TK 10
.S65
R33
1993

**PRENTICE HALL SIGNAL PROCESSING SERIES**

*Alan V. Oppenheim, Series Editor*

ANDREWS AND HUNT *Digital Image Restoration*
BRIGHAM *The Fast Fourier Transform*
BRIGHAM *The Fast Fourier Transform and Its Applications*
BURDIC *Underwater Acoustic System Analysis, 2/E*
CASTLEMAN *Digital Image Processing*
COWAN AND GRANT *Adaptive Filters*
CROCHIERE AND RABINER *Multirate Digital Signal Processing*
DUDGEON AND MERSEREAU *Multidimensional Digital Signal Processing*
HAMMING *Digital Filters, 3/E*
HAYKIN, ED. *Advances In Spectrum Analysis and Array Processing, Vols. I & II*
HAYKIN, ED. *Array Signal Processing*
JAYANT AND NOLL *Digital Coding Of Waveforms*
JOHNSON AND DUDGEON *Array Signal Processing: Concepts and Techniques*
KAY *Fundamentals Of Statistical and Signal Processing: Estimation Theory*
KAY *Modern Spectral Estimation*
KINO *Acoustic Waves: Devices, Imagining, and Analog Signal Processing*
LEA, ED. *Trends In Speech Recognition*
LIM *Two-Dimensional Signal and Image Processing*
LIM, ED. *Speech Enhancement*
LIM AND OPPENHEIM, EDS. *Advanced Topics In Signal Processing*
MARPLE *Digital Spectral Analysis with Applications*
MCCLELLAN AND RADER *Number Theory In Digital Signal Processing*
MENDEL *Lessons In Digital Estimation Theory*
NIKIAS AND PETROPULU *Higher-Order Spectra Analysis: A Nonlinear Signal-Processing Framework*
OPPENHEIM, ED. *Applications Of Digital Signal Processing*
OPPENHEIM AND NAWAB, EDS. *Symbolic and Knowledge-Based Signal Processing*
OPPENHEIM, WILLSKY, WITH YOUNG *Signals and Systems*
OPPENHEIM AND SCHAFER *Digital Signal Processing*
OPPENHEIM AND SCHAFER *Discrete-Time Signal Processing*
PICINBONO *Random Signals and Systems*
QUACKENBUSH ET AL. *Objective Measures Of Speech Quality*
RABINER AND GOLD *Theory and Applications Of Digital Signal Processing*
RABINER AND JUANG *Fundamentals of Speech Recognition*
RABINER AND SCHAFER *Digital Processing Of Speech Signals*
ROBINSON AND TREITEL *Geophysical Signal Analysis*
STEARNS AND DAVID *Signal Processing Algorithms in Fortran and C*
STEARNS AND HUSH *Digital Signal Anaysis, 2/E*
THERRIEN *Discrete Random Signal and Statistical Signal Processing*
TRIBOLET *Seismic Applications Of Homomorphic Signal Processing*
VAIDYANATHAN *Multirate Systems and Filter Banks*
WIDROW AND STEARNS *Adaptive Signal Processing*

# FUNDAMENTALS OF SPEECH RECOGNITION

Lawrence Rabiner
Biing-Hwang Juang

中央大學 書藏

Prentice-Hall International, Inc.

1280065
93-3U093

0-13-285826-6

This edition may be sold only in those countries to which it is consigned by Prentice-Hall International. It is not to be re-exported and it is not for sale in the U.S.A., Mexico, or Canada.

©1993 by AT&T. All rights reserved.

Published by PTR Prentice-Hall, Inc.
A Simon & Schuster Company
Englewood Cliffs, New Jersey 07632

All rights reserved. No part of this book may be reproduced, in any form or by any means, without permission in writing from the publisher.

