino, F.K. Soong, L.R. Rabiner, and D.H. Klatt, "Comparative study of several distortion measures for speech recognition," *Proc. ICASSP-85*, pp. 25-28, Tampa, USA, 1985.

79. A.V. Oppenheim and R.W. Schafer, "Homomorphic analysis of speech," *IEEE Trans. Audio, Electroacoust.*, vol. AU-16, pp. 221-226, 1968.

80. S.M. Peeling, R.K. Moore, and M.J. Tomlinson, "The multi-layer perceptron as a tool for speech pattern processing research," *Proceedings of the Institute of Acoustics Autumn Conference*, 1986.

81. D.A. Pierre, *Optimization Theory with Applications*, Dover Edition, 1986.

82. A.B. Poritz, "Linear predictive hidden Markov models and the speech signal," *Proc. ICASSP-82*, pp. 1291-1294, Paris, France, 1982.

83. R. W. Prager, T.D. Harrison, and F. Fallside, "Boltzman machines for speech recognition," *Computer Speech & Language*, vol. 1, pp. 3-27, 1986.

84. L.R. Rabiner and R.W. Schafer, *Digital Processing of Speech Signals*, Prentice Hall, 1978.

85. L.R. Rabiner and S. Levinson, "Isolated and connected word recognition-theory and selected applications," *IEEE Trans. Communication*, vol. COM-29, 1981.

86. 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.

87. L.R. Rabiner, J.G. Wilpon, and F.K. Soong, "High performance connected digit recognition using hidden Markov models," *Proc. ICASSP-88*, New York, USA, 1988.

88. S. Renals and R. Rohwer, "Learning phoneme recognition using neural networks," *Proc. ICASSP-89*, pp. 413-416, Glasgow, Scotland, 1989.

89. R.A. Roberts and C.T. Mullis, *Digital Signal Processing*, Addison Wesley, 1987.

90. D.E. Rumelhart and J.L. McClelland, *Parallel Distributed Processing; Explorations in the Microstructure of Cognition*, vols. 1 and 2, MIT Press , 1986.

91. M.J. Russell and R.K. Moore, "Explicit modelling of state occupancy in hidden Markov models for automatic speech recognition," *Proc. ICASSP-85*, pp. 5-8, Tampa, USA, 1985.

92. F. Sadaoki, *Digital Speech Processing, Synthesis, and Recognition*, Marcel Dekker , 1989.

93. 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.

94. 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.

95. 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.

96. C.E. Shannon, "A mathematical theory of communications," *Bell System Technical J.*, vol. 27, pp. 379-423, 623-656, 1948.

97. C.E. Shannon, "Prediction and entropy of printed English," *Bell System Technical J.*, vol. 30, pp. 50-64, 1951.

98. D. Shipman and V.W. Zue, "Properties of large lexicons; implication for advanced isolated word recognition system," *Proc. ICASSP-82*, pp. 546-549, Paris, France, 1982.

99. M.M. Sondhi and S.E. Levinson, "Computing relative redundancy to measure grammatical constraint in speech recognition tasks," *Proc. ICASSP-78*, pp. 409-412, USA, 1978.

100. 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.

101. M. Tomita, *Efficient Parsing for Natural Language - A Fast Algorithm for Practical Systems*, Kluwer Academic, 1986.

102. J.D. Ullman and J.E. Hopcroft, *Introduction to Automata Theory, Language and Computation*, Addison Wesley, 1979.

103. A.J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. formation Theory*, vol. IT-13, pp. 260-269, 1967.

104. A. Waibel, "Recognition of lexical stress in a continuous speech understanding system – a pattern recognition approach," *Proc. ICASSP-86*, pp. 2287-2290, Tokyo, Japan, 1986.

105. A. Waibel, K. Lang, and G. Hinton, "Speech recognition using time-delay neural networks," *IEEE Workshop on Speech Recognition, Arden House*, 1988.

106. R.L. Watrous, "Connectionist speech recognition using the temporal flow model," *IEEE Workshop on Speech Recognition, Arden House*, 1988.

107. 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.

108. W.A. Woods, "Language processing for speech understanding ," in *Chapter 12 in Computer Speech Processing*, ed. F. Fallside and W.A. Woods, pp. 305-334, Prentice Hall International, 1985.

109. J.Z. Young, *Programmes of the Brain*, Oxford University Press, 1975.

CHAPTER FOUR

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 the 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. Conventional pattern recognition techniques have been well used to solve the quantisation or data compression problem with successful application in speech coding, image coding, and speech recognition [12,17].

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 hidden Markov modelling. In particular, we will 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.

4.1. Conventional Vector Quantisation

Vector quantisation (VQ) reduces the data redundancy to be transmitted. This inevitably causes distortion between original data and transmitted data. A key point of VQ is to minimise the distortion. In this section, the distortion caused by VQ is considered; then two typical VQ techniques are shown which can minimise the distortion.

4.1.1. Vector quantisation and distortion

Assume that $\mathbf{x} = (x_1,x_2,\ldots,x_d)^t \in R^d$ 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 $\mathbf{z}$. It is then said that $\mathbf{x}$ is quantised to $\mathbf{z}$.

$$\mathbf{z} = q(\mathbf{x}) \tag{4.1.1}$$

In Eq. (4.1.1) $q()$ is the quantisation operator. Typically, $\mathbf{z}$ takes one of a finite set of values $\mathbf{Z} = \{\mathbf{z}_i,\ 1 \le i \le L\}$, where $\mathbf{z}_i = (z_1, z_2,\ldots,z_d)$. The set $\mathbf{Z}$ is referred to as the codebook, L is the size of the codebook, and $\{\mathbf{z}_i\}$ is 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 $\mathbf{z}_i$. The quantiser then assigns the codeword $\mathbf{z}_i$ if $\mathbf{x}$ lies in C_i

$$q(\mathbf{x}) = \mathbf{z}_i, \text{ if } \mathbf{x} \in C_i \tag{4.1.2}$$

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 4.1.1. 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 $\mathbf{z}_i$. The shapes of the various cells can be different, and the positions of the codewords corresponding to the cells are determined by minimising the average distortion associated with the corresponding cells. The positions of the codewords within each cell are shown by dots in Figure 4.1.1.

When $\mathbf{x}$ is quantised as $\mathbf{z}$, a quantisation error results and a distortion measure $d(\mathbf{x},\mathbf{z})$ can be defined between $\mathbf{x}$ and $\mathbf{z}$ to measure the quantisation quality. The distortion measure between $\mathbf{x}$ and $\mathbf{z}$ 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 distance measures discussed in Chapter 3 can be used here as distortion measures for vector quantisation. A number of perceptually based distortion measures, and others that correlate well with subjective judgements, have also been used in speech coding [2,18]. 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 $\mathbf{z}$.

$$d(\mathbf{x},\mathbf{z}) = (\mathbf{x}-\mathbf{z})^t \Sigma^{-1} (\mathbf{x}-\mathbf{z}) \tag{4.1.3}$$

This distortion measure, known as the *Mahalanobis distance*, is actually a simplified Gaussian density

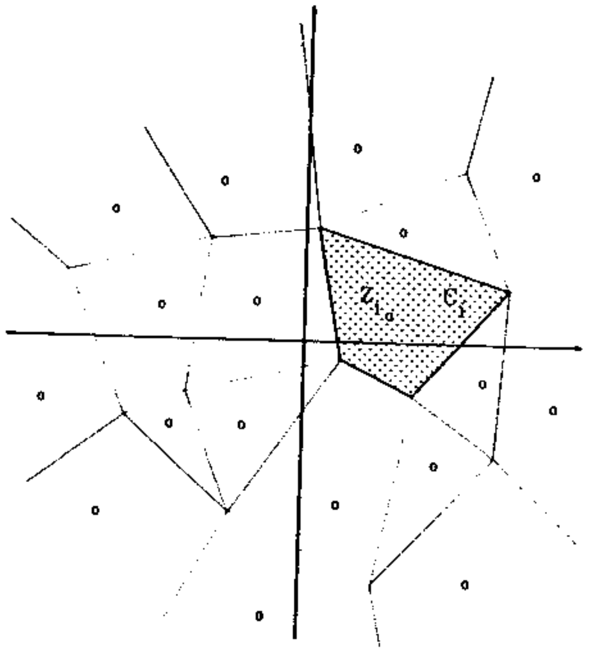

Figure 4.1.1. Partitioning of two-dimensional space into 18 cells.

representation as discussed previously in Chapter 3.

To design an L-level codebook, it is necessary 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

$$D = E[d(\mathbf{x},\mathbf{z})] \tag{4.1.4}$$

$$= \sum_{i=1}^{L} Pr(\mathbf{z}_i)E[d(\mathbf{x},\mathbf{z}_i)|x \in C_i]$$

$$= \sum_{i=1}^{L} \int_{x \in C_i} d(\mathbf{x},\mathbf{z}_i)f(\mathbf{x},\mathbf{z}_i)d\mathbf{x}$$

$$= \sum_{i=1}^{L} Pr(\mathbf{z}_i)\int_{x \in C_i} d(\mathbf{x},\mathbf{z}_i)f(\mathbf{x}|\mathbf{z}_i)d\mathbf{x}$$

$$= \sum_{i=1}^{L} D_i$$

where $E[.]$ denotes the expectation; the integral is taken over all components of the vector $\mathbf{x}$; $Pr(\mathbf{z}_i)$ denotes the discrete probability of the codeword $\mathbf{z}_i$; $f(\mathbf{x}|\mathbf{z}_i)$ denotes the multidimensional probability density function of $\mathbf{x}$ given $\mathbf{z}_i$; and D_i is the average distortion in cell C_i.

No analytic solution exists to guarantee global minimisation of the average distortion measure for a given set of speech data. However, an iterative algorithm, which can guarantee a local minimum, exists and works well in practice. We will discuss several such algorithms that are useful in codebook design.

4.1.2. The k-means algorithm

If the overall average distortion is used as a criterion in codebook design, we say a quantiser is optimal if the overall average distortion 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 nearest neighbour selection rule

$$q(\mathbf{x}) = \mathbf{z}_i, \text{ if and } \mathit{only} \text{ if } d(\mathbf{x},\mathbf{z}_i) \le d(\mathbf{x},\mathbf{z}_j), \quad j \ne i,\ 1 \le j \le L \tag{4.1.5}$$

Note that

$$E[d(\mathbf{x},\mathbf{z}_i)|\mathbf{x} \in C_i] \qquad (4.1.6)$$

can be minimised when $\mathbf{z}_i$ is selected such that $d(\mathbf{x},\mathbf{z}_i)$ is minimised for $\mathbf{x}$. This means that the quantiser must choose the codeword that results in the minimum distortion with respect to $\mathbf{x}$, i.e. $\mathbf{x}$ is selected for the corresponding cell C_i.

The second condition for optimality is that each codeword $\mathbf{z}_i$ is chosen to minimise the average distortion in cell C_i. That is, $\mathbf{z}_i$ is that vector $\mathbf{z}$ which minimises

$$D_i = Pr(\mathbf{z}_i)\, E[d(\mathbf{x},\mathbf{z}_i)|\mathbf{x} \in C_i] \qquad (4.1.7)$$

Since the overall average distortion D is a linear combination of average distortions in cell C_i, they can be independently computed after classification of $\mathbf{x}$. The vector $\mathbf{z}_i$ is called the centroid of the cell C_i, and is written

$$\mathbf{z}_i = \text{cent}(C_i) \qquad (4.1.8)$$

Computing the centroid for a particular region (cell) will depend on the definition of the distortion measure [10]. In practice, given a set of training vectors $\{\mathbf{x}_k,\ 1 \le k \le T\}$, a subset of K_i vectors will be located in cell C_i. In this case, $f(\mathbf{x}|\mathbf{z}_i)$ can be assumed to be $1/K_i$, and $Pr(\mathbf{z}_i)$ becomes K_i/T. The average distortion D_i in cell C_i can then be given by

$$D_i = \frac{1}{T} \sum_{\mathbf{x} \in C_i} d(\mathbf{x},\mathbf{z}_i) \qquad (4.1.9)$$

Given the average distortion of cluster C_i as in Eq. (4.1.9), and $d(\mathbf{x},\mathbf{z})$ as in Eq. (4.1.3), the minimisation of D with respect to $\mathbf{z}_i$ is given by setting the derivative of D_i to zero.

$$\begin{aligned} \nabla_{\mathbf{z}_i} D_i &= \nabla_{\mathbf{z}_i} \frac{1}{T} \sum_{\mathbf{x} \in C_i} (\mathbf{x}-\mathbf{z}_i)^t \Sigma^{-1} (\mathbf{x}-\mathbf{z}_i) \\ &= \frac{1}{T} \sum_{\mathbf{x} \in C_i} \nabla_{\mathbf{z}_i} (\mathbf{x}-\mathbf{z}_i)^t \Sigma^{-1} (\mathbf{x}-\mathbf{z}_i) \qquad (4.1.10) \\ &= \frac{-2}{T} \sum_{\mathbf{x} \in C_i} \Sigma^{-1} (\mathbf{x}-\mathbf{z}_i) = 0 \end{aligned}$$

Finally centroid $\mathbf{z}_i$ is obtained from

$$\mathbf{z}_i = \frac{1}{K_i} \sum_{\mathbf{x} \in C_i} \mathbf{x} \qquad (4.1.11)$$

where $\mathbf{z}_i$ 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 [17].

To minimise iteratively the average distortion measure, the most widely used method is the k-means algorithm [1,8,9]. In the k-means algorithm, 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:

Example 4.1.1. The k-means algorithm

Step 1: Initialisation. Choose some adequate method [8] to derive an initial VQ codebook ($\mathbf{z}_i$, $1 \le i \le L$).

Step 2: Classification. Classify each element of training vectors $\{\mathbf{x}_k\}$ into one of the clusters C_i by choosing the nearest codeword $\mathbf{z}_i$ ($\mathbf{x} \in C_i$, iff $d(\mathbf{x},\mathbf{z}_i) \le d(\mathbf{x},\mathbf{z}_j)$ for all $j \ne i$). This classification is called minimum distance classifier.

Step 3: Codebook updating. Update the codeword of every cluster by computing the centroid of the training vectors in each cluster ($\mathbf{z}_i = \text{cent}(C_i)$, $1 \le i \le L$).

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 $\mathbf{z}_i$ (or mean) for each cluster C_i has been found, then the minimisation process is found simply by partitioning all the training vectors into their corresponding closest cluster according to the distortion measure. 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 average distortion D_i. By iterating over these two steps, a new value of overall distortion D which is smaller than that of the previous step can be obtained. However, the k-means algorithm can only converge to a local optimum [1,16]. Furthermore, any such solution is, in general, not unique [11]. 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 [13].

Example 4.1.1. The LBG algorithm

An extended k-means algorithm, the LBG algorithm proposed by Linde, Buzo, and Gray [16], is also commonly used. The LBG algorithm 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.

Step 1: Initialisation. Set L (number of partitions or clusters) $=1$. Find the 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 minimum distance classifier.

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, go to 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.

4.2. VQ Codebook with 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 feature space into separate regions according to some distortion measure regardless of the probability distributions of the original data. This introduces errors in the partition operations which may destroy the original signal structure.

As an alternative, the VQ codebook can be modelled as a family of Gaussian probability density functions (pdfs) such that each cell will be represented as a probability density function as shown in Figure 4.2.1, where dotted lines show conventional VQ partitions. These probability density functions can be overlapped, rather than partitioned, to represent the acoustic feature space. The centroid obtained via such a representation may be quite different from that obtained 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 Gaussian pdfs within the VQ operations can be closely combined with the HMM methodology leading

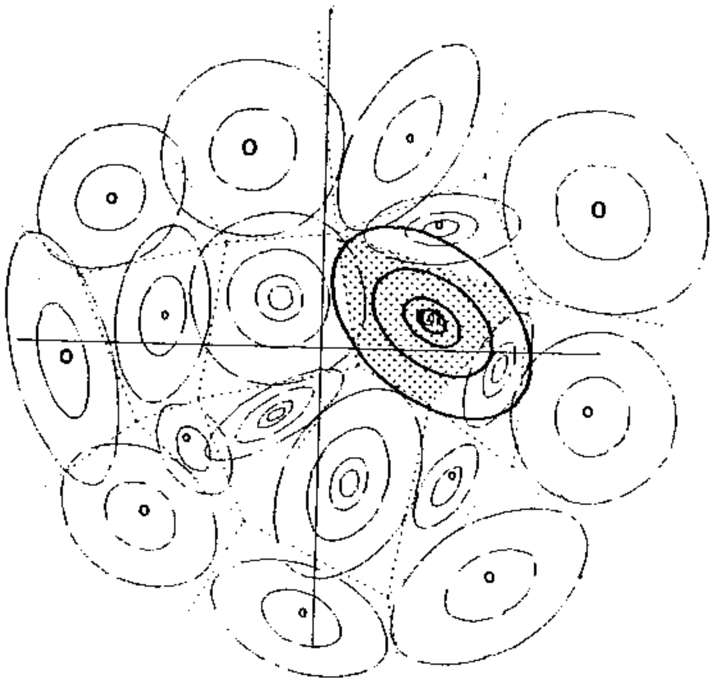

Figure 4.2.1. Partitioning of two-dimensional space with densities

to the unified modelling framework as will be discussed in Chapter 7.

Problems of estimating the parameters which determine a mixture Gaussian pdf have been the subject of a large, diverse body of literature [19]. The most distinctive solution to the problem is the EM algorithm [7]. This technique has in fact been defined in an earlier publication by Baum *et al.* [3] and has been widely used in HMM-based speech recognition. Re-estimation of these parameters can be viewed as a process of unsupervised learning of the mixture Gaussian pdf, like those described in Chapter 2, which in fact include the conventional VQ technique of the k-means algorithm as a special case.

4.2.1. Estimation of the mixture pdf

In this section, we will apply the EM algorithm to the estimation problem of the mixture Gaussian pdf. As discussed in Chapter 2, the EM algorithm is a maximisation algorithm of the log-likelihood of incomplete data with respect to parameters in the probability model, by iteratively maximising expectation of the log-likelihood of complete data via a Q-function. Here incomplete data indicate the observed data (training data), and complete data are composed of observed data and unobserved data which specify hidden components from which the observed data come.

In the mixture pdf, we can assume that observable data $\mathbf{x}_k$ occurs from one of component densities $C_{y_k}=f_{y_k}(\mathbf{x}_k|\varphi_{y_k})$.[1] y_k is unobservable and specifies the pdf with parameter φ_{y_k}. The complete data is $(\mathbf{x}_k,y_k)$ and its probability density function is given as joint probability

$$f(\mathbf{x}_k,y_k|\Phi)=Pr_{y_k}f_{y_k}(\mathbf{x}_k|\varphi_{y_k}) \tag{4.2.1}$$

where Pr_{y_k} is the probability of the unobserved data y_k used to specify the component density C_{y_k} from which the observed data $\mathbf{x}_k$ comes. Φ is a vector of all parameters $(\varphi_1,\ldots,\varphi_L,Pr_1,\ldots,Pr_L)$ contained in the probability model. The number of components is assumed to be L. Eq. (4.2.1) means that the probability that the data $\mathbf{x}_k$ occurs from the component density C_{y_k} is the joint probability of two independent actions: specifying the component density C_{y_k} and emitting the data $\mathbf{x}_k$ from the component. The probability density function of incomplete (observed) data $\mathbf{x}_k$

[1] In Chapter 2, we denoted the category-conditional pdf by $f(\mathbf{x}_k|\omega_{y_k},\varphi_{y_k})$; here for simplicity it is denoted by $f_{y_k}(\mathbf{x}_k|\varphi_{y_k})$

is therefore given as the marginal probability:

$$f(\mathbf{x}_k)=\sum_{y_k} f(\mathbf{x}_k,y_k|\Phi)$$
$$=\sum_{y_k} Pr_{y_k} f_{y_k}(\mathbf{x}_k|\varphi_{y_k}) \qquad (4.2.2)$$

Eq. (4.2.2) is the commonly used mixture probability density function of observed data.

