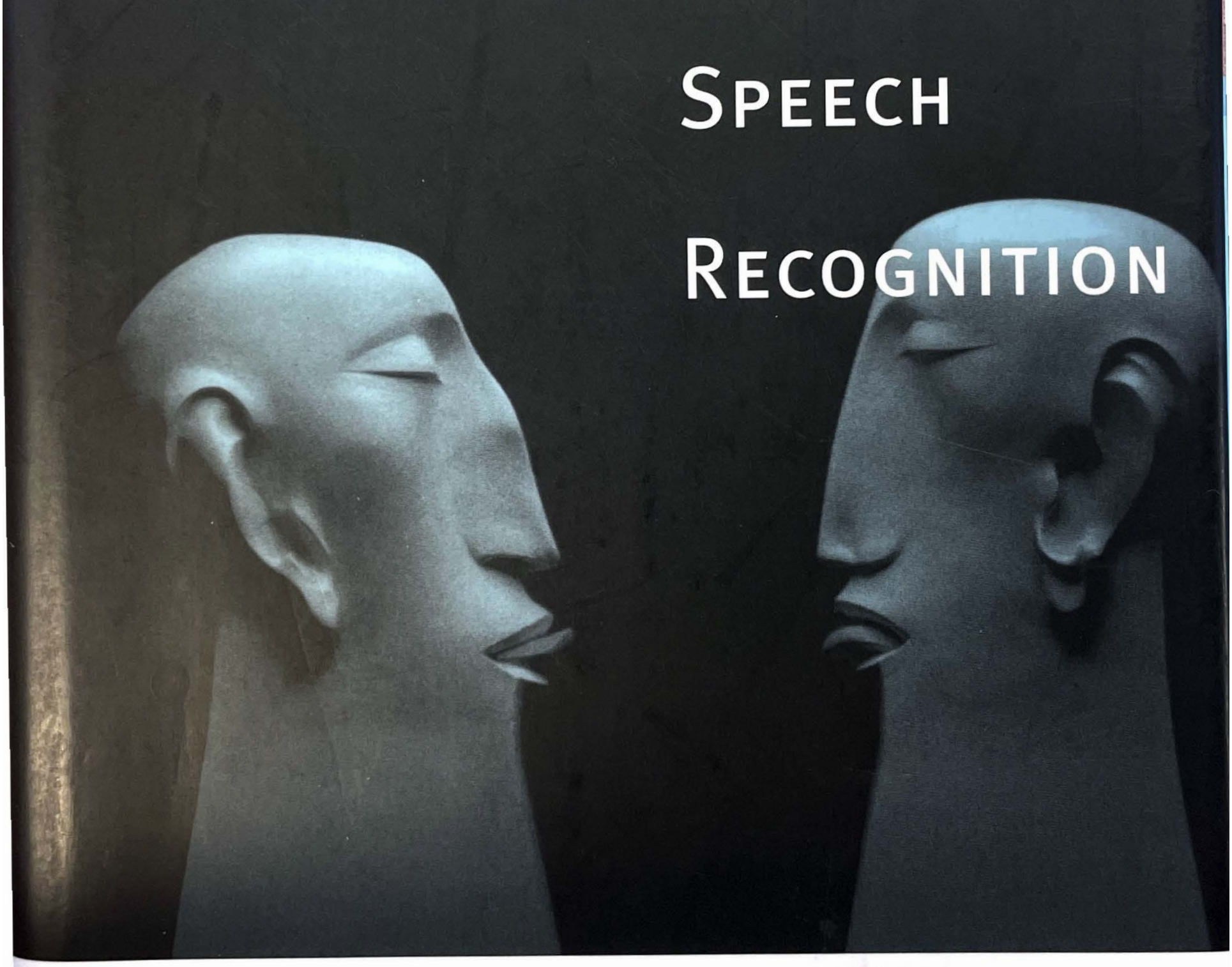
Statistical Methods for Speech Recognition



Frederick Jelinek

Statistical Methods for Speech Recognition

Frederick Jelinek

The MIT Press
Cambridge, Massachusetts
London, England

Third printing, 2001

This book was set in Times New Roman by Asco Typesetters, Hong Kong.

Printed and bound in the United States of America.

Library of Congress Cataloging-in-Publication Data

Jelinek, Frederick, 1932–
Statistical methods for speech recognition / Frederick Jelinek.
p. cm. — (Language, speech, and communication)
Includes bibliographical references and index.
ISBN 0-262-10066-5 (alk. paper)
1. Automatic speech recognition—Statistical methods. I. Title.
II. Series.
TK7895.S65J45 1998 97-34860
006.4′54—dc21 CIP

Contents

Chapter 1

The Speech Recognition Problem

1.1 Introduction

A speech recognizer is a device that automatically transcribes speech into text. It can be thought of as a voice-actuated "typewriter" in which a computer program carries out the transcription and the transcribed text appears on a workstation display. The recognizer is usually based on some finite vocabulary that restricts the words that can be "printed" out. Until we state otherwise, the designation *word* denotes a *word form* defined by its spelling. Two differently spelled inflections or derivations of the same stem are considered different words (e.g., *table* and *tables*). Homographs having different parts of speech (e.g., *absent* [VERB] and *absent* [ADJECTIVE]) or meanings (e.g., *bank* [FINANCIAL INSTITUTION] and *bank* [OF A RIVER]) constitute the same word.

Figure 1.1, the speech waveform corresponding to an utterance of the chess move "*Bishop moves to king knight five*," illustrates our problem. The waveform was produced at Stanford University in the late 1960s in a speech recognition project undertaken by Raj Reddy [1].

Because the field was then in its infancy, Reddy tried to give himself all the conceivable but fair advantages. Recognition of spoken chess moves could be based on a small vocabulary, a restricted syntax, and a relatively complete world knowledge: The system knew which chess moves were legal, so that, for instance, a recognition hypothesis corresponding to a move of a piece to a square occupied by another piece of the same color could immediately be rejected. The speech was recorded by a very good microphone in a quiet environment, and the system was adjusted (as well as the state of the art allowed) to the speaker.

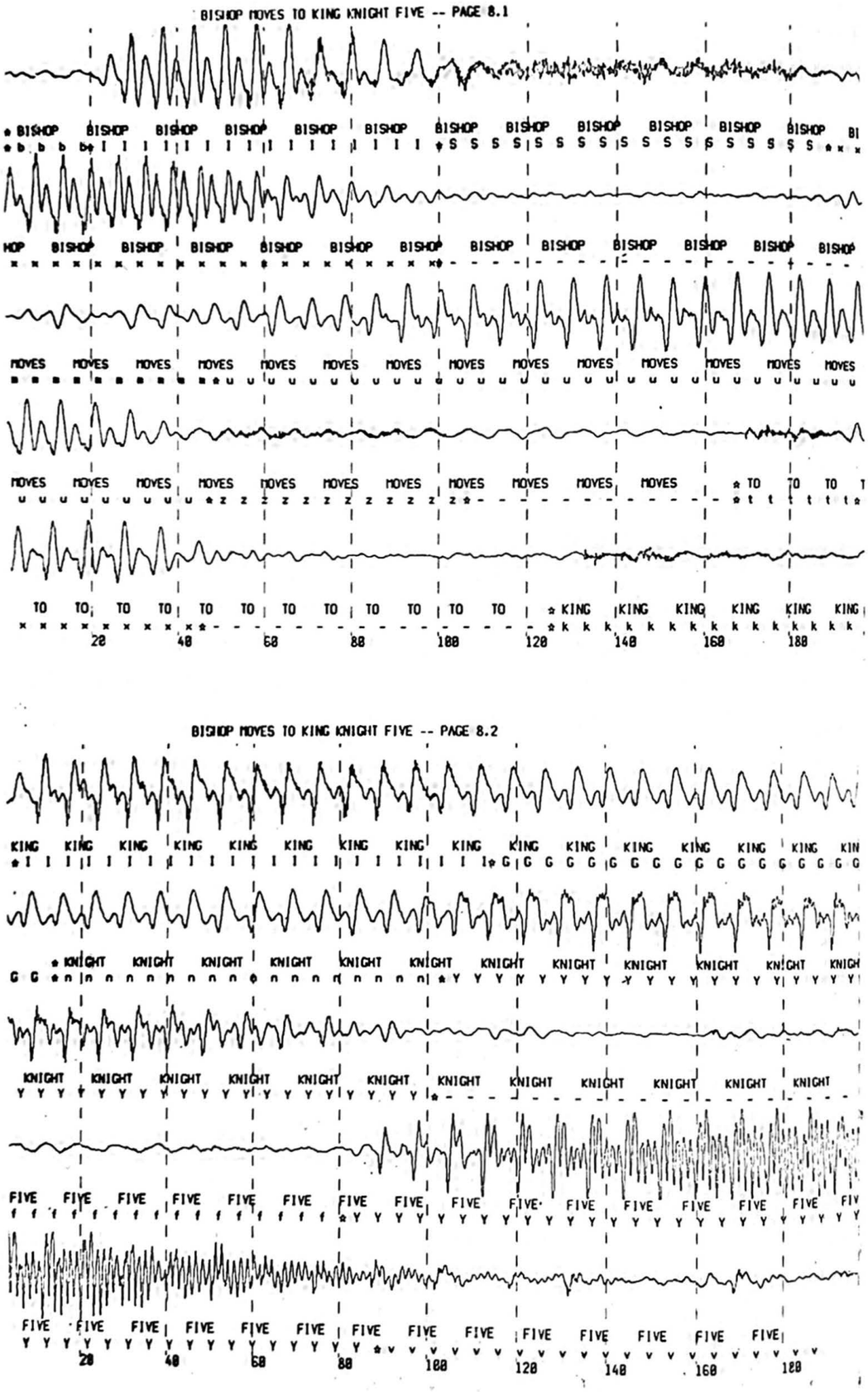


Figure 1.1
The sentence "Bishop moves to king knight five" aligned with its speech waveform

In early design strategy of the field, a recognizer would segment the speech into successive phones (basic pronunciation units [2]), then identify the particular phones corresponding to the segments, and finally transcribe the recognized phone strings into an English text.

Figure 1.1 aligns the speech waveform with the spoken words. Inspecting it, we can see that although the boundary between different speech events cannot be accurately placed, distinct events certainly appear in succession. So even though the boundaries are fuzzy, most of the segments seem to contain repeating nuclei that are candidates for recognition. Although this does not hold for stops[1] like *b* (see the beginning of BISHOP) or *k* (beginning of KING), it does seem to hold for vowels such as *i* (BISHOP and KING) or *u* (the first vowel of MOVES). But inspecting figure 1.1 more closely, we encounter a very big problem: The *i* of KING looks much more like the *u* of MOVES than it does like the *i* of BISHOP! So visually similar waveforms do not necessarily indicate perceptually similar sounds.

Actually, context is involved here, and the apparent waveform similarities are due to the influence of the nasality of the sound of *ng* that follows the *i* and of the sound of *m* that precedes the *u* sound.

Having now indicated an aspect of why speech recognition is not an easy task, we are ready to begin its serious consideration. In this chapter we will formulate the large vocabulary[2] speech recognition problem mathematically, which will result in a recognizer's natural breakup into its components. The chapter will conclude with the first example of self-organization from data: the basic vector quantization algorithm [11] that can be used to transform the speech signal into a sequence of symbols from a relatively limited (and automatically selected!) alphabet.

1. Stops, also called plosives, are sounds that consist of two parts, a stop portion and a release portion. English stops are *b*, *d*, *g*, *k*, *p*, *t*. [2]

2. Although no hard and fast distinction between small and large vocabulary tasks exists, here are some examples of each:

- Small vocabulary: recognition of digits, yes-no answers to questions, inventory control, etc.
- Large vocabulary: text creation by dictation, transcription of e-mail, transcription of telephone conversations, text creation for the handicapped.

Small vocabulary tasks are of great economic importance. They are just not the subject of this book.

1.2 A Mathematical Formulation

To discuss the problem of speech recognizer design, we need its mathematical formulation. A precise statement of the problem leads directly to a fruitful decomposition into easier to treat subproblems. Our approach is statistical,[3] so the formulation will involve probabilities. Here it is [3] [4]:

Let **A** denote the acoustic evidence (data) on the basis of which the recognizer will make its decision about which words were spoken. Because we are dealing with digital computers, then without loss of generality we may assume that **A** is a sequence of symbols taken from some (possible very large) alphabet $\mathscr{A}$:

$$\mathbf{A} = a_1, a_2, \ldots, a_m \qquad a_i \in \mathscr{A} \tag{1}$$

The symbols a_i can be thought of as having been generated in time, as indicated by the index i.

Let

$$\mathbf{W} = w_1, w_2, \ldots, w_n \qquad w_i \in \mathscr{V} \tag{2}$$

denote a string of n words, each belonging to a fixed and known vocabulary $\mathscr{V}$.

If $P(\mathbf{W}|\mathbf{A})$ denotes the probability that the words **W** were spoken, given that the evidence **A** was observed, then the recognizer should decide in favor of a word string $\hat{\mathbf{W}}$ satisfying

$$\hat{\mathbf{W}} = \arg \max_{\mathbf{W}} P(\mathbf{W}|\mathbf{A}) \tag{3}$$

That is, the recognizer will pick the most likely word string given the observed acoustic evidence.

Of course, underlying the target formula (3) is the tacit assumption that all words of a message are equally important to the user, that is, that misrecognition does not carry a different penalty depending on which word was misrecognized. Under this philosophy the warning "*Fire!*"

3. No advanced results of probability or statistical theory will be used in this self-contained text. The student is required simply to be comfortable with statistical concepts and be able to manipulate them intuitively. So although nothing in this text presumes more than the knowledge of the first four chapters of a book like [5], the required sophistication can probably be gained only by completing an entire course.

carries no more importance in a crowded theater than an innocuous commercial announcement. But let that possible criticism pass.[4]

The well known Bayes' formula of probability theory allows us to rewrite the right-hand side probability of (3) as

$$P(\mathbf{W}|\mathbf{A}) = \frac{P(\mathbf{W})P(\mathbf{A}|\mathbf{W})}{P(\mathbf{A})} \tag{4}$$

where $P(\mathbf{W})$ is the probability that the word string $\mathbf{W}$ will be uttered, $P(\mathbf{A}|\mathbf{W})$ is the probability that when the speaker says $\mathbf{W}$ the acoustic evidence $\mathbf{A}$ will be observed, and $P(\mathbf{A})$ is the average probability that $\mathbf{A}$ will be observed. That is,

$$P(\mathbf{A}) = \sum_{\mathbf{W}'} P(\mathbf{W}')P(\mathbf{A}|\mathbf{W}') \tag{5}$$

Since the maximization in (3) is carried out with the variable $\mathbf{A}$ fixed (there is no other acoustic data save the one we are given), it follows from (3) and (4) that the recognizer's aim is to find the word string $\hat{\mathbf{W}}$ that maximizes the product $P(\mathbf{W})P(\mathbf{A}|\mathbf{W})$, that is, it satisfies

$$\hat{\mathbf{W}} = \arg \max_{\mathbf{W}} P(\mathbf{W})P(\mathbf{A}|\mathbf{W}) \tag{6}$$

1.3 Components of a Speech Recognizer

Formula (6) determines what processes and components are of concern in the design of a speech recognizer.