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-285826-6

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall Hispanoamericana, S.A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Simon & Schuster Asia Pte. Ltd., *Singapore*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*
Prentice-Hall, Inc., *Englewood Cliffs, New Jersey*

*To the memories of our parents*

***Dr. Nathan Rabiner***
*and*
***Mrs. L. Y. Juang***
*for teaching us that anything worthwhile requires dedication and hard work.*
*And to our families*
***Suzanne, Sheri, Wendi and Joni Rabiner***
*and*
***Hsiao-Yun, Lynn and Ryan Juang***
*for their love and the time they gave up in order for us to work on this book.*

# CONTENTS


LIST OF FIGURES xiii

LIST OF TABLES xxix

PREFACE xxxi

1 FUNDAMENTALS OF SPEECH RECOGNITION 1

1.1 Introduction 1
1.2 The Paradigm for Speech Recognition 3
1.3 Outline 3
1.4 A Brief History of Speech-Recognition Research 6

2 THE SPEECH SIGNAL: PRODUCTION, PERCEPTION, AND ACOUSTIC-PHONETIC CHARACTERIZATION 11

2.1 Introduction 11
2.1.1 The Process of Speech Production and Perception in Human Beings 11
2.2 The Speech-Production Process 14
2.3 Representing Speech in the Time and Frequency Domains 17
2.4 Speech Sounds and Features 20

2.4.1 The Vowels 21
2.4.2 Diphthongs 28
2.4.3 Semivowels 29
2.4.4 Nasal Consonants 30
2.4.5 Unvoiced Fricatives 31
2.4.6 Voiced Fricatives 32
2.4.7 Voiced and Unvoiced Stops 33
2.4.8 Review Exercises 37
2.5 Approaches to Automatic Speech Recognition by Machine 42
2.5.1 Acoustic-Phonetic Approach to Speech Recognition 45
2.5.2 Statistical Pattern-Recognition Approach to Speech Recognition 51
2.5.3 Artificial Intelligence (AI) Approaches to Speech Recognition 52
2.5.4 Neural Networks and Their Application to Speech Recognition 54
2.6 Summary 65

3 SIGNAL PROCESSING AND ANALYSIS METHODS FOR SPEECH RECOGNITION 69

3.1 Introduction 69
3.1.1 Spectral Analysis Models 70
3.2 The Bank-of-Filters Front-End Processor 73
3.2.1 Types of Filter Bank Used for Speech Recognition 77
3.2.2 Implementations of Filter Banks 80
3.2.3 Summary of Considerations for Speech-Recognition Filter Banks 92
3.2.4 Practical Examples of Speech-Recognition Filter Banks 93
3.2.5 Generalizations of Filter-Bank Analyzer 95
3.3 Linear Predictive Coding Model for Speech Recognition 97
3.3.1 The LPC Model 100
3.3.2 LPC Analysis Equations 101
3.3.3 The Autocorrelation Method 103
3.3.4 The Covariance Method 106
3.3.5 Review Exercise 107
3.3.6 Examples of LPC Analysis 108
3.3.7 LPC Processor for Speech Recognition 112
3.3.8 Review Exercises 117
3.3.9 Typical LPC Analysis Parameters 121
3.4 Vector Quantization 122
3.4.1 Elements of a Vector Quantization Implementation 123
3.4.2 The VQ Training Set 124
3.4.3 The Similarity or Distance Measure 125
3.4.4 Clustering the Training Vectors 125
3.4.5 Vector Classification Procedure 128
3.4.6 Comparison of Vector and Scalar Quantizers 129



3.4.7 Extensions of Vector Quantization 129
3.4.8 Summary of the VQ Method 131
3.5 Auditory-Based Spectral Analysis Models 132
3.5.1 The EIH Model 134
3.6 Summary 139

4 PATTERN-COMPARISON TECHNIQUES 141

4.1 Introduction 141
4.2 Speech (Endpoint) Detection 143
4.3 Distortion Measures—Mathematical Considerations 149
4.4 Distortion Measures—Perceptual Considerations 150
4.5 Spectral-Distortion Measures 154
4.5.1 Log Spectral Distance 158
4.5.2 Cepstral Distances 163
4.5.3 Weighted Cepstral Distances and Liftering 166
4.5.4 Likelihood Distortions 171
4.5.5 Variations of Likelihood Distortions 177
4.5.6 Spectral Distortion Using a Warped Frequency Scale 183
4.5.7 Alternative Spectral Representations and Distortion Measures 190
4.5.8 Summary of Distortion Measures—Computational Considerations 193
4.6 Incorporation of Spectral Dynamic Features into the Distortion Measure 194
4.7 Time Alignment and Normalization 200
4.7.1 Dynamic Programming—Basic Considerations 204
4.7.2 Time-Normalization Constraints 208
4.7.3 Dynamic Time-Warping Solution 221
4.7.4 Other Considerations in Dynamic Time Warping 229
4.7.5 Multiple Time-Alignment Paths 232
4.8 Summary 238