The EM algorithm maximises the logarithm of Eq. (4.2.2) by iteratively maximising the expectation of the logarithm of Eq. (4.2.1) over parameters Φ. As shown in Chapter 2, this leads to maximisation of the following Q-function:

$$Q(\Phi,\overline{\Phi})=\sum_{k=1}^{T} Q_k(\Phi,\overline{\Phi})$$
$$=\sum_{k=1}^{T}\sum_{y_k}\frac{f(\mathbf{x}_k,y_k|\Phi)}{f(\mathbf{x}_k|\Phi)}\log f(\mathbf{x}_k,y_k|\overline{\Phi}) \qquad (4.2.3)$$

where T is the number of observed data samples and $\overline{\Phi}$ is the newly estimated parameters in any iteration. By inserting Eq. (4.2.1) into (4.2.3), the following formula is obtained (see Section 2.4.3):

$$Q(\Phi,\overline{\Phi})=\sum_{i=1}^{L} a_i \log \overline{Pr_i}+\sum_{i=1}^{L} Q_\varphi(\Phi,\overline{\varphi}_i) \qquad (4.2.4)$$

where

$$a_i=\sum_{k=1}^{T}\frac{Pr_i f_i(\mathbf{x}_k|\varphi_i)}{f(\mathbf{x}_k|\Phi)} \qquad (4.2.5)$$

$$Q_\varphi(\Phi,\overline{\varphi}_i)=\sum_{k=1}^{T}\frac{Pr_i f_i(\mathbf{x}_k|\varphi_i)}{f(\mathbf{x}_k|\Phi)}\log f_i(\mathbf{x}_k|\overline{\varphi}_i) \qquad (4.2.6)$$

The maximisation of the first term of Eq. (4.2.4) is obtained from linearly constrained optimisation, Example 2.5.3 (see Section 2.5). Then

$$\overline{Pr_i}=\frac{a_i}{\sum_{i=1}^{L} a_i}$$

$$=\sum_{k=1}^{T}\frac{Pr_i f_i(\mathbf{x}_k|\varphi_i)}{f(\mathbf{x}_k|\Phi)} \Bigg/ \left[\sum_{k=1}^{T}\frac{\sum_{i=1}^{L} Pr_i f_i(\mathbf{x}_k|\varphi_i)}{f(\mathbf{x}_k|\Phi)}\right] \qquad (4.2.7)$$

$$=\frac{1}{T}\sum_{k=1}^{T}\frac{Pr_i f_i(\mathbf{x}_k|\varphi_i)}{f(\mathbf{x}_k|\Phi)}$$

The maximisation of the second term of Eq. (4.2.4) is obtained by setting its derivative with respect to $\overline{\varphi}_i$ to zero:

$$\nabla_{\overline{\varphi}_i} Q_\varphi(\Phi,\overline{\varphi}_i)=\sum_{k=1}^{T}\frac{Pr_i f_i(\mathbf{x}_k|\varphi_i)}{f(\mathbf{x}_k|\Phi)}\nabla_{\overline{\varphi}_i}\log f_i(\mathbf{x}_k|\overline{\varphi}_i)$$
$$=\sum_{k=1}^{T}\frac{Pr_i f_i(\mathbf{x}_k|\varphi_i)}{f(\mathbf{x}_k|\Phi)}\,\frac{\nabla_{\overline{\varphi}_i} f_i(\mathbf{x}_k|\overline{\varphi}_i)}{f_i(\mathbf{x}_k|\overline{\varphi}_i)}=0 \qquad (4.2.8)$$

In the mixture Gaussian pdf, each component pdf is given as a normal pdf

$$f_i(\mathbf{x}_k|\overline{\varphi}_i)=N(\mathbf{x}_k,\overline{\mu}_i,\overline{\Sigma}_i) \qquad (4.2.9)$$

Then the partial derivatives in Eq. (4.2.8) with respect to mean vector $\overline{\mu}_i$ and inverse covariance matrix $\overline{\Sigma}_i^{-1}$ are (see Example 2.5.1):

$$\nabla_{\overline{\mu}_i} N(\mathbf{x}_k,\overline{\mu}_i,\overline{\Sigma}_i)=N(\mathbf{x}_k,\overline{\mu}_i,\overline{\Sigma}_i)\overline{\Sigma}_i^{-1}(\mathbf{x}_k-\overline{\mu}_i) \qquad (4.2.10)$$

$$\nabla_{\overline{\Sigma}_i^{-1}} N(\mathbf{x}_k,\overline{\mu}_i,\overline{\Sigma}_i)=\tfrac{1}{2}N(\mathbf{x}_k,\overline{\mu}_i,\overline{\Sigma}_i)(\overline{\Sigma}_i-(\mathbf{x}_k-\overline{\mu}_i)(\mathbf{x}_k-\overline{\mu}_i)^t)$$

Substituting Eq. (4.2.10) into (4.2.8), it can be seen that the re-estimates $\overline{\mu}_i$ and $\overline{\Sigma}_i$, can be given by

$$\bar{\mu}_i = \frac{\sum_{k=1}^{T} \mathbf{x}_k \frac{Pr_i f_i(\mathbf{x}_k|\varphi_i)}{f(\mathbf{x}_k|\Phi)}}{\sum_{k=1}^{T} \frac{Pr_i f_i(\mathbf{x}_k|\varphi_i)}{f(\mathbf{x}_k|\Phi)}} \qquad (4.2.11)$$

$$\bar{\Sigma}_i = \frac{\sum_{k=1}^{T} (\mathbf{x}_k - \bar{\mu}_i)(\mathbf{x}_k - \bar{\mu}_i)^t \frac{Pr_i f_i(\mathbf{x}_k|\varphi_i)}{f(\mathbf{x}_k|\Phi)}}{\sum_{k=1}^{T} \frac{Pr_i f_i(\mathbf{x}_k|\varphi_i)}{f(\mathbf{x}_k|\Phi)}} \qquad (4.2.12)$$

Here, $Pr_i f_i(\mathbf{x}_k|\varphi_i)/f(\mathbf{x}_k|\Phi)$ is the *a posteriori* probability and can be considered as the probability that observation $\mathbf{x}_k$ belongs to component C_i. The information as to whether a given observation $\mathbf{x}$ should belong to component C_i (the *i*th Gaussian pdf) is *hidden*, and can only be seen via y. The solution of the EM algorithm is to compute when and how often a given observation $\mathbf{x}$ will be expected to be in each component. These expected statistics can then be used to compute new estimates of new parameters $\bar{\Phi}$. No assumption is imposed on how each component pdf should be organised with any other. As will be discussed later, the Markov properties can be imposed on these component pdfs 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. 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 7.

4.2.2. Simplified mixture pdf estimation

Several types of VQ algorithms can be derived by simplifying the process of mixture Gaussian pdf estimation. Figure 4.2.2 shows their relations and the algorithms are

described below as examples.

Example 4.2.1. Mixture Gaussian VQ algorithm

The estimation of mixture Gaussian pdf is iteratively carried out, and the resultant mean vectors $\bar{\mu}_i$ are interpreted as codeword $\mathbf{z}_i$. We call this vector quantisation here *mixture Gaussian VQ*. The algorithm is described as followed:

Step 1: Initialisation. Choose some adequate method to derive an initial VQ codebook (μ_i, $1 \le i \le L$), related covariance matrix Σ_i and *a priori* probability Pr_i.

Step 2: Contribution computation. Compute contributions from each element of training vectors $\{\mathbf{x}_k\}$ to each component C_i by computing the *a posteriori* probability $Pr_i f_i(\mathbf{x}_k|\varphi_i)/f(\mathbf{x}_k|\Phi)$.

Step 3: Codebook updating. Update the codeword by computing the centroid in each component pdf by using the contribution of each training vector based on Eq. (4.2.11). Update the related covariance matrix and *a priori* probability based on Eq. (4.2.12) and (4.2.7).

Step 4: Termination. If the increase in the value of the Q-function at the current iteration relative to the value of the Q-function at the previous iteration is below a chosen threshold, **STOP**; otherwise go to Step 2.

The *mixture Gaussian VQ* can be further simplified. For example, in Step 2 of Example 4.2.1, one can select the component to which the contribution of the training vector $\mathbf{x}_k$ is the largest. Such a *single* classification has the same strategy as the conventional minimum-error-rate classifier (see Section 2.2.3) based on *a posteriori* probability. Based on the single classification approach, many alternatives exist by replacing distortion measures. For example, a distortion measure used in *mixture Gaussian VQ* may be:

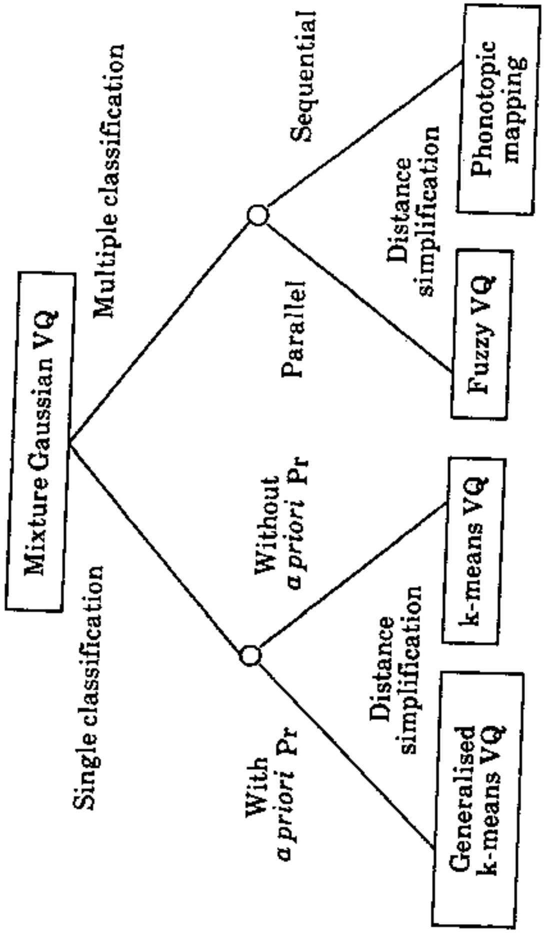

Figure 4.2.2. Relation of several VQ techniques.

$$d(\mathbf{x}_k,\mu_i) = -\log f(\mathbf{x}_k,y_k|\overline{\Phi})$$
$$= -\log Pr_i + \tfrac{1}{2}\log(2\pi)^d|\Sigma_i| \qquad (4.2.13)$$
$$+\tfrac{1}{2}(\mathbf{x}_k-\mu_i)^t\Sigma_i^{-1}(\mathbf{x}_k-\mu_i)$$

Based on such a distortion measure, one can assume that elements of training vectors $\mathbf{x}=(x_1,x_2,...,x_d)$ are independent with a fixed variance σ^2. This leads to a simplified distortion:

$$d(\mathbf{x}_k,\mu_i) = -\log Pr_i + \tfrac{1}{2}\log(2\pi)^d\sigma^{2d}$$
$$+\tfrac{1}{2}(\mathbf{x}_k-\mu_i)^t(\mathbf{x}_k-\mu_i)/\sigma^2 \qquad (4.2.14)$$

The above distortion measure can be expressed as:

$$d(\mathbf{x}_k,\mu_i) = (\mathbf{x}_k-\mu_i)^t(\mathbf{x}_k-\mu_i) - \lambda\log Pr_i \qquad (4.2.15)$$

When this distortion measure is used, it is called *entropy constrained VQ* [6], or *generalised k-means VQ* in which *a priori* probability Pr_i is taken into consideration in *k-means VQ*, and the term $-\lambda\log Pr_i$ is related to the entropy of the codewords in coding [6]. If the *a priori* probability term in Eq. (4.2.15) is discarded, it is exactly the same as the conventional *k-means VQ* algorithm. In addition, if the covariance matrix Σ_i is kept, it leads to *k-means VQ* with *Mahalanobis* distortion measure as shown in Section 4.1.

Example 4.2.2. Fuzzy VQ algorithm [4,20,22]

Fuzzy VQ can also be viewed as a simplified case of the mixture Gaussian VQ. When *k-means VQ* with Euclidean distance is used in multiple classification mode, the overall average distortion can be written as:

$$D = \sum_{k=1}^{T}\sum_{i=1}^{L} m_{ki}^{F} d(\mathbf{x}_k,\mu_i) \qquad (4.2.16)$$

where m_{ki} corresponds to the *a posteriori* probability or contribution of training vector $\mathbf{x}_k$ to component C_i in Step 2 of the *mixture Gaussian VQ*, therefore

$$\sum_{i=1}^{L} m_{ki} = 1 \tag{4.2.17}$$

It should be noticed that Eq. (4.2.16) corresponds to the negative of the Q-function (4.2.3). Eq. (4.2.16) is called the fuzzy objective function, and the parameter $F>1$ is called the degree of fuzziness. The optimal m_{ki} which minimises Eq. (4.2.16) under the condition Eq. (4.2.17) is given as (see Example 2.5.4):

$$m_{ki} = \left[\sum_{j=1}^{L} [d(\mathbf{x}_k, \mu_i)/d(\mathbf{x}_k, \mu_j)]^{1/(F-1)} \right]^{-1} \tag{4.2.18}$$

When the above contribution m_{ki} is used with Euclid distance, it is called *Fuzzy VQ* or *c-means VQ*.

Step 1: Initialisation. Choose some adequate method to derive an initial VQ codebook (μ_i, $1 \le i \le L$).

Step 2: Contribution computation. Compute contribution m_{ki} of each element of training vectors $\{\mathbf{x}_k\}$ to one of the components C_i by using Eq. (4.2.18).

Step 3: Codebook updating. Update the codeword by computing the centroid by using the contribution of each training vector:

$$\mu_i = \frac{\sum_{k=1}^{T} m_{ki}^F \mathbf{x}_k}{\sum_{k=1}^{T} m_{ki}^F} \tag{4.2.19}$$

Step 4: Termination. If the decrease in the value of the fuzzy objective function at the current iteration relative to the value of the fuzzy objective function at the previous iteration is below a chosen threshold, **STOP**; otherwise go to Step 2.

The above described VQ algorithms can be viewed as parallel algorithms since processing of previous training data $\mathbf{x}_{k-1}$ does not affect the processing of the present training data $\mathbf{x}_k$. On the other hand, there is a sequential VQ algorithm in which the present data $\mathbf{x}_k$ is affected by the previous data processing. The typical algorithm is called *phonotopic mapping*, described as follows:

Example 4.2.3. Phonotopic mapping algorithm [14,15]

Step 1: Initialisation. Choose some adequate method to derive an initial VQ codebook (μ_i, $1 \le i \le L$), and set k equal to 1.

Step 2: Contribution computation. Select the nearest codeword μ_i to the kth training vector $\mathbf{x}_k$ using Euclidean distance. Collect the N_i codewords which are nearest to μ_i. Compute the contribution of data $\mathbf{x}_k$ to each of the N_i codewords by:

$$\alpha_k(\mathbf{x}_k - \mu_i^k) \tag{4.2.20}$$

where α_k is a monotonically decreasing function of k, and μ_i^k is the codeword μ_i updated at the kth data training.

Step 3: Codebook updating. Update the N_i codewords by updating the new centroid by using the contribution of the training vector based on the following expression:

$$\mu_i^{k+1} = \mu_i^k + \alpha_k(\mathbf{x}_k - \mu_i^k) \tag{4.2.21}$$

Step 4: Termination. If the all data are used for updating the codeword, **STOP**; otherwise go to Step 2 for the next data ($k = k+1$). Steps 2, 3 and 4 are repeated for the same training data set.

The above mentioned VQ techniques were mainly used in speech or image coding. There are a sender and a receiver in the communication. The sender encodes speech/image signal into codes and transmits them to the receiver. The receiver decodes the transmitted codes into a speech/image signal. A codebook is the information pair of code number and the corresponding codewords (vectors) which are designed to minimise the overall average distortion (by Step 1 to Step 4 in the VQ algorithm).

Sharing this codebook, the sender encodes the input speech/image data by selecting the code number whose codeword is closest to the data, and transmitting it to the receiver. The receiver reproduces the speech/image data by looking up the same codebook using the transmitted code number and getting the corresponding codeword. VQ techniques are also used in speech recognition where a typical example is a discrete HMM in which codewords are used as sample space to represent the probability mass function of speech data. The relationship between VQ and HMM will be described in Chapter 7.

4.3. VQ for Category Discrimination

In the previous sections, we have discussed several kinds of VQ techniques for generating the codebook which is mainly used in speech/image coding. For speech recognition, recent researches show that these VQ techniques should incorporate the linguistic information and classification information in the optimisation procedure. Here we will give two examples which are specifically designed for speech recognition by incorporating phoneme category (or other linguistic category) information in the optimisation procedure. A more general unified theory which can optimise the VQ codebook as well as HMM parameters is within this scheme, and will be described separately in Chapter 7. The purpose of the VQ, in this case, is to design the codewords which can discriminate the phoneme category, instead of minimising overall average distortion. We assume that each training data $\mathbf{x}_k$ has a label indicating its phoneme category y_k.

Example 4.3.1. Learning vector quantisation (LVQ) [14]

This algorithm proceeds sequentially and is similar to phonotopic mapping. The difference is that only one codeword μ_i is selected for updating instead of N_i codewords in Step 2 of phonotopic mapping. A phoneme category is given to the training data and the selected codeword, whose category is determined by majority rule, for the data used to compute the codeword. If the phoneme category y_k of the data $\mathbf{x}_k$ is equal to the phoneme category of the closest codeword μ_i (correct case), the codeword of the component is updated (rewarded) as follows:

$$\mu_i^{k+1}=\mu_i^k+\alpha_k(\mathbf{x}_k-\mu_i^k) \qquad (4.3.1)$$

otherwise punished (incorrect case):

$$\mu_i^{k+1}=\mu_i^k-\alpha_k(\mathbf{x}_k-\mu_i^k) \qquad (4.3.2)$$

The other codewords are not updated by this training data. The effect of this VQ lies in the ability to learn the decision boundary optimally between two categories by using Eq. (4.3.1) in the correct case and by using Eq. (4.3.2) in the incorrect case.

Another example to be discussed here is *mutual information based VQ*. This algorithm also proceeds sequentially and is similar to the inverse of *LBG VQ*. Instead of splitting each cluster, it merges sequentially two clusters which can keep the mutual information defined between codewords and phoneme as high as possible after provisional merge, among all the cluster combinations. This makes it possible to get codewords which have good correspondence to phonemes, or can discriminate phonemes.

Example 4.3.2. Mutual information based VQ [21]

This algorithm starts with producing many clusters by the LBG algorithm, then reduces the number of clusters by merging them, while keeping mutual information as high as possible.

Let z_i and y_k denote the codeword of cluster C_i and phoneme categories Z and Y denote the codebook $\{z_i\}$ and phoneme set $\{y_k\}$ respectively. The number of codewords is assumed to be L at the initial step. $H(Y)$ and $H(Y|Z)$ denote *a priori* entropy of phoneme category and conditional entropy of phoneme category after observing the codeword respectively. Mutual information between acoustic codeword and phoneme category is denoted as $I(Y;Z)=H(Y)-H(Y|Z)$ (see Section 2.6). The merging process is as follows:

Step 1: Select two codewords $\mathbf{z}_i$ and $\mathbf{z}_j$ among all codewords, and compute the following probability after provisionally merging them.

$$Pr(y_k|z_{ij})=Pr(y_k|z_i)+Pr(y_k|z_j) \tag{4.3.3}$$

where $Pr(y_k|z_{ij})$ is the *a posteriori* probability after provisionally merging the two codewords $\mathbf{z}_i$ and $\mathbf{z}_j$. Then compute the mutual information in the following way:

$$\begin{aligned} I_{ij}(Y;Z) &= H(Y)-H_{ij}(Y|Z) \\ &= \sum_k \sum_{l=1}^{L-1} P(y_k,z_l)\log\frac{Pr(y_k,z_l)}{Pr(y_k)Pr(z_l)} \end{aligned} \tag{4.3.4}$$

where $H_{ij}(Y|Z)$ and $I_{ij}(Y;Z)$ are the conditional entropy and mutual information after provisionally merging two codewords $\mathbf{z}_i$ and $\mathbf{z}_j$.

Step 2: Merge two codewords $\mathbf{z}_i$ and $\mathbf{z}_j$ which can show the maximum mutual information $I_{ij}(Y;Z)$ $i \neq j$ after their provisional merge in Step 1, and also merge the *a posteriori* probabilities $Pr(y_k|z_i)$ and $Pr(y_k|z_j)$. Decrease the total number of codewords $L=L-1$, and go to Step 1.

The final codewords are obtained by computing the mean vector of merged clusters after the above merging process finishes.

The same technique as shown in Example 4.3.2 is applicable without using the phoneme category in an unsupervised way. In this case, the VQ codeword is itself regarded as a category, and merged one by one according to the mutual information criteria [5].

4.4. Summary

In this chapter we have discussed several prerequisites for discrete 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 (codewords) 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 relationship between them is shown through simplification of the classification process and distortion measure.

The assumption of probability density function 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 re-estimate the unobservable 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.

References

1. M.R. Anderberg, *Cluster Analysis for Applications*, Academic Press, 1973.

2. 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.

3. L.E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains," *Ann. Math. Stat.*, vol. 41, pp. 164-171 , 1970.

4. J.C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum Press, 1981.

5. P.F. Brown, "Acoustic-phonetic modeling problem in automatic speech recognition," Ph.D. thesis, Department of Computer Science, Carnegie-Mellon University, 1987.

6. P. Chou, T. Lookabaugh, and R. Gray, "Entropy-constrained vector quantization," *IEEE Trans. Acoustic, Speech, and Signal Processing*, vol. ASSP-37, pp. 31-42, 1989.

7. 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.

8. R.O. Duda and R.E. Hart, *Pattern Classification and Scene Analysis*, John Wiley, 1973.

9. E.W. Forgy, "Cluster analysis of multivariate data: Efficiency vs. interpretability of classifications," *Biometrics*, vol. 21, p. 768, abstract, 1965.

10. A. Gersho, "On the structure of vector quantizers," *IEEE Trans. Information Theory*, vol. IT-28, pp. 157-166, 1982.

11. R.M Gray and E.D. Karnin, "Multiple local optima in vector quantizers," *IEEE Trans. Information Theory*, vol. IT-28, pp. 256-261, 1982.

12. R.M. Gray, "Vector quantization," *IEEE ASSP Magazine*, pp. 4-29, April, 1984.

13. 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.

14. T. Kohonen, *Self-Organization and Associative Memory*, Springer-Verlag, 1984.

15. T. Kohonen, "The "neural" phonetic typewriter," *Computer*, vol. March, pp. 11-22, 1988.

16. Y. Linde, A. Buzo, and R.M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Commun.*, vol. COM-28, pp. 84-95, 1980.

17. J. Makhoul, S. Roucos, and H. Gish, "Vector quantisation in speech coding," *Proc. IEEE*, vol. 73, pp. 1551-1588, 1985.

18. D.B. Paul, "An 800 bps adaptive vector quantization vocoder using a perceptual distance measure," *Proc. ICASSP-83*, pp. 73-76, USA, 1983.

19. R.A. Redner and H.F. Walker, "Mixture densities, maximum likelihood and the EM algorithm," *SIAM review*, vol. 26, pp. 195-239, 1984.

20. S.Z. Selim and M.A. Ismail, "Soft clustering of multidimensional data; a semi-fuzzy approach," *Pattern Recognition*, vol. 17, pp. 559-568, 1984.

21. K. Shirai, N. Aoki, and N. Hosaka, "Multi-level clustering of acoustic features for phoneme recognition based on mutual information," *Proc. ICASSP-89*, pp. 604-607, Glasgow, UK, 1989.

22. 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.

CHAPTER FIVE

HIDDEN MARKOV MODELS AND BASIC ALGORITHMS

As pointed out in previous chapters, there are many uncertainties in speech recognition. Stochastic modelling is a flexible general method for modelling such problems. Hidden Markov modelling is such a stochastic technique for the study of the complete-incomplete data problems associated with time series. It permits modelling with many of the classical probability distributions and is well suited to the incorporation of temporal information. 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.

5.1. 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 high 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 by either of the other symbols, a or c, be 1/4. Finally, let the probability that the symbol c is followed by a c be 1/2, and by either of the other two, a or b, be 1/4. We have

$$Pr(a|a) = \frac{1}{3},\ Pr(b|a) = \frac{1}{3},\ Pr(c|a) = \frac{1}{3}$$

$$Pr(a|b) = \frac{1}{4},\ Pr(b|b) = \frac{1}{2},\ Pr(c|b) = \frac{1}{4}$$

$$Pr(a|c) = \frac{1}{4},\ Pr(b|c) = \frac{1}{4},\ Pr(c|c) = \frac{1}{2}$$

It is conventional to use a transition graph to illustrate a Markov process. There are,(of course) three states in this example (for a, b, and c respectively) indicated by the circles shown in Figure 5.1.1.