1.3.1 Acoustic Processing

First, it is necessary to decide what acoustic data $\mathbf{A}$ will be observed. That is, one needs to decide on a *front end* that will transform the pressure waveform (which is what sound is) into the symbols a_i with which the recognizer will deal. So in principle, this front end includes a microphone whose output is an electric signal, a means of sampling that signal, and a manner of processing the resulting sequence of samples.

4. Strictly speaking, formula (3) is appropriate only if we are after a perfect transcription of the utterance, that is, if one error is as bad as many. Were we to accept that errors are inevitable (which they certainly are) and aim explicitly at minimizing their number, a much more complex formula would be required. So our formula only approximates (it turns out very fruitfully) what we are intuitively after.

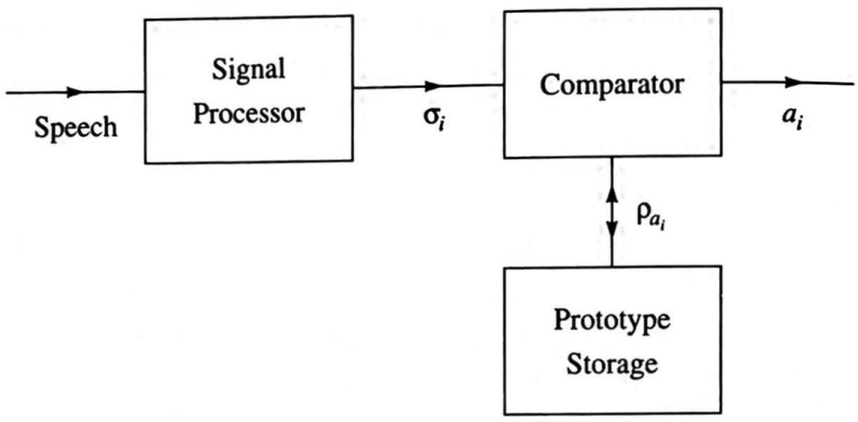


Figure 1.2
A schematic diagram of a recognizer front end (acoustic processor)

In this book, dedicated to statistical models of speech recognition, we will not address the front-end problem, except in the penultimate section of this chapter.[5] For the present, we will assume that the alphabet $\mathscr{A}$ is simply given. Those interested in front-end design should consult the many books and articles that thoroughly discuss *signal processing* [6] [7]. For the reader to gain some idea of what might be involved, however, we present in figure 1.2 a schematic diagram of a rudimentary front end (*acoustic processor*).

We can think of the *signal processor* as a device that at regular intervals of time (e.g., a hundred times per second[6]) generates real-valued vectors σ_i. The components of σ_i could be sample values of outputs of band-pass filters applied to the signal coming out of the microphone.

The *prototype storage* contains a set of vector prototypes $\mathscr{R} = \{\rho_1, \rho_2, \ldots, \rho_K\}$ of the same kind as σ_i. The *comparator* finds the closest element of $\mathscr{R}$ to σ_i, and the index of that element *is* the acoustic symbol a_i. To be precise,

$$\hat{J} = \arg \min_{j=1}^{K} d(\sigma_i, \rho_j) \tag{7}$$

and

5. Obviously, good signal processing is crucial and is the subject of intensive research. This book is about extracting information from the processed signal. Bad processing means loss of information: There is less of it to extract.

6. This happens to be the prevalent standard in our field.

$$a_i = \hat{J} \tag{8}$$

In (7), $d(\ ,\ )$ denotes a suitable distance function.[7]

In the penultimate section of this chapter we introduce a simple method, called *vector quantization*, that can derive the prototype set $\mathcal{R} = \{\rho_1, \rho_2, \ldots, \rho_K\}$ from speech data. There, the intuition behind the output symbol selection rule (7) and (8) will become apparent.

In most state-of-the-art large vocabulary recognition systems the comparator of figure 1.2 is omitted and the rest of the recognizer handles directly the signal processor outputs σ_i. They then constitute the observable symbols a_i. To introduce the relevant methods in the simplest setting, however, we will assume an acoustic symbol alphabet size of the order of hundreds (the size 200 is very common), which will still allow us to deal with the essence of the problem. We will generalize certain important results to "continuous" vector spaces (for signal processor outputs σ) in chapter 9. The added complication is that statistical estimation for vector spaces requires parametric methods. Section 2.9.1 of the next chapter will briefly introduce an appropriate model.

1.3.2 Acoustic Modeling

Returning now to formula (6), the recognizer needs to be able to determine the value $P(\mathbf{A}|\mathbf{W})$ of the probability that when the speaker uttered the word sequence $\mathbf{W}$ the acoustic processor produced the data $\mathbf{A}$. Since this number must be made available for all possible pairings of $\mathbf{W}$ with $\mathbf{A}$, it follows that it must be *computable "on the fly."* The number of different possible values of $\mathbf{A}$ and $\mathbf{W}$ is just too large to permit a lookup.

Thus to compute $P(\mathbf{A}|\mathbf{W})$ we need a statistical *acoustic model* of the speaker's interaction with the acoustic processor. The total process we are modeling involves the way the speaker *pronounces* the words of $\mathbf{W}$, the ambience (room noise, reverberation, etc.), the microphone placement and characteristics, and the acoustic processing performed by the front end.

The usual acoustic model employed in speech recognizers, the *hidden Markov model*, will be discussed in the next chapters. Other models are possible, for instance those based on *artificial neural networks* [8] or on *dynamic time warping* [9]. These methods are not treated in this book.

7. A very adequate distance is Euclidean:

$$d(\mathbf{x}, \mathbf{y}) \doteq \sqrt{\sum_i (x_i - y_i)^2}$$

1.3.3 Language Modeling

Formula (6) further requires that we be able to compute for every word string $\mathbf{W}$ the a priori probability $P(\mathbf{W})$ that the speaker wishes to utter $\mathbf{W}$. Bayes' formula allows many decompositions of $P(\mathbf{W})$, but because the recognizer "naturally" wishes to convey the text in the sequence in which it was spoken, we will use the decomposition

$$P(\mathbf{W}) = \prod_{i=1}^{n} P(w_i|w_1, \ldots, w_{i-1}) \tag{9}$$

The recognizer must thus be able to determine estimates of the probabilities $P(w_i|w_1, \ldots, w_{i-1})$. We use the term *estimate* on purpose, because even for moderate values of i and vocabularies of reasonable size, the probability $P(w_i|w_1, \ldots, w_{i-1})$ has just too many arguments. In fact, if $|\mathcal{V}|$ denotes the size of the vocabulary, then for $|\mathcal{V}| = 20{,}000$ and $i = 3$, the number of arguments is 8×10^{12}.

It is, of course, absurd to think that the speaker's choice of his i^{th} word depends on the entire *history* $w_1, \ldots, w_{i-1}$ of all of his previous speech. It is therefore natural that for purposes of the choice of w_i, the history be put into *equivalence classes* $\Phi(w_1, \ldots, w_{i-1})$. Thus in reality formula (9) becomes

$$P(\mathbf{W}) = \prod_{i=1}^{n} P(w_i|\Phi(w_1, \ldots, w_{i-1})) \tag{10}$$

and the art of *language modeling* consists of determining the appropriate equivalence classification Φ and a method of estimating the probabilities $P(w_i|\Phi(w_1, \ldots, w_{i-1}))$.

It is worth stressing that the language model used should depend on the use to which the recognizer will be put. The transcription of dictated radiological reports requires different language models than the writing of movie reviews. If text is to be produced, then the language model may reasonably be constructed by processing examples of corresponding written materials. It will then depend on text only and not in any way on speech.

1.3.4 Hypothesis Search

Finally, to find the desired transcription $\hat{\mathbf{W}}$ of the acoustic data $\mathbf{A}$ by formula (6), we must search over all possible word strings $\mathbf{W}$ to find the maximizing one. This search cannot be conducted by brute force: The space of $\mathbf{W}$s is astronomically large.

A parsimonious hypothesis search is needed that will not even consider the overwhelming number of possible candidates **W** and will examine only those word strings in some way suggested by the acoustics **A**. We will devote chapters 5 and 6 to showing two ways to proceed.

1.3.5 The Source-Channel Model of Speech Recognition

Our formulation leads to the schematic diagram of figure 1.3. The human speaker is shown as consisting of two parts: The source of the communication is his mind, which specifies the words **W** that will be pronounced by his vocal apparatus, the speech producer. The recognizer also consists of two parts, the acoustic processor and the linguistic decoder, the latter containing the acoustic and language models and the hypothesis search algorithm. Figure 1.3 embeds the process into the communication theory [10] framework: the source (modeled by the language model), the "noisy" channel (consisting of the tandem of the speech producer and the acoustic processor and modeled by the acoustic model), and the *linguistic decoder*.

The only difference between our communication situation and the standard one is that in the standard, the system designer can introduce an encoder and a modulator (signal generator) between the source and the channel. We must make do with the coder-modulator that evolution has bequeathed to us: human language and speech.

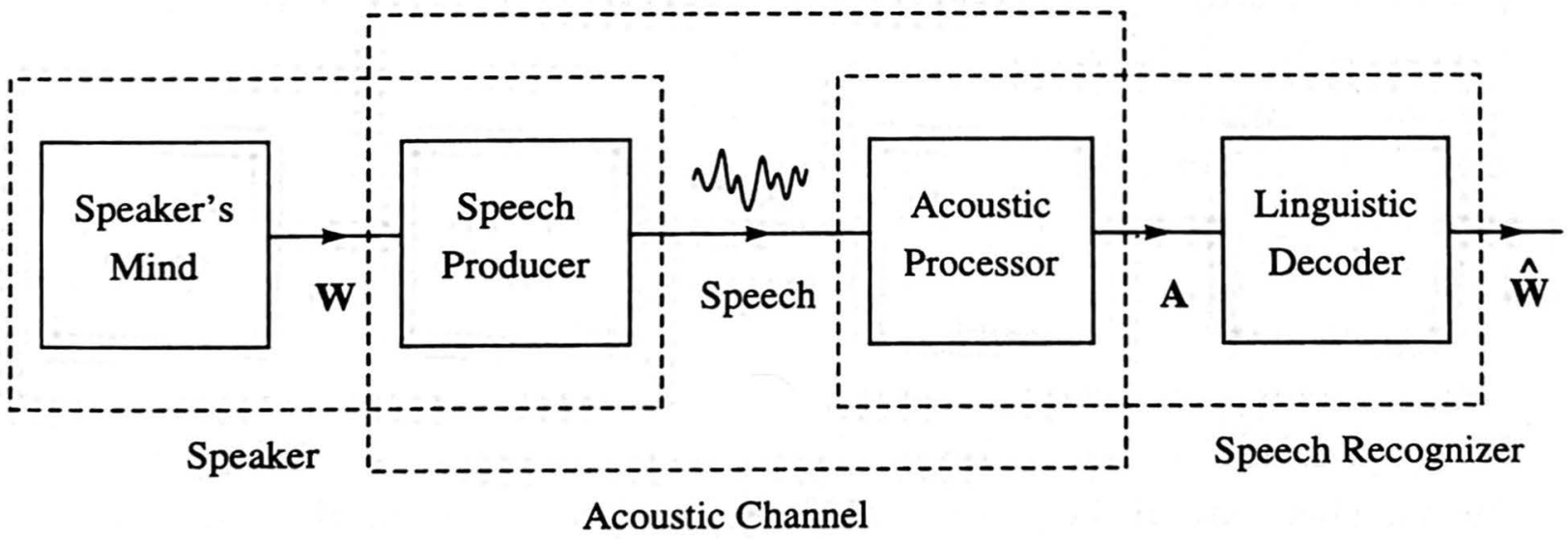


Figure 1.3
Source-channel model of speech recognition

1.4 About This Book

The previous section revealed the topics with which speech recognition research must be concerned. This book investigates methods that would accomplish the tasks outlined in sections 1.3.2 through 1.3.4. The headings of the above sections use the expression *modeling* because the required values of $P(\mathbf{A}|\mathbf{W})$ and $P(\mathbf{W})$ can only be produced by evaluating probabilities of output strings generated by abstract models of corresponding processes. These models will have no more than a mathematical reality. No claims whatever can conceivably be made about their relation to humans' actual speech production or recognition.

For all practical purposes the rest of the book is devoted to the structure of models of progressively higher sophistication, to their parameters, and to the problem of estimating the parameter values from actual speech data.

The plan of this text is to present first (chapters 1 through 5) each basic component of a large vocabulary speech recognizer (acoustic model, language model, hypothesis search) and devote the rest of the book (chapters 6 through 15) to refinements.

1.5 Vector Quantization

We conclude this chapter by introducing a simple method that can be used to find appropriate vector prototypes $\mathcal{R} = \{\rho_1, \rho_2, \ldots, \rho_K\}$ for the prototype storage of the acoustic processor of figure 1.2 (see the discussion of section 1.3.1) [11].

Consider the real vectors σ_i put out periodically by the signal processor. If their dimension is L, then they can be regarded as points in the real L-dimensional space. As speech is input to the signal processor, the space is being occupied by the points σ_i. If speech can be represented as a succession of *phones* [2] produced by the speaker (see the pronunciation specification of words in any dictionary), then the points σ_i will *cluster* in regions of the L-space characteristic of the particular phones the speaker produced. Therefore, if the prototypes $\rho_j \in \mathcal{R}$ are selected to be the centers of the σ-clusters, then the nearest prototype to an observed σ_i will be an appropriate *representative* of σ_i. This in fact is the idea behind the output symbol selection rule (7) and (8).

Vector quantization is a simple and effective method for finding cluster centers. Along with the rule (7) it presupposes a distance measure $d(\ ,\ )$ between points of the L-space. It turns out that a very adequate measure

is the Euclidean distance

$$d(\sigma,\rho) = \sqrt{\sum_{l=1}^{L}(\sigma(l) - \rho(l))^2} \tag{11}$$

where $\sigma(l)$ and $\rho(l)$ denote the l^{th} components of the vectors σ and ρ.

The Vector Quantization Algorithm[8]

1. Select K, the number of clusters.
2. Send speech through the signal processor (see figure 1.2) obtaining vectors σ_i for $i = 1, 2, \ldots, N$ (the total amount of speech, proportional to N, must be judiciously chosen).[9]
3. Select uniformly at random K initial candidate cluster centers ρ_j^0 from among the speech vectors $\{\sigma_1, \sigma_2, \ldots, \sigma_N\}$, that is,

$$P\{\rho_j^0 = \sigma_i\} = \frac{1}{N}$$

4. Partition $\{\sigma_1, \sigma_2, \ldots, \sigma_N\}$ into K regions $\mathcal{S}_j^0$ comprising all σ_i nearer to ρ_j^0 than to any other ρ_h^0, $h \neq j$. That is,

$$\sigma_i \in \mathcal{S}_j^0 \qquad \text{if } d(\sigma_i, \rho_j^0) \leq d(\sigma_i, \rho_h^0) \text{ for all } h$$

5. Find the *center of gravity* ρ_j^1 of each collection $\mathcal{S}_j^0$. That is,

$$\rho_j^1 = \arg\min_{\rho} \sum_{\sigma_i \in \mathcal{S}_j^0} d(\sigma_i, \rho)$$

6. Repartition $\{\sigma_1, \sigma_2, \ldots, \sigma_N\}$ into K regions $\mathcal{S}_j^1$ comprising all σ_i nearer to ρ_j^1 than to any other ρ_h^1, $h \neq j$.
7. Find the center of gravity ρ_j^2 of each collection $\mathcal{S}_j^1$.
8. And so on.