5 SPEECH RECOGNITION SYSTEM DESIGN AND IMPLEMENTATION ISSUES 242

5.1 Introduction 242
5.2 Application of Source-Coding Techniques to Recognition 244
5.2.1 Vector Quantization and Pattern Comparison Without Time Alignment 244
5.2.2 Centroid Computation for VQ Codebook Design 246
5.2.3 Vector Quantizers with Memory 254
5.2.4 Segmental Vector Quantization 256
5.2.5 Use of a Vector Quantizer as a Recognition Preprocessor 257
5.2.6 Vector Quantization for Efficient Pattern Matching 263
5.3 Template Training Methods 264
5.3.1 Casual Training 265

5.3.2 Robust Training 266
5.3.3 Clustering 267
5.4 Performance Analysis and Recognition Enhancements 274
5.4.1 Choice of Distortion Measures 274
5.4.2 Choice of Clustering Methods and kNN Decision Rule 277
5.4.3 Incorporation of Energy Information 280
5.4.4 Effects of Signal Analysis Parameters 282
5.4.5 Performance of Isolated Word-Recognition Systems 284
5.5 Template Adaptation to New Talkers 285
5.5.1 Spectral Transformation 286
5.5.2 Hierarchical Spectral Clustering 288
5.6 Discriminative Methods in Speech Recognition 291
5.6.1 Determination of Word Equivalence Classes 294
5.6.2 Discriminative Weighting Functions 297
5.6.3 Discriminative Training for Minimum Recognition Error 302
5.7 Speech Recognition in Adverse Environments 305
5.7.1 Adverse Conditions in Speech Recognition 306
5.7.2 Dealing with Adverse Conditions 309
5.8 Summary 317

6 THEORY AND IMPLEMENTATION OF HIDDEN MARKOV MODELS 321

6.1 Introduction 321
6.2 Discrete-Time Markov Processes 322
6.3 Extensions to Hidden Markov Models 325
6.3.1 Coin-Toss Models 326
6.3.2 The Urn-and-Ball Model 328
6.3.3 Elements of an HMM 329
6.3.4 HMM Generator of Observations 330
6.4 The Three Basic Problems for HMMs 333
6.4.1 Solution to Problem 1—Probability Evaluation 334
6.4.2 Solution to Problem 2—"Optimal" State Sequence 337
6.4.3 Solution to Problem 3—Parameter Estimation 342
6.4.4 Notes on the Reestimation Procedure 347
6.5 Types of HMMs 348
6.6 Continuous Observation Densities in HMMs 350
6.7 Autoregressive HMMs 352
6.8 Variants on HMM Structures—Null Transitions and Tied States 356
6.9 Inclusion of Explicit State Duration Density in HMMs 358
6.10 Optimization Criterion—ML, MMI, and MDI 362
6.11 Comparisons of HMMs 364
6.12 Implementation Issues for HMMs 365
6.12.1 Scaling 365
6.12.2 Multiple Observation Sequences 369
6.12.3 Initial Estimates of HMM Parameters 370





6.12.4 Effects of Insufficient Training Data 370
6.12.5 Choice of Model 371
6.13 Improving the Effectiveness of Model Estimates 372
6.13.1 Deleted Interpolation 372
6.13.2 Bayesian Adaptation 373
6.13.3 Corrective Training 376
6.14 Model Clustering and Splitting 377
6.15 HMM System for Isolated Word Recognition 378
6.15.1 Choice of Model Parameters 379
6.15.2 Segmental K-Means Segmentation into States 382
6.15.3 Incorporation of State Duration into the HMM 384
6.15.4 HMM Isolated-Digit Performance 385
6.16 Summary 386

7 SPEECH RECOGNITION BASED ON CONNECTED WORD MODELS 390

7.1 Introduction 390
7.2 General Notation for the Connected Word-Recognition Problem 393
7.3 The Two-Level Dynamic Programming (Two-Level DP) Algorithm 395
7.3.1 Computation of the Two-Level DP Algorithm 399
7.4 The Level Building (LB) Algorithm 400
7.4.1 Mathematics of the Level Building Algorithm 401
7.4.2 Multiple Level Considerations 405
7.4.3 Computation of the Level Building Algorithm 407
7.4.4 Implementation Aspects of Level Building 410
7.4.5 Integration of a Grammar Network 414
7.4.6 Examples of LB Computation of Digit Strings 416
7.5 The One-Pass (One-State) Algorithm 416
7.6 Multiple Candidate Strings 420
7.7 Summary of Connected Word Recognition Algorithms 423
7.8 Grammar Networks for Connected Digit Recognition 425
7.9 Segmental K-Means Training Procedure 427
7.10 Connected Digit Recognition Implementation 428
7.10.1 HMM-Based System for Connected Digit Recognition 429
7.10.2 Performance Evaluation on Connected Digit Strings 430
7.11 Summary 432