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, an example given earlier in this text. Let state a have output, saying *sunny*, state b

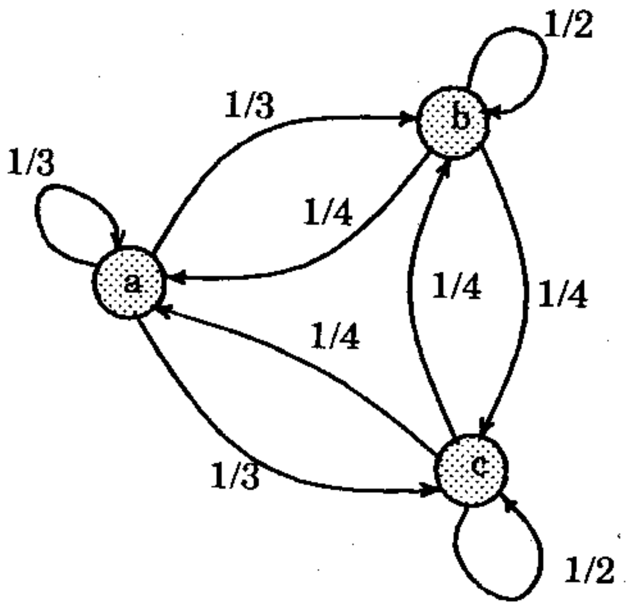

Figure 5.1.1. An example of a Markov process.

have output *cloudy*, and state *c* have output *rainy*. Given today as being 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 there 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 labelled as 1, 2, and 3, then the transition graph shown in Figure 5.1.1 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 be 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 $\pi_1 + \pi_2 + \pi_3 = 1$. The case may arise where one value $\pi_i = 1$ 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 a Markov model, 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 corresponds to an output probability distribution instead of a deterministic event. 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 powerful in certain applications such as speech recognition in particular. In the following sections, we will discuss a theory to lift such a veil, i.e. the theory of hidden Markov models.

5.2. 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 before 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 analysing segment. If it is assumed that these periods of steady state behaviour can be identified, and that the temporal variations within each of these steady periods are, in a sense, statistical, then a statistical model may be used to represent 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 steady 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 [9] 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 the transition between states, which corresponds to 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 characterised by the output probabilities.

To understand the concept of the HMM, consider the following example illustrated as Figure 5.2.1. A person is performing a 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 an 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 (the urns) conducted behind the veil.

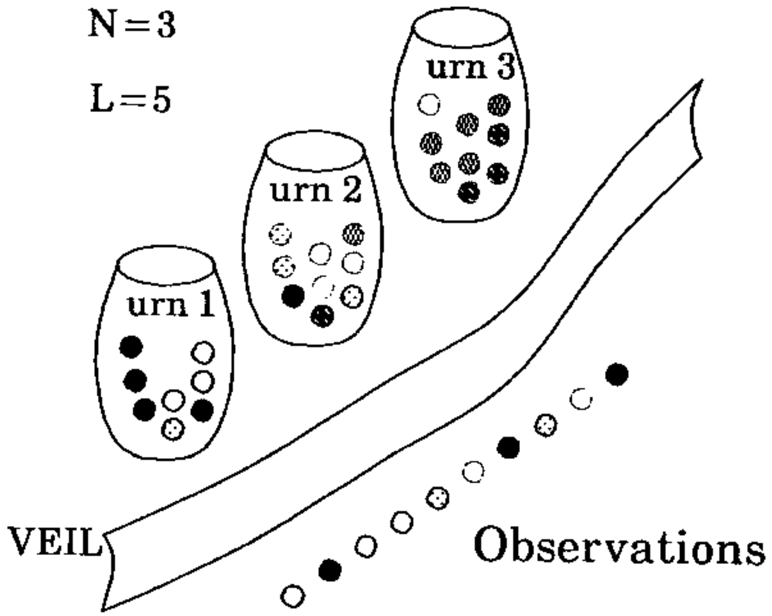

Figure 5.2.1. The urn and ball experiment.

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

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

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

L = number of observation symbols (number of colours in use with the balls)

$S = \{s\}$, 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.

$v = \{v_1, v_2, \ldots, v_L\}$, 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[1], 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 re-estimation algorithms for the model parameters.

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

[1] The state-dependent output probability is a special case of the transition-dependent one.

An HMM can be represented by using the compact notation $\lambda = (A, B, \pi)$. Specification of an HMM involves the choice of the number of states, N, 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 also 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 unity.

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, $b_i(O)$, 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 5.2.2. shows an example of such an HMM. The transition probabilities are labelled in the figure;

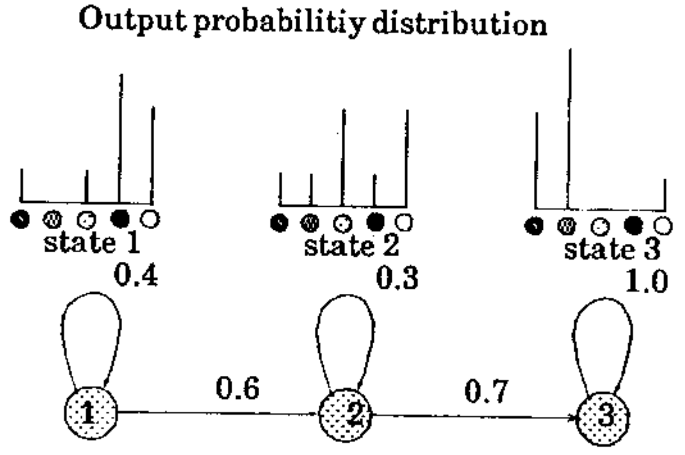

Figure 5.2.2. Example of a simple hidden Markov model.

the output probabilities associated with each state are also illustrated. If the model is looked at generatively, the Markov chain syntheses a sequence of states (urns), and the output probability distributions then turn the sequence of states into a time series (observed 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 assumptions severely limit the local memory of first-order HMMs, they reduce the number of *free* parameters, and also make learning and decoding algorithms extremely efficient. Efforts to model time correlations explicitly can be found in [5,14].

In statistical modelling, free parameters refer to those parameters that will be estimated from observations, and correspond to the most important factor in statistical system design. Even though inadequate assumptions are made, the system may provide better performance if such assumptions can drastically reduce the overall 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 have been 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 the same parabola as well as the points at hand. As noise exists, if the number of sample points is limited, a straight line may fit the given data even better than the parabola, though curves fitted from a larger data set might be quite different. On the other hand, a higher order 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 higher order polynomial than a low order polynomial because more parameters need to be estimated for the higher order polynomial.

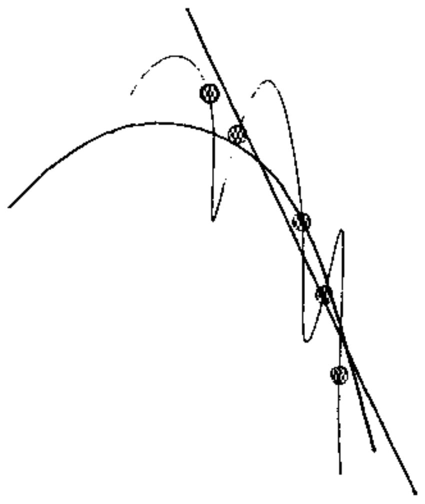

Figure 5.2.3. Fitting curves to a set of samples.

5.3. Basic Algorithms for HMMs

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

(1) The Evaluation Problem: Given the observation sequence $\mathbf{O} = O_1, O_2, \ldots, O_T$, and the model $\lambda = (A, B, \pi)$, the problem is how to compute $Pr(\mathbf{O}|\lambda)$, the probability that this observed sequence was produced by the model. This problem can also be viewed as: given several

competing models and a sequence of observations, how do we choose the model which best matches the observations for the purpose of classification or recognition.

(2) The Estimation Problem: Given the observation sequence **O**, how do we 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 observations have come about.

(3) The Decoding Problem: Given the observation sequence **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 can be closely related under the same probabilistic framework.

5.3.1. 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 $S = s_1,s_2,...,s_T$, because of our assumptions, the probability of the observation sequence **O** is $Pr(\mathbf{O}|S,\lambda)$, where

$$Pr(\mathbf{O}|S,\lambda) = b_{s_1}(O_1)b_{s_2}(O_2)\cdots b_{s_T}(O_T) \tag{5.3.1}$$

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

$$\begin{aligned} Pr(S|\lambda) &= \pi_{s_1}a_{s_1s_2}a_{s_2s_3}\cdots a_{s_{T-1}s_T} \\ &= a_{s_0s_1}a_{s_1s_2}a_{s_2s_3}\cdots a_{s_{T-1}s_T} \end{aligned} \tag{5.3.2}$$

where $a_{s_0s_1}$ 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.

$$Pr(\mathbf{O},S|\lambda) = Pr(\mathbf{O}|S,\lambda)Pr(S|\lambda) \tag{5.3.3}$$

The probability $Pr(\mathbf{O}|\lambda)$ is the summation of Eq. (5.3.3) over all possible state sequences:

$$\begin{aligned} Pr(\mathbf{O}|\lambda) &= \sum_{all\ S} Pr(\mathbf{O}|S,\lambda)Pr(S|\lambda) \\ &= \sum_{all\ S}\prod_{t=1}^{T} a_{s_{t-1}s_t}b_{s_t}(O_t) \end{aligned} \tag{5.3.4}$$

From Eq. (5.3.4) it can be seen that a transition starts from an initial state (at time $t=1$) with probability $a_{s_0s_1}$ (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 $a_{s_1s_2}$, 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 is of 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, given that there are only N states, all 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 [11].

The forward variable can be first defined as:

$$\alpha_t(i) = Pr(O_1,O_2,...,O_t,s_t=i|\lambda) \tag{5.3.5}$$

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:

$$\alpha_t(j) = [\sum_i \alpha_{t-1}(i) a_{ij}] b_j(O_t) \tag{5.3.6}$$

Step 3: Final probability is given by:

$$Pr(\mathbf{O}|\lambda) = \sum_{i \in S_F} \alpha_T(i) \tag{5.3.7}$$

In the above forward iteration, Step 1 initialises the forward probabilities with the initial probability for all states. Eq. (5.3.6) illustrates that state j can be reached at time t from all the possible states i at time $t-1$. Note that $\alpha_{t-1}(i)$ is the probability of the joint event that the sequence $O_1, O_2, ..., O_{t-1}$ is observed and the last state is i: thus the product $\alpha_{t-1}(i) a_{ij}$ is then the probability that the joint events $O_1, O_2, ..., O_{t-1}$ 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 $O_1, O_2, ..., O_{t-1}, O_t$ 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 $\alpha_T(i) = Pr(O_1, O_2, ..., O_T, s_T = i|\lambda)$ and the transitions must end at one of S_F.

The computation in the calculation of $\alpha_t(j)$ is of the order of $O(N^2T)$. An example of the recursive computation for the forward variable using the model given in Figure 5.2.2 is illustrated in Figure 5.3.1. The computation leads to a lattice 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 considered, the final state in the final column contains the probability of generating the given observation sequence $\mathbf{O}$.

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

$$\beta_t(i) = Pr(O_{t+1}, O_{t+2}, ..., O_T | s_t = i, \lambda) \tag{5.3.8}$$

i.e. the probability of the partial observation sequence from $t+1$ to the final observation T, given state i at time t and

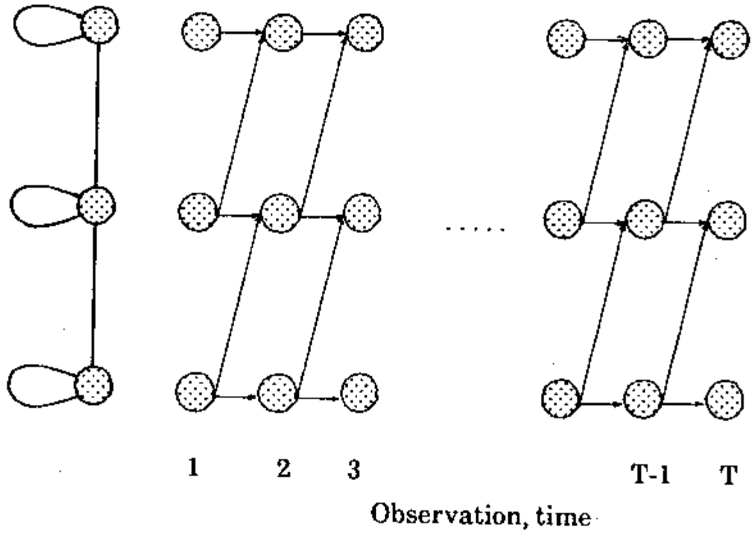

Figure 5.3.1. Illustration of computation for the forward variables.

the model λ. This backward variable can also be 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 $t=T-1,T-2,...,1$ and all states j, compute:

$$\beta_t(j) = [\sum_i a_{ji}b_i(O_{t+1})\beta_{t+1}(i)] \qquad (5.3.9)$$

Step 3: Final probability is given by:

$$Pr(O|\lambda) = \sum_{i \in S_I} \pi_i b_i(O_1)\beta_1(i) \qquad (5.3.10)$$

Step 1 arbitrarily defines $\beta_T(i)$ to be $1/N_F$ for all final states. Step 2 shows that in order to have been in state j at time t, and to accounts 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 accounts for the rest of the observation sequence. The computation complexity of $\beta_t(i)$ 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 in Section 5.3.3.

5.3.2. 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 [13]. It is very similar to the DTW algorithm discussed in Chapter 3, where the transition information is neglected and speech data such as the LPC cepstrum are usually stored without further parameterisation.

Viterbi algorithm

Step 1: Initialisation. For all states i,

$$\delta_1(i)=\pi_i b_i(O_1)$$
$$\Psi_1(i)=0;$$

Step 2: Recursion. From time $t=2$ to T, for all states j,

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

Step 3: Termination. (* indicates the optimised results).

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

Step 4: Path (state sequence) backtracking. From time $T-1$ to 1

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

The Viterbi algorithm can also be 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 $\underset{S}{\mathrm{Max}}[Pr(\mathbf{O},S|\lambda)]$ may not represent well the summation for $Pr(\mathbf{O}|\lambda)$, especially in the HMM parameter estimation procedure, although the probabilities obtained from the forward and Viterbi algorithms may be very close [12]. In such cases, the forward–backward algorithm may work more robustly than the Viterbi algorithm. This is achieved for an increase in computational complexity since in the forward–backward algorithm 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. Also 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 [7,12].

5.3.3. Baum-Welch re-estimation 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 this analytically for a maximum likelihood model as discussed in Chapter 2. Therefore an iterative algorithm or gradient technique for optimisation is used. The iterative algorithm used in HMM-based speech recognition is known as the Baum–Welch algorithm [3]. This has the same optimisation techniques as the EM algorithm for the mixture density problems discussed in Chapter 2. Here, the unobservable data are a sequence of hidden states, S. 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 an estimation of original model parameters based on current probabilities.

Consider any model whose parameters λ contain no zero values. 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 *a posterior* probability of transitions γ_{ij}, from state i to state j, conditioned on the observation sequence and the model can be computed as:

$$\begin{aligned}\gamma_t(i,j) &= Pr(s_t=i, s_{t+1}=j|\mathbf{O},\lambda)\\ &= \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{Pr(\mathbf{O}|\lambda)}\\ &= \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{\sum_{k\in S_F}\alpha_T(k)}\end{aligned} \tag{5.3.11}$$

As illustrated in Figure 5.3.2, $\gamma_t(i,j)$ 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, multiplied by $a_{ij}b_j(O_{t+1})$, which accounts for the local transition from state i, multiplied by $\beta_{t+1}(j)$, which accounts for the path being in state j at time $t+1$.

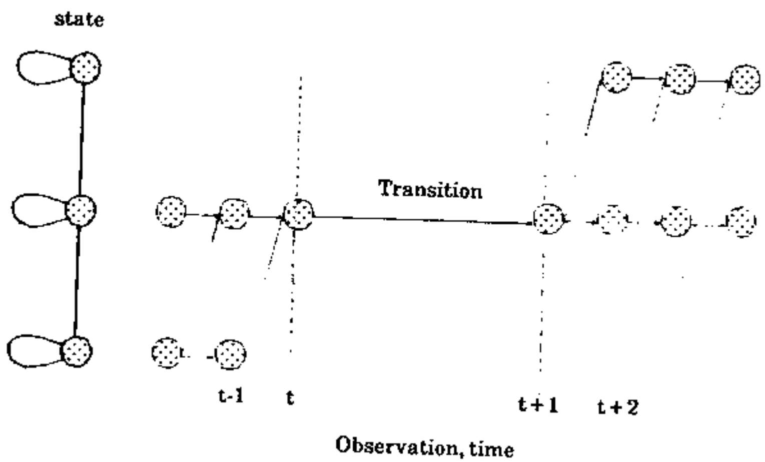

Figure 5.3.2. Illustration of computation for gamma values.

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

$$\gamma_t(i) = Pr(s_t=i|\mathbf{O},\lambda) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{k\in S_F}\alpha_T(k)} \tag{5.3.12}$$

From Eq. (5.3.12), 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.

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 improve estimates iteratively.

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

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1}\gamma_t(i,j)}{\sum_{t=1}^{T-1}\sum_j \gamma_t(i,j)} = \frac{\sum_{t=1}^{T-1}\gamma_t(i,j)}{\sum_{t=1}^{T-1}\gamma_t(i)} \tag{5.3.13}$$

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 re-estimated as:

$$\bar{b}_j(k) = \frac{\sum\limits_{t \epsilon O_t = v_k} \gamma_t(j)}{\sum\limits_{t=1}^{T} \gamma_t(j)} \tag{5.3.14}$$

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

$$\bar{\pi}_i = \gamma_1(i) \tag{5.3.15}$$

It can be shown that either:

1. the initial model λ defines a critical point of the likelihood function, where new estimates equal 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 are more likely to produce the given observation sequence $\mathbf{O}$.

Thus if $\bar{\lambda}$ is iteratively used to replace λ and repeat the above re-estimation calculation, it can be guaranteed that $Pr(\mathbf{O}|\lambda)$ can be improved until some limiting point is reached. Eq. (5.3.12) to (5.3.14) are instances of the Baum – Welch re-estimation algorithm [3], which has the same form as the EM algorithm discussed in Chapter 2. 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 system debugging as a *divide – conquer* strategy. Another useful debugging method is based on the fact that $Pr(\mathbf{O}|\lambda)$ should be the same whether it is calculated from forward or backward probabilities, i.e.

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

It is implied by the content of Baum and Eagon [1] and Petrie [10] that the true model can be recovered from a sufficiently long observation sequence except for 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 under certain conditions be successfully *lifted*.

Note that a single observation sequence is not enough for re-estimation 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 a word-based speech recognition system, several utterances for the same word are generally required to re-estimate the HMM parameters. The re-estimation formulas of Eq. (5.3.13) to (5.3.15) can be easily extended to such multiple observation sequences. Let $\mathbf{O}^M = [\mathbf{O}^1, \mathbf{O}^2, \ldots, \mathbf{O}^m]$ 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

$$\log Pr(\mathbf{O}^M|\lambda) = \sum_{n=1}^{m} \log Pr(\mathbf{O}^n|\lambda) \tag{5.3.16}$$

Let $\sum\limits_t \gamma_t^n(i,j)$ denote the expected number of transitions from state i to state j estimated from $\mathbf{O}^n$. The average expected number of transitions, $\sum\limits_t \gamma_t(i,j)$ is the summation of $\gamma_t^n(i,j)$

with respect to n. Thus, the re-estimation equation for the transition probability, a_{ij}, can be computed:

$$\bar{a}_{ij} = \frac{\sum_{n} \sum_{t=1}^{T_n-1} \gamma_t^n(i,j)}{\sum_{n} \sum_{t=1}^{T_n-1} \sum_{j} \gamma_t^n(i,j)} \tag{5.3.17}$$

Let $\sum_{t} \gamma_t^n(i)$ denote the expectation of being in state i at time t estimated from $\mathbf{O}^n$. Similarly, Eq. (5.3.14) can be extended to multiple observation sequences as:

$$\bar{b}_j(k) = \frac{\sum_{n} \sum_{t \in O_t^n = v_k} \gamma_t^n(j)}{\sum_{n} \sum_{t=1}^{T_n} \gamma_t^n(j)} \tag{5.3.18}$$

5.4. Proof of the Re-estimation Algorithm

The original proof of the Baum–Welch algorithm, which dealt specifically with a finite alphabet and general output distributions, appeared in [1]. A generalised proof was then based on constructing an information-theoretic Q-function, i.e. Kullback–Leibler number [3,6]. This are actually the same as the Q-function of the EM algorithm for mixture densities discussed in Chapter 4. Here, the unobservable data are a sequence of hidden states, S. For models λ and $\bar{\lambda}$, the Q-function can be defined as:

$$Q(\lambda,\bar{\lambda}) = \frac{1}{Pr(\mathbf{O}|\lambda)} \sum_{all\ S} Pr(\mathbf{O},S|\lambda) \log Pr(\mathbf{O},S|\bar{\lambda}). \tag{5.4.1}$$

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 5.4.1.
$Q(\lambda,\bar{\lambda}) \geq Q(\lambda,\lambda) \Rightarrow Pr(\mathbf{O}|\bar{\lambda}) \geq Pr(\mathbf{O}|\lambda)$. 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

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

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 6). From Theorem 5.4.1, we have

$$\log\frac{Pr(\mathbf{O}|\bar{\lambda})}{Pr(\mathbf{O}|\lambda)} \geq Q(\lambda,\bar{\lambda}) - Q(\lambda,\lambda). \tag{5.4.2}$$

If a new model $\bar{\lambda}$ that makes the right-hand side of Eq. (5.4.2) positive can be found, it means that the model re-estimation algorithm can be guaranteed to improve the $Pr(\mathbf{O}|\lambda)$. Clearly, the guaranteed improvement by this method results in $\bar{\lambda}$, 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. (5.3.13) to (5.3.15). To show this let the state sequence be $S = s_1, s_2,, \ldots, s_T$. Then

$$\log Pr(\mathbf{O},S|\bar{\lambda}) = \log\bar{\pi}_{s_1} + \sum_{t=1}^{T-1} \log\bar{a}_{s_t s_{t+1}} + \sum_{t=1}^{T} \log\bar{b}_{s_t}(O_t) \tag{5.4.3}$$

Substituting Eq. (5.4.3) in (5.4.1) and regrouping terms in the summations according to state transitions and observed symbols, it can be seen that

$$Q(\lambda,\bar{\lambda})=\sum_i\sum_j c_{ij}\log\bar{a}_{ij}+\sum_j\sum_{k=1}^{L}d_{jk}\log\bar{b}_j(k)+\sum_i e_i\log\bar{\pi}_i \qquad (5.4.4)$$

Here

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

$$= \sum_{t=1}^{T-1}\gamma_t(i,j) \qquad (5.4.5)$$

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

$$= \sum_{t\in O_t=v_k}\gamma_t(j) \qquad (5.4.6)$$

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

$$= \gamma_1(i) \qquad (5.4.7)$$

Thus, according to Example 2.5.3 (see Section 2.5), $Q(\lambda,\bar{\lambda})$ can be maximised if

$$\bar{a}_{ij} = \frac{c_{ij}}{\sum_j c_{ij}} = \frac{\sum_{t=1}^{T-1}\gamma_t(i,j)}{\sum_{t=1}^{T-1}\sum_j\gamma_t(i,j)} \qquad (5.4.8)$$

$$\bar{b}_j(k) = \frac{d_{jk}}{\sum_k d_{jk}} = \frac{\sum_{t\in O_t=v_k}\gamma_t(j)}{\sum_t\gamma_t(j)} \qquad (5.4.9)$$

$$\bar{\pi}_i = \frac{e_i}{\sum_i e_i} = \gamma_1(i) \qquad (5.4.10)$$

These are recognised as the Baum–Welch re-estimation formulas.