The particular method of finding clusters by alternately determining nearest neighbors of candidate centers and then locating neighborhood centers is also referred to as *K-means clustering*.

8. As presented here, the algorithm contains the idea's essence. In practice, to obtain good recognition, many refinements are necessary that are the result of intensive experimentation. For instance, the selection of initial cluster centers in step 3 may not in fact be carried out at random. This is the case with the vast majority of algorithms presented in this book: We describe the basic idea that must then be worked out in practice.

9. Five minutes of speech is usually adequate.

1.6 Additional Reading

The reader interested in finding out something about the human aspects of the speech production and perception process might start with Denes and Pinson [12] and continue with a more up-to-date article by Allen [13].

As pointed out in section 3.1, even though speech recognition depends crucially on the appropriate choice of signal processing, the latter is not a subject of this book. In addition to the references provided in that section [6] [7], it may be worthwhile to browse through a more recent book [14] or to consult articles by Cohen [15] and Picone [16] aimed specifically at signal processing for speech.

To simplify exposition of speech recognition's basic ideas and algorithms, we have limited ourselves to discrete outputs from the acoustic processor. The discretization is generally obtained by vector quantization (section 1.5) [11], about which it is possible to prove many interesting mathematical properties [17]. Unfortunately, vector quantization does in general lose some important information that would be available to the rest of the recognizer if it were fed by the signal processor's raw output.[10] More sophisticated methods of vector quantization can alleviate such loss [18] [19] [20]. Furthermore, a very interesting discretization method called *ranks* [21] both is robust and facilitates recognition results comparable to the very best in the state of the art. So dealing with discrete but sophisticated acoustic recognizer outputs turns out not to be much of a compromise after all.

In this book, we present what we think are our field's most fruitful approaches. Some leading contributors maintain that these are inadequate and that radical innovations are necessary to even approach the solution of the problem [22]. Twenty-eight years ago, a very famous communication engineer, N.R. Pierce, felt that ours was a hopeless endeavor [23].

References

[1] D.R. Reddy, "An approach to computer speech recognition by direct analysis of the speech wave," Tech. Report No. CS49, Computer Science Department, Stanford University, Sept. 1966.

[2] P. Ladefoged, *A Course in Phonetics*, Harcourt Brace Jovanovich, New York, 1975.

10. We do take up this eventuality in chapter 9.

[3] L.R. Bahl, F. Jelinek, and R.L. Mercer, "A maximum likelihood approach to continuous speech recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-5, pp. 179–90, March 1983.

[4] F. Jelinek, L.R. Bahl, and R.L. Mercer, "Design of a linguistic statistical decoder for the recognition of continuous speech," *IEEE Transactions on Information Theory*, vol. IT-21, pp. 250–56, 1975.

[5] A.F. Karr: *Probability*, Springer Verlag, New York, 1993.

[6] L.R. Rabiner and B-H Juang: *Fundamentals of Speech Recognition*, Prentice-Hall, Englewood Cliffs, NJ, 1993.

[7] L.R. Rabiner and R.W. Schafer, *Digital Processing of Speech Signals*, Prentice-Hall, Englewood Cliffs, NJ, 1978.

[8] R.P. Lippmann, "Review of neural networks for speech recognition," in *Readings in Speech Recognition*, A. Waibel and K.F. Lee, eds., Morgan Kaufmann, San Mateo, CA, 1990.

[9] L.R. Rabiner and S.E. Levinson, "Isolated and Connected Word Recognition—Theory and Selected Applications," *IEEE Transactions on Communications*, vol. COM-29, no. 5, pp. 621–69, May 1981.

[10] R.E. Ziemer and W.H. Tranter, *Principles of Communications*, John Wiley, New York, 1995.

[11] R.M. Gray, "Vector quantization," *IEEE ASSP Magazine*, vol. 1, pp. 4–29, April 1984.

[12] P.B. Denes and E.N. Pinson, *The Speech Chain: The Physics and Biology of Spoken Language*, Doubleday, New York, 1973.

[13] J.B. Allen, "How do humans process and recognize speech?" *IEEE Transactions on Speech and Audio Processing*, vol. 2, no. 4, pp. 567–77, October 1994.

[14] J.R. Deller, J.G. Proakis, and J.H.L. Hansen, *Discrete-Time Processing of Speech Signals*, Macmillan, New York, 1993.

[15] J.R. Cohen, "Application of an auditory model to speech recognition," *Journal of the Acoustic Society of America*, vol. 85, no. 6, pp. 2623–29, June 1989.

[16] J.W. Picone, "Signal modeling techniques in speech recognition," *Proceedings of the IEEE*, vol. 81, no. 9, pp. 1215–47.

[17] R.M. Gray and A. Gersho, *Vector Quantization and Signal Compression*, Klewer, Boston, MA, 1991.

[18] D. Rtischev, *Speaker Adaptation in a Large Vocabulary Speech Recognition System*, M.S. thesis, Massachussetts Institute of Technology, Cambridge, January 1989.

[19] A. Nadas, D. Nahamoo, and M.A. Picheny, "Speech recognition using noise adaptive prototypes," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 10, pp. 1495–1503, October 1989.

[20] L.R. Bahl, P.V. deSouza, P.S. Gopalakrishnan, and M.A. Picheny, "Context dependent vector quantization for continuous speech recognition," *Proceedings of*

the International Conference on Acoustics, Speech, and Signal Processing, vol. II, pp. 632–35, Minneapolis, MN, 1993.

[21] L.R. Bahl, P.V. deSouza, P.S. Gopalakrishnan, D. Nahamoo, and M.A. Picheny, "Robust methods for using context-dependent features and models in a continuous speech recognizer," *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, vol. I, pp. 533–36, Adelaide, 1994.

[22] H. Bourlard, H. Hermansky, and N. Morgan, "Towards increasing speech recognition error rates," *Speech Communication*, vol. 18, no. 3, pp. 205–31, 1996.

[23] J.R. Pierce, "Whither speech recognition?" *Journal of the Acoustic Society of America*, vol. 46, pp. 1049–51, 1969.

Chapter 3
The Acoustic Model

3.1 Introduction

In this chapter we will show two methods by which to construct the acoustic model that the recognizer may use to compute the probabilities $P(\mathbf{A}|\mathbf{W})$ for any acoustic data string $\mathbf{A} = a_1, a_2, \ldots, a_m$ and hypothesized word string $\mathbf{W} = w_1, w_2, \ldots, w_n$. As chapter 1 pointed out, these probabilities are needed so the recognizer can search for the desired transcribed word string $\hat{\mathbf{W}}$ defined by

$$\hat{\mathbf{W}} \doteq \arg \max_{\mathbf{W}} P(\mathbf{A}|\mathbf{W})P(\mathbf{W})$$

We will restrict ourselves to finite acoustic data alphabets $\mathscr{A}$ with several hundreds of symbols (200 is standard). We will make the required generalization to real vector alphabets currently used in the most advanced continuous speech recognition systems after we introduce the Expectation-Maximization algorithm in chapter 9.

The acoustic model will be based on the hidden Markov model (HMM) concept introduced in chapter 2. The general approach will be as follows [1] [2] [3]:

The model for a word string $\mathbf{W}$ will be made up of a *concatenation* of models pertaining to the individual words w_i (in a more sophisticated approach these models may be influenced by the *context* w_{i-1} and w_{i+1}, but we will not concern ourselves with this potential complication at this time). The models for the individual words w_i belonging to the basic vocabulary $\mathscr{V}$ will themselves be made up as a concatenation of yet smaller HMMs, the basic building blocks of the acoustic model system. There are many ways to select such building blocks. This chapter will introduce two.

The need for building blocks is obvious: The vocabulary is large (in the ten thousands) and changeable, so it is practically impossible to tailor

models separately for individual words. The two types of acoustic models discussed here differ exactly in the nature of their building blocks. The *phonetic* acoustic model is based on an intuitive linguistic concept. On the other hand, the *fenonic* model is completely self-organized from data and at this stage of the discussion is based on whole words. The former model does not take into account intraword context; neither model as presented is capable of dealing with interword context (coarticulation). The consideration of the latter is deferred to chapter 12.

3.2 Phonetic Acoustic Models

The basic way of constructing HMMs for words is as follows:

1. Create a phonetic dictionary [4] [5] [6] for the vocabulary in question, that is, make available the correspondence between each word v and a sequence $\Phi(v) = \varphi_1, \varphi_2, \ldots, \varphi_{l_v}$, of symbols from a predetermined phonetic alphabet ϕ. $\Phi(v)$ is then an encoding of the pronunciation of v and is referred to as the *phonetic base form* of v.

This step involves making a decision about the phonetic alphabet ϕ to be used. Although the international phonetic alphabet (IPA) is prevalent [5], one could use an ordinary dictionary as a guide. For instance, the *American Heritage Dictionary* [7] gives the following pronunciations:

aluminum = ə l oo m ə n ə m

green = g r ē n

Worcester = w o͝o s t ər

For some words (e.g., *either, the*) that have several fundamentally different valid pronunciations, multiple base forms must be provided. The phonetic alphabet usually distinguishes between stressed and unstressed vowels and includes silence and end-of-word symbols, and its size is of the order of 100.

2. To each symbol of the phonetic alphabet let there correspond an *elementary* HMM with distinguished starting and ending states.

Figure 3.1 shows the transition structure of an appropriate elementary HMM. Note that it generates at least one output symbol and possibly an unlimited number of them.

3. The HMM for a word v is a concatenation of the elementary HMMs specified by the sequence $\Phi(v)$, where the final state of one HMM is con-

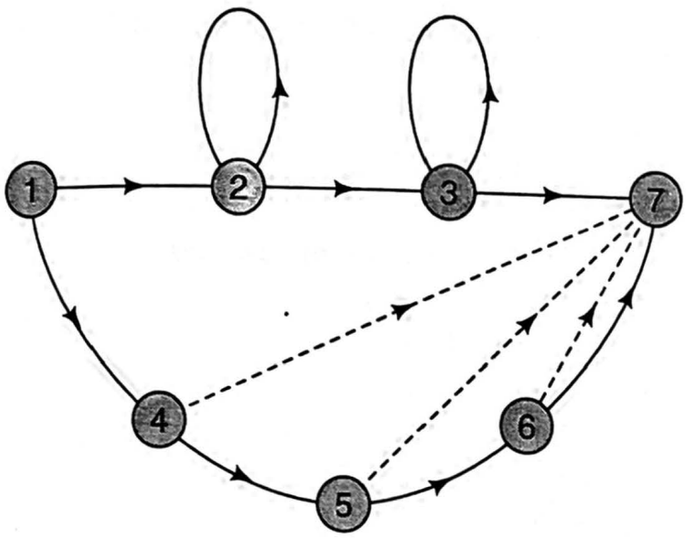


Figure 3.1
Hidden Markov model of acoustic string production by a phone

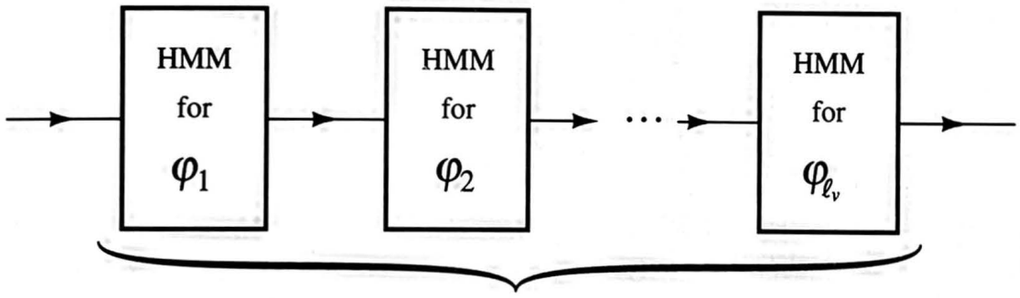


Figure 3.2
HMM for the word v as determined by its phonetic baseform

nected to the initial state of the following HMM by a null transition (see figure 3.2).

4. To get a *composite* model for a transcription **W** of some given speech data **A**, word models for the individual words w_i are concatenated by inserting between them elementary HMMs corresponding to silence symbols and/or end-of-word symbols (see figure 3.3).

5. The Baum-Welch algorithm (see section 2.7 of the preceding chapter) estimates the HMM parameters by letting users read a prepared text **W**, observing the acoustic processor's output **A** and using the composite HMM corresponding to **W** as a model of the production mechanism that resulted in the observed **A**.

Figure 3.3
Composite HMM for the training text $w_1, w_2, \ldots, w_n$

3.3 More on Acoustic Model Training

It may be useful to specify in somewhat more detail what is meant by step 5 in section 3.2. The literal description of HMM training (synonym for estimation of statistical parameters from data) in section 2.7 concerned one HMM that was to account for the production of all the training data **Y** from which the parameters were to be estimated. So, for instance, for each different transition t of the given HMM, one was to estimate the output probability $q(a|t)$. Clearly, this would be possible with any accuracy only if t were used repeatedly as **Y** was being produced.

But this is not necessarily the situation for the composite HMM constructed in step 4 of section 3.2. If all of its transitions were regarded as different, then (in one training iteration) many of them would be used very few times, and most not at all. So this is yet another aspect in which the building block construction is important. During the estimation process, those transitions of the composite HMM that correspond to the same transition in any given building block must be considered the same.[1] Only in this way will most transitions be used many times when the acoustic data **A** is produced.

To carry out training properly we can proceed as follows:

1. Establish an inventory of elementary HMMs. In the rudimentary case described in section 3.2, these correspond to the different phones of the phonetic alphabet ϕ plus the silence and end-of-word HMMs.
2. Give a different index to each different transition of the elementary HMM set. For instance, let $t_{j,k}$ denote the k^{th} transition of the j^{th} building block HMM.
3. For each different $t_{j,k}$ establish different counters $c(t_{j,k})$ and $c(a, t_{j,k})$ to be used as specified in equations (27) and (28) of chapter 2.
4. Identify the composite HMM's transitions by the proper index $t_{j,k}$ of the building block elementary HMM to which they belong.
5. Train the composite HMM as specified in section 2.7, contributing the amounts $P\{t^i = t_{j,k}\}$ and $P\{t^i = t_{j,k}\}\delta(a_{i+1}, a)$ to the $c(t_{j,k})$ and $c(a, t_{j,k})$ counters, respectively:

1. That is, if the j^{th} building block appears M times in the construction of the composite HMM, then each transition t of the j^{th} building block will appear in exactly M places in the composite HMM. All M of these will correspond to only one set of accumulators $c(a, t), a \in \mathscr{A}$ for the purpose of estimating the parameters $p(t)$ and $q(a|t)$ of the j^{th} building block.

$$c(t_{j,k}) = \sum_{i=0}^{k-1} P\{t^i = t_{j,k}\}$$

$$c(a, t_{j,k}) = \sum_{i=0}^{k-1} P\{t^i = t_{j,k}\}\,\delta(a_{i+1}, a)$$

3.4 The Effect of Context

The problem with the phonetic model[2] proposed in section 3.2 is that it does not take into account the influence of context on pronunciation. Thus in American speech an initial *t* (e.g., in *table*) is usually aspirated, while a final *t* (e.g., in *nest*) is not. One could try to fix this problem by making the phonetic alphabet allophonic,[3] but that would mean deciding what the allophonic alphabet should be, and hiring a golden-eared (and very conscientious and patient) phonetician to create an allophonic dictionary. We will revisit the allophonic problem in chapter 12 when we derive an allophonic inventory directly from speech data using decision tree methodology.