8 LARGE VOCABULARY CONTINUOUS SPEECH RECOGNITION 434

8.1 Introduction 434
8.2 Subword Speech Units 435
8.3 Subword Unit Models Based on HMMs 439
8.4 Training of Subword Units 441

8.5 Language Models for Large Vocabulary Speech Recognition 447
8.6 Statistical Language Modeling 448
8.7 Perplexity of the Language Model 449
8.8 Overall Recognition System Based on Subword Units 450
8.8.1 Control of Word Insertion/Word Deletion Rate 454
8.8.2 Task Semantics 454
8.8.3 System Performance on the Resource Management Task 454
8.9 Context-Dependent Subword Units 458
8.9.1 Creation of Context-Dependent Diphones and Triphones 460
8.9.2 Using Interword Training to Create CD Units 461
8.9.3 Smoothing and Interpolation of CD PLU Models 462
8.9.4 Smoothing and Interpolation of Continuous Densities 464
8.9.5 Implementation Issues Using CD Units 464
8.9.6 Recognition Results Using CD Units 467
8.9.7 Position Dependent Units 469
8.9.8 Unit Splitting and Clustering 470
8.9.9 Other Factors for Creating Additional Subword Units 475
8.9.10 Acoustic Segment Units 476
8.10 Creation of Vocabulary-Independent Units 477
8.11 Semantic Postprocessor for Recognition 478
8.12 Summary 478

**9 TASK ORIENTED APPLICATIONS OF AUTOMATIC SPEECH RECOGNITION 482**

9.1 Introduction 482
9.2 Speech-Recognizer Performance Scores 484
9.3 Characteristics of Speech-Recognition Applications 485
9.3.1 Methods of Handling Recognition Errors 486
9.4 Broad Classes of Speech-Recognition Applications 487
9.5 Command-and-Control Applications 488
9.5.1 Voice Repertory Dialer 489
9.5.2 Automated Call-Type Recognition 490
9.5.3 Call Distribution by Voice Commands 491
9.5.4 Directory Listing Retrieval 491
9.5.5 Credit Card Sales Validation 492
9.6 Projections for Speech Recognition 493

**INDEX 497**

# LIST OF FIGURES

1.1 General block diagram of a task-oriented speech-recognition system. 3

2.1 Schematic diagram of speech-production/speech-perception process (after Flanagan [unpublished]). 12
2.2 Alternative view of speech-production/speech-perception process (after Rabiner and Levinson [1]). 13
2.3 Mid-sagittal plane X-ray of the human vocal apparatus (after Flanagan et al. [2]). 15
2.4 Schematic view of the human vocal mechanism (after Flanagan [3]). 16
2.5 Glottal volume velocity and resulting sound pressure at the start of a voiced sound (after Ishizaka and Flanagan [4]). 16
2.6 Schematic representation of the complete physiological mechanism of speech production (after Flanagan [3]). 17
2.7 Waveform plot of the beginning of the utterance "It's time." 18
2.8 Wideband and narrowband spectrograms and speech amplitude for the utterance "Every salt breeze comes from the sea." 19