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

$$\sum_i\pi_i = 1, \qquad (5.4.11)$$

$$\sum_j a_{ij} = 1, \qquad (5.4.12)$$

and

$$\sum_{k=1}^{L}b_j(k) = 1 \qquad (5.4.13)$$

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 $Pr(\mathbf{O}|\lambda)$ [4,8]. The maximisation of $\log Pr(\mathbf{O}|\lambda)$ can be solved by the classical Lagrange method. Let Q be the Lagrangian of $\log Pr(boldO|\lambda)$ with respect to the constraints Eq. (5.4.12).

$$Q = \log Pr(\mathbf{O}|\lambda)+\sum_i\kappa_i(\sum_j a_{ij}-1) \qquad (5.4.14)$$

where the κ_i are the as yet undetermined Lagrange multipliers. At a critical point of $\log Pr(\mathbf{O}|\lambda)$, it will be the case that, for all i,j,

$$\frac{\partial Q}{\partial a_{ij}} = \frac{1}{Pr(\mathbf{O}|\lambda)} \frac{\partial Pr(\mathbf{O}|\lambda)}{\partial a_{ij}} + \kappa_i = 0 \qquad (5.4.15)$$

Multiplying Eq. (5.4.15) by a_{ij} and summing over j

$$\sum_j a_{ij} \frac{1}{Pr(\mathbf{O}|\lambda)} \frac{\partial Pr(\mathbf{O}|\lambda)}{\partial a_{ij}} = -(\sum_j a_{ij})\kappa_i =$$
$$= -\kappa_i = \frac{1}{Pr(\mathbf{O}|\lambda)} \frac{\partial Pr(\mathbf{O}|\lambda)}{\partial a_{ij}} \qquad (5.4.16)$$

From Eq. (5.4.16) it may be seen that $\log Pr(\mathbf{O}|\lambda)$ is maximised when

$$a_{ij} = \frac{a_{ij} \frac{\partial Pr(\mathbf{O}|\lambda)}{\partial a_{ij}} \frac{1}{Pr(\mathbf{O}|\lambda)}}{\sum_k a_{ik} \frac{\partial Pr(\mathbf{O}|\lambda)}{\partial a_{ik}} \frac{1}{Pr(\mathbf{O}|\lambda)}} \qquad (5.4.17)$$

A similar argument can be made for the π and B parameters. While it is true that solving (5.4.17) for a_{ij} is analytically intractable, it can be used to provide some useful insights into the Baum−Welch re-estimation formulas and alternatives to them for solving the training problem. Since $Pr(\mathbf{O}|\lambda)$ can be expressed as:

$$Pr(\mathbf{O}|\lambda) = \sum_i \sum_j \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j) \qquad (5.4.18)$$

by differentiating Eq. (5.4.18) (note that since α_t and β_t also contain a_{ij}, the formula for differentiating a product must be used here), $\partial Pr(\mathbf{O}|\lambda)/\partial a_{ij}$ is:

$$\frac{\partial Pr(\mathbf{O}|\lambda)}{\partial a_{ij}} = \sum_{t=1}^{T-1} \alpha_t(i) b_j(O_{t+1}) \beta_{t+1}(j) \qquad (5.4.19)$$

Using Eq. (5.4.19) to substitute for $\partial Pr(\mathbf{O}|\lambda)/\partial a_{ij}$ in Eq. (5.4.17) gives:

$$a_{ij} = \frac{\frac{1}{Pr(\mathbf{O}|\lambda)} \sum_{t=1}^{T-1} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\frac{1}{Pr(\mathbf{O}|\lambda)} \sum_j \sum_{t=1}^{T-1} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)} \qquad (5.4.20)$$

If the left-hand side is considered as the re-estimate and the right-hand side is computed using the current values of the variables, this leads to the Baum−Welch re-estimation algorithm. Similarly, if π_i and b_{jk} are differentiated, then

$$\frac{\partial Pr(\mathbf{O}|\lambda)}{\partial \pi_i} = \sum_j b_i(O_1) a_{ij} b_j(O_2) \beta_2(j) = b_i(O_1) \beta_1(i) \qquad (5.4.21)$$

and

$$\frac{\partial Pr(\mathbf{O}|\lambda)}{\partial b_{jk}} = \sum_{t \in O_t = v_k} \sum_i \alpha_t(i) a_{ij} \beta_{t+1}(j) + \delta(O_1, v_k) \pi_j \beta_1(j) \qquad (5.4.22)$$

can be obtained respectively. In Eq. (5.4.22), δ denotes the Kronecker δ function. By substituting Eq. (5.4.21) and (5.4.22) into their respective analogues of Eq. (5.4.17), the re-estimation can be obtained respectively.

In general, let Λ be a parameter space. A transformation T of the parameter space onto itself, i.e. $T{:}\Lambda \rightarrow \Lambda$, is defined as:

$$T(x)_{ij} = \frac{x_{ij} \frac{\partial P}{\partial x_{ij}} \frac{1}{P}}{\sum_k x_{ik} \frac{\partial P}{\partial x_{ik}} \frac{1}{P}} \qquad (5.4.23)$$

where $T(x)_{ij}$ denotes the i,jth coordinate of the image of $\mathbf{x}$ under T. The parameter space is restricted to be the manifold such that $x_{ij} \geq 0$ for all i,j and $\sum_j x_{ij} = 1$ for all j.

Thus the re-estimation algorithm discussed above is a special case of the transformation Eq. (5.4.23) with P a particular homogeneous polynomial in values x_{ij} having positive coefficients. Here the x_{ij} include (A,B,π) of the HMM. Baum and Eagon [1] have shown that for any such polynomial $P[T(\mathbf{x})] > P(\mathbf{x})$ except if $\mathbf{x}$ is a critical point of P. Thus the transformation T is appropriately called a growth transformation. The conditions under which T is a growth transformation were relaxed later to include all polynomials with positive coefficients [2], and $P[\eta T(\mathbf{x}) + (1-\eta)\mathbf{x}] \geq P(\mathbf{x})$ for $0 \leq \eta \leq 1$ was also proved.

For multiple observation sequences, the Q-function can be defined as the summation of each Q-function corresponding to the nth observation sequence $\mathbf{O}^n$. Following Theorem 5.4.1, it can be seen that maximisation of the Q-function will lead to maximisation of $Pr(\mathbf{O}^M|\lambda)$. From Eq. (5.4.4), it can be easily verified that re-estimation formulas Eq. (5.3.17) and (5.3.18) stand.

5.5. 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 modelled effectively. The forward–backward algorithm and Baum–Welch algorithm can be used to optimise automatically 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 the re-estimation algorithm is based on constructing an information-theoretic Q-function, which is actually identical to the EM algorithm discussed in chapter 4 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.

References

1. 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.
2. L.E. Baum and G.R. Sell, "Growth transformations for functions on manifolds," *Pac. J. Math.*, vol. 27, pp. 211-227, 1968.
3. L.E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains," *Ann. Math. Stat.*, vol. 41, pp. 164-171, 1970.
4. 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.
5. P.F. Brown, "Acoustic-phonetic modeling problem in automatic speech recognition," Ph.D. thesis, Department of Computer Science, Carnegie-Mellon University, 1987.
6. S. Kullback and R.A. Leibler, "On information and sufficiency," *Ann. Math. Stat*, vol. 22, pp. 79-86, 1951.
7. K.F. Lee, "Large-vocabulary speaker-independent continuous speech recognition: The SPHINX system," Ph.D. thesis, Department of Computer Science, Carnegie-Mellon University, 1988; also Automatic Speech Recognition: The Development of the SPHINX System, Kluwer Academic Publishers, 1989.
8. L.R. Liporace, "Maximum likelihood estimation for multivariate observations of Markov sources," *IEEE Trans. Information theory*, vol. IT-28, pp. 729-734, 1982.
9. 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.
10. T Petrie, "Probabilistic functions of finite state Markov chains," *Ann. Math. Stat.*, vol. 40, pp. 97-115, 1969.
11. A.B. Poritz, "Hidden Markov models: a guided tour," *Proc. ICASSP-88*, pp. 7-13, New York, USA, 1988.

12. L.R. Rabiner and B.H. Juang, "An introduction to hidden Markov models," *IEEE ASSP Magazine*, pp. 4-16, Jan. 1986.

13. A.J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. formation Theory*, vol. IT-13, pp. 260-269, 1967.

14. C. Wellekens, "Explicit time correlation in hidden Markov models for speech recognition," *Proc. ICASSP-87*, pp. 384-386, Dallas, USA, 1987.

CHAPTER SIX

CONTINUOUS HIDDEN MARKOV MODELS

In this chapter we will discuss theories related to continuous mixture HMMs. If the observation does not come from a finite set, but from some set of 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 re-estimation algorithm discussed in Section 5.3.3 can be extended to estimate continuous probability density functions with the help of the Kullback–Leibler statistic, the Q-function [1,2]. As discussed in Chapter 5, the Q-function is one of the main mathematical foundations of the theory, in that the parameter re-estimates can be characterised as the critical point of it. Baum *et al.* [1,2] 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 [7] redefined the Q-function, and successfully relaxed the requirement so as to accommodate a broad class of elliptically symmetric density functions. Juang [6] further expanded the estimation algorithm to cope with finite mixtures of strictly log concave and/or elliptically symmetric density functions. In this chapter we will first review the Liporace proof to the general continuous parameter re-estimation formulas, and then discuss continuous mixture models, which are necessary to understand the unified modelling theory presented in the next chapter.

6.1. Continuous HMM

In this section we will first discuss general re-estimation 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.

6.1.1. General case

In a manner similar to the discrete HMM, for a given continuous observation sequence $\mathbf{X} \epsilon R^d$ 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

$$f(\mathbf{X}|\lambda) = \sum_{all\ S} f(\mathbf{X},S|\lambda)$$
$$= \sum_{S} \prod_{t=1}^{T} a_{s_{t-1}s_t} b_{s_t}(\mathbf{x}_t) \quad . \tag{6.1.1}$$

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

$$|\Sigma_i|^{-1/2} f_i(q_i(\mathbf{x})) \tag{6.1.3}$$

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

$$q_i(\mathbf{x}) = (\mathbf{x}-\mu_i)^t \Sigma_i^{-1} (\mathbf{x}-\mu_i). \tag{6.1.3}$$

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 the Euclidean d-space. Following Liporace's results [7], one extra assumption for elliptically symmetric densities is necessary: if the elliptically symmetric density $b(\mathbf{x})$ satisfies the consistency conditions of Kolmogorov [4], it can be represented as

$$b(\mathbf{x}) = \int_0^\infty N(\mathbf{x}, \mu, u^2\Sigma) dG(u) \tag{6.1.4}$$

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

$$f(\mathbf{X},S|\lambda) = \int_{\ddot{U}} \prod_{t=1}^{T} a_{s_{t-1}s_t} N(\mathbf{x}_t, \mu_{s_t}, u_t^2 \Sigma_{s_t}) dG(u_1) \cdots dG(u_T) \tag{6.1.5}$$
$$= E_U f(\mathbf{X},S,U|\lambda),$$

where $U = (u_1,\ldots,u_T)^t$, E_U denotes the average with respect to T-fold distribution $G(u_1) \cdots G(u_T)$ 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 re-estimation transformation is based on an auxiliary function $Q(\lambda,\bar{\lambda})$ of current parameters λ and new parameters $\bar{\lambda}$ defined by

$$Q(\lambda,\bar{\lambda}) = \frac{1}{f(\mathbf{X}|\lambda)} \sum_{S} [f(\mathbf{X},S|\lambda) \log f(\mathbf{X},S|\bar{\lambda})] \tag{6.1.6a}$$

$Q(\lambda,\bar{\lambda})$ 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. (6.1.6a) 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

$$Q(\lambda,\bar{\lambda}) = \frac{1}{f(\mathbf{X}|\lambda)}\sum_{S} E_U[f(\mathbf{X},S,U|\lambda)\log f(\mathbf{X},S,U|\bar{\lambda})] \quad (6.1.6b)$$

The following discussion will be based on Eq. (6.1.6b); it can be simplified to Eq. (6.1.6a) which is only a special case of Eq. (6.1.6b).

The utility of $Q(\lambda,\bar{\lambda})$ is similar to that discussed in Theorem 5.4.1, i.e.

$$Q(\lambda,\bar{\lambda}) \geq Q(\lambda,\lambda) \Rightarrow f(\mathbf{X}|\bar{\lambda}) \geq f(\mathbf{X}|\lambda). \quad (6.1.7)$$

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

Theorem 6.1.1.
If for each state j,

(i) $b_j(\mathbf{x})$ has the representation as Eq. (6.1.4); 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 [7].

Theorem 6.1.2.
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 $\nabla f(\mathbf{X}|\lambda) = 0$.

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

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

Based on Theorems 6.1.1 and 6.1.2, the re-estimation 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

$$\begin{aligned}
Q(\lambda,\bar{\lambda}) = \frac{1}{f(\mathbf{X}|\lambda)}\sum_{S} E_U\ f(\mathbf{X},S,U|\lambda)\sum_{t=1}^{T}[&\log\bar{a}_{s_{t-1}s_t}\\
&+ (1/2)\log|\bar{\Sigma}_{s_t}^{-1}| - d\log u_t - (d/2)\log(2\pi) \quad (6.1.8)\\
&- (1/2u_t^2)(\mathbf{x}_t-\bar{\mu}_{s_t})^t\bar{\Sigma}_{s_t}^{-1}(\mathbf{x}_t-\bar{\mu}_{s_t})]
\end{aligned}$$

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

$$0 = \sum_{t=1}^{T} \sum_{S \in S_j(t)} E_U(f(\mathbf{X},S,U|\lambda)/u_t^2)\bar{\Sigma}_j^{-1}(\mathbf{x}_t - \bar{\mu}_j) \qquad (6.1.9)$$

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

$$\int_0^\infty u_t^{-2} N(\mathbf{x}_t,\mu_j,u_t^2\Sigma_j) dG(u_t) \qquad (6.1.10)$$

According to Eq. (6.1.4), Eq. (6.1.10) is equal to $-2\partial b_j(\mathbf{x})/\partial q_j(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_t}$. 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

$$\rho_t(j) = \sum_i \alpha_{t-1}(i) a_{ij} [-2 \frac{\partial b_j(\mathbf{x})}{\partial q_j(\mathbf{x})}|_{\mathbf{x}=\mathbf{x}_t}] \qquad (6.1.11)$$

Thus

$$\bar{\mu}_j = \frac{\sum_{t=1}^{T} \rho_t(j)\beta_t(j)\mathbf{x}_t}{\sum_{t=1}^{T} \rho_t(j)\beta_t(j)} \qquad (6.1.12)$$

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

$$\bar{\Sigma}_j = \frac{\sum_{t=1}^{T} \rho_t(j)\beta_t(j)(\mathbf{x}_t - \bar{\mu}_j)(\mathbf{x}_t - \bar{\mu}_j)^t}{\sum_{t=1}^{T} \rho_t(j)\beta_t(j)} \qquad (6.1.13)$$

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 6.1.1.

6.1.2. Gaussian density function

When the above re-estimation formulas are applied to the multivariate Gaussian densities, $\rho_t(j) = \alpha_t(j)$. Therefore, Eq. (6.1.12) and (6.1.13) become

$$\bar{\mu}_j = \frac{\sum_{t=1}^{T} \gamma_t(j)\mathbf{x}_t}{\sum_{t=1}^{T} \gamma_t(j)} \qquad (6.1.14)$$

and

$$\bar{\Sigma}_j = \frac{\sum_{t=1}^{T} \gamma_t(j)(\mathbf{x}_t - \bar{\mu}_j)(\mathbf{x}_t - \bar{\mu}_j)^t}{\sum_{t=1}^{T} \gamma_t(j)} \tag{6.1.15}$$

where $\gamma_t(j)$ is the *a posteriori* probability which has the same meaning as used for the discrete HMM. The interpretation of Eq. (6.1.14) and Eq. (6.1.15) 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. (5.3.17) and Eq. (5.3.18), summation with respect to variables of multiple independent observations can be applied to the denominator and numerators respectively. Notice that Eq. (6.1.15) relies on the re-estimated mean vector $\bar{\mu}_j$, which is inconvenient to implement. Eq. (6.1.15) can also be expressed as

$$\bar{\Sigma}_j = \frac{\sum_{t=1}^{T} \gamma_t(j)\mathbf{x}_t\mathbf{x}_t^t}{\sum_{t=1}^{T} \gamma_t(j)} - \bar{\mu}_j\bar{\mu}_j^t \tag{6.1.16}$$

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 re-estimation equation can be written as [6]

$$\bar{\Sigma}_j = \frac{\sum_{t=1}^{T} \gamma_t(j)(\mathbf{x}_t - \mu_j)(\mathbf{x}_t - \mu_j)^t}{\sum_{t=1}^{T} \gamma_t(j)} \tag{6.1.17}$$

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

Referring to Eq. (4.2.11) and (4.2.12), the mixture densities problem discussed in Chapter 4 can be considered as a special case of this model in which the state transitions are Markov of order zero with L states ($N = L$),

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

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

6.2. 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 form 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 are 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 [3]. In contrast 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, the 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 [5], as well as the Parzen estimation of the probability density functions [8] have also been reported in the literature. These are not considered further here becáuse 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 reduce drastically 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 degrade recognition accuracy significantly. 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 although 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, $1 \le j \le N$, if chosen as for Gaussian mixture densities, can be written as:

$$\begin{aligned} b_j(\mathbf{x}) &= \sum_{k=1}^{M} c_{jk} b_{jk}(\mathbf{x}) \\ &= \sum_{k=1}^{M} c_{jk} N(\mathbf{x},\mu_{jk},\Sigma_{jk}) \end{aligned} \tag{6.2.1}$$

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

$$\sum_{k=1}^{M} c_{jk} = 1 \tag{6.2.2}$$

so that

$$\int_{-\infty}^{\infty} b_j(\mathbf{x}) d\mathbf{x} = 1. \tag{6.2.3}$$

6.3. Continuous Mixture HMM

When $b_j(\mathbf{x})$ is represented by a mixture of densities as Eq. (6.2.1), the summand in Eq. (6.1.1) over all S is, in fact, the joint density $f(\mathbf{X},S|\lambda)$, which can be expressed as

$$f(\mathbf{X},S|\lambda) = \prod_{t=1}^{T} a_{s_{t-1}s_t} b_{s_t}(\mathbf{x}_t)$$
$$= \sum_{k_1=1}^{M}\sum_{k_2=1}^{M} \cdots \sum_{k_T=1}^{M} \qquad (6.3.1)$$
$$[\prod_{t=1}^{T} a_{s_{t-1}s_t} b_{s_t k_t}(\mathbf{x}_t)] c_{s_1k_1} c_{s_2k_2} \cdots c_{s_Tk_T}$$

Let Ω^T be the Tth Cartesian product of $\Omega = \{1,2,...,M\}$, which is defined as the branch alphabet. In the summand of Eq. (6.3.1), it can be considered as the summation of:

$$f(\mathbf{X},S,K|\lambda) = \prod_{t=1}^{T} a_{s_{t-1}s_t} b_{s_t k_t}(\mathbf{x}_t) c_{s_t k_t} \qquad (6.3.2)$$

Therefore, the joint probability density of the truncated stochastic process **X** is

$$f(\mathbf{X}|\lambda) = \sum_{S}\sum_{K\in\Omega^T} f(\mathbf{X},S,K|\lambda) \qquad (6.3.3)$$

Eq. (6.3.3) can be interpreted such that there are N^T possible stochastic state sequences that may lead to the observation **X**, with each possible state sequence being a superposition of M^T branch layers.