One way the speech recognition community is attacking the context problem is via *triphones* [8]. This amounts to deciding that the contextual influence of the preceding and the following phones is most important. Or, in allophonic terms, a phone φ is realized by allophones denoted by $x\varphi y$, where x and y range over the phonetic alphabet. The problem with *this* solution is that other, wider contexts may be important and that even in this relatively narrow case, there are potentially K^3 allophones, where K is the size of the phonetic alphabet. For the purposes of HMM parameter estimation this is too many, so triphones must be clustered, and that is again best done via the decision tree methodology discussed in chapter 12.

In this chapter we will solve the phonetic context problem by encoding base forms into a natural, data-driven alphabet that takes into account all intraword context. This solution is thus particularly suitable for isolated word recognition where (short) pauses between words effectively insulate their pronunciation from the surroundings. Compared to the phonetic base form approach of section 3.2, the method will lead to improved speech

2. That is, the problem with the inventory of the building blocks.

3. Allophones are the perceptually different realizations of the same phone [5].

recognition. Furthermore, it will be the basis for our "ultimate" approach to coarticulation (defined as acoustic influence of words on each other) introduced in chapter 12.

We must now make a slight detour and introduce the concept of *Viterbi alignment*.

3.5 Viterbi Alignment

Consider any acoustic data string $\mathbf{A} = a_1, a_2, \ldots, a_m$ and a particular HMM having an inventory of transitions $\mathcal{T}$. *Viterbi decoding* (see section 2.4) refers to finding the most likely sequence of transitions $\mathbf{T}_s = t_{s_1}, t_{s_2}, \ldots, t_{s_k}, (t_{s_i} \in \mathcal{T})$ to have generated $\mathbf{A}$.[4]

The subsequence $\mathbf{T}_o = t_{o_1}, t_{o_2}, \ldots, t_{o_m}$ consisting of the non-null (output producing) transitions of $\mathbf{T}_s$ is the basis of the Viterbi alignment of $\mathbf{A}$ with the HMM. It constitutes a labeling of the symbols a_i of $\mathbf{A}$ by the transitions t_{o_i} that can be thought of as having "caused" the outputs a_i.

If the HMM in question is made up of a concatenation of n elementary HMMs (e.g., the phonetic HMMs of section 3.2 or the word models resulting from their concatenation) then the labeling $\mathbf{T}_o$ effectively segments $\mathbf{A}$ into subsequences of symbols $\mathbf{A}_j^* = a_{l_{j-1}+1}, a_{l_{j-1}+2}, \ldots, a_{l_j}$ that were labeled by transitions belonging to the same elementary HMM. The label $t_{o_i}, i = l_{j-1}$ of the preceding symbol $a_{l_{j-1}}$ (i.e., the corresponding transition) belongs to a different elementary HMM and so does the label t_{o_h}, $h = l_j + 1$ of the succeeding symbol a_{l_j+1} (whereas the transitions $t_{o_{i+1}}, \ldots, t_{o_{h-1}}$, belong to the same elementary HMM).

The term *Viterbi (forced) alignment* refers to this segmentation $\mathbf{A} = \mathbf{A}_1^* \| \mathbf{A}_2^* \| \ldots \| \mathbf{A}_n^*$.

3.6 Singleton Fenonic Base Forms

In spite of the superior performance they facilitate, *fenonic* base forms are not widely used in current speech recognizers. We include their description because they are an excellent illustration of complete self-organization from data. They do not presuppose any phonetic concepts whatever. They are also a useful tool for modeling new words that do not belong to the prepared vocabulary $\mathcal{V}$.

4. Because the most likely path may involve null transitions, $k \geq m$.

Having explained Viterbi alignment, we can now address the problem of phonetic context. To establish fenonic base forms [9], an acoustic processor with a finite output alphabet is needed. This does not mean that the method applies only to systems that use such alphabets at run time. The method establishes *word models* from data before any actual recognition takes place. In fact, even in finite alphabet systems it might be useful to have a different acoustic processor at run time from the one used to define these base forms.

Speak a word, say *table*. The output of the acoustic processor is going to be a particular string $\mathbf{A}(table) = a_{i_1}, a_{i_2}, \ldots, a_{i_m}$ with a_{i_j} belonging to the acoustic alphabet $\mathscr{A}$.[5] The string $\mathbf{A}(table)$ is surely characteristic of the pronunciation of the word *table* and can thus be thought of as an *encoding* of that pronunciation.

Speak another word, say *famous*, and consider the consequent acoustic processor output string $\mathbf{A}(famous) = a_{j_1}, a_{j_2}, \ldots, a_{j_l}$. If it should turn out that $a_{i_k} = a_{j_n}$ for some k and n, then that means that the sound of *table* at time k was acoustically similar to the sound of *famous* at time n. It is in this sense that the string $\mathbf{A}(\langle word \rangle)$ constitutes an acoustically faithful encoding of the sound of $\langle word \rangle$.

It thus makes sense to use $\mathbf{A}(\langle word \rangle)$ as a *fenonic* baseform of $\langle word \rangle$, and create the HMM for $\langle word \rangle$ as a concatenation of elementary HMMs corresponding to the individual symbols making up the codeword $\mathbf{A}(\langle word \rangle)$.[6]

Caution: Do not be confused by the fact that the codeword $\mathbf{A}(\langle word \rangle)$ originated as the output string of an acoustic processor! The individual symbols a_{i_h} will be used strictly as abstract identifiers of the particular elementary HMMs whose concatenation will form the composite HMM for $\langle word \rangle$. These identifiers simply serve to identically index acoustically similar segments. The nature of the identifiers' acoustic realization will be determined by training.

5. The string $a_{i_1}, a_{i_2}, \ldots, a_{i_m}$ can be obtained from a Viterbi alignment of speech data $\mathbf{A}$ that included the utterance of the word *table*. This alignment can be based on a composite HMM consisting of previously trained *phonetic acoustic word models*. This composite HMM is thought of as having generated $\mathbf{A}$, with the subsequence $a_{i_1}, a_{i_2}, \ldots, a_{i_m}$ aligned with the transitions belonging to the HMM for *table*.

6. The "fe" in the made-up term *fenonic* stands for "front end". The suffix "nonic" is intended to lend the term scientific respectability.

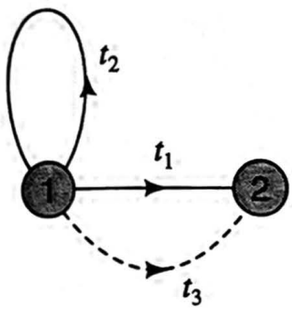


Figure 3.4
Structure of an elementary hidden Markov model of a fenon

Figure 3.4 shows the structure of the elementary HMMs to be used as building blocks. Their statistical parameters will be estimated from data as section 3.3 outlined. Since the length of the fenonic base form is equal to the number of centiseconds (it is assumed that outputs are produced at the rate of 100 per second, but other rates are acceptable) it took to pronounce the word, it is reasonable to expect that after training, the probability $P(t_1)$ of the direct transition in figure 3.4 will turn out to be close to 1 in all the elementary HMMs of the set. The loop and the null transitions in figure 3.4 provide the flexibility that allows the duration of subsequent pronunciations of $\langle word \rangle$ to be different from the one that gave rise to the fenonic baseform.

Different elementary HMMs will have different output probabilities $q(a|t_i)$ associated with the two output producing transitions in figure 3.4. Their values will be derived from training. We will take what comes, but it will not be a coincidence if it should turn out for both non-null transitions t_i belonging to the HMM designated by the *fenonic identifier a* that[7]

$$q(a|t_i) > q(a'|t_i) \qquad \text{for all } a' \neq a$$

3.7 A Needed Generalization

Singleton fenonic base forms have three problems:

1. Base forms are obtained from a single pronunciation of a word that may turn out to have been irregular (coincidental).
2. The base-form creation process requires the user to pronounce each word of a large vocabulary—an impossibly time-consuming task.

7. This should hold assuming the same acoustic processor is used at run time as was used to establish fenonic base forms.

3. The base forms do not take into account word context: the influence of the preceding and following words on the pronunciation of the current one.

In fact, it is clear that

1. To avoid atypicality, base forms should be produced from several sample pronunciations of the word.
2. Base forms for a large vocabulary should be produced once and for all and must be the same for each user. Only the statistical parameters $p(t_i)$ and $q(y|t_i)$ can conceivably be speaker dependent.

We are about to see how to construct speaker independent fenonic base forms from multiple recordings of vocabulary words by a variety of speakers. This will obviate the first two problems. The third problem either can be tolerated (this is certainly fine for isolated word recognition) or must be treated in some way similar to that discussed in chapter 12.

Suppose that we have recorded M utterances of some word v, $\mathbf{A}_i(v) = a_{i,1}, a_{i,2}, \ldots, a_{i,m_i}$, $i = 1, 2, \ldots M$. Then the logical thing to seek is a fenonic base form $B^*(v)$ satisfying

$$\mathbf{B}^*(v) = \arg\max_{\mathbf{B}} \prod_{i=1}^{M} P_{\mathbf{B}}(\mathbf{A}_i(v)) \tag{1}$$

In equation (1) the strings $\mathbf{A}_i$ denote ordinary acoustic processor outputs and the subscript to the probability $P_{\mathbf{B}}$ signifies that the HMM producing them is constructed from elementary HMMs specified by the base form $\mathbf{B}$. Of course, this probability is well defined only after the statistical parameters of the HMMs have been estimated, but that is not the main problem.

The unfortunate fact is that the finding of $\mathbf{B}^*$ in (1) involves too large a search space: The duration of an average word exceeds 25 centiseconds, so the number of possible base forms exceeds L^{25} where L is the size of the fenonic alphabet. One possibility is the construction of *synthetic base forms* described in section 3.8. The process will again be based on Viterbi alignment.

3.8 Generation of Synthetic Base Forms

We start the construction by the following training bootstrap:

1. Record all data necessary for speaker independent base form creation for the entire vocabulary $\mathcal{V}$. Each word $v \in \mathcal{V}$ will be recorded by M different speakers.

2. Establish an acoustic processor output alphabet via an appropriate form of vector quantization (see section 1.5) applied to the entire data.
3. Create singleton base forms (by the method of section 3.6) by choosing at random from among the M recordings $\{\mathbf{A}_1(v), \ldots, \mathbf{A}_M(v)\}$ of each word v.
4. Train statistical parameters of the elementary HMMs making up the selected singleton base forms. (This is done on the entire recorded data for all words.)
5. For each word v, choose the "best" new base form $\mathbf{C}^*(v)$ from among the recorded set $\{\mathbf{A}_1(v), \ldots, \mathbf{A}_M(v)\}$:

$$\mathbf{C}^*(v) = \arg \max_{j=1}^{M} \prod_{i=1}^{M} P_{\mathbf{A}_j(v)}(\mathbf{A}_i(v)) \tag{2}$$

6. Using the new base forms $\mathbf{C}^*(v)$, reestimate the statistical parameters of the elementary HMMs (as in step 4).
7. Reselect "best" new base forms (as in step 5) based on the new statistical parameter values obtained in the previous step. Continue in a loop from step 5 until the base-form selection process has converged.

Comparing (2) with (1) we see that whereas we are after the best conceivable base form $\mathbf{B}^*(v)$ giving rise to the data, $\mathbf{C}^*(v)$ is only the best from among the alternative singleton baseforms $\{\mathbf{A}_1(v), \ldots, \mathbf{A}_M(v)\}$. We must therefore push on further.

In the above algorithm we have estimated the statistical parameters of the elementary HMMs that will be used to build up word HMMs according to the specification of the obtained fenonic base forms $\mathbf{C}^*(v)$. We are now ready to construct base forms more nearly satisfying (1) than do the base forms $\mathbf{C}^*(v)$ we selected so far.

The process starts by mutually aligning the symbols of the strings $\mathbf{A}_i = a_{i,1}, a_{i,2}, \ldots, a_{i,m_i}$[8] as follows:

Using $\mathbf{C}^*$ defined by equation (2) as the base-form specifier for word v, Viterbi-align the symbols of the strings $\mathbf{A}_i$. That is, for each $i = 1, 2, \ldots, M$ find the elementary HMM of the concatenation defined by the base form $\mathbf{C}^*$ that has most likely "produced" the symbol $a_{i,j}$ for each $j = 1, 2, \ldots, m_i$, where m_i is the number of symbols in $\mathbf{A}_i$.

8. To avoid complicating our notation unnecessarily, we are dropping the explicit argument v.

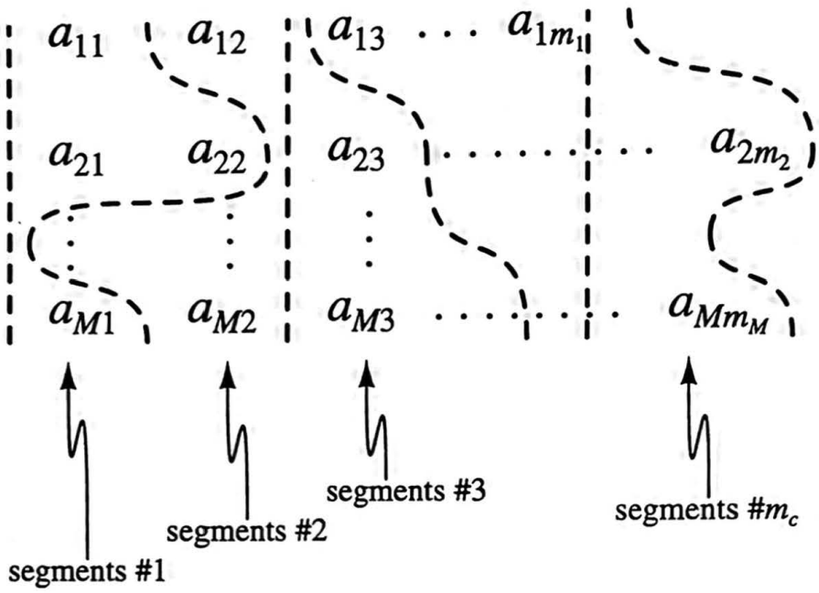


Figure 3.5
Mutual alignment of acoustic output strings corresponding to M different pronunciations of the same word

If m_C is the length of $\mathbf{C}^*$, the alignment will have segmented each string $\mathbf{A}_i$ into m_C segments (some of them possibly of length 0), one segment per symbol of $\mathbf{C}^*$. Figure 3.5 shows an example of this schematically.

Let $r_j(\mathbf{A}_i)$ denote the concatenation of the first j segments of the string $\mathbf{A}_i$. In the following, let the inventory of elementary HMMs include the *null HMM* consisting of a single null transition. We now proceed:

1. Find the best concatenation of elementary HMMs, $\mathbf{b}_1^* = b_{1,1}^*, b_{1,2}^*$ producing independently of each other all the first segments of the strings $\mathbf{A}_i$. That is,

$$\mathbf{b}_1^* = \arg\max_{\mathbf{b}_1} \prod_{i=1}^{M} P_{\mathbf{b}_1}(r_1(\mathbf{A}_i))$$

where $\mathbf{b}_1 = b_{1,1}, b_{1,2}$ runs over all pairs of elementary HMMs including the null HMM. Note that $\mathbf{b}_1^*$ may in fact consist of the null HMM (the concatenation of two null HMMs is equivalent to a single null HMM), or of one or two "real" (i.e., output producing) elementary HMMs.