2.9 Wideband spectrogram and formant frequency representation of the utterance "Why do I owe you a letter" (after Atal and Hanauer [5]). 21
2.10 Wideband spectrogram and intensity contour of the phrase "Should we chase." 22
2.11 The speech waveform and a segmentation and labeling of the constituent sounds of the phrase "Should we chase." 23
2.12 Chart of the classification of the standard phonemes of American English into broad sound classes. 25
2.13 Articulatory configurations for typical vowel sounds (after Flanagan [3]). 25
2.14 Acoustic waveform plots of typical vowel sounds. 26
2.15 Spectrograms of the vowel sounds. 27
2.16 Measured frequencies of first and second formants for a wide range of talkers for several vowels (after Peterson & Barney [7]). 27
2.17 The vowel triangle with centroid positions of the common vowels. 28
2.18 Spectrogram plots of four diphthongs. 30
2.19 Time variation of the first two formants for the diphthongs (after Holbrook and Fairbanks [9]). 31
2.20 Waveforms for the sequences /ə-m-a/ and /ə-n-a/. 32
2.21 Spectrograms of the sequences /ə-m-a/ and ə-n-a/. 33
2.22 Waveforms for the sounds /f/, /s/ and /sh/ in the context /ə-x-a/ where /x/ is the unvoiced fricative. 34
2.23 Spectrogram comparisons of the sounds /ə-f-a/, /ə-s-a/ and /ə-sh-a/. 34
2.24 Waveforms for the sequences /ə-v-a/ and /ə-zh-a/. 35
2.25 Spectrograms for the sequences /ə-v-a/ and /ə-zh-a/. 36
2.26 Waveform for the sequence /ə-b-a/. 36
2.27 Waveforms for the sequences /ə-p-a/ and /ə-t-a/. 37
2.28 Spectrogram comparisons of the sequences of voiced (/ə-b-a/) and voiceless (/ə-p-a/ and /ə-t-a/) stop consonants. 38
2.29 Spectrograms of the 11 isolated digits, 0 through 9 plus oh, in random sequence. 40
2.30 Spectrograms of two connected digit sequences. 41
2.31 Phoneme lattice for word string. 43
2.32 Block diagram of acoustic-phonetic speech-recognition system. 45
2.33 Acoustic-phonetic vowel classifier. 47
2.34 Binary tree speech sound classifier. 48
2.35 Segmentation and labeling for word sequence "seven–six." 49
2.36 Segmentation and labeling for word sequence "did you." 50
2.37 Block diagram of pattern-recognition speech recognizer. 51



2.38 Illustration of the word correction capability of syntax in speech recognition (after Rabiner and Levinson [1]). 54
2.39 A bottom-up approach to knowledge integration for speech recognition. 55
2.40 A top-down approach to knowledge integration for speech recognition. 55
2.41 A blackboard approach to knowledge integration for speech recognition (after Lesser et al. [11]). 56
2.42 Conceptual block diagram of a human speech understanding system. 56
2.43 Simple computation element of a neural network. 57
2.44 McCullough-Pitts model of neurons (after McCullough and Pitts [12]). 58
2.45 Single-layer and three-layer perceptrons. 59
2.46 A multilayer perceptron for classifying steady vowels based on $F_1$, $F_2$ measurements (after Lippmann [13]). 59
2.47 Model of a recurrent neural network. 60
2.48 A fixed point interpretation of the Hopfield network. 60
2.49 The time delay neural network computational element (after Waibel et al. [14]). 63
2.50 A TDNN architecture for recognizing /b/, /d/ and /g/ (after Waibel et al. [14]). 64
2.51 A combination neural network and matched filter for speech recognition (after Tank & Hopfield [15]). 65
2.52 Example illustrating the combination of a neural network and a set of matched filters (after Tank & Hopfield [15]). 66
2.53 The hidden control neural network (after Levin [16]). 67

3.1 (a) Pattern recognition and (b) acoustic phonetic approaches to speech recognition. 71
3.2 Bank-of-filters analysis model. 72
3.3 LPC analysis model. 72
3.4 Complete bank-of-filters analysis model. 74
3.5 Typical waveforms and spectra for analysis of a pure sinusoid in the filter-bank model. 75
3.6 Typical waveforms and spectra of a voice speech signal in the bank-of-filters analysis model. 76
3.7 Ideal (a) and realistic (b) set of filter responses of a $Q$-channel filter bank covering the frequency range $F_s/N$ to $(Q + {}^1/_2)F_s/N$. 77
3.8 Ideal specifications of a 4-channel octave band-filter bank (a), a 12-channel third-octave band filter bank (b), and a 7-channel critical band scale filter bank (c) covering the telephone bandwidth range (200–3200 Hz). 79