Similar to Eq. (6.1.6), an auxiliary function $Q(\lambda,\bar{\lambda})$ of two model points, λ and $\bar{\lambda}$, given an observation **X** can be written as:

$$Q(\lambda,\bar{\lambda}) = \sum_{S}\sum_{K\in\Omega^T} \frac{f(\mathbf{X},S,K|\lambda)}{f(\mathbf{X}|\lambda)} \log f(\mathbf{X},S,K|\bar{\lambda}) \qquad (6.3.4)$$

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.(6.3.2), the following decomposition can be shown:

$$\log f(\mathbf{X},S,K|\bar{\lambda})$$
$$= \sum_{t=1}^{T}\log\bar{a}_{s_{t-1}s_t} + \sum_{t=1}^{T}\log\bar{b}_{s_t k_t}(\mathbf{x}_t) + \sum_{t=1}^{T}\log\bar{c}_{s_t k_t} \qquad (6.3.5)$$
$$= \log\bar{\pi}_{s_1} + \sum_{t=1}^{T-1}\log\bar{a}_{s_t s_{t+1}} + \sum_{t=1}^{T}\log\bar{b}_{s_t k_t}(\mathbf{x}_t) + \sum_{t=1}^{T}\log\bar{c}_{s_t k_t}$$

Maximisation of the likelihood by way of re-estimation can be accomplished on individual parameter sets owing to the separability shown in Eq. (6.3.5). The separation of Eq. (6.3.5) is the key to the increased versatility of a re-estimation algorithm in accommodating mixture observation densities. The auxiliary function can be rewritten in a separated form:

$$Q(\lambda,\bar{\lambda}) = \sum_{S}\sum_{K} \frac{f(\mathbf{X},S,K|\lambda)}{f(\mathbf{X}|\lambda)} \log f(\mathbf{X},S,K|\bar{\lambda})$$
$$= \sum_{S}\sum_{K} \frac{f(\mathbf{X},S,K|\lambda)}{f(\mathbf{X}|\lambda)} [\log\bar{\pi}_{s_1} + \sum_{t=1}^{T-1}\log\bar{a}_{s_t s_{t+1}}$$
$$+ \sum_{t=1}^{T}\log\bar{b}_{s_t k_t}(\mathbf{x}_t) + \sum_{t=1}^{T}\log\bar{c}_{s_t k_t}] \qquad (6.3.6)$$
$$= Q_{\pi}(\lambda,\bar{\pi}) + \sum_{i} Q_{a_i}(\lambda,\bar{a}_{ij}) + \sum_{j}\sum_{k=1}^{M} Q_{b_j}(\lambda,\bar{\mathbf{b}}_{jk})$$
$$+ \sum_{j} Q_{c_j}(\lambda,\bar{c}_{jk}),$$

where

$$Q_{\pi}(\lambda,\bar{\pi}) = \sum_{S}\sum_{K} f(S,K|\mathbf{X},\lambda)\log\bar{\pi}_{s_1}$$
$$= \sum_{i}\sum_{K} f(s_1 = i,\ K|\mathbf{X},\lambda)\log\bar{\pi}_i \qquad (6.3.7)$$

$$Q_{a_i}(\lambda,\bar{a}_{ij}) = \sum_{j}\sum_{t=1}^{T-1}\sum_{K} f(s_t = i, s_{t+1} = j, K|\mathbf{X},\lambda)\log\bar{a}_{ij} \qquad (6.3.8)$$

$$Q_{b_j}(\lambda,\bar{\mathbf{b}}_{jk})=\sum_{t=1}^{T} f(s_t=j,k_t=k|\mathbf{X},\lambda)\log\bar{b}_{jk}(\mathbf{x}_t), \tag{6.3.9}$$

and

$$Q_{c_j}(\lambda,\bar{c}_{jk})=\sum_{k=1}^{M}\sum_{t=1}^{T} f(s_t=j,k_t=k|\mathbf{X},\lambda)\log\bar{c}_{jk} \tag{6.3.10}$$

Formally, individual maximisation of Q_π, Q_{a_i} (for all i), Q_{c_j} (for all j) subject to the respective stochastic constraints 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 the discussion in Section 6.1, 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 re-estimates 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:

$$\bar{\pi}_i = \frac{\sum_K \frac{f(\mathbf{X},s_1=i,K|\lambda)}{f(\mathbf{X}|\lambda)}}{\sum_K \frac{f(\mathbf{X},K|\lambda)}{f(\mathbf{X}|\lambda)}} = \frac{f(\mathbf{X},s_1=i|\lambda)}{f(\mathbf{X}|\lambda)} \tag{6.3.11}$$

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1}\sum_K \frac{f(\mathbf{X},s_t=i,s_{t+1}=j,K|\lambda)}{f(\mathbf{X}|\lambda)}}{\sum_{t=1}^{T-1}\sum_K \frac{f(\mathbf{X},s_t=i,K|\lambda)}{f(\mathbf{X}|\lambda)}} = \frac{\sum_{t=1}^{T-1} \frac{f(\mathbf{X},s_t=i,s_{t+1}=j|\lambda)}{f(\mathbf{X}|\lambda)}}{\sum_{t=1}^{T-1} \frac{f(\mathbf{X},s_t=i|\lambda)}{f(\mathbf{X}|\lambda)}} \tag{6.3.12}$$

$$c_{jk} = \frac{\sum_{t=1}^{T} \frac{f(\mathbf{X},s_t=j,k_t=k|\lambda)}{f(\mathbf{X}|\lambda)}}{\sum_{t=1}^{T} \frac{f(\mathbf{X},s_t=j|\lambda)}{f(\mathbf{X}|\lambda)}} \tag{6.3.13}$$

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

$$\begin{aligned}\gamma_t(i,j) &= f(s_t=i,s_{t+1}=j|\mathbf{X},\lambda)\\ &= \frac{f(\mathbf{X},s_t=i,s_{t+1}=j|\lambda)}{f(\mathbf{X}|\lambda)}\\ &= \frac{\alpha_t(i)a_{ij}[\sum_{k=1}^{M} c_{jk}b_{jk}(\mathbf{x}_{t+1})]\beta_{t+1}(j)}{\sum_{k\in S_F}\alpha_T(k)} \quad \text{for } 1\le t\le T-1\end{aligned} \tag{6.3.14}$$

$$\begin{aligned}\gamma_t(i) &= f(s_t=i|\mathbf{X},\lambda)\\ &= \frac{f(\mathbf{X},s_t=i|\lambda)}{f(\mathbf{X}|\lambda)}\\ &= \frac{\alpha_t(i)\beta_t(i)}{\sum_{k\in S_F}\alpha_T(k)} \quad \text{for } 1\le t\le T\end{aligned} \tag{6.3.15}$$

and

$$\begin{aligned}\zeta_t(j,k) &= f(s_t=j,k_t=k\,|\,\mathbf{X},\lambda)\\ &= \frac{f(\mathbf{X},s_t=j,k_t=k\,|\,\lambda)}{f(\mathbf{X}|\lambda)}\\ &= \frac{\sum_i \alpha_{t-1}(i)a_{ij}c_{jk}b_{jk}(\mathbf{x}_t)\beta_t(j)}{\sum_{i\in S_F}\alpha_T(i)} \quad \text{for } 1<t\le T \end{aligned} \tag{6.3.16}$$

As a result, the new estimates of initial probabilities, $\overline{\pi}_i$, and transition probabilities, $\overline{a}_{ij}$, can be expressed in a similar way to Eq. (5.3.15) and (5.3.13). The only difference from Eq.(5.3.15) and (5.3.13) is that $b_j(O_{t+1})$ in Eq. (5.3.15) and (5.3.13) is replaced by the continuous mixture probability density function of observation vector $\mathbf{x}$. Similarly, for the weighting coefficients, c_{jk} can be replaced by:

$$\overline{c}_{jk} = \frac{\sum_{t=1}^{T}\zeta_t(j,k)}{\sum_{t=1}^{T}\gamma_t(j)} \tag{6.3.17}$$

Maximisation of $Q_{b_j}(\lambda,\overline{\mathbf{b}}_{jk})$ with respect to $\overline{\mathbf{b}}_{jk}$ 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 $\{\overline{\mathbf{b}}_{jk}\}$ that satisfies:

$$\begin{aligned}&\nabla_{\overline{\mathbf{b}}_{jk}}Q_{b_j}(\lambda,\overline{\mathbf{b}}_{jk})\\ &= \sum_{t=1}^{T} f(s_t=j,k_t=k\,|\,\mathbf{X},\lambda)\frac{\nabla_{\overline{\mathbf{b}}_{jk}}\overline{b}_{jk}(\mathbf{x}_t)}{\overline{b}_{jk}(\mathbf{x}_t)} = 0\end{aligned} \tag{6.3.18}$$

where $\nabla_{\overline{\mathbf{b}}_{jk}}\overline{b}_{jk}(\mathbf{x}_t)$ denotes derivatives with respect to parameters of $\overline{b}_{jk}$. Thus, for the Gaussian mixture density functions represented in Eq. (6.2.1), the partial derivatives are with respect to $\overline{\mu}_{jk}$ and $\overline{\Sigma}_{jk}^{-1}$, and they are:

$$\begin{aligned}\frac{\partial\overline{b}_{jk}(\mathbf{x})}{\partial\overline{\mu}_{jk}} &= N(\mathbf{x},\overline{\mu}_{jk},\overline{\Sigma}_{jk})\overline{\Sigma}_{jk}^{-1}(\mathbf{x}-\overline{\mu}_{jk})\\ \frac{\partial\overline{b}_{jk}(\mathbf{x})}{\partial\overline{\Sigma}_{jk}^{-1}} &= \frac{1}{2}N(\mathbf{x},\overline{\mu}_{jk},\overline{\Sigma}_{jk})[\overline{\Sigma}_{jk}-(\mathbf{x}-\overline{\mu}_{jk})(\mathbf{x}-\overline{\mu}_{jk})^t]\end{aligned} \tag{6.3.19}$$

Substituting Eq. (6.3.19) in Eq. (6.3.18), it can be seen that the solutions to Eq. (6.3.19), i.e. re-estimates $\overline{\mu}_{jk}$ and $\overline{\Sigma}_{jk}$, can be given by

$$\begin{aligned}\overline{\mu}_{jk} &= \frac{\sum_{t=1}^{T}\frac{f(\mathbf{X},s_t=j,k_t=k|\lambda)}{f(\mathbf{X}|\lambda)}\mathbf{x}_t}{\sum_{t=1}^{T}\frac{f(\mathbf{X},s_t=j,k_t=k|\lambda)}{f(\mathbf{X}|\lambda)}}\\ &= \frac{\sum_{t=1}^{T}\zeta_t(j,k)\mathbf{x}_t}{\sum_{t=1}^{T}\zeta_t(j,k)}\end{aligned} \tag{6.3.20}$$

$$\begin{aligned}\overline{\Sigma}_{jk} &= \frac{\sum_{t=1}^{T}\frac{f(\mathbf{X},s_t=j,k_t=k|\lambda)}{f(\mathbf{X}|\lambda)}(\mathbf{x}_t-\overline{\mu}_{jk})(\mathbf{x}_t-\overline{\mu}_{jk})^t}{\sum_{t=1}^{T}\frac{f(\mathbf{X},s_t=j,k_t=k|\lambda)}{f(\mathbf{X}|\lambda)}}\\ &= \frac{\sum_{t=1}^{T}\zeta_t(j,k)(\mathbf{x}_t-\overline{\mu}_{jk})(\mathbf{x}_t-\overline{\mu}_{jk})^t}{\sum_{t=1}^{T}\zeta_t(j,k)}\end{aligned} \tag{6.3.21}$$

The interpretation of Eq. (6.3.17), (6.3.20) and (6.3.21) is similar to that of the discrete HMM. The re-estimation 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 re-estimation of $\overline{\mu}_{jk}$ 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. (6.3.20) and Eq. (6.3.21) have identical forms to Eq. (4.2.11) and Eq. (4.2.12) respectively except that the Markov property is imposed on the posterior probability here.

6.4. Summary

The original Q-function of Baum *et al.* provides a proof for 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 the most widely used density functions such as Gaussian. Without loss of generality, the finite mixture continuous HMM has 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 the 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 compared with the discrete HMM. A better usage of the continuous mixture HMM will be discussed in the following chapters.

References

1. L.E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains," *Ann. Math. Stat.*, vol. 41, pp. 164-171, 1970.
2. 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.
3. G.R. Doddington, "Phonetically sensitive discriminants for improved speech recognition," *Proc. ICASSP-89*, pp. 556-559, Glasgow, Scotland, 1989.
4. J. Doob, *Stochastic Processes*, p. 10, Jhon Wiley, 1953.
5. 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, pp. 539-542, Florence, Italy, 1987.
6. B.H. Juang, "Maximum-likelihood estimation for mixture multivariate stochastic observations of Markov chain," *AT&T Technical Journal*, vol. 64, pp. 1235-1249, 1985.
7. L.R. Liporace, "Maximum likelihood estimation for multivariate observations of Markov sources," *IEEE Trans. Information theory*, vol. IT-28, pp. 729-734, 1982.
8. S. Soudoplatoff, "Markov modeling of continuous parameters in speech recognition," *Proc. ICASSP-86*, pp. 45-48, Tokyo, Japan, 1986.

CHAPTER SEVEN

UNIFIED THEORY: SEMI-CONTINUOUS HIDDEN MARKOV MODELS

Vector quantisation, the discrete HMM, and the continuous mixture HMM have been introduced in previous chapters, where the EM algorithm plays a key role in their development. In practice, 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. Like the continuous mixture HMM, the semi-continuous HMM also has 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. Compared with 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 in conventional VQ. In practice the speech recognition accuracy of the semi-continuous HMM is better than 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. The Q-function, once again, serves here as the major mathematical underpinning. It will be seen that all these theories can be developed based on the general Q-function with various simplifications.

7.1. 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 [9,13,14,16,21,23]. A distinctive technique is hidden Markov modelling based on multiple VQ codebooks, which has been shown to offer improved speech recognition accuracy [6,14]. 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 [12]. 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 compared with the discrete HMM [4,18,19]. For speaker-independent speech recognition, a mixture of a large number of probability density functions [17,20] or a large number of states in the single-mixture case [5] is generally required to model the characteristics of different speakers. In addition, it has been observed that some difficulties due to modelling assumptions that introduce singularities may arise in the continuous single mixture HMM [15], 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 [4,19]. 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 [4] has been shown to improve significantly 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 functions 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.

7.2. Semi-Continuous HMM

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 partitioning 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.

7.2.1. Basic principles

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:

$$
\begin{aligned}
b_{s_t}(\mathbf{x}) &= f(\mathbf{x}|s_t) \\
&= \sum_{j=1}^{L} f(\mathbf{x}|v_j,s_t)Pr(v_j|s_t) \qquad (7.2.1)
\end{aligned}
$$

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. (7.2.1) can be written as:

$$
\begin{aligned}
b_i(\mathbf{x}) &= \sum_{j=1}^{L} f(\mathbf{x}|v_j)Pr(v_j|s_t=i) \\
&= \sum_{j=1}^{L} f(\mathbf{x}|v_j)b_i(j) \qquad (7.2.2)
\end{aligned}
$$

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. (7.2.2) works in a similar way to other non-parametric or heuristic methods, such as the HMM based on the Parzen estimator [22], the fuzzy VQ [23], and the multi-labelling VQ [16]. 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 is depicted in Figure 7.2.1. 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 $\mathbf{x}$. 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 $(1 \le j \le L)$. These codebook density functions are subsequently used by the semi-continuous output probability (Eq. (7.2.2)). The structure of the semi-continuous HMM can be exactly the same as the discrete HMM. However, the output probabilities 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. (7.2.2) 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 re-estimate 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. This is exactly the same as the tied mixture continuous HMMs used in some speech recognition systems [3].

Compared with 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 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. Compared with 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-labelling 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 in conventional VQ, which cannot be obtained from fuzzy VQ or multi-labelling VQ.

In practice, Eq. (7.2.2) can be simplified by using the 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 keeping 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. (7.2.2) can be rewritten as

$$b_i(\mathbf{x}) = \sum_{v_j \in \eta(\mathbf{x})} f(\mathbf{x}|v_j) b_i(j) \tag{7.2.3}$$

Since the number of VQ codewords in $\eta(\mathbf{x})$ is of lower order than the VQ level, Eq. (7.2.3) can significantly reduce the amount of computational load for subsequent modelling compared with Eq. (7.2.2). 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. (7.2.3) 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

$$b_i(j) = \begin{cases} 1 & \text{if codeword } j \in \text{state } i \\ 0 & \text{otherwise} \end{cases} \tag{7.2.4}$$

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-

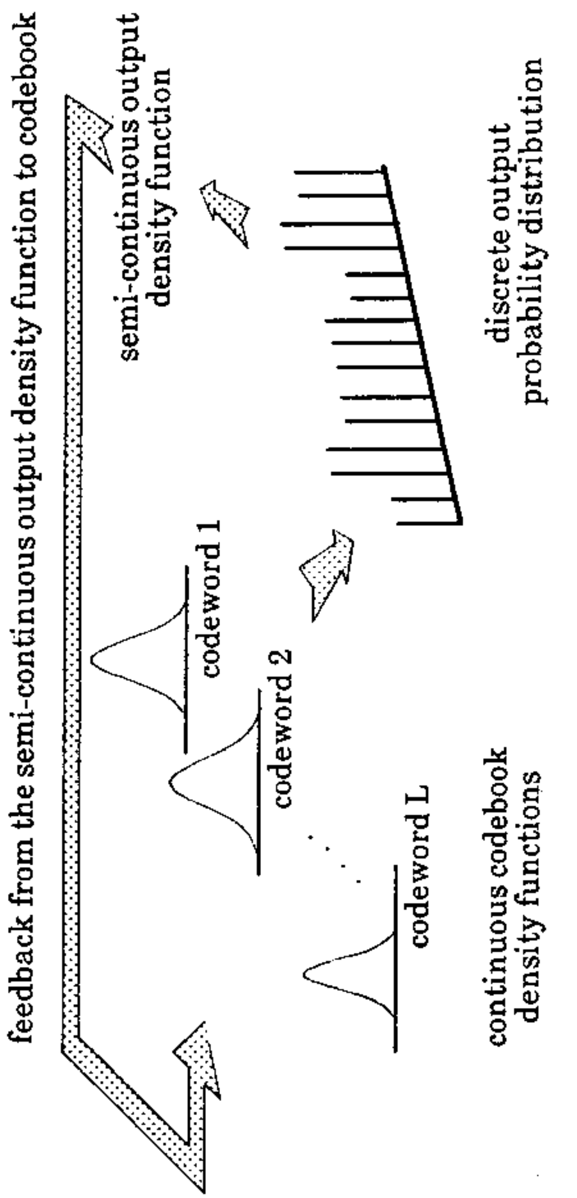
feedback from the semi-continuous output density function to codebook
semi-continuous output density function
codeword 1
codeword 2
codeword L
discrete output probability distribution
continuous codebook density functions

Figure 7.2.1. Relationship between the semi-continuous HMM and codebook

mixture continuous hidden Markov modelling. Of course, Eq. (7.2.4) can be modified to let each state contain several disjoint codewords extending to the case of the continuous mixture HMM. From the 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 in a similar manner to those described in Chapter 6), 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 4;

(2) sample estimates of the covariance matrices based on the conventional VQ codebook [9];

(3) Gaussian clustering techniques [7];

(4) feedback from semi-continuous hidden Markov modelling, variants including collection of output density functions in the continuous HMM to form a codebook, and simplified feedback techniques [8,10].

As pointed out by Bahl, Jelinek, and Mercer [2],

> *It is possible to allow feedback from the decoder to the acoustic processor but the mathematical consequences of such a step are not well understood.*

The unified modelling approach can be viewed as preliminary attempts to describe such a feedback. Feedback from semi-continuous hidden Markov modelling to VQ will be discussed in the following sections.

7.2.2. Re-estimation formulas

Feedback from hidden Markov modelling to the VQ codebook is a re-estimation 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 owing to its advantages. Other probability density functions can be applied in principle in a similar way to those described in Chapter 6.

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 re-estimation algorithm for the weighting coefficients (as discussed in Chapter 6) can be applied to re-estimate $b_i(k)$. In a manner similar to the conventional HMM, re-estimation 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. (7.2.2) or Eq. (7.2.3) is used for the output probabilities. Further to the forward and backward probabilities, the intermediate probabilities, $\chi_t(i,j,k)$, $\gamma_t(i,j)$, $\gamma_t(i)$, $\zeta_t(i,k)$, and $\zeta_t(k)$ can be defined as follows for efficient re-estimation of the model parameters:

$$\chi_t(i,j,k) = f(s_t=i,\ s_{t+1}=j,\ \mathbf{x}_{t+1}\sim v_k|\mathbf{X},\lambda) = \frac{\alpha_t(i)a_{ij}b_j(k)f(\mathbf{x}_{t+1}|v_k)\beta_{t+1}(j)}{f(\mathbf{X}|\lambda)}, \quad 1\le t\le T-1 \tag{7.2.5}$$

$$\gamma_t(i,j) = f(s_t=i,\ s_{t+1}=j|\mathbf{X},\lambda), \quad 1\le t\le T-1 \tag{7.2.6}$$

$$\gamma_t(i) = f(s_t=i|\mathbf{X},\lambda) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{k\in S_F}\alpha_T(k)}, \quad 1\le t\le T \tag{7.2.7}$$

$$\zeta_t(i,k) = f(s_t=i,\mathbf{x}_t\sim v_k|\mathbf{X},\lambda), \quad 1\le t\le T \tag{7.2.8}$$

$$\zeta_t(k) = f(\mathbf{x}_t\sim v_k|\mathbf{X},\lambda), \quad 1\le t\le T. \tag{7.2.9}$$

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 $\chi_t()$ since

$$\gamma_t(i,j) = \sum_{k=1}^{L} \chi_t(i,j,k), \quad 1 \le t \le T-1 \tag{7.2.10}$$

$$\gamma_t(i) = \sum_{j} \gamma_t(i,j), \quad 1 \le t \le T-1 \tag{7.2.11}$$

In Eq. (7.2.11), $\gamma_T(i)$ must be computed from Eq. (7.2.7).

$$\zeta_t(i,k) = \begin{cases} \displaystyle\sum_j \chi_{t-1}(j,i,k) & \text{if } 1 < t \le T \\ \dfrac{\pi_i b_i(k) f(\mathbf{x}_1|v_k)\beta_1(i)}{f(\mathbf{X}|\lambda)} & \text{if } t = 1 \end{cases} \tag{7.2.12}$$

$$\zeta_t(k) = \sum_i \zeta_t(i,k), \quad 1 \le t \le T \tag{7.3.13}$$

Using Eq. (7.2.10) to (7.3.13), re-estimation equations for $\bar{\pi}_i$, $\bar{a}_{ij}$, and $\bar{b}_i(k)$, can be derived (detailed proof will be presented in Section 7.5). In fact, the re-estimation formulas have the same form as the continuous mixture HMM since:

$$\bar{\pi}_i = \gamma_1(i); \tag{7.2.14}$$

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \gamma_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}; \tag{7.2.15}$$

$$\bar{b}_j(k) = \frac{\sum_{t=1}^{T} \zeta_t(j,k)}{\sum_{t=1}^{T} \gamma_t(j)}. \tag{7.2.16}$$

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 re-estimate the parameters of the VQ codebook, i.e. the mean vectors, μ_j, and covariance matrices, Σ_j, of the codeword v_j, they can be written as:

$$\bar{\mu}_j = \frac{\sum_{t=1}^{T} \zeta_t(j)\mathbf{x}_t}{\sum_{t=1}^{T} \zeta_t(j)}, \quad 1 \le j \le L; \tag{7.2.17}$$

$$\bar{\Sigma}_j = \frac{\sum_{t=1}^{T} \zeta_t(j)(\mathbf{x}_t - \bar{\mu}_j)(\mathbf{x}_t - \bar{\mu}_j)^t}{\sum_{t=1}^{T} \zeta_t(j)}, \quad 1 \le j \le L. \tag{7.2.18}$$

As discussed in Chapter 5, Eq. (7.2.14) to (7.2.18) can be extended to multiple independent observation sequences by summing numerators and denominators respectively corresponding to each observation sequence. Re-estimation formulas for transition and output probabilities are, here, formally, the same as for conventional ones. However, to re-estimate parameters of the VQ codebook, observations for different models will be used together since different models use the same VQ codebook. Thus, re-estimation of mean vectors and covariance matrices of different models will involve interdependencies. According to Eq. (7.2.17) and (7.2.18), if any observation $\mathbf{x}_t$ (no matter what model it is designated for) has a large value of *a posteriori* probability $\zeta_t(j)$, it will have a large contribution on re-estimation of parameters of codeword v_j. Different VQ density functions to be re-estimated in such a manner will be strongly correlated owing to large values of $\zeta_t(j)$. Let F denote the set of models to be re-estimated. There are multiple independent observation sequences for each model. In general, re-estimation formulas for the VQ codebook can be written as:

$$\bar{\mu}_j = \frac{\sum_{m \in F}[\sum_n \sum_{t=1}^{T} \zeta_t^{m,n}(j)\mathbf{x}_t^{m,n}]}{\sum_{m \in F}[\sum_n \sum_{t=1}^{T} \zeta_t^{m,n}(j)]}, \quad 1 \le j \le L; \tag{7.2.19}$$

and

$$\bar{\Sigma}_j = \frac{\sum_{m \in F}[\sum_n \sum_{t=1}^{T} \zeta_t^{m,n}(j)(\mathbf{x}_t^{m,n} - \bar{\mu}_j)(\mathbf{x}_t^{m,n} - \bar{\mu}_j)^t]}{\sum_{m \in F}[\sum_n \sum_{t=1}^{T} \zeta_t^{m,n}(j)]}, \tag{7.2.20}$$

$$1 \le j \le L.$$

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 $\zeta_t^{m,n}(j)$ is computed from observation sequence n of model m.