2. Find the best HMM pair $b_{2,1}^*, b_{2,2}^*$ to concatenate with $\mathbf{b}_1^*$ so that the resulting HMM specified by $\mathbf{b}_2^* = \mathbf{b}_1^*, b_{2,1}^*, b_{2,2}^*$ is most likely to produce independently the initial segments $r_2(\mathbf{A}_i)$. That is, limited by the fixed choice $\mathbf{b}_1^*$ of the initial HMMs, $\mathbf{b}_2^*$ is chosen to maximize the value of the

product

$$\prod_{i=1}^{M} P_{\mathbf{b}_2}(r_2(\mathbf{A}_i))$$

where $\mathbf{b}_2 = \mathbf{b}_1^*, b_{2,1}, b_{2,2}$.

3. Continue in this vein, selecting the HMM pair $b_{j,1}^*, b_{j,2}^*$ so that $\mathbf{b}_j^* = \mathbf{b}_{j-1}^*, b_{j,1}^*, b_{j,2}^*$ will maximize the product

$$\prod_{i=1}^{M} P_{\mathbf{b}_j}(r_j(\mathbf{A}_i))$$

where $\mathbf{b}_j = \mathbf{b}_{j-1}^*, b_{j,1}, b_{j,2}$.

4. The string $\mathbf{B}^* = \mathbf{b}_{m_C}^*$ obtained in this way is the desired synthetic baseform derived from the pronunciations $\mathbf{A}_i = a_{i,1}, a_{i,2}, \ldots, a_{i,m_i}$. Simplify it by eliminating from it all null HMMs.

For best results, the above process should be iterated. That is, once the base forms $\mathbf{B}^*$ are obtained in the last step, the elementary HMMs are retrained on their basis. Then the basic strings $\mathbf{A}_i = a_{i,1}, a_{i,2}, \ldots, a_{i,m_i}$ are aligned with respect to the symbols of $\mathbf{B}^*$ (rather than $\mathbf{C}^*$) and the process starts again from step 1.

3.9 A Further Refinement

Synthetic base form construction can be improved by carrying on in the spirit of the following steps:

1. Find the best concatenation of elementary HMMs, $\mathbf{b}_1^* = b_{1,1}^*, b_{1,2}^*, b_{2,1}, b_{2,2}$ producing independently all the first two segments of the strings $\mathbf{A}_i$. That is,

$$\mathbf{b}_1^* = \arg\max_{\mathbf{b}_1} \prod_{i=1}^{M} P_{\mathbf{b}_1}(r_2(\mathbf{A}_i))$$

Note that $\mathbf{b}_1^*$ may in fact consist of the null HMM or of one to four "real" elementary HMMs.

2. Find the best HMM sequence $b_{2,1}^*, b_{2,2}^*, b_{3,1}, b_{3,2}$ to concatenate with $b_{1,1}^*, b_{1,2}^*$ so that the resulting HMM specified by $\mathbf{b}_2^* = b_{1,1}^*, b_{1,2}^*, b_{2,1}^*, b_{2,2}^*, b_{3,1}, b_{3,2}$ is most likely to produce independently the initial segments $r_3(\mathbf{A}_i)$. That is, limited by the fixed choice $b_{1,1}^*, b_{1,2}^*$ of the initial HMMs, $\mathbf{b}_2^*$ maximizes the value of the product

$$\prod_{i=1}^{M} P_{\mathbf{b}_2}(r_3(\mathbf{A}_i))$$

3. Continue as in step 2 for subsequent segments.

We need not elaborate the reasons why this improves the resulting synthetic baseform at the price of increased complexity of the search for it.

3.10 Singleton Base Forms for Words Outside the Vocabulary

Finally a remark on the virtues of singleton base forms. They provide an easy means for adding a model of new words to the vocabulary. The user need only say the word and provide its spelling, and the acoustic processor outputs directly specify the required HMM. In fact, this can be done at run time as part of the proofreading of the recognized text. Thus if I said *serendipity* and the system recognized *property* because *serendipity* was not in the vocabulary, I need only make the correction in the recognized text and I gain a singleton fenonic base form for *serendipity* if the recognizer is able to identify (by Viterbi alignment of the speech with the text as currently recognized—see section 3.5) the speech segment that it erroneously recognized as *property*.

3.11 Additional Reading

A very useful review of how to apply HMMs to speech recognition can be found in reference [10]. For a discussion of related practical issues encountered in hidden Markov modeling, see the article by Juang and Rabiner [11].

The inspiration for fenonic word models was no doubt the early work of Bakis [12] who derived HMMs directly from aligned data. His HMM structures are referred to as *Bakis models* and constitute the earliest use of continuous feature vector outputs of the signal processor as inputs to the linguistic decoder.[9]

The mainstream modeling presented in this chapter is based on acoustic processors generating outputs at regular time intervals. This need not be so. Mari Ostendorf and colleagues continue to make fruitful attempts to deal with other natural segmental units [13] [14] [15].

9. More on continuous processing will be found in chapter 9.

The Baum algorithm aims at estimating HMM parameters in a maximum likelihood way. That is, it adjusts the parameters of the models of the training script so as to maximize the probability of the models' producing the observed acoustic processor output string. This does not necessarily lead to the best discrimination, that is, speech recognition performance. What one would really wish is to maximize the a posteriori probability $P(\mathbf{W}|\mathbf{A})$ where **W** is the spoken word string and **A** is the observed acoustic processor output. This turns out to be a rather difficult problem requiring in practice various assumptions and approximations. The first attempts at solution can be found in [16] and [17]. A fundamental step forward was taken by Kanevsky and colleagues who came up with an adjustment to the Baum algorithm that allows training aimed at optimization of the a posteriori probability criterion [18]. A survey of these approaches can be found in [19].

Worth mentioning is also a different attempt to go beyond maximum likelihood in optimizing speech recognizer performance that iteratively adjusts the HMM parameter values so as to make correct and incorrect words more and less probable, respectively [20].

As we mentioned in chapter 1, it is possible to use models other than HMMs as a basis of speech recognition. Among these, artificial neural networks (ANNs) and dynamic time warping (DTW) are preeminent. Bourlard and Wellekens have pointed out interesting connections between HMMs and ANNs [21]. Richard and Lippmann showed (among other properties) that ANNs can be used to estimate a posteriori probabilities [22], and Bourlard and Morgan have taken advantage of this fact in formulating a combined HMM and ANN approach to speech recognition [23]. The system constructed by Robinson, Hochberg, and Renals achieves state-of-the-art results in speaker independent large vocabulary speech recognition [24].

Finally an excursion into the brief history of speech recognition. In the 1970s the dominant paradigm for small vocabulary isolated speech recognition was DTW. Its basic idea was to warp a prototype observation of an utterance into the unknown observed string and reach a decision among competing word candidates according to the warping penalty incurred. A very popular type of this penalty was *Itakura distance* [25]. DTW gave very good results, in fact, for its field of application, better ones than HMMs did (that is, until DTW was essentially abandoned) [26]. The main problems with this approach were (a) incorporation of language models was not natural, (b) the problem of construction of

synthetic prototypes remained unsolved,[10] and (c) a unified recognizer statistical formulation incorporating all speech recognizer modules was never found. Two good discussions of DTW applications can be found in references [27] and [28].

References

[1] L.R. Bahl and F. Jelinek, "Decoding for channels with insertions, deletions, and substitutions with applications to speech recognition," *IEEE Transactions on Information Theory*, IT-21, pp. 404–11, 1975.

[2] J.K. Baker, "The Dragon system—an overview," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-23, no. 1, pp. 24–29, February 1975.

[3] J.K. Baker, "Stochastic modeling for automatic speech understanding," *Speech Recognition*, D.R. Reddy, ed., Academic Press, New York, 1975.

[4] J.E. Shoup, "Phonological aspects of speech recognition," ch. 6 in *Trends in Speech Recognition*, W.A. Lea, ed., Prentice Hall, Englewood Cliffs, NJ, 1980.

[5] P. Ladefoged, *A Course in Phonetics*, Harcourt Brace Jovanovich, New York, 1975.

[6] P.S. Cohen and R.L. Mercer, "The phonological component of an automatic speech-recognition system," in *Speech Recognition*, D.R. Reddy, ed., Academic Press, New York, 1975.

[7] *The American Heritage Dictionary of the English Language*, Houghton Mifflin, Boston, MA, 1973.

[8] R. Schwartz, C. Barry, Yen-Lu Chow, A. Derr, Ming-Whei Feng, O. Kimball, F. Kubala, J. Makhoul, J. Vandegrift, "The BBN BYBLOS continuous speech recognition system," *Proceedings of the DAPPA Speech and Natural Language Workshop*, pp. 94–99, Morgan Kaufmann Publishers, San Mateo, CA, February 1989.

[9] L.R. Bahl, P. Brown, P. deSouza, R.L. Mercer, and M. Picheny, "Acoustic Markov models used in the Tangora speech recognition system," *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, New York, April 1988.

[10] S.E. Levinson, L.R. Rabiner, and M.M. Sondhi, "An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition," *Bell System Technical Journal*, vol. 62, no. 4, pp. 1035–74, 1983.

10. The required prototypes had to be constructed from actual samples of word utterances. No smaller building blocks gave adequate performance. Thus the method seemed inapplicable to large vocabulary speech recognition.

[11] B.H. Juang and L.R. Rabiner, " Issues in using hidden Markov models for speech recognition," in *Advances in Signal Processing*, S. Furui and M.M. Sondhi, eds., pp. 509–54, Marcel Dekker, New York, 1992.

[12] R. Bakis, "Continuous-speech word spotting via centisecond acoustic states," IBM Research Report RC 4788, Yorktown Heights, NY, April 2, 1974.

[13] M. Ostendorf and S. Roukos, "A stochastic segment model for phoneme-based continuous speech recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*. vol. 37, no. 12, pp. 1857–69, December 1989.

[14] M. Ostendorf, V.V. Digalakis, and O.A. Kimball, "From HMMs to segment models: A unified view of stochastic modeling for speech recognition," *IEEE Transactions on Speech and Audio Processing*, vol. 4, no. 5, pp. 360–78, September 1996.

[15] M. Ostendorf, "From HMMs to segment models: Stochastic modeling for CSR," in *Automatic Speech and Speaker Recognition*, C-H. Lee, F.K. Soong, and K.K. Paliwal, eds., pp. 185–210, Klewer Academic Publishers, Norwell, MA, 1996.

[16] L.R. Bahl, P. Brown, P. deSouza, and R.L. Mercer, "Maximum mutual information estimation of hidden Markov model parameters for speech recognition," *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 49–52, Tokyo, 1986.

[17] S.J. Young, "Competitive training in hidden Markov models," *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 681–84, Albuquerque, NM April 1990.

[18] P.S. Gopalakrishnan, D. Kanevsky, A. Nadas, and D. Nahamoo, "An inequality for rational functions with applications to some statistical estimation problems," *IEEE Transactions on Information Theory*, vol. 37, no. 1, pp. 107–13, January 1991.

[19] Y. Normandin, "Maximum mutual information estimation of hidden Markov models," in *Automatic Speech and Speaker Recognition*, C-H. Lee, F.K. Soong, and K.K. Paliwal, eds., pp. 58–81, Klewer Academic Publishers, Norwell, MA, 1996.

[20] L.R. Bahl, P. Brown, P. deSouza, and R.L. Mercer, "Estimating hidden Markov model parameters so as to maximize speech recognition accuracy," *IEEE Transactions on Speech and Audio Processing*, vol. 1, no. 1, pp. 77–83, January 1993.

[21] H. Bourlard and C.J. Wellekens, "Links between Markov models and multilayer perceptrons," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 6, pp. 1167–78, December 1990.

[22] M. Richard and R. Lippmann, "Neural network classifiers estimate Bayesian à posteriori probabilities," *Neural Computation*, vol. 3, no. 4, pp. 461–83, 1991.

[23] H. Bourlard and N. Morgan, "Hybrid connectionist models for continuous speech recognition," in *Automatic Speech and Speaker Recognition*, C-H. Lee,

F.K. Soong, and K.K. Paliwal, eds., pp. 259–83, Klewer Academic Publishers, Norwell, MA, 1996.

[24] T. Robinson, M. Hochberg, and S. Renals, "The use of recurrent neural networks in continuous speech recognition," in *Automatic Speech and Speaker Recognition*, C-H. Lee, F.K. Soong, and K.K. Paliwal, eds., pp. 233–58, Klewer Academic Publishers, Norwell, MA, 1996.

[25] F. Itakura, "Minimum prediction residual principle applied to speech recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-23, no. 1, pp. 67–72, February 1975.

[26] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-26, no. 1, pp. 43–49, February 1978.

[27] L.R. Rabiner, A.E. Rosenberg, and S.E. Levinson, "Considerations in dynamic time warping algorithms for discrete word recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-26, no. 6, pp. 575–82, December 1978.

[28] C-H. Lee, "Applications of dynamic programming to speech and language processing," *AT&T Technical Journal*, pp. 114–30, Murray Hill, NJ, May/June 1989.

Chapter 9

The Expectation-Maximization Algorithm and Its Consequences

9.1 Introduction

In this chapter we will derive the Expectation-Maximization (EM) algorithm [1] and use it to justify the convergence and optimality of the Baum-Welch algorithm that estimates maximum likelihood HMM parameter values from data.[1] We will then generalize the Baum-Welch algorithm to apply to continuous, normally distributed acoustics. We will finally introduce the *tied mixture* acoustic model used in many current continuous speech recognizers. The EM algorithm's applicability is, of course, not restricted to the training of HMMs: It is much more general.

The development below is based on a special case of Jensen's inequality, which we already proved in section 7.3.[2]

LEMMA 9.1. If $p(x)$ and $q(x)$ are two discrete probability distributions, then

$$\sum_x p(x) \log p(x) \geq \sum_x p(x) \log q(x)$$

with equality if and only if $p(x) = q(x)$ for all x.