3.9 The variation of bandwidth with frequency for the perceptually based critical band scale. 79
3.10 The signals $s(m)$ and $w(n-m)$ used in evaluation of the short-time Fourier transform. 81
3.11 Short-time Fourier transform using a long (500 points or 50 msec) Hamming window on a section of voiced speech. 82
3.12 Short-time Fourier transform using a short (50 points or 5 msec) Hamming window on a section of voiced speech. 82
3.13 Short-time Fourier transform using a long (500 points or 50 msec) Hamming window on a section of unvoiced speech. 83
3.14 Short-time Fourier transform using a short (50 points or 5 msec) Hamming window on a section of unvoiced speech. 83
3.15 Linear filter interpretation of the short-time Fourier transform. 84
3.16 FFT implementation of a uniform filter bank. 89
3.17 Direct form implementation of an arbitrary nonuniform filter bank. 89
3.18 Two arbitrary nonuniform filter-bank ideal filter specifications consisting of either 3 bands (part a) or 7 bands (part b). 90
3.19 Tree structure implementation of a 4-band, octave-spaced, filter bank. 92
3.20 Window sequence, $w(n)$, (part a), the individual filter response (part b), and the composite response (part c) of a $Q = 15$ channel, uniform filter bank, designed using a 101-point Kaiser window smoothed lowpass window (after Dautrich et al. [4]). 94
3.21 Window sequence, $w(n)$, (part a), the individual filter responses (part b), and the composite response (part c) of a $Q = 15$ channel, uniform filter bank, designed using a 101-point Kaiser window directly as the lowpass window (after Dautrich et al. [4]). 95
3.22 Individual channel responses (parts a to d) and composite filter response (part c) of a $Q = 4$ channel, octave band design, using 101-point FIR filters in each band (after Dautrich et al. [4]). 96
3.23 Individual channel responses and composite filter response of a $Q = 12$ channel, 1/3 octave band design, using 201-point FIR filters in each band (after Dautrich et al. [4]). 97
3.24 Individual channel responses (parts a to g) and composite filter response (part h) of a $Q = 7$ channel critical band filter bank design (after Dautrich et al. [4]). 98
3.25 Individual channel responses and composite filter response of a $Q = 13$ channel, critical band spacing filter bank, using highly overlapping filters in frequency (after Dautrich et al. [4]). 99
3.26 Generalization of filter-bank analysis model. 99
3.27 Linear prediction model of speech. 100
3.28 Speech synthesis model based on LPC model. 101



3.29 Illustration of speech sample, weighted speech section, and prediction error for voiced speech where the prediction error is large at the beginning of the section. 104
3.30 Illustration of speech sample, weighted speech section, and prediction error for voiced speech where the prediction error is large at the end of the section. 104
3.31 Illustration of speech sample, weighted speech section, and prediction error for unvoiced speech where there are almost no artifacts at the boundaries of the section. 105
3.32 Typical signals and spectra for LPC autocorrelation method for a segment of speech spoken by a male speaker (after Rabiner et al. [8]). 108
3.33 Typical signals and spectra for LPC autocorrelation method for a segment of speech spoken by a female speaker (after Rabiner et al. [8]). 109
3.34 Examples of signal (differentiated) and prediction error for several vowels (after Strube [9]). 110
3.35 Variation of the RMS prediction error with the number of predictor coefficients, $p$ (after Atal and Hanauer [10]). 110
3.36 Spectra for a vowel sound for several values of predictor order, $p$. 111
3.37 Block diagram of LPC processor for speech recognition. 113
3.38 Magnitude spectrum of LPC preemphasis network for $\tilde{a} = 0.95$. 113
3.39 Blocking of speech into overlapping frames. 114
3.40 Block diagram of the basic VQ training and classification structure. 124
3.41 Partitioning of a vector space into VQ cells with each cell represented by a centroid vector. 126
3.42 Flow diagram of binary split codebook generation algorithm. 127
3.43 Codebook distortion versus codebook size (measured in bits per frame) for both voiced and unvoiced speech (after Juang et al. [12]). 128
3.44 Codebook vector locations in the $F_1 - F_2$ plane (for a 32-vector codebook) superimposed on the vowel ellipses (after Juang et al. [12]). 128
3.45 Model and distortion error spectra for scalar and vector quantizers (after Juang et al. [12]). 130
3.46 Plots and histograms of temporal distortion for scalar and vector quantizers (after Juang et al. [12]). 131
3.47 Physiological model of the human ear. 132
3.48 Expanded view of the middle and inner ear mechanics. 133
3.49 Block diagram of the EIH model (after Ghitza [13]). 135