In Eq. (7.2.19) and (7.2.20), re-estimation formulas for the mean vectors and covariance matrices of the VQ codebook can be regarded as re-estimation 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. Comparison with Eq. (4.2.11) and (4.2.12) shows that the EM algorithm for the mixture density VQ codebook design is merely a special form of these re-estimation formulas. Here, a unified modelling approach to VQ and hidden Markov modelling of speech signals is established. In the EM algorithm, the *a posteriori* probability density is used as the weight in the re-estimation formulas of the VQ codebook. In the unified modelling approach, the weight is also the *a posteriori* 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. (7.2.3) is used as the semi-continuous output probability densities, the re-estimation computational complexity can also be significantly reduced. With such a simplification, Eq. (7.2.5) can be written as:

$$\chi_t(i,j,k) = \begin{cases} \dfrac{\alpha_t(i)a_{ij}b_j(k)f(\mathbf{x}_{t+1}|v_k)\beta_{t+1}(j)}{f(\mathbf{X}|\lambda)} & \text{if } v_k \in \eta(\mathbf{x}_t) \\ 0 & \text{otherwise} \end{cases} \tag{7.2.21}$$

Similarly, Eq. (7.2.6) to (7.2.9) can be treated in the same manner by using Eq. (7.2.10) to (7.2.13).

7.2.3. Semi-continuous decoder

Whether the models are re-estimated 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 semi-continuous output probability density function. For example, the Viterbi algorithm can be modified as follows:

Modified Viterbi algorithm

```
Step 1: 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 $t=2$ to $T$, for all states $j$,
        $\delta_t(j) = \mathrm{Max}_i[\delta_{t-1}(i)a_{ij}] \sum_{v_l \in \eta(\mathbf{x}_t)} [f(\mathbf{x}_t|v_l)b_j(v_l)]$
        $\Psi_t(j) = \mathrm{argmax}_i[\delta_{t-1}(i)a_{ij}]$
Step 3: Termination. (* indicates the optimised
```

results).

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

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

Step 4: Path (state sequence) backtracking. From time $T-1$ 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, $f(\mathbf{x}|v_j)$, is empirically required to be normalised within [0,1] to retain consistency with the discrete probabilities. The normalisation of $f(\mathbf{x}|v_j)$ can be achieved using

$$f(\mathbf{x}|v_j) \leftarrow \frac{f(\mathbf{x}|v_j)}{\sum_{k=1}^{L} f(\mathbf{x}|v_k)} \qquad (7.2.22)$$

or

$$f(\mathbf{x}|v_j) \leftarrow \frac{f(\mathbf{x}|v_j)}{\sum_{v_k \in \eta(\mathbf{x})} f(\mathbf{x}|v_k)} \qquad (7.2.23)$$

Experiments show that both Eq. (7.2.22) and Eq. (7.2.23) work well if discrete HMM parameters are used. However, if the model is re-estimated based on the semi-continuous HMM, such a normalisation is unnecessary since training and decoding can be done in a consistent way.

7.3. Proof of the Unified Re-estimation

The proof in the main mathematical underpinning of the unified modelling theory is to define the general Q-function [11]. To re-estimate 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 of being maximised in the unified modelling approach should then be the summation of each individual likelihood of the HMM:

$$\sum_{m \in F} \sum_{n} \log f(\mathbf{X}^{m,n}|\lambda^m) \qquad (7.3.1)$$

where $\mathbf{X}^{m,n}$ denotes the observation sequence n for model m; and 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 general Q-function can be defined as:

$$Q(\Lambda,\bar{\Lambda}) =$$

$$\sum_{m \in F} \sum_{n} \sum_{S} \sum_{V} \frac{f(\mathbf{X}^{m,n},S,\,V|\lambda^m)}{f(\mathbf{X}^{m,n}|\lambda^m)} \log f(\mathbf{X}^{m,n},S,\,V|\bar{\lambda}^m) \qquad (7.3.2)$$

where Λ and $\bar{\Lambda}$ denote $\{\lambda^m\}$ and $\{\bar{\lambda}^m\}$ respectively. Following the discussion in Chapters 5 and 6, it can be seen that maximisation of the general Q-function will lead to maximisation of the likelihood function defined in Eq. (7.3.1).

The Q-function defined in Eq. (7.3.2) can be well separated into Q_π^m, Q_a^m, Q_b^m, and Q_v in a manner similar to Eq. (6.3.6). 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 6. They can be maximised independently for each m. However, for re-estimation 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:

$$\zeta_t^{m,n}(j) = f(\mathbf{x}_t^{m,n} \sim v_j|\mathbf{X}^{m,n},\lambda^m) \qquad (7.3.3)$$

Q_v can be written as:

$$Q_v(\Lambda,\bar{v}_j) = \sum_{m\in F}\sum_{n}\sum_{t=1}^{T} f(\mathbf{x}_t^{m,n}\sim v_j|\mathbf{X}^{m,n},\lambda^m)\log f(\mathbf{x}_t^{m,n}|\bar{v}_j) \quad (7.3.4)$$

Notice the similarity between Eq. (7.3.4) and the Q-function defined for the continuous mixture HMM (Eq. (6.3.9)). If the VQ codeword density is assumed to be Gaussian, it is not difficult to extend the re-estimation formulas of Eq. (6.3.20) and (6.3.21) to Eq. (7.2.19) and (7.2.20). In general, a broad class of elliptically symmetric probability density functions can also be accommodated in a similar manner to that discussed in Chapter 6.

When the EM algorithm is applied to VQ, the discrete HMM, and the continuous mixture HMM, the Q-function has played a key role in the development of the theories. The Q-function defined above are a general form of those discussed in Chapters 4 to 6. For example, if hidden state information S and Markov properities are excluded, Eq. (7.3.2) becomes Eq. (4.2.3). Similarly, if hidden VQ codebook information V is excluded, Eq. (7.3.2) becomes the Q-function used in the discrete HMM. Compared with the Q-function used in the continuous mixture HMM, the difference is that Eq. (7.3.2) introduces output probability density tying. With various simplifications of the general Q-function, we can conveniently return to other modelling techniques.

7.4. Summary

Compared with 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-labelling VQ. Unlike fuzzy VQ or multi-labelling, 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. Several variants of unified modelling, such as feedback VQ [8] or supervised VQ [1], can be found in the literature. If the Viterbi algorithm is used in training instead of the forward−backward algorithm and the covariance matrices are not re-estimated, supervised VQ [1] can also be viewed as a special case of the semi-continuous HMM.

The semi-continuous HMM incorporates 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. 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 the computational complexity are reduced compared with 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 the assumption of the continuous probability density function. Acoustic features must be well represented for the chosen probability density function.

The Q-function has played an important role in the unified modelling theory. Like a number of variants such as time duration models, introducing hidden parameters in the Q-function paves the way to solve the incomplete data problem by unobservable complete data.

References

1. L. Bahl *et al.*, "Large vocabulary natural language continuous speech recognition," *Proc. ICASSP-89*, pp. 465-467, Glasgow, Scotland, 1989.

2. 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.

3. J. Bellegarda and D. Nahamoo, "Tied mixture continuous parameter models for large vocabulary isolated speech recognition," *Proc. ICASSP-89*, pp. 13-16, Glasgow, Scotland, 1989.

4. P.F. Brown, "Acoustic-phonetic modeling problem in automatic speech recognition," Ph.D. thesis, Department of Computer Science, Carnegie-Mellon University, 1987.

5. G.R. Doddington, "Phonetically sensitive discriminants for improved speech recognition," *Proc. ICASSP-89*, pp. 556-559, Glasgow, Scotland, 1989.

6. V.N. Gupta, M. Lennig, and P. Mermelstein, "Integration of acoustic information in a large vocabulary word recognizer," *Proc. ICASSP-87*, pp. 697-700, Dallas, USA, 1987.

7. 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*, p. 71, Kobe, Japan, 1988.

8. 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.

9. 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.

10. 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.

11. X.D. Huang, "Semi-continuous hidden Markov models for speech recognition," Ph.D. thesis, Department of Electrical Engineering, University of Edinburgh, 1989.

12. 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*, pp. 639-642, Glasgow, Scotland, 1989.

13. F. Jelinek, "Continuous speech recognition by statistical methods," *Proc. IEEE*, vol. 64, pp. 532-556, 1976.

14. K.F. Lee, "Large-vocabulary speaker-independent continuous speech recognition: The SPHINX system," Ph.D. thesis, Department of Computer Science, Carnegie-Mellon University, 1988; also Automatic Speech Recognition: The Development of the SPHINX System, Kluwer Academic Publishers, 1989.

15. 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.

16. M. Nishimura and K. Toshioka, "HMM-based speech recognition using multi-dimensional multi-labeling," *Proc. ICASSP-87*, pp. 1163-1166, Dallas, USA, 1987.

17. D.B. Paul, "The Lincoln robust continuous speech recognizer," *Proc. ICASSP-89*, pp. 449-452, Glasgow, Scotland, 1989.

18. A.B. Poritz and A.G. Richter, "On hidden Markov models in isolated word recognition," *Proc. ICASSP-86*, pp. 705-708, Tokyo, Japan, 1986.

19. 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.

20. L.R. Rabiner, J.G. Wilpon, and F.K. Soong, "High performance connected digit recognition using hidden Markov models," *Proc. ICASSP-88*, New York, USA, 1988.

21. 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*, pp. 548-551, Glasgow, Scotland, 1989.

22. S. Soudoplatoff, "Markov modeling of continuous parameters in speech recognition," *Proc. ICASSP-86*, pp. 45-48, Tokyo, Japan, 1986.

23. 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.

CHAPTER EIGHT

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 system relies on the 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 interdependent. For example, detailed models usually require more parameters, which result in a requirement for more training data to estimate these parameters reliably.

In this chapter we will discuss practical issues and various improved modelling techniques related to these.

8.1. Problems of Insufficient Data

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 tokens tends to infinity) [12]. 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 parameters being inadequately trained and the value of these parameters will tend 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 re-estimated. 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.

8.1.1. Parameter tying

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 the continuous output densities of the continuous mixture HMM. The fenone models [8] can also be viewed as a kind of tying of 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 [5]. In this manner the number of free parameters can be reduced, and parameters can be well estimated. This is particularly suitable to re-

estimation of covariance matrices in the continuous mixture HMM [29,50].

Let us take the discrete HMM as an 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 re-estimated as follows:

$$\bar{a}_{ij} = \frac{\sum\limits_{i',j' \in \tau(i,j)} \sum\limits_{t=1}^{T-1} \gamma_t(i', j')}{\sum\limits_{i' \in \tau(i,j)} \sum\limits_{t=1}^{T-1} \sum\limits_{k} \gamma_t(i', k)} \tag{8.1.1}$$

$$\bar{b}_j(k) = \frac{\sum\limits_{j' \in \tau(j)} \sum\limits_{t \in O_t = v_k} \gamma_t(j')}{\sum\limits_{j' \in \tau(j)} \sum\limits_{t=1}^{T} \gamma_t(j')} = \frac{\sum\limits_{j' \in \tau(j)} \sum\limits_{t \in O_t = v_k} \sum\limits_{i} \gamma_t(j', i)}{\sum\limits_{j' \in \tau(j)} \sum\limits_{t=1}^{T} \sum\limits_{i} \gamma_t(j', i)} \tag{8.1.2}$$

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 [5,37]. 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. (8.1.2) can thus be regarded as a tied transition-dependent HMM, where output probabilities associated with transitions to state j are tied together.

8.1.2. 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 those parameters that are less than a certain value to be equal to the floor value 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 [33,37].

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 a 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 a 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:

$$b_j = \kappa^1 b_j^1 + \kappa^2 b_j^2, \qquad \kappa^1 + \kappa^2 = 1 \tag{8.1.3}$$

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 4.2.1 discussed in Chapter 4, re-estimation of κ has indeed the same form as the mixture density estimation. Thus, it can be done via the EM algorithm. In Eq. (8.1.3), the 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 so 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 4, the maximum likelihood estimates for κ^1 can be written as:

$$\bar{\kappa}^1 = \frac{1}{K}\sum_{i=1}^{K} \frac{\kappa^1 Pr^1(\mathbf{y}_i)}{\kappa^1 Pr^1(\mathbf{y}_i) + (1-\kappa^1)Pr^2(\mathbf{y}_i)} \tag{8.1.4}$$

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 predict unseen events better. 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 re-estimation 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 re-estimation) 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 [37], with $\bar{\kappa}$ being re-estimated 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 a discrete output probability distribution with a uniform distribution as in Eq. (8.1.3). To improve the performance of a speaker-independent speech recognition system, we can divide the training data into male and female groups, and build male and female models 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

$$b_j = \kappa^1 b_j^1 + \kappa^2 b_j^2 + \kappa^3 b_j^3, \qquad \kappa^1+\kappa^2+\kappa^3 = 1 \qquad (8.1.5)$$

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.

8.2. Estimation Criteria

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. In HMM-based speech recognition, this amounts

to the assertion that the observed speech is genuinely produced by the HMM being used, and that only the values of the model parameters are unknown. However, this can well be challenged. Typical HMMs make many inaccurate assumptions about the speech production process, such as the output independence assumption, Markov assumption, and the continuous probability density assumption. Such inaccurate assumptions substantially weaken the rationale for maximum-likelihood criteria. For instance, although maximum-likelihood estimation is consistent (convergence to the true value), it is meaningless to have such a property if the wrong model is used. The true parameters in such cases will be the true parameters of the wrong models. Therefore, an estimation criterion that can work well in spite of these inaccurate assumptions should offer improved recognition accuracy compared with the maximum likelihood criterion. Other re-estimation criteria, such as maximum mutual information criteria [14], minimum discrimination information criteria [22], H-criteria [27], and corrective training [7,38], have been proposed to improve the maximum likelihood criterion. In this section we will discuss two distinctive methods: maximum mutual information estimation and corrective training procedures, both of which have been shown to work well in speech recognition.

8.2.1. Maximum mutual information criteria

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 [14]. Therefore, the invalid argument for maximum likelihood estimation can be, to an extent, 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 $(\lambda_1,\lambda_2,\ldots,\lambda_\upsilon)$, and the criterion can be written as

$$I(\mathbf{O},\Lambda) = \sum_{v}[\log Pr(\mathbf{O}^v|\lambda_v) - \log\sum_{w}Pr(\mathbf{O}^v|\lambda_w)Pr(\lambda_w)] \quad (8.2.1)$$

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. To maximise the above equation, traditional maximisation techniques, such as the gradient descent methods, can be used [6,26]. Alternatively, the Baum–Welch algorithm can be generalised to rational objective functions [28], and applied to maximum mutual information criteria.

The forward–backward algorithm described in Chapter 5 can be used in the same way as for the maximum likelihood criterion, but re-estimation formulas must be obtained from the gradient search or generalised Baum–Welch algorithm [28]. If the objective function is maximised with a gradient-based hill-climbing algorithm, the derivative of $I(\mathbf{O},\lambda)$ with respect to model parameters λ, namely, (A,B,π), must be computed. As probability $Pr(O|\lambda)$ can be represented by the forward variables and backward variables, the derivatives can also be expressed by $\gamma_t(i,j)$ in a similar manner to those discussed in Chapter 5 [14]. Here, the forward–backward computation must be performed for all the speech samples with every model. Since the forward–backward computation is only needed for the speech sample corresponding to model v in maximum likelihood methods, if the total number of models is large, the hill-climbing step will involve much more computation than maximum likelihood methods. One way to reduce computational complexity is to replace the second term $(\sum_{w})$ in Eq. (8.2.1) by the summation for only a set of acoustically confusable models.

When parameters are iteratively re-estimated, stochastic constraints must be satisfied in each iteration. However, this is not the case when the gradient descent algorithm is applied directly [14]. In addition, the gradient itself may not be bounded [44]. Implementational issues to solve these problems can be found in [14,44].

8.2.2. Corrective training

The heuristic *corrective* re-estimation procedure [3,7,38] attempts to maximise recognition accuracy on the training data. This algorithm tests the decoder using re-estimated models according to training data in the training procedure, and subsequently improves the correct models and suppresses misrecognised or near-miss models. The basic training procedure is simple, and can be described as follows:

Corrective training

Step 1: Using training data and the forward–backward algorithm iteratively obtain $\gamma_t()$ probabilities and expected frequencies used in the re-estimation of HMM parameters.

Step 2: Perform speech recognition on the training data based on currently estimated parameters.

Step 3: If any word is misrecognised, or near-misrecognised, adjust the estimated model parameters to reduce the probability of misrecognised or near-misrecognised words. Update the model parameters.

Step 4: If any adjustments are made in Step 3, return to Step 2.

Although such a procedure cannot guarantee convergence, it has many merits. Firstly, the model assumption may not be required to be as accurate as for the maximum likelihood criteria. For any set of models, corrective training attempts to find statistics that make the models work rather than maximising the likelihood. Corrective training acts to minimise the recognition error rate, even if it reduces the likelihood or the mutual information of the training data. Secondly, if the conventional EM algorithm leads to statistics that are stranded on a local maximum, corrective training may offer the potential to hoist them off, allowing continued optimisation [7]. In practice, it is reported that corrective training yields better recognition accuracy than either maximum likelihood criteria or maximum mutual information criteria [7].

In a similar way to corrective training, *discriminant analysis* can be used in continuous hidden Markov modelling in which the *between-class* covariance matrices can be obtained from the *misrecognised* data [21]. The between-class and within-class covariance matrices are used to project each continuous output probability density function in the continuous HMM, and such a projection is reported to reduce recognition error rate significantly in comparison with the continuous HMM [21].

8.3. 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 [14,24,37,50,53,61].

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 [14,52,53]. Since different features may have different physical meanings, or even be strongly correlated, it is often necessary to use appropriate probability density functions. In addition, dimension reduction approaches based on principal component or discriminant analysis projection are

required [14,31]. Alternatively, if the semi-continuous HMM or the discrete HMM is used, each feature representation can be quantised by its own VQ codebook [29,30,37].

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 [39]. 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 in Eq. (7.2.2), which consists of L-mixture continuous density functions. In other words, the semi-continuous output probability could be modified as:

$$b_i(\mathbf{x}) = \prod_{c} \sum_{j=1}^{L} f^c(\mathbf{x}|v_j^c) b_i^c(v_j^c) \qquad (8.3.1)$$

where c denotes the codebook used. The re-estimation algorithm for the multiple codebook based HMM could be extended if Eq. (7.2.5) 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 7, for codebook c_l, $\chi_t(i,j,k^{c_l})$ could be extended as:

$$\chi_t(i,j,k^{c_l}) = \qquad (8.3.2)$$

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

Other intermediate probabilities can also be computed in a manner similar to Eq. (8.3.1), and it can be easily proved that this is consistent with the maximum likelihood criterion.

8.4. 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

$$d_i(t) = a_{ii}^t(1-a_{ii}) \tag{8.4.1}$$

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 [40,56], 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 8.4.1. In (a), the state duration probability has an exponential form as in Eq. (8.4.1). In (b), the self-transition probabilities are replaced with an explicit duration probability distribution. At time t, the process enters state i 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 state i. Thus, by setting the time duration probability density to be the exponential density of Eq. (8.4.1) the time duration HMM can be made equivalent to the standard HMM.