9.2 The EM Theorem

We will now develop the main theorem. Let y denote observable data. Let $P_{\theta'}(y)$ be the probability distribution of y under some model whose parameters are denoted by θ'.[3] Let $P_\theta(y)$ be the corresponding distribu-

1. See section 2.7.
2. Property **(7)** in section 7.3.
3. θ' denotes the totality of all the parameters whose values are needed to specify the distribution $P_{\theta'}$.

tion under a different setting θ of the same parameters. We are interested in developing conditions for which y is more likely under θ than it is under θ'.[4]

Let t be another random variable whose value is determined in the same process (t is therefore governed by the value of the parameters θ' or θ) that generates y (for instance, in the HMM setting, y might correspond to the observed output sequence and t to state or to transition sequences). Then, because $P_{\theta'}(t|y)$ is a probability distribution that sums to 1,

$$\log P_\theta(y) - \log P_{\theta'}(y) = \sum_t P_{\theta'}(t|y) \log P_\theta(y) - \sum_t P_{\theta'}(t|y) \log P_{\theta'}(y)$$

Since we can multiply by 1 without changing anything, the above is equal to

$$\begin{aligned}
&\log P_\theta(y) - \log P_{\theta'}(y) \\
&\quad = \sum_t P_{\theta'}(t|y) \log P_\theta(y) \frac{P_\theta(t,y)}{P_\theta(t,y)} - \sum_t P_{\theta'}(t|y) \log P_{\theta'}(y) \frac{P_{\theta'}(t,y)}{P_{\theta'}(t,y)} \\
&\quad = \sum_t P_{\theta'}(t|y) \log \frac{P_\theta(t,y)}{P_\theta(t|y)} - \sum_t P_{\theta'}(t|y) \log \frac{P_{\theta'}(t,y)}{P_{\theta'}(t|y)} \\
&\quad = \sum_t P_{\theta'}(t|y) \log P_\theta(t,y) - \sum_t P_{\theta'}(t|y) \log P_{\theta'}(t,y) \\
&\qquad + \sum_t P_{\theta'}(t|y) \log P_{\theta'}(t|y) - \sum_t P_{\theta'}(t|y) \log P_\theta(t|y) \\
&\quad \geq \sum_t P_{\theta'}(t|y) \log P_\theta(t,y) - \sum_t P_{\theta'}(t|y) \log P_{\theta'}(t,y)
\end{aligned}$$

where the inequality follows from lemma 9.1. Thus if the last quantity in the above equation is positive, so is the first. We have thus proven

THEOREM 9.1. If

$$\sum_t P_{\theta'}(t|y) \log P_\theta(t,y) > \sum_t P_{\theta'}(t|y) \log P_{\theta'}(t,y) \tag{1}$$

then

$$P_\theta(y) > P_{\theta'}(y) \tag{2}$$

This is the basic EM theorem. It says that if we start with the parameter setting θ' and find a parameter setting θ for which the inequality (1) holds,

4. In that case, θ represents an improvement over θ'.

then the observed data y will be more probable under the regime θ than they were under θ'.

To take best advantage of this hill-climbing theorem, we should thus endeavor to find the setting θ that will maximize the left-hand side of (1). Such setting will, of course, satisfy (1) since both of its sides have the same form, which is maximized by that choice of θ. The reason for the name *Expectation-Maximization* is that we take the expectation of the random variable $\log P_\theta(t, y)$ with respect to the old distribution $P_{\theta'}(t|y)$ and then maximize that expectation as a function of the argument θ. It is an algorithm, because the natural way to run it is to choose an initial value of θ', then compute the maximizing θ, then set θ' to θ, compute a new θ, etc. As the process continues, the value of $P_{\theta'}(y)$ will keep climbing toward a limit, since the upper bound $P_{\theta'}(y) \leq 1$ necessarily applies.

The secret of success in applying the EM algorithm is a judicious choice of the auxiliary variable t that will allow finding the maximum of the expectation on the left-hand side of (1). Such a choice is possible for HMMs.

9.3 The Baum-Welch Algorithm

We will now use the EM theorem to derive the Baum-Welch algorithm [2] for a discrete output alphabet $\mathcal{Y}$.[5] To do this with ease, we will use that formulation of HMMs in which various transitions are deterministically related to the observed outputs (as formulated in section 2.2). That is, a function $Y(t)$ assigns outputs y to transitions t.

We will naturally deal with strings of outputs, denoted by $\mathbf{y} = y_1 y_2 \ldots y_n$, and strings of transitions $\mathbf{t} = t_{i_1 j_1}, t_{i_2 j_2} \ldots t_{i_k j_k}$. Here $k \geq n$ because some of the transitions may be null. The expression $t_{i_l j_l}$ denotes the $(j_l)^{th}$ transition out of state i_l. Since $\mathbf{t}$ defines a path, then necessarily $R(t_{i_l j_l}) = i_{l+1}$.[6] (We are following the notation introduced in section 2.2.) The parameter θ of interest consists of the totality of transition probabilities

$$p_{ij} = P(t_{ij})$$

5. See section 2.7.

6. If $P(\mathbf{t}) > 0$, then our notation is such that the transition j_k out of state i_k leads to state i_{k+1}, or equivalently $L(t_{i_k j_k}) = i_k$ and $R(t_{i_k j_k}) = i_{k+1}$.

that define the HMM in question. Thus to vary θ is equivalent to varying the transition probabilities p_{ij}.

We will wish to find the maximum of $\sum_{\mathbf{t}} P_{\theta'}(\mathbf{t}|\mathbf{y}) \log P_{\theta}(\mathbf{t}, \mathbf{y})$ with respect to θ, which we will obtain by differentiating with respect to the probabilities p_{ij} and equating the result to 0. We thus get[7]

$$\frac{\partial}{\partial p_{ij}} \left[\sum_{\mathbf{t}} P_{\theta'}(\mathbf{t}|\mathbf{y}) \log P_{\theta}(\mathbf{t}, \mathbf{y}) - \sum_{m} \lambda_m \sum_{n} p_{mn} \right] = \sum_{\mathbf{t}} P_{\theta'}(\mathbf{t}|\mathbf{y}) \frac{(\partial/\partial p_{ij}) P_{\theta}(\mathbf{t}, \mathbf{y})}{P_{\theta}(\mathbf{t}, \mathbf{y})} - \lambda_i \tag{3}$$

Since $\mathbf{t}$ determines $\mathbf{y}$, then either $P_{\theta}(\mathbf{t}, \mathbf{y}) = 0$ (if $\mathbf{y}$ is incompatible with $\mathbf{t}$), or

$$P_{\theta}(\mathbf{t}, \mathbf{y}) = P_{\theta}(\mathbf{t}) = \prod_{l=1}^{K} p_{i_l j_l} \tag{4}$$

If $c_{ij}(\mathbf{t})$ denotes the number of times the transition t_{ij} takes place in the string $\mathbf{t}$, then since

$$\frac{\partial}{\partial p_{ij}} P_{\theta}(\mathbf{t}, \mathbf{y}) = \frac{\partial}{\partial p_{ij}} \prod_{l=1}^{k} p_{i_l j_l}$$

we get

$$\frac{(\partial/\partial p_{ij}) P_{\theta}(\mathbf{t}, \mathbf{y})}{P_{\theta}(\mathbf{t}, \mathbf{y})} = \frac{c_{ij}(\mathbf{t})}{p_{ij}} \tag{5}$$

Thus equating (3) to 0 we end up with the equation

$$\sum_{t} P_{\theta'}(\mathbf{t}|\mathbf{y}) \frac{c_{ij}(\mathbf{t})}{p_{ij}} = \lambda_i \tag{6}$$

or,

$$p_{ij} = \frac{1}{\lambda_i P_{\theta'}(\mathbf{y})} \sum_{t} P_{\theta'}(\mathbf{t}, \mathbf{y}) c_{ij}(\mathbf{t}) = \frac{1}{K_i} \sum_{t} P_{\theta'}(\mathbf{t}, \mathbf{y}) c_{ij}(\mathbf{t}) \tag{7}$$

where K_i plays the role of a normalizing constant that assures that

7. As elsewhere, we use the method of undetermined lagrangian multipliers.

$\sum_j p_{ij} = 1$. But, using the Kronecker delta notation[8] and recalling that $\mathbf{t} = t_{i_1 j_1}, t_{i_2 j_2}, \ldots, t_{i_k j_k}$,

$$c_{ij}(\mathbf{t}) = \sum_{l=1}^{k} \delta(t_{ij}, t_{i_l j_l})$$

so that if t_{ij} is not a null transition then

$$\begin{aligned}
&\sum_t P_{\theta'}(\mathbf{t}, \mathbf{y}) c_{ij}(\mathbf{t}) \\
&\quad = \sum_{l=1}^{k} \sum_t P_{\theta'}(\mathbf{t}, \mathbf{y})\, \delta(t_{ij}, t_{i_l j_l}) \\
&\quad = \sum_{l=1}^{k} P_{\theta'}(y_1, \ldots, y_{l-1}, s_{l-1} = i)\, p'_{ij}\, P_{\theta'}(y_{l+1}, \ldots, y_k | s_l = R(t_{ij})) \\
&\quad = \sum_{l=1}^{k} \alpha_{l-1}^{\theta'}(i)\, p'_{ij}\, \beta_l^{\theta'}(R(t_{ij}))
\end{aligned} \tag{8}$$

In (8) we have used the definitions

$$\begin{aligned}
\alpha_l(s) &\doteq P(y_1, \ldots, y_l, s_l = s) \\
\beta_l(s) &\doteq P(y_{l+1}, \ldots, y_k | s_l = s)
\end{aligned} \tag{9}$$

introduced in section 2.7.

Similarly, if t_{ij} is a null transition then

$$\begin{aligned}
&\sum_t P_{\theta'}(\mathbf{t}, \mathbf{y}) c_{ij}(\mathbf{t}) \\
&\quad = \sum_{l=1}^{k} \sum_t P_{\theta'}(\mathbf{t}, \mathbf{y})\, \delta(t_{ij}, t_{i_l j_l}) \\
&\quad = \sum_{l=1}^{k} P_{\theta'}(y_1, \ldots, y_{l-1}, s_{l-1} = i)\, p'_{ij}\, P_{\theta'}(y_l, \ldots, y_k | s_{l-1} = R(t_{ij})) \\
&\quad = \sum_{l=1}^{k} \alpha_{l-1}^{\theta'}(i)\, p'_{ij}\, \beta_{l-1}^{\theta'}(R(t_{ij}))
\end{aligned} \tag{10}$$

8.

$$\delta(a, b) \doteq \begin{cases} 1 & \text{if } a = b \\ 0 & \text{otherwise} \end{cases}$$

The right-hand sides of (8) and (10) are in fact the contributions the Baum-Welch algorithm makes to the counters that determine the next value of p_{ij}. We have thus deduced the validity of the Baum-Welch algorithm from the EM theorem. Of course, the values of $\alpha_l(s)$ and $\beta_l(s)$ are obtained iteratively by formulas (32) and (34) of chapter 2.

9.4 Real Vector Outputs of the Acoustic Processor

We will now generalize HMMs to the case where their output symbols are normally distributed vectors of real numbers. To avoid unnecessary complications, we will develop our reestimation formulas for two-dimensional vectors. The reader will then immediately accept the obvious generalization to k dimensions.

9.4.1 Development for Two Dimensions

The general setup is this: We have the usual HMM with (possibly multiple) transitions between states. The non-null transitions generate outputs that are normally distributed two-dimensional real vectors $\mathbf{y}$. The unknown parameters then are (a) the transition probabilities $p(t)$ satisfying[9]

$$\sum_{t:L(t)=s} p(t) = 1 \qquad \text{for all states } s \tag{11}$$

and (b) the parameters $\mathbf{m}(t)$, and $\mathbf{U}(t)$ of the normal density

$$\mathcal{N}_t(\mathbf{y}) = \frac{1}{2\pi\sqrt{|\mathbf{U}(t)|}} \exp\left\{-\frac{1}{2}(\mathbf{y}-\mathbf{m}(t))\mathbf{U}(t)^{-1}(\mathbf{y}-\mathbf{m}(t))'\right\} \tag{12}$$

where $(\mathbf{y}-\mathbf{m}(t))'$ denotes the transpose of $(\mathbf{y}-\mathbf{m}(t))$. $\mathbf{U}(t)$ is the process's covariance matrix. To simplify the notation, we will frequently omit the argument t and use it only when necessary to avoid confusion.

If

$$\mathbf{U} = \begin{bmatrix} (\sigma_1)^2 & \rho \\ \rho & (\sigma_2)^2 \end{bmatrix}$$

then the determinant

$$D \doteq |\mathbf{U}| = (\sigma_1)^2(\sigma_2)^2 - \rho^2 \tag{13}$$

9. We are using the $L(t)$ and $R(t)$ notation introduced in section 2.2.

If further $\mathbf{y} = (y_1, y_2)$ and $\mathbf{m} = (m_1, m_2)$ then we can rewrite (12) as

$$\mathcal{N}_t(\mathbf{y}) = \frac{1}{2\pi\sqrt{D(t)}} \exp\left\{-\frac{G(t)}{2D(t)}\right\} \tag{14}$$

where

$$G \doteq (\sigma_2)^2(y_1 - m_1)^2 - 2\rho(y_1 - m_1)(y_2 - m_2) + (\sigma_1)^2(y_2 - m_2)^2 \tag{15}$$

It follows that the parameter estimation task for real vector HMMs concerns the values of $\theta(t) \doteq \{p(t), m_1(t), m_2(t), \sigma_1(t)^2, \sigma_2(t)^2, \rho(t)\}$. Denoting by $\phi(t)$ any of the members of the set $\theta(t)$, we have shown in section 9.3 that we seek the solution to the equation[10]

$$\sum_t P_{\theta'}(\mathbf{t}|\mathbf{Y}) \frac{(\partial/\partial\phi(t))P_\theta(\mathbf{t}, \mathbf{Y})}{P_\theta(\mathbf{t}, \mathbf{Y})} - \sum_m \lambda_m \sum_{t:L(t)=m} p_\theta(t) = 0 \tag{16}$$

for all non-null t and $\phi(t) \in \theta(t)$

where θ' and θ denote the old and new (reestimated) parameter set values.

Now[11]

$$P_\theta(\mathbf{t}, \mathbf{Y}) = \prod_{l=1}^{n} p(t_l) \prod_t \mathcal{N}_t(\mathbf{y}_l)^{\delta(t,t_l)} \tag{17}$$

where the observed sequence $\mathbf{Y}$ is of length n, and to simplify notation we have allowed no null transitions in the sequence $\mathbf{t}$. (Null transitions would make the indexing below more complex, but would change nothing of significance.)

First, observe that the transition probabilities $p(t)$ are not involved in the factor $\prod_t \mathcal{N}_t(\mathbf{y}_l)^{\delta(t,t_l)}$ at all; the latter is simply a constant that cancels out from the numerator and denominator of (16). Therefore, the estimation problem for $p(t)$ is essentially the same as that posed in (4), so that the reestimated $p_\theta(t)$ will be given by (compare with (8))

$$p_\theta(t) = \frac{1}{K} \sum_{l=1}^{n} \alpha_{l-1}^{\theta'}(L(t))\, p_{\theta'}(t)\, \mathcal{N}_t^{\theta'}(\mathbf{y}_l)\, \beta_l^{\theta'}(R(t)) \tag{18}$$

where K is a normalizing constant, and α and β were defined in (9).

10. We are using the sequence notation $\mathbf{Y} = \mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n$ where $\mathbf{y}_l = (y_{l,1}, y_{l,2})$.

11. Of course, $\prod_t \mathcal{N}_t(\mathbf{y}_l)^{\delta(t,t_l)} = \mathcal{N}_{t_l}(\mathbf{y}_l)$. We use the seemingly more complex product to be able to differentiate later with respect to functions of the transition variable t.