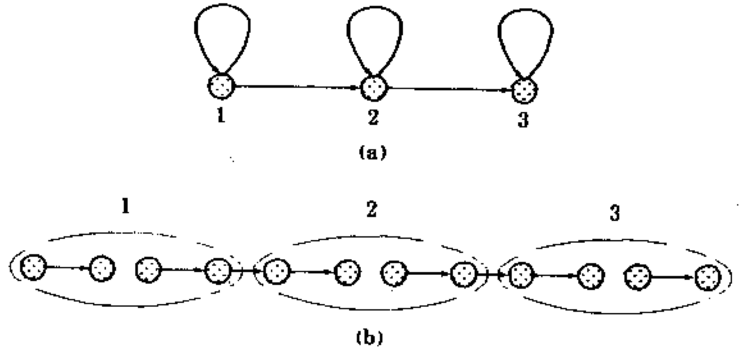

Figure 8.4.1 A time duration HMM

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 re-estimate the parameters of the HMM with time duration modelling, the forward recursion must be modified as follows:

$$\alpha_t(j) = \sum_{\tau} \sum_{\substack{all\ i \\ i \neq j}} \alpha_{t-\tau}(i) a_{ij} d_j(\tau) \prod_{l=1}^{\tau} b_j(O_{t-\tau+l}) \tag{8.4.2}$$

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 j. Intuitively, Eq. (8.4.2) 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 the output probabilities. Similarly, the backward recursion can be written as:

$$\beta_t(i) = \sum_{\tau} \sum_{\substack{all\ j \\ j \neq i}} a_{ij} d_j(\tau) \prod_{l=1}^{\tau} b_j(O_{t+l}) \beta_{t+\tau}(j) \tag{8.4.3}$$

The modified Baum – Welch algorithm can then be used based on Eq. (8.4.2) and (8.4.3). The proof of the re-estimation algorithm can be based on the modified Q-function as in Eq. (5.4.1) except that $Pr(\mathbf{O},S|\lambda)$ should be replaced with $Pr(\mathbf{O},S,T|\lambda)$, which denotes the joint probability of observation, $\mathbf{O}$, state sequence, $S=\{s_1,...,s_k,...,s_N\}$ in terms of state s_k with time duration τ_k, and the corresponding duration sequence, $T=\{\tau_1,...,\tau_k,...,\tau_N\}$.

$$Q(\lambda,\bar{\lambda}) = \frac{1}{Pr(\mathbf{O}|\lambda)}\sum_T\sum_S Pr(\mathbf{O},S,T|\lambda)\log Pr(\mathbf{O},S,T|\bar{\lambda}) \quad (8.4.4)$$

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:

$$\gamma_{t,\tau}(i,j) = \alpha_t(i)a_{ij}d_j(\tau)\prod_{l=1}^{\tau}b_j(O_{t+l})\beta_{t+\tau}(j) \quad (8.4.5)$$

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

$$\gamma_{t,\tau}(i) = \sum_j \gamma_{t,\tau}(j,i) \quad (8.4.6)$$

In a manner similar to Eq. (5.4.4) and after some algebraic operations, the re-estimation algorithm can be written as follows

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1}\sum_{\tau}\gamma_{t,\tau}(i,j)}{\sum_{t=1}^{T-1}\sum_{\tau}\sum_{j}\gamma_{t,\tau}(i,j)} \quad (8.4.7)$$

$$\bar{d}_j(\tau) = \frac{\sum_{t=1}^{T}\gamma_{t,\tau}(j)}{\sum_{t=1}^{T}\sum_{\tau}\gamma_{t,\tau}(j)} \quad (8.4.8)$$

$$\bar{b}_j(k) = \frac{\sum_{t=1}^{T}\sum_{\tau}\gamma_{t,\tau}(j)c_{t,\tau}(k)}{\sum_{t=1}^{T}\sum_{\tau}\gamma_{t,\tau}(j)\tau} \quad (8.4.9)$$

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:

$$\delta_t(j) = \underset{i}{\text{Max}}\ \underset{\tau}{\text{Max}}[\delta_{t-\tau}(i)a_{ij}d_j(\tau)\prod_{k=1}^{\tau}b_j(O_{t-\tau+k})] \quad (8.4.10)$$

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 to 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 fewer training data to estimate $d_j(\tau)$ 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 [57]. One proposal to alleviate some of these problems is to use a continuous density function instead of the discrete distribution $d_j(\tau)$ [40,56]. A more interesting alternative is to smooth the discrete time duration distribution by the Gaussian probability density function [4,37], which can significantly improve the performance when limited training data are used.

8.5. 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 the above described system, 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.

8.5.1. Whole-word models

The most natural unit of speech is the word, and it has been widely used for many speech recognition systems

[21,29,41,48,53]. One of the most distinctive advantages of using 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.

8.5.2. Subword models

Instead of using whole-word models, various subword models can be used so 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 [34,58], word-dependent phone models [16,37,45], and triphone models [2,17];

(2) acoustically defined subword models, such as fenone models [8] and segment models [35]; and

(3) hybrid models such as generalised triphone models [32,37], and allophonic models [1].

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 [8,36,51].

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 [32]. 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 [19,20]. A more interesting approach is to identify and merge triphones automatically in similar acoustic realisations, with clustering procedures being used to produce generalised triphone models from triphone models [37]. The distance measure between two models, λ_1 and λ_2, of the same phone in similar context is based on the following distance measure:

$$D(\lambda_1,\lambda_2) = \frac{\prod_i (Pr(i|\lambda_1)^{N_{\lambda_1}(i)})(\prod_i (Pr(i|\lambda_2)^{N_{\lambda_2}(i)})}{\prod_i (Pr(i|\lambda_{1+2})^{N_{\lambda_{1+2}}(i)})} \qquad (8.5.1)$$

where $Pr(i|\lambda_k)$ and $N_{\lambda_k}(i)$ are the discrete output probability of codeword i and the corresponding count of codeword i in λ_k respectively. This equation measures the ratio between the probability that the individual distributions (before merging) 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 re-estimation.

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 [38]. 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 [1].

8.6. Isolated vs Continuous Speech Recognition

For isolated word recognition, the training and recognition can be implemented directly using the basic algorithms introduced in Chapter 5. 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 forward–backward algorithm and 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 the 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 re-estimation 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 [5,42,46,53,63].

The Viterbi search [63] 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 [10]. 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 found first, and each state with smaller probability than 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 the stack decoding algorithm [5], can be used, with however considerable increase in computational complexity.

As pointed out in Chapter 3, a decoder must evaluate $Pr(\mathbf{W})Pr(\mathbf{O}|\mathbf{W})$ for all the possible word sequences, **W**, given the observation **O**. An optimal decoder should choose a word sequence **W** that maximises $Pr(\mathbf{W})Pr(\mathbf{O}|\mathbf{W})$. The search process can be represented by a tree where branches correspond to words, and nodes correspond to sentences as shown in Figure 8.6.1. Hence, non-terminal nodes correspond to incomplete sentences, and terminal nodes to complete sentences. Techniques, such as depth-first or breadth-first searching, can be useed to obtain an optimal word sequence. In depth-first searching, nodes are first expanded based on most recently generated nodes; and in breadth-first searching, nodes are first expanded in the order in which they are generated. However, these blind-search methods do not take into account how close we are getting to the goal, and are usually computationally infeasible in practical speech recognition systems. Therefore, heuristic search methods, which are based on some evaluation functions, can be applied here. Heuristic search theory has been well studied in Artificial Intelligence [11,47]. One of the key problems in heuristic search is the definition of an evaluation function. The evaluation function of a node may include the cost up to the node and estimate of cost to the target node from the node. For example, stack decoding [5] is a variant of the heuristic A* search based on the forward algorithm, where the evaluation function is based on the forward probability. In contrast to the Viterbi search, it is not time-synchronous, but extends paths of different lengths. The search begins by adding all possible one-word hypotheses to the *OPEN* list. Then the *best* hypothesis is removed from the *OPEN* list, and all paths from it are extended, evaluated, and placed back in the *OPEN* list. This search continues until a complete path that is guaranteed to be better than all paths in the *OPEN* list has been found. In general, the evaluation function that estimates the score of the complete path as the sum of the known score of the partial path and the expected score of the remaining path is needed to select the *best* path. If the expected score of the remaining path is always an underestimate of the actual score, then the solution found will be optimal. In practice, an underestimating evaluation function is difficult to find for speech recognition. The ones that are guaranteed to underestimate will result in a very large *OPEN* list. So, a heuristic function that may over-estimate has to be used to prune more hypotheses. The best-first search is usually carried out. In addition, two types of pruning can be used:

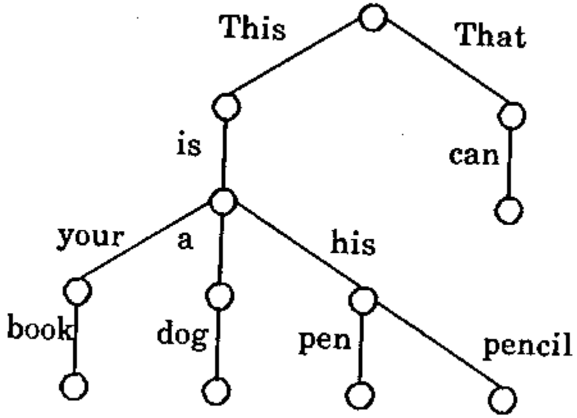

Figure 8.6.1. Search tree

(1) a fast-match that reduces candidates for detailed decoding [9], and

(2) the use of a *stack* that saves only a fixed number of hypotheses in the *OPEN* list [5].

In summary, Viterbi search is a graph search, and paths cannot be summed because they may have different word histories. Stack decoding is a tree search, so each node has a unique history, and the forward algorithm can be used within word hypotheses when the word is extended. With stack decoding, it is possible to use an objective function that searches for the optimal word string, rather than the optimal state sequence. However, unlike Viterbi search where acoustic probabilities being compared are always based on the same partial input, partial paths of different lengths are allowed in stack decoding. Normalisation must be used in order to compare these probabilities.

8.7. Speaker Adaptation

In speaker-dependent speech recognition, every speaker is required to dictate a specific script in order to train the recognition system. As there are anatomical differences (the length of the vocal tract and the size of nasal cavity) and speaking habit differences (accent, speed, and loudness) among different speakers, performance of such systems degrades significantly when the speech being recognised is not well characterised by the speech sample being used in training. To avoid fully re-training the recognition system for the new user, it is necessary to employ a set of speech data of new speakers to adapt speech recognition systems trained from other speakers. In contrast, in speaker-independent speech recognition, the use of speech from many speakers enables reliable and robust estimation of a large number of parameters to model speaker-independence. Different speakers can use the system without re-training the system. However, owing to inherent differences of speakers, it is still desirable to enhance the robustness of speaker-independent systems by considering different speaker characteristics through speaker adaptation.

Most speaker adaptation algorithms require a few adaptation sentences from the new speaker, and adapt model parameters from these adaptation speech data, thereby alleviating tedious re-training procedure with a full set of training data. Techniques, such as transformation of original speech data of a new speaker according to that of a reference

speaker [15,18,54], and combination of *a priori* knowledge with the speaker-specific knowledge obtained from adaptation data using Bayesian learning [13,55,62], have been proposed. Various techniques to modify the VQ codebook [25,49,60] and HMM parameters of a reference speaker according to adaptation data [23,43,49,59], have also been widely investigated.

In various adaptation techniques, unified modelling to VQ and HMM based on the semi-continuous HMM presents a novel method for speaker adaptation. The VQ codebook characterises the most significant speaker differences, but the number of free parameters is modest. Modification of the VQ codebook according to the HMM parameters that are ultimately used for speech recognition offers a unique way to modify the VQ codebook. The semi-continuous re-estimation formulas, Eq. (7.2.19) and Eq. (7.2.20), can be well used for such an application. Using the unified modelling theory, the VQ codebook can be re-estimated by the adaptation data while the HMM parameters are fixed. It is reported [55] that speaker adaptation based on the unified modelling theory yields excellent adaptation results. It is substantially better than either Bayesian learning of the VQ codebook or VQ mapping techniques [60].

8.8. Summary

In this chapter we have 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, such as time duration modelling, requires substantial training data. Tying and interpolation are two important techniques in solving this problem. Estimation criteria become much more important if there exist too many inadequate assumptions. Speaker adaptation helps practical applications of speech recognition. The use of these techniques can be found in many practical speech recognition systems.

References

1. L. Bahl *et al.*, "Large vocabulary natural language continuous speech recognition," *Proc. ICASSP-89*, pp. 465-467, Glasgow, Scotland, 1989.
2. L.R. Bahl *et al.*, "Further results on the recognition of a continuously read natural corpus," *Proc. ICASSP-80*, USA, 1980.
3. T. Applebaum and B. Hanson, "Enhancing the discrimination of speaker independent hidden Markov models with corrective training," *Proc. ICASSP-89*, pp. 302-305, Glasgow, Scotland, 1989.
4. Y. Ariki and M.A. Jack, "Enhanced time duration constraints in hidden Markov modelling for phoneme recognition," *IEE Electronics Letters*, vol. 25, 1989.
5. 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.
6. 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*, pp. 49-52, Tokyo, Japan, 1986.
7. 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*, New York, USA, 1988.
8. 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*, New York, USA, 1988.
9. L.R. Bahl, P. Gopalakrishnan, D. Kanevsky, and D. Nahamoo, "Matrix fast match: a fast method for identifying a short list of candidate," *Proc. ICASSP-89*, pp. 345-348, Glasgow, Scotland, 1989.
10. J. Baker, "Stochastic modeling as a means of automatic speech recognition," Ph.D. thesis, Department of Computer Science, Carnegie-Mellon University, 1975.

11. A. Barr and E. Feigenbaum, *The Handbook of Artificial Intelligence*, Pitman Books Limited, 1981.
12. 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.
13. P.F. Brown, C.-H. Lee, and J.C. Spohrer, "Bayesian Adaptation in Speech Recognition," *Proc. ICASSP-83*, pp. 761-764, USA, 1983.
14. P.F. Brown, "Acoustic-phonetic modeling problem in automatic speech recognition," Ph.D. thesis, Department of Computer Science, Carnegie-Mellon University, 1987.
15. K. Choukri and G. Chollet, "Adaptation of automatic speech recognisers to new speakers using canonical correlation techniques," *Computer Speech and Language*, vol. 1, pp. 95-107, 1986.
16. 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.
17. 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*, pp. 89-92, Dallas, USA, 1987.
18. S. Cox and J. Bridle, "Unsupervised speaker adaptation by probabilistic spectrum fitting," *Proc. ICASSP-89*, pp. 294-297, Glasgow, Scotland, 1989.
19. L. Deng, M. Lennig, V. Gupta, and P. Mermelstein, "Modeling acoustic-phonetic detail in an HMM-based large-vocabulary speech recognizer," *Proc. ICASSP-88*, pp. 509-512, New York, USA, 1988.
20. A. Derouault, "Context-dependent phonetic Markov models for large vocabulary speech recognition," *Proc. ICASSP-87*, pp. 360-363, Dallas, USA, 1987.
21. G.R. Doddington, "Phonetically sensitive discriminants for improved speech recognition," *Proc. ICASSP-89*, pp. 556-559, Glasgow, Scotland, 1989.
22. Y. Ephraim, A. Dembo, and L.R. Rabiner, "A minimum discrimination information approach for hidden Markov

modeling," *Proc. ICASSP-87*, pp. 25-28, Dallas, USA, 1987.

23. M. Feng, F. Kubala, R. Schwartz, and J. Makhoul, "Improved speaker adaptation using text-dependent spectral mappings," *Proc. ICASSP-88*, pp. 131-134, New York, USA, 1988.

24. 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.

25. S. Furui, "Unsupervised speaker adaptation method based on hierarchical spectral clustering," *Proc. ICASSP-89*, pp. 286-287, Glasgow, Scotland, 1989.

26. P.E. Gill, W. Murray, and M.H. Wright, *Practical Optimization*, Academic Press, 1981.

27. P. Gopalakrishnan, D. Kanevsky, A. Nadas, D. Nahamoo, and M. Picheny, "Decoder selection based on cross-entropies," *Proc. ICASSP-88*, pp. 20-23, New York, USA, 1988.

28. P. Gopalakrishnan, D. Kanevsky, A. Nadas, and D. Nahamoo, "A generalization of the Baum algorithm to rational objective functions," *Proc. ICASSP-89*, pp. 631-634, Glasgow, Scotland, 1989.

29. V.N. Gupta, M. Lennig, and P. Mermelstein, "Integration of acoustic information in a large vocabulary word recognizer," *Proc. ICASSP-87*, pp. 697-700, Dallas, USA, 1987.

30. 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.

31. M.J. Hunt and C. Lefebvre, "A comparison of several acoustic representation for speech recognition with degraded and undegraded speech," *Proc. ICASSP-89*, pp. 262-265, Glasgow, Scotland, 1989.

32. M. Hwang, H. Hon, and K. Lee, "Modelling between-word coarticulation in continuous speech recognition," *Eurospeech 89*, Paris, France, 1989.

33. F. Jelinek and R.L. Mercer, "Interpolated estimation of Markov source parameters from sparse data," *Proc. the Workshop on Pattern Recognition in Practice*, Amsterdam, The Netherlands, North-Holland, 1980.

34. D. Klatt, "Problem of variability in speech recognition and in models of speech perception," in *Variability and Invariance in Speech Processes*, ed. J. Perkell and D. Klatt, pp. 300-320, Lawrence Erlbaum Assoc, 1986.

35. C.H. Lee, F.K. Soong, and B.H. Juang, "A segment model based approach to speech recognition," *Proc. ICASSP-88*, New York, USA, 1988.

36. 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.

37. K.F. Lee, "Large-vocabulary speaker-independent continuous speech recognition: The SPHINX system," Ph.D. thesis, Department of Computer Science, Carnegie-Mellon University, 1988; also Automatic Speech Recognition: The Development of the SPHINX System, Kluwer Academic Publishers, 1989.

38. K.F. Lee, "Hidden Markov models: past, present, and future," *Eurospeech 89*, Paris, France, 1989.

39. K.F. Lee, H.W. Hon, and R. Reddy, "The SPHINX speech recognition system," *Proc. ICASSP-89*, pp. 445-449, Glasgow, Scotland, 1989.

40. S.E. Levinson, "Continuously variable duration hidden Markov models for automatic speech recognition," *Computer Speech and Language*, vol. 1, pp. 29-45, 1986.

41. R.P. Lippmann, "An introduction to computing with neural nets," *IEEE ASSP Magazine*, pp. 4-22, 1987.

42. B.T. Lowerre and D.R. Reddy, "The harpy speech understanding system," in *Trends in Speech Recognition*, Prentice Hall, 1980.

43. E.A. Martin, R.P. Lippmann, and D.B. Paul, "Dynamic adaptation of hidden Markov models for robust isolated-word speech recognition," *Proc. ICASSP-88*, pp. 52-54, New York, USA, 1988.

44. B. Merialdo, "Phonetic recognition using hidden Markov models and maximum mutual information training," *Proc. ICASSP-88*, pp. 111-114, New York, USA, 1988.

45. H. Murveit and M. Weintraub, "1000-word speaker-independent continuous-speech recognition using hidden Markov models," *Proc. ICASSP-88*, pp. 115-118, New York,

USA, 1988.

46. 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*, pp. 833-836, Dallas, USA, 1987.

47. N. Nilsson, "Principles of artificial intelligence," *Palo Alto, Calif. Tioga*, 1980.

48. M. Nishimura and K. Toshioka, "HMM-based speech recognition using multi-dimensional multi-labeling," *Proc. ICASSP-87*, pp. 1163-1166, Dallas, USA, 1987.

49. M. Nishimura and K. Sugawara, "Speaker adaptation method for HMM-based speech recognition," *Proc. ICASSP-88*, pp. 207-210, New York, USA, 1988.

50. 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.

51. D.B. Paul and E.A. Martin, "Speaker stress-resistant continuous speech recognition," *Proc. ICASSP-88*, New York, USA, 1988.

52. D.B. Paul, "The Lincoln robust continuous speech recognizer," *Proc. ICASSP-89*, pp. 449-452, Glasgow, Scotland, 1989.

53. L.R. Rabiner, J.G. Wilpon, and F.K. Soong, "High performance connected digit recognition using hidden Markov models," *Proc. ICASSP-88*, New York, USA, 1988.

54. G. Rigoll, "Speaker adaptation for large vocabulary speech recognition systems using "speaker Markov models"," *Proc. ICASSP-89*, pp. 5-8, Glasgow, Scotland, 1989.

55. D. Rtischev, "Speaker adaptation in a large-vocabulary speech recognition system," M.Sc. thesis, Department of Electrical Engineering and Computer Science, MIT, 1989.

56. M.J. Russell and R.K. Moore, "Explicit modelling of state occupancy in hidden Markov models for automatic speech recognition," *Proc. ICASSP-85*, pp. 5-8, Tampa, USA, 1985.

57. M.J. Russell and A.E. Cook, "Experimental evaluation of duration modelling techniques for automatic speech recognition," *Proc. ICASSP-87*, pp. 2376-2379, Dallas, USA, 1987.

58. R. Schwartz, J. Klovstad, J. Makhoul, and J. Sorensen, "A preliminary design of a phonetic vocoder based on a diphone model," *Proc. ICASSP-80*, pp. 32-35, USA, 1980.

59. R. Schwartz, Y. Chow, and F. Kubala, "Rapid speaker adaptation using a probabilistic spectral mapping," *Proc. ICASSP-87*, pp. 633-636, Dallas, USA, 1987.

60. K. Shikano, K.-F. Lee, and R. Reddy, "Speaker adaptation through vector quantization," *Proc. ICASSP-86*, pp. 2643-2646, Tokyo, Japan, 1986.

61. F.K. Soong and A.E. Rosenberg, "On the use of instantaneous and transitional spectral information in speaker recognition," *Proc. ICASSP-86*, pp. 877-880, Tokyo, Japan, 1986.

62. 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.

63. A.J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. formation Theory*, vol. IT-13, pp. 260-269, 1967.

CHAPTER NINE

EXPERIMENTAL EXAMPLES

In this chapter we will discuss implementational issues and experimental results based on the theories introduced so far. In particular, SPHINX, a state-of-the-art large-vocabulary speaker-independent continuous speech recognition system developed at Carnegie Mellon University [7], will be used here as a typical example to illustrate principles in hidden Markov modelling of speech signals.

9.1. Implementational Issues

This section will briefly introduce several implementation issues, such as initial estimates, model structures, scaling, logarithmic representation, and thresholding. Some of these, like thresholding or logarithmic representation, are not necessarily required.

9.1.1. Initial estimates

In theory, the re-estimation 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. In particular, if a probability is initialised to be zero, it will remain zero with every iteration. 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 by using hand-marked data, by using a segmental k-means clustering [11], and by using discrete HMM parameters [5].

9.1.2. HMM structures

The choice of model topology depends on available training data and what the model is used to represent. If each HMM is used to represent a phone or a triphone, one possible topology is shown in Figure 9.1.1. There are seven states and twelve transitions with transition-dependent output probabilities. Three groups of transition-dependent output probabilities and transition probabilities are tied and represented as B, M, and E in the figure. The model assumes that there are at least three steady states for a

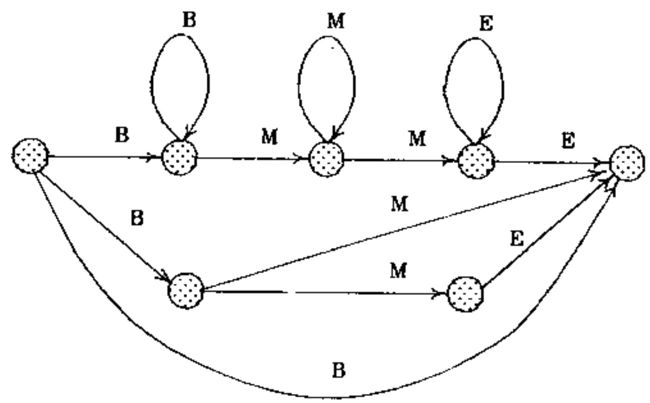

Figure 9.1.1. HMM used in SPHINX

phone, which are indicated by the self-loops. This topology has the advantage of better modelling time duration as there is only one legal path for input of length 1 to 4. It has been successfully used in SPHINX [7].