We will next take partial derivatives with respect to the means m_i. We have

$$\frac{\partial}{\partial m_i}\mathcal{N}_t(\mathbf{y}) = \frac{\partial}{\partial m_i}\left(\frac{1}{2\pi\sqrt{D}}\exp\left\{-\frac{G}{2D}\right\}\right)$$
$$= \left(\frac{1}{2\pi\sqrt{D}}\exp\left\{-\frac{G}{2D}\right\}\right)\left(-\frac{(\partial/\partial m_i)G}{2D}\right)$$

and

$$\frac{\partial}{\partial m_i}G = -2(\sigma_i)^2(y_{l,i} - m_i) + 2\rho(y_{l,j} - m_j) \qquad j \neq i$$

so

$$\frac{\partial}{\partial m_i}\mathcal{N}_t(\mathbf{y}) = \left(\frac{1}{2\pi\sqrt{D}}\exp\left\{-\frac{G}{2D}\right\}\right)\left(\frac{(\sigma_i)^2(y_{l,i} - m_i) - \rho(y_{l,j} - m_j)}{D}\right)$$

It then follows directly from (14) and (17) that

$$\frac{(\partial)/\partial m_i(t'))P_\theta(\mathbf{t},\mathbf{y})}{P_\theta(\mathbf{t},\mathbf{y})}$$
$$= \sum_{l=1}^{n}\frac{(\partial/\partial m_i(t'))\mathcal{N}_{t'}(\mathbf{y}_l)^{\delta(t',t_l)}}{\mathcal{N}_{t'}(\mathbf{y}_l)^{\delta(t',t_l)}} \tag{19}$$
$$= \frac{1}{D(t')}\sum_{l=1}^{n}\delta(t',t_l)(\sigma_i(t')^2(y_{l,i} - m_i(t')) - \rho(t')(y_{l,j} - m_j(t'))) \qquad j \neq i$$

We next want to derive the expressions for the partial derivatives with respect to σ_i and ρ. Denoting either by φ, we get

$$\frac{\partial}{\partial\varphi}\mathcal{N}_t(\mathbf{y}) = \frac{\partial}{\partial\varphi}\left(\frac{1}{2\pi\sqrt{D}}\exp\left\{-\frac{G}{2D}\right\}\right)$$
$$= \left(\frac{1}{2\pi\sqrt{D}}\exp\left\{-\frac{G}{2D}\right\}\right)\left(-\frac{1}{2D^2}\left((D-G)\frac{\partial}{\partial\varphi}D + D\frac{\partial}{\partial\varphi}G\right)\right) \tag{20}$$

so that

$$\frac{(\partial/\partial\varphi(t'))P_\theta(\mathbf{t},\mathbf{y})}{P_\theta(\mathbf{t},\mathbf{y})} = \sum_{l=1}^{n}\frac{(\partial/\partial\varphi(t'))\mathcal{N}_{t'}(\mathbf{y}_l)^{\delta(t',t_l)}}{\mathcal{N}_{t'}(\mathbf{y}_l)^{\delta(t',t_l)}}$$
$$= \sum_{l=1}^{n}\delta(t',t_l)\left(-\frac{1}{2D^2}\left((D-G)\frac{\partial}{\partial\varphi}D + D\frac{\partial}{\partial\varphi}G\right)\right)$$

Since

$$\frac{\partial}{\partial \varphi} D = \begin{cases} 2\sigma_i(\sigma_j)^2 & j \neq i \quad \text{if } \varphi = \sigma_i \\ -2\rho & \text{if } \varphi = \rho \end{cases} \tag{21}$$

$$\frac{\partial}{\partial \varphi} G = \begin{cases} 2\sigma_i(y_{l,j} - m_j)^2 & j \neq i \quad \text{if } \varphi = \sigma_i \\ -2(y_{l,1} - m_1)(y_{l,2} - m_2) & \text{if } \varphi = \rho \end{cases} \tag{22}$$

then we finally get

$$\begin{aligned} &\frac{(\partial/\partial\sigma_i(t'))P_\theta(\mathbf{t},\mathbf{y})}{P_\theta(\mathbf{t},\mathbf{y})} \\ &= \sum_{l=1}^{n} \delta(t', t_l) \frac{-2\sigma_i}{2((\sigma_1)^2(\sigma_2)^2 - \rho^2)^2} \\ &\quad \times \Big[(\sigma_j)^2\Big((\sigma_1)^2(\sigma_2)^2 - \rho^2 - (\sigma_2)^2(y_{l,1} - m_1)^2 \\ &\qquad + 2\rho(y_{l,1} - m_1)(y_{l,2} - m_2) - (\sigma_1)^2(y_{l,2} - m_2)^2\Big) \\ &\quad + ((\sigma_1)^2(\sigma_2)^2 - \rho^2)(y_{l,j} - m_j)^2\Big] \end{aligned}$$

where we have omitted on the right-hand side the argument t' to keep the notation as simple as possible. Therefore

$$\begin{aligned} &\frac{(\partial/\partial\sigma_1(t'))P_\theta(\mathbf{t},\mathbf{y})}{P_\theta(\mathbf{t},\mathbf{y})} \\ &= \frac{-\sigma_1}{((\sigma_1)^2(\sigma_2)^2 - \rho^2)^2} \\ &\quad \times \sum_{l=1}^{n} \delta(t', t_l)\Big((\sigma_1)^2(\sigma_2)^4 - \rho^2(\sigma_2)^2 - (\sigma_2)^4(y_{l,1} - m_1)^2 \\ &\qquad + 2\rho(\sigma_2)^2(y_{l,1} - m_1)(y_{l,2} - m_2) - \rho^2(y_{l,2} - m_2)^2\Big) \end{aligned} \tag{23}$$

$$\begin{aligned} &= \frac{-\sigma_1}{((\sigma_1)^2(\sigma_2)^2 - \rho^2)^2} \\ &\quad \times \sum_{l=1}^{n} \delta(t', t_l)\Big((\sigma_2)^4[(\sigma_1)^2 - (y_{l,1} - m_1)^2] \\ &\qquad - 2\rho(\sigma_2)^2[\rho - (y_{l,1} - m_1)(y_{l,2} - m_2)] \\ &\qquad + \rho^2[(\sigma_2)^2 - (y_{l,2} - m_2)^2]\Big) \end{aligned}$$

and

$$\frac{(\partial/\partial\sigma_2(t'))P_\theta(\mathbf{t},\mathbf{y})}{P_\theta(\mathbf{t},\mathbf{y})}$$

$$= \frac{-\sigma_2}{((\sigma_1)^2(\sigma_2)^2-\rho^2)^2} \times \sum_{l=1}^{n} \delta(t',t_l)\Big((\sigma_1)^4[(\sigma_2)^2-(y_{l,2}-m_2)^2] - 2\rho(\sigma_1)^2[\rho-(y_{l,1}-m_1)(y_{l,2}-m_2)] + \rho^2[(\sigma_1)^2-(y_{l,1}-m_1)^2]\Big) \tag{24}$$

Finally

$$\frac{(\partial/\partial\rho(t'))P_\theta(\mathbf{t},\mathbf{y})}{P_\theta(\mathbf{t},\mathbf{y})}$$

$$= \frac{-1}{2((\sigma_1)^2(\sigma_2)^2-\rho^2)^2} \times \sum_{l=1}^{n} \delta(t',t_l)\Big\{(-2\rho)\Big[(\sigma_1)^2(\sigma_2)^2-\rho^2 - \Big((\sigma_2)^2(y_{l,1}-m_1)^2 - 2\rho(y_{l,1}-m_1)(y_{l,2}-m_2) + (\sigma_1)^2(y_{l,2}-m_2)^2\Big)\Big] + ((\sigma_1)^2(\sigma_2)^2-\rho^2)(-2(y_{l,1}-m_1)(y_{l,2}-m_2))\Big\}$$

$$= \frac{1}{((\sigma_1)^2(\sigma_2)^2-\rho^2)^2} \times \sum_{l=1}^{n} \delta(t',t_l)\Big[\rho((\sigma_1)^2(\sigma_2)^2-\rho^2) - \Big(\rho(\sigma_2)^2(y_{l,1}-m_1)^2 - ((\sigma_1)^2(\sigma_2)^2+\rho^2)(y_{l,1}-m_1)(y_{l,2}-m_2) + \rho(\sigma_1)^2(y_{l,2}-m_2)^2\Big)\Big]$$

$$= \frac{-1}{((\sigma_1)^2(\sigma_2)^2-\rho^2)^2} \times \sum_{l=1}^{n} \delta(t',t_l)\Big[((\sigma_1)^2(\sigma_2)^2+\rho^2)(\rho-(y_{l,1}-m_1)(y_{l,2}-m_2)) + \rho(\sigma_2)^2((\sigma_1)^2-(y_{l,1}-m_1)^2) + \rho(\sigma_1)^2((\sigma_2)^2-(y_{l,2}-m_2)^2)\Big]$$

It follows directly from (16) and the right-hand sides of (19), (23), (24), and (25) that the desired parameter setting $\theta(t')$ is one satisfying the following equations:[12]

$$\sum_t P_{\theta'}(\mathbf{t},\mathbf{y}) \sum_{l=1}^{n} \delta(t', t_l)(y_{l,i} - m_i(t')) = 0 \quad i = 1,2 \tag{26}$$

$$\sum_t P_{\theta'}(\mathbf{t},\mathbf{y}) \sum_{l=1}^{n} \delta(t', t_l)[(y_{l,i} - m_i(t'))^2 - \sigma_i(t')^2] = 0 \quad i = 1,2 \tag{27}$$

$$\sum_t P_{\theta'}(\mathbf{t},\mathbf{y}) \sum_{l=1}^{n} \delta(t', t_l)((y_{l,1} - m_1(t'))(y_{l,2} - m_2(t')) - \rho(t')) = 0 \tag{28}$$

Thus (repeating (8) for convenience) the desired reestimation formulas are

$$p_\theta(t) = \frac{1}{K} \sum_{l=1}^{n} \alpha_{l-1}^{\theta'}(L(t))\, p_{\theta'}(t)\, \mathcal{N}_t^{\theta'}(\mathbf{y}_l)\, \beta_l^{\theta'}(R(t)) \tag{29}$$

$$m_i^\theta(t) = \frac{1}{K^*(t)} \sum_{l=1}^{n} \alpha_{l-1}^{\theta'}(L(t))\, p_{\theta'}(t)\, \mathcal{N}_t^{\theta'}(\mathbf{y}_l)\, \beta_l^{\theta'}(R(t)) \times y_{l,i} \quad i = 1,2 \tag{30}$$

$$\sigma_i^\theta(t)^2 = \frac{1}{K^*(t)} \sum_{l=1}^{n} \alpha_{l-1}^{\theta'}(L(t))\, p_{\theta'}(t)\, \mathcal{N}_t^{\theta'}(\mathbf{y}_l)\, \beta_l^{\theta'}(R(t)) \times (y_{l,i} - m_i^\theta(t))^2 \quad i = 1,2 \tag{31}$$

$$\rho^\theta(t') = \frac{1}{K^*(t)} \sum_{l=1}^{n} \alpha_{l-1}^{\theta'}(L(t))\, p_{\theta'}(t)\, \mathcal{N}_t^{\theta'}(\mathbf{y}_l)\, \beta_l^{\theta'}(R(t)) \times (y_{l,1} - m_1^\theta(t))(y_{l,2} - m_2^\theta(t)) \tag{32}$$

In (29) the normalizing constant K assures that the sum of the probabilities of transitions leaving any state equals 1. The normalizing function $K^*(t)$ is given by

$$K^*(t) = \sum_{l=1}^{n} \alpha_{l-1}^{\theta'}(L(t))\, p_{\theta'}(t)\, \mathcal{N}_t^{\theta'}(\mathbf{y}_l)\, \beta_l^{\theta'}(R(t))$$

As always, we provide for each parameter to be estimated a counter (or a set of counters) that accumulates the terms of the above sums. This

12. In which, for convenience of actual computation, we have replaced $P_{\theta'}(\mathbf{t}|\mathbf{y})$ by $P_{\theta'}(\mathbf{t},\mathbf{y})$.

is somewhat complicated in the case of formulas (31) and (32), which involve the results $m_i^\theta(t)$ of the computation of (30).

9.4.2 The Generalization to k Dimensions

It is obvious both from the above derivation and from its results (29) through (32) how to estimate parameters in the k-dimensional general case. In fact, if $\mathbf{U}(t) = [\rho_{i,j}(t)]$ denotes the covariance $k \times k$ matrix then the general reestimation formulas are simply (29), (30), and

$$\rho_{i,j}^\theta(t') = \frac{1}{K^*(t)} \sum_{l=1}^{n} \alpha_{l-1}^{\theta'}(L(t))\, p_{\theta'}(t) \mathcal{N}_t^{\theta'}(\mathbf{y}_l), \beta_l^{\theta'}(R(t)) \times (y_{l,i} - m_i^\theta(t))(y_{l,j} - m_j^\theta(t)) \tag{33}$$

9.5 Constant and Tied Parameters

9.5.1 Keeping Some Parameters Constant

In many applications, we wish to estimate the values of only some HMM parameters. For instance, in smoothing trigram models (see section 4.4) we wanted to find the probabilities λ_i of the model's null transitions while keeping the output distributions $f(\ |\ )$ as determined from the main part of the text training data. It is obvious from our derivation of the Baum-Welch algorithm for both symbolic (section 9.3) and real vector (section 9.4) outputs that

1. The formulas estimating the transition probabilities out of one state do not involve explicitly the transition probabilities out of another state. (Of course, the totality of all parameters affects the values of $\alpha_l(s)$ and $\beta_l(s)$ at various states s.)
2. The formulas estimating the output distributions or output density parameters associated with one transition do not involve explicitly the corresponding parameters associated with other transitions.

Thus the sole difference between estimating all the parameters and only *some* of them is that the latter case involves fewer counters (but of the same type).

In the case of real vectors, it is even possible to keep constant some means and covariances associated with a particular transition and estimate that transition's remaining parameters. Inspection of (19), (23), (24), and (25) shows what must be done with the contents of the counters in that case (to which (29), (30), and (33) no longer apply).

In current practice, the most common parameters to be kept constant are the off-diagonal covariances. Because of lack of sufficient data and the computational expense of their estimation, these covariances are usually set to equal 0 (i.e., $\rho_{i,j}(t) = 0$ for $i \neq j$). For this special case (29), (30), and (33) remain applicable.

9.5.2 Tying of Parameter Sets

In the vast majority of cases of interest, the HMMs used are so complex (and there are so many of them) that there is insufficient data to estimate the values of all their parameters individually. Neither does there exist enough outside knowledge to fix the values of certain parameters and estimate the others. The designer invariably partitions the set of parameters into subsets and insists that all the parameters belonging to the same subset have the same value,[13] which he sets out to estimate. The parameters of any given subset are then said to be *tied*.

Let us first see what the consequences of tying are in the case of the symbolic output alphabet treated in section 9.3. Suppose the transition probabilities out of states i and i' are to be tied in such a way that $p_{ij} = p_{i'j}$ for $j = 1, 2, \ldots, l_i$. Then, using the notation of the development of (3) through (7), the relation (5) becomes

$$\frac{(\partial/\partial p_{ij})P_\theta(\mathbf{t}, \mathbf{y})}{P_\theta(\mathbf{t}, \mathbf{y})} = \frac{c_{ij}(\mathbf{t}) + c_{i'j}(\mathbf{t})}{p_{ij}}$$

and therefore (compare with (7))

$$p_{ij} = \frac{1}{K_i} \sum_t P_{\theta'}(\mathbf{t}, \mathbf{y})[c_{ij}(\mathbf{t}) + c_{i'j}(\mathbf{t})]$$

The treatment of the general case is obvious. If the transition probabilities out of states $i, i', \ldots$ are to be tied, we simply pool the contents of their corresponding counters and after each iteration set the new probabilities to be proportional to the pool's final contents.[14]

13. The membership of these tied subsets is either determined intuitively or from data, usually by the method of *decision trees* discussed in chapter 10.