Another possible representation of such a model can be shown in Figure 9.1.2. There are five states and twelve transitions with state-dependent output probabilities. This is the so-called left-to-right model and has been widely used in many speech recognition systems [2,5,12]. If such a model is used to represent a word, more states are generally required.

No matter what kind of structure is used, the parameters of the model must satisfy stochastic constraints. In the discrete HMM, for example, the following equations must be satisfied.

$$\sum_i \pi_i = 1,$$

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

and

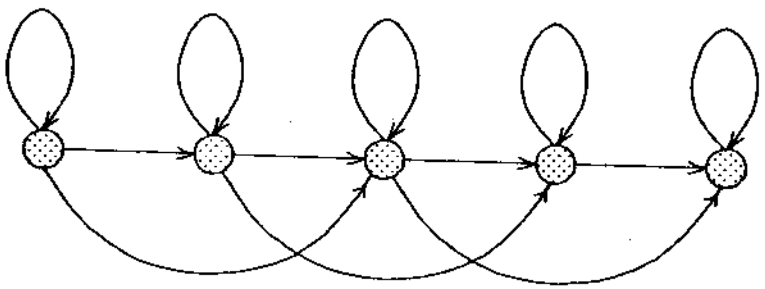

Figure 9.1.2. A left-to-right HMM

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

9.1.3. 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 computations will exceed the precision range of essentially any machine. Thus in practice, the number of observations necessary to train adequately a model and/or compute its probability will result in underflow on the 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. (5.3.6) and then be multiplied by a scaling coefficient, c_t

$$c_t = [\sum_i \alpha_t(i)]^{-1} \tag{9.1.1}$$

so that $\sum_i c_t \alpha_t(i) = 1$ for $1 \le t \le T$. $\beta_t(i)$ can also be multiplied by c_t for $1 \le t \le T$ and $1 \le i \le N$. 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

$$C_t = \prod_{i=1}^{t} c_i \tag{9.1.2}$$

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

$$D_t = \prod_{i=t}^{T} c_i \tag{9.1.3}$$

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

$$\begin{aligned}\sum_{i \in S_F} \alpha_T'(i) &= C_T \sum_{i \in S_F} \alpha_T(i) \\ &= C_T Pr(\mathbf{O}|\lambda)\end{aligned} \tag{9.1.4}$$

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

$$\begin{aligned}\gamma_t'(i,j) &= \frac{C_t \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j) D_{t+1}}{C_T \sum_{i \in S_F} \alpha_T(i)} \\ &= \gamma_t(i,j)\end{aligned} \tag{9.1.5}$$

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. (9.1.5). Therefore, re-estimation formulas can be kept exactly the same as discussed in Chapter 5 except that $Pr(\mathbf{O}|\lambda)$ should be computed according to

$$Pr(\mathbf{O}|\lambda) = \frac{\sum_{i \in S_F} \alpha_T'(i)}{C_T} \tag{9.1.6}$$

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.

9.1.4. Logarithmic computation

An alternative way to avoid underflow is to use a logarithmic representation for all the probabilities. This not only ensures that scaling is unnecessary as 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 unity. 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 $P_1 \geq P_2$:

$$\begin{aligned}&\log_b(P_1+P_2) \\ &= \log_b(b^{\log_b P_1} + b^{\log_b P_2}) \\ &= \log_b P_1 + \log_b(1+b^{\log_b P_2 - \log_b P_1})\end{aligned} \tag{9.1.6}$$

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 $\log_b P_2 - \log_b P_1$, $\log_b(1+b^{\log_b P_2 - \log_b P_1})$ could be stored as a table, $T(n)$, where

$$T(n) = \begin{cases} \log_b(1+b^n) & \text{if } T(n) \geq 0.5 \\ 0 & \text{otherwise} \end{cases} \tag{9.1.7}$$

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 until $\log_b(1+b^n)$ approaches zero. Varying b from 1.0001 to 1.00001, the size of $T(n)$ increases from 99041 to

1220614 when 32-bit integers are used for logarithms [3,7].

With the aid of this table,

$$\log_b(P_1+P_2) = \begin{cases} \log_b P_1 + T(\log_b P_2 - \log_b P_1) & \text{if } P_1 > P_2 \\ \log_b P_2 + T(\log_b P_1 - \log_b P_2) & \text{otherwise} \end{cases} \tag{9.1.8}$$

This implements the addition of two probabilities as one integer addition, one subtraction, two comparisons, and one table look-up. 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.

9.1.5. Thresholding

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 re-estimation formulas, if certain γ become very small relative to other γ, it is observed that these small γ have little effect on the final re-estimates. 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,

$$\hat{\alpha}_t = \operatorname*{Max}_i \alpha_t(i) \tag{9.1.9}$$

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 will be used for estimation of model parameters, which will usually deteriorate final recognition accuracy.

9.1.6. Examples of C programs

This section will demonstrate how the theories introduced so far can be implemented on computers. C programming language will be used to explain implementational details for several selected examples. It is intended to help readers in writing computer programs. The C program routines presented here are unnecessary optimal solutions.

The mathematical model has been described previously. To convert such models into computer programs, the data structure must be defined first. For example, each state and transition arch can be defined as

```
struct state
{
   short    num_out, num_in, initial, final;
   float    output[VQ_LEVEL];
   int      trans_out[MAX_NUM_TRANS_OUT],
            trans_in[MAX_NUM_TRANS_TO_STATE];
};

struct transition
{
   int      destination, origin;
   float    trans_prob;
};
```

```
struct HMM
{
   int       num_states, num_arcs, num_initial,
             num_final, num_omatrix;
   struct    state *states;
   struct    transition *transitions;
};
```

Therefore, each HMM consists of states (HMM.states), transitions (HMM.transitions), number of state (HMM.num_states), number of arcs (HMM.num_arcs), number of initial states (HMM.num_initial), number of final states (HMM.num_final), and number of output probability distributions (HMM.num_omatrix) in the HMM. Each state is specified by an output probability distributions (states.output[k]), number of arcs going out of the state (states.num_out), number of arcs going into the state (states.num_in), and indicator if the state is an initial state (states.initial) or a final state (states.final). Each transition arc is specified by the destination and origin of the arc, and associated transition probability (transitions.trans_prob). With data structures defined above, the forward algorithm, for instance, can be conveniently written as:

```
extern float Discard_Alpha; /* thresholding */
extern float Alpha[MAX_TIME][MAX_STATE]; /* α t×s */
extern float *Pi,          /* Initial probabilities */
             *Scale;       /* Scaling */

forward_computation (hmm, train_data)
struct HMM *hmm;
int   *train_data;  /* quantised speech starting from [1] */
{
 int    *stop, *cptr, i, s;
 float  best_alpha, *current_alpha_ptr, threshold, o_prob;
 struct  transition *tptr;

 for (i = 1; i <= T; i++)
   for (s = 0; s < hmm -> num_states; s++)
     Alpha[i][s] = 0.0;
```

```
for (s = 0; s < hmm -> num_states; s++)
{
  if (hmm -> states[s].initial)
    Alpha[1][s] = Pi[s];
}


for (i = 1; i <= T; i++)
{
  best_alpha = Scale[i] = 0.0;
  for (s = 0; s < hmm -> num_states; s++)
  {
    if (i == 1 && hmm -> states[s].initial)
    { Scale[1] + = Alpha[1][s];
      continue;
     }
    current_alpha_ptr = &Alpha[i][s];
    stop = &(hmm -> states[s].trans_in[
               hmm -> states[s].num_in]);
    o_prob = hmm -> states[s].output[train_data[i]];
    for (cptr = &(hmm -> states[s].trans_in[0]);
               cptr < stop; cptr++)
    {
      tptr = &(hmm -> transitions[*cptr]);
      if (i != 1) {
       *current_alpha_ptr + =
        Alpha[i-1][tptr -> origin]*
        (tptr -> trans_prob)*o_prob;}
    }
    Scale[i] + = *current_alpha_ptr;
    if (*current_alpha_ptr > best_alpha)
      best_alpha = *current_alpha_ptr;
 }

threshold = best_alpha * Discard_Alpha;
for (s = 0; s < hmm -> num_states; s++)
{
   current_alpha_ptr = &Alpha[i][s];
   if (*current_alpha_ptr < threshold)
   {
      Scale[i] -= *current_alpha_ptr;
      *current_alpha_ptr = 0.0;
      }
}

if (Scale[i]>MIN_FLOAT)
```

```
        for (s = 0; s < hmm -> num_states; s++)
          Alpha[i][s] /= Scale[i];
        else
          printf("ERROR: increase threshold ");
    }
  }
```

The backward algorithm, the Baum – Welch algorithm, as well as the Viterbi algorithm can be developed in a similar manner.

9.2. Database and Analysis Conditions

The speech database is crucial to the success of hidden Markov modelling. A compromise has to be made between enough detailed system configuration to model variabilities, and available training data to estimate the free parameters reliably. For example, the database for the task of *resource management*, which is designed for inquiry of naval resources and used for evaluation of systems in the DARPA speech recognition projects [4,7,9], contains 4358 sentences from 105 speakers. It is reported that a decrease in the number of training data with the fixed system configuration (fixed number of free parameters) results in much faster deterioration than that with data-dependent configuration (reduce the number of free parameters according to the training data, such as by using fewer generalised triphones) [8]. Although recognition accuracy can be greatly improved from increased training data, the data-dependent configuration is still important to optimise the recognition accuracy. With respect to use of a large speech database, a speaker-independent HMM speech recognition is very suitable since the database can be collected incrementally. In a database with millions of sentences, an automatic learning technique that does not need explicitly to segment and label the phonemes or words such as the HMM is necessary. As

pointed out by Lee [8],

> *We believe larger databases are essential for HMM systems, and HMM systems ideal for larger databases.*

In the resource management database, at the lexical level, there are 997 words with 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 994009 word pairs, only 57878 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 from the given HMM to the following word HMM is $1/K$, 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 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: 1-0.97z^{-1}

9.3. Experimental Examples

In SPHINX, the signal processing stage is based on the bilinear transformed LPC cepstrum, which converts the linear frequency axis into a form of mel-scale [7,13]. 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.

9.3.1. Discrete HMM results

SPHINX was developed using the discrete HMM [7]. Phone or triphone models used here are the same as that shown in Figure 9.1.1, and are concatenated to form a word model according to the pronunciation dictionary in training and recognition procedure.

Three VQ codebooks are used for twelve LPC bilinear transformed LPC cepstral coefficients; twelve differenced bilinear transformed LPC cepstral coefficients; and energy and differenced energy. Use of multiple features and multiple codebooks can substantially reduce the recognition error rate. When phone models are used, the word recognition error rate can be reduced by 40−50% [7]. Although time duration modelling can significantly improve the recognition accuracy for speaker-dependent speech recognition [1,5], use of word duration constraints in SPHINX as a part of the Viterbi beam search has no significant effects when grammatic constraints exist. This indicates that, for speaker-independent speech recognition, word duration constraints may well be used as a method for speaker adaptation.

One problem with continuous speech recognition is the unclear articulation of function words, such as *a*, *the*, *in*, *of*, etc. These function words in English are limited, but occur frequently. A detailed modelling approach is to treat each phone in each function word separately. By explicitly modelling the most difficult sub-vocabulary, the recognition accuracy can be increased considerably. In a similar manner, function phrases, such as *that are of, is the,* and *that are*, can be explicitly modelled. In the vocabulary of resource management task, 42 function words and 12 function phrases are identified and modelled separately. Modelling these frequently occurring words and phrases increased the number of free parameters by a factor of 5 compared with to phone models, and the word recognition error rate can be reduced by about 25% [7].

The function-word and function-phrase dependent phone models provide better representation of the function words. However, simpler phone models for the non-function words are inadequate, because the realisation of a phone crucially depends on context. 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 can be used in the clustering procedure along with triphones [7]. In addition, between-word coarticulation modelling can also be incorporated [6]. The generalised triphone modelling can empirically determine how many models could be trained given a set of training data. Generalised triphone models increased the number of free parameters by a factor of 25 compared with phone models, and reduced the word recognition error rate by more than 70% when 1100 generalised triphone models were used [8]. Further improvement can be obtained by using corrective training on the optimal models, where the error rate can be reduced by more than 15% [8].

If between-word coarticulation modelling is excluded, for the test data used here, the error rate increases by about 15%. The word recognition accuracy is 91.0% when 1000 generalised triphone models are used. If function-word and function-phrase modelling are further excluded, and a lower number of generalised triphone models (200) are applied, the word recognition accuracy becomes 88.0%. Several algorithms based on such a configuration will be investigated hereafter. 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.

The training procedure in SPHINX can be briefly described as follows: the phone model is first re-estimated from the training data, and these models are then used as initial parameters for generalised triphones. The discrete output probabilities are finally smoothed by employing deleted interpolation with the phone models and uniform distribution.

9.3.2. Continuous mixture HMM results

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 [7]. 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 four 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 segment and label training speech phonetically. These labelled 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 word accuracy of the continuous mixture HMM is 81.3%, which is significantly lower than that 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 [10], for the experiments conducted here, it is *significantly* worse than the discrete HMM. Explanations for this paradox are as follows.

(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 in 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 distinct 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.

9.3.3. Semi-continuous HMM results

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 algorithms are iteratively used to re-estimate simultaneously 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 the M most significant codewords are used for subsequent processing. Experiments with the top one and top four codewords were conducted.

Under the same analysis conditions as previously, the word accuracies for the semi-continuous HMM are shown in Table 9.1. The results for the discrete HMM and the continuous mixture HMM are also listed for comparison.

Table 9.1.
Word accuracy using 200 generalised triphones, 4358 training sentences, 300 test sentences

Types	*Word accuracy*
Discrete HMM	88.0%
Continuous mixture HMM	81.3%
Semi-continuous HMM + top 1	84.0%
Semi-continuous HMM + top 4	89.1%

From Table 9.1, 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 compared with the standard SPHINX version [7], smoothing by multiple codewords can still play an important role. As the diagonal Gaussian assumption may be inappropriate, the covariance matrices need not be re-estimated. Indeed, fixed covariance matrices are marginally better than re-estimated ones owing to inappropriate assumptions.

9.3.4. Less correlated data results

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. However, only the insignificant improvements are obtained based on such a projection [5]. This may be because the covariance for each codeword is quite different, and such a projection only makes *average* covariance diagonal, which is inadequate. As bilinear transformed cepstral coefficients cannot be modelled well by diagonal Gaussian probability density functions,

experiments without bilinear transformation were conducted. 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 [5]. 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 [13]. 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 9.2.
Word accuracy of 18th order cepstrum
(200 generalised triphones)
4358 training sentences, 300 test sentences

Types	*Word accuracy*
Discrete HMM	83.8%
Semi-continuous HMM + top 1	85.5%
Semi-continuous HMM + top 2	87.6%
Semi-continuous HMM + top 4	88.5%
Semi-continuous HMM + top 6	88.6%
Semi-continuous HMM + top 8	88.2%

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 9.2.

Here, the recognition accuracy of the semi-continuous HMM is significantly better than the discrete HMM, and error reduction is over 29%. Even if the top one codeword is used, the semi-continuous HMM is better than the discrete HMM (85.5% vs 83.8%). Use of multiple codewords (top 4 and top 6) 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 9.2, it can be seen that the semi-continuous HMM with the top four codewords is adequate (88.5%). In contrast, when bilinear transformed data were used (Table 9.1), the error reduction was less than 10% compared with 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 a continuous probability density function, especially the one 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 improve recognition accuracy further. Regarding use of full covariance, the semi-continuous HMM has a distinct advantage. Since Gaussian probability density functions are tied to the VQ codebook, by choosing the 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.

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 has been strongly indicated by the experimental results based on the bilinear transformed cepstrum and cepstrum.

9.4. Summary

In this chapter we have discussed several implementational issues and experimental examples in hidden Markov modelling. As introduced previously, sufficient training data, automatic learning algorithms, and detailed modelling are three very important factors of a successful speech recognition system. From the continuous HMM point of view, detailed acoustic modelling can be achieved in increasing mixture densities. However, the performance will suffer from insufficient training data if there are too many free parameters in increasing mixture densities. 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. 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. These merits of the unified modelling can be viewed as a good solution to the conflict between detailed acoustic modelling and insufficient training data.

While it can be concluded that the theory of HMMs is powerful for modelling speech signals, by itself it has not totally solved the general speech recognition problem, but has provided some insight for future researchers to visit, enhance and report.

References

1. Y. Ariki and M.A. Jack, "Enhanced time duration constraints in hidden Markov modelling for phoneme recognition," *IEE Electronics Letters*, vol. 25, 1989.
2. R. Bakis, "Continuous speech recognition via centisecond acoustic states," *The 91st Meeting of the Acoustic Society of America, April*, 1976.
3. P.F. Brown, "Acoustic-phonetic modeling problem in automatic speech recognition," Ph.D. thesis, Department of Computer Science, Carnegie-Mellon University, 1987.
4. 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*, pp. 89-92, Dallas, USA, 1987.
5. X.D. Huang, "Semi-continuous hidden Markov models for speech recognition," Ph.D. thesis, Department of Electrical Engineering, University of Edinburgh, 1989.
6. M. Hwang, H. Hon, and K. Lee, "Modelling between-word coarticulation in continuous speech recognition," *Eurospeech 89*, Paris, France, 1989.
7. K.F. Lee, "Large-vocabulary speaker-independent continuous speech recognition: The SPHINX system," Ph.D. thesis, Department of Computer Science, Carnegie-Mellon University, 1988; also Automatic Speech Recognition: The Development of the SPHINX System, Kluwer Academic Publishers, 1989.

8. K.F. Lee, "Hidden Markov models: past, present, and future," *Eurospeech 89*, Paris, France, 1989.
9. D.B. Paul, "The Lincoln robust continuous speech recognizer," *Proc. ICASSP-89*, pp. 449-452, Glasgow, Scotland, 1989.
10. 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.
11. L.R. Rabiner, J.G. Wilpon, and B.H. Juang, "A segmental k-means training procedure for connected word recognition," *AT&T Technical Journal*, vol. 65, pp. 21-31, 1986.
12. L.R. Rabiner, J.G. Wilpon, and F.K. Soong, "High performance connected digit recognition using hidden Markov models," *Proc. ICASSP-88*, New York, USA, 1988.
13. K. Shikano, "Evaluation of LPC spectral matching measures for phonetic unit recognition," CMU Technical Report CMU-CS-86-108, Computer Science Department, 1986.

APPENDIXES

Appendix 1:
Show the maximum likelihood estimation of a mixture Gaussian pdf, in a case where the *a priori* probability $Pr(\omega_i)$ is unknown.

The equality constraint is given as:

$$\sum_{i=1}^{s} Pr(\omega_i) = 1 \tag{A1.1}$$

Using Lagrange's multiplier λ (see Section 2.5.3), Eq. (2.4.8) is modified to the following partial derivative of augmented log-likelihood with respect to $Pr(\omega i)$:

$$\nabla_{Pr(\omega_i)}\, l_a(\varphi) \tag{A1.2}$$

$$= \sum_{k=1}^{n} Pr(\omega_i | x_k,\ \varphi_i) \nabla_{Pr(\omega_i)} \log(f(x_k | \omega_i,\ \varphi_i) Pr(\omega_i))$$

$$+ \nabla_{Pr(\omega_i)} \lambda \sum_{i=1}^{s} Pr(\omega_i)$$

$$= \sum_{k=1}^{n} Pr(\omega_i | x_k,\ \varphi_i) \frac{1}{Pr(\omega_i)} + \lambda = 0$$

By multiplying the above expression by $Pr(\omega_i)$, and summing over i, the following expression is obtained:

$$-\lambda = \sum_{i=1}^{s} \sum_{k=1}^{n} Pr(\omega_i | x_k, \varphi_i) = n \tag{A1.3}$$

Then, from Eq. (A1.2), the following expression is obtained:

$$Pr(\omega_i) = \frac{1}{n} \sum_{k=1}^{n} Pr(\omega_i | x_k, \varphi_i) \tag{A1.4}$$

Appendix 2:
Show Jensen's inequality $H(\Phi,\Phi) \geq H(\Phi,\bar{\Phi})$, where

$$H(\Phi,\Phi) = \int \log f(\mathbf{y}|\mathbf{x},\Phi) f(\mathbf{y}|\mathbf{x},\Phi) d\mathbf{y}$$

$$H(\Phi,\bar{\Phi}) = \int \log f(\mathbf{y}|\mathbf{x},\bar{\Phi}) f(\mathbf{y}|\mathbf{x},\Phi) d\mathbf{y}$$

This is shown by subtracting the above two expressions:

$$H(\Phi,\bar{\Phi}) - H(\Phi,\Phi) \tag{A2.1}$$

$$= \int (\log f(\mathbf{y}|\mathbf{x},\bar{\Phi}) - \log f(\mathbf{y}|\mathbf{x},\Phi)) f(\mathbf{y}|\mathbf{x},\Phi) d\mathbf{y}$$

$$= \int (\log \frac{f(\mathbf{y}|\mathbf{x},\bar{\Phi})}{f(\mathbf{y}|\mathbf{x},\Phi)}) f(\mathbf{y}|\mathbf{x},\Phi) d\mathbf{y}$$

$$\leq \int (\frac{f(\mathbf{y}|\mathbf{x},\bar{\Phi})}{f(\mathbf{y}|\mathbf{x},\Phi)} - 1) f(\mathbf{y}|\mathbf{x},\Phi) d\mathbf{y} \quad \text{since } \log x \leq x - 1$$

$$= \int f(\mathbf{y}|\mathbf{x},\bar{\Phi}) d\mathbf{y} - \int f(\mathbf{y}|\mathbf{x},\Phi) d\mathbf{y} = 0$$

Equality is given when $f(\mathbf{y}|\mathbf{x},\bar{\Phi}) = f(\mathbf{y}|\mathbf{x},\Phi)$

INDEX