14. Note that this approach implies that the tied parameters are truly identical, that is, interchangeable. So the fact that the statistics for transition t may be derived mainly from observations of transition t' is presumably not bothersome. We would undoubtedly much rather not have the parameters identical but use one as a help in deriving a more robust estimate of the other. This is not what tying accomplishes.

We next focus our attention on the case of real vector outputs. Then tying of output parameters associated with two transitions t' and t'' means that $\mathbf{m}(t') = \mathbf{m}(t'')$ and $\mathbf{U}(t') = \mathbf{U}(t'')$. Therefore, $\theta^* \doteq \theta(t') = \theta(t'')$. (See the notation introduced following (15)). Again, let ϕ^* denote any of the members of the parameter set θ^*.

Using formula (17) we get

$$\begin{aligned}\frac{(\partial/\partial\phi^*)P_\theta(\mathbf{t},\mathbf{y})}{P_\theta(\mathbf{t},\mathbf{y})} &= \frac{(\partial/\partial\phi^*)\prod_{l=1}^n p(t_l)\prod_t \mathcal{N}_t(\mathbf{y}_l)^{\delta(t,t_l)}}{\prod_{l=1}^n p(t_1)\prod_t \mathcal{N}_t(\mathbf{y}_l)^{\delta(t,t_l)}} \\ &= \sum_{l=1}^n \left(\frac{(\partial/\partial\phi^*)\mathcal{N}_{t'}(\mathbf{y}_l)^{\delta(t',t_l)}}{\mathcal{N}_{t'}(\mathbf{y}_l)^{\delta(t',t_l)}} + \frac{(\partial/\partial\phi^*)\mathcal{N}_{t''}(\mathbf{y}_l)^{\delta(t'',t_l)}}{\mathcal{N}_{t''}(\mathbf{y}_l)^{\delta(t'',t_l)}} \right) \\ &= \sum_{l=1}^n (\delta(t',t_l) + \delta(t'',t_l)) \frac{(\partial/\partial\phi^*)\mathcal{N}_{t_l}(\mathbf{y}_l)}{\mathcal{N}_{t_l}(\mathbf{y}_l)}\end{aligned}$$

As a result, condition (26) becomes

$$\sum_t P_{\theta'}(\mathbf{t}|\mathbf{y}) \sum_{l=1}^n (\delta(t',t_l) + \delta(t'',t_l))(y_{l,i} - m_i(t_l)) = 0 \qquad i = 1,2$$

and so (30) is replaced by

$$\begin{aligned}m_i^\theta(t') &= m_i^\theta(t'') \\ &= \frac{1}{K^*(t') + K^*(t'')} \sum_{l=1}^n \Big(\alpha_{l-1}^{\theta'}(L(t'))\, p_{\theta'}(t')\, \mathcal{N}_{t'}^{\theta'}(\mathbf{y}_l)\, \beta_l^{\theta'}(R(t')) \\ &\quad + \alpha_{l-1}^{\theta'}(L(t''))\, p_{\theta'}(t'')\, \mathcal{N}_{t''}^{\theta'}(\mathbf{y}_l)\, \beta_l^{\theta'}(R(t'')) \Big) \times y_{l,i} \\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad i = 1,2\end{aligned}$$

and similarly (31) is replaced by

$$\begin{aligned}\sigma_i^\theta(t')^2 &= \sigma_i^\theta(t'')^2 \\ &= \frac{1}{k^*(t') + k^*(t'')} \sum_{l=1}^n \Big(\alpha_{l-1}^{\theta'}(L(t'))\, p_{\theta'}(t')\, \mathcal{N}_{t'}^{\theta'}(\mathbf{y}_l)\, \beta_l^{\theta'}(R(t')) \\ &\quad + \alpha_{l-1}^{\theta'}(L(t''))\, p_{\theta'}(t'')\, \mathcal{N}_{t''}^{\theta'}(\mathbf{y}_l)\, \beta_l^{\theta'}(R(t'')) \Big) \times (y_{l,i} - m_i^\theta(t'))^2\end{aligned}$$

and (32) by

$$\rho^{\theta}(t') = \rho^{\theta}(t'')$$
$$= \frac{1}{k^*(t') + k^*(t'')} \sum_{l=1}^{n} \Big(\alpha_{l-1}^{\theta'}(L(t'))\, p_{\theta'}(t')\, \mathcal{N}_{t'}^{\theta'}(\mathbf{y}_l)\, \beta_l^{\theta'}(R(t'))$$
$$+ \alpha_{l-1}^{\theta'}(L(t''))\, p_{\theta'}(t'')\, \mathcal{N}_{t''}^{\theta'}(\mathbf{y}_l)\, \beta_l^{\theta'}(R(t'')) \Big)$$
$$\times (y_{l,1} - m_1^{\theta}(t'))(y_{l,2} - m_2^{\theta}(t'))$$

We thus see that all that needs to be done, even for estimating real vector output parameters, is simply the pooling of contents of counters corresponding to those transitions whose output probabilities are to be tied.

9.6 Tied Mixtures

The current continuous speech recognizers use as acoustic processor outputs vectors of cepstral coefficients [3]. Their production probabilities must be tied not only because there is insufficient data to estimate them, but also because the calculation at recognition time of candidate probabilities pertaining to the observed vectors $\mathbf{y}_l$ may be computationally too expensive.

The ingenious accepted remedy is referred to as the method of *tied mixtures* [4]. The number of different normal output densities $\mathcal{N}_i(\mathbf{y})$, $i = 1, 2, \ldots, M$ is limited to some tolerable number M, and the output density corresponding to any particular transition t is specified by the formula

$$P(\mathbf{y}|t) = \sum_{i=1}^{M} q(i|t)\mathcal{N}_i(\mathbf{y}) \tag{34}$$

The computational saving is obvious. As the acoustic processor generates the successive output vectors $\mathbf{y}_l$, the recognizer computes (possibly in parallel) the values $\mathcal{N}_i(\mathbf{y})$, $i = 1, 2, \ldots, M$. These values are then employed in formulas (34) when evaluating the likelihood of transitions through various HMMs. Of course, there is no requirement that for any particular i, t combination, $q(i|t) > 0$. So although M may even be of the order of several hundred, the number of nonzero terms in (34) would typically be of the order of ten.

There is an additional important reason for employing the mixture formula. Although the Baum-Welch algorithm conveniently estimates the

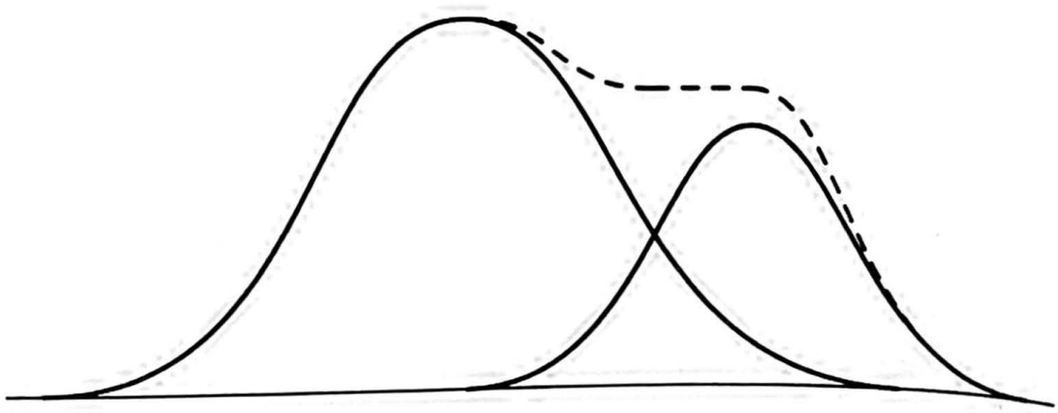


Figure 9.1
A density function resulting from the superposition of two Gaussian densities

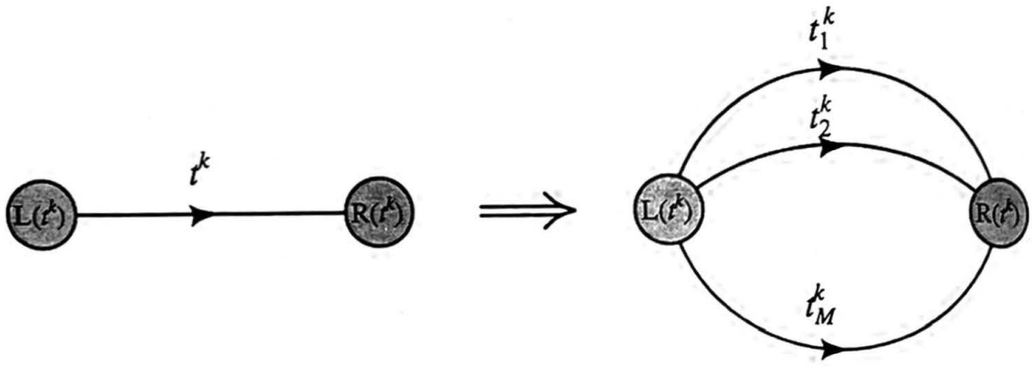


Figure 9.2
Replacement of a single transition by a set illustrating an analysis of probability weight estimation for tied mixture distributions

parameters of normal densities, these do not adequately reflect the speech process. Because of their squared exponent, *Gaussian* density values *decay* too fast as the observable variable deviates from its mean. The simple expedient of mixing several densities having different means can slow this rate of decay. Figure 9.1 illustrates this for the one-dimensional case.

In principle, it is easy to estimate the parameters of HMMs whose outputs are generated according to the formula (34). We need only to replace in the HMM a single transition t^k by M parallel transitions $t_i^k, i = 1, 2, \ldots, M$ having the same source and target states $L(t^k)$ and $R(t^k)$.[15] The output density associated with the transition t_i^k is then $\mathcal{N}_i(\mathbf{y})$. During the parameter estimation process all the output densities for dif-

15. See figure 9.2.

ferent values of k at transitions t_i^k are tied while the transition probabilities $p(t_i^k) = q(i|t^k)$ are determined in the usual way by formula (29).

9.7 Additional Reading

An easy to read exposition of the EM algorithm with some accompanying examples can be found in reference [5]. Since the EM algorithm is in many cases computationally expensive, the possibilities of acceleration [6] as well as the question of the rate of convergence [7] are of overriding interest.

We pointed out in section 9.6 the usefulness of tied mixtures [4]. Independent mixtures, however, give potentially even better results provided enough training material is available for accurate estimation [8]–[12]. State-of-the-art methods of tying [13] are based on growing decision trees as described in chapter 10. We will return to the specific problem of HMM state tying in chapter 12.

References

[1] A.P. Dempster, N.M. Laird, and D.B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm", *Journal of the Royal Statistical Society*, ser. B, vol. 39, pp. 1–38, 1977.

[2] L. Baum, "An inequality and associated maximization technique in statistical estimation of probabilistic functions of a Markov process," *Inequalities*, vol. 3, pp. 1–8, 1972.

[3] S.B. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-28, no. 4, pp. 357–66, August 1980.

[4] J.R. Bellegarda and D. Nahamoo, "Tied mixture continuous parameter modeling for speech recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-38, no, 12, pp. 2033–45, December 1990.

[5] T.K. Moon, "The Expectation-Maximization algorithm," *IEEE Signal Processing Magazine*, vol. 13, no. 6, pp. 47–60, November 1996.

[6] Isaac Meilijson, "A fast improvement to the EM algorithm on its own terms," *Journal of Royal Statistical Society*, ser. B, vol. 51, no. 1, pp. 127–38, 1989.

[7] C.F.J. Wu, "On the convergence properties of the EM algorithm," *The Annals of Statistics*, vol. 11, no. 1, pp. 95–103, 1983.

[8] A. Nadas and D. Nahamoo, "Automatic speech recognition via pseudo-independent marginal mixtures," *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 1285–87, Dallas, Tx, April 1987.

[9] R.A. Redner and H.F. Walker, "Mixture densities, maximum likelihood and the EM algorithm," *SIAM Review*, vol. 26, no. 2, pp. 195–39, April 1984.

[10] L.R. Rabiner, B.H. Juang, S.E. Levinson, and M.M. Sondhi, "Recognition of isolated digits using hidden Markov models with continouous mixture densities," *AT&T Technical Journal*, vol. 64, no. 6, pp. 1211–34, July-August 1985.

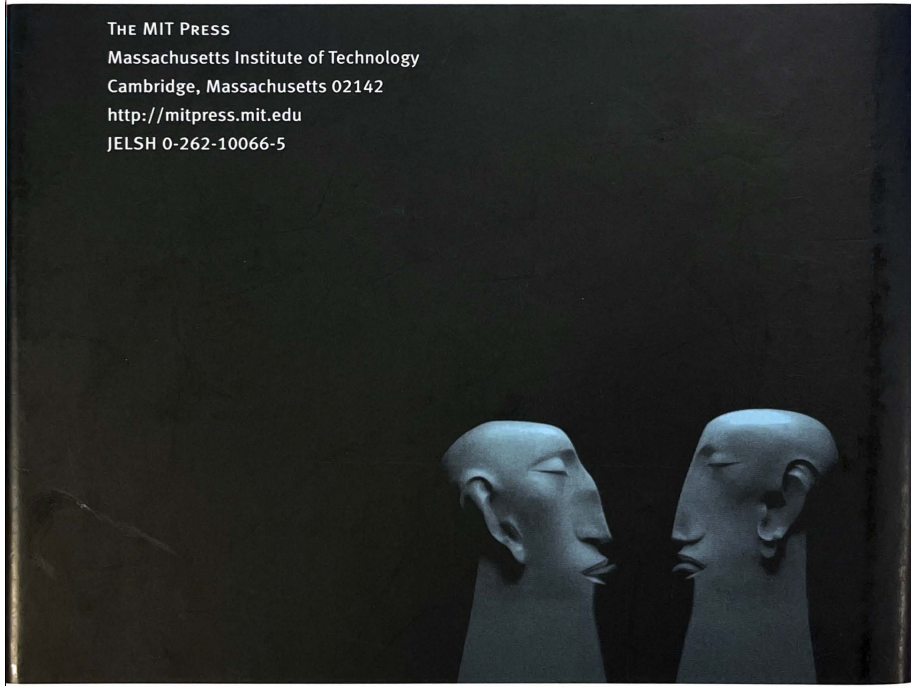
[11] B.H Juang, "Maximum-likelihood estimation for mixture multivariate stochastic observations of Markov chains," *AT&T Technical Journal*, vol. 64, no. 6, pp. 1235–49, July-August 1985.

[12] L.R. Rabiner, B.H. Juang, S.E. Levinson, and M.M. Sondhi, "Some properties of continuous hidden Markov model representations," *AT&T Technical Journal*, vol. 64, no. 6, pp. 1251–70, July-August 1985.

[13] S.J. Young, J.J. Odell, P.C. Woodland, "Tree-based state tying for high accuracy acoustic modelling," *Proceedings of the DARPA Speech and Natural Language Processing Workshop*, 307–12, Plainsboro, March 1994.

"For the first time, researchers in this field will have a book that will serve as 'the bible' for many aspects of language and speech processing. Frankly, I can't imagine a person working in this field not wanting to have a personal copy."
—Victor Zue, MIT Laboratory for Computer Science

The MIT Press
Massachusetts Institute of Technology
Cambridge, Massachusetts 02142
http://mitpress.mit.edu
JELSH 0-262-10066-5



9 780262 100663 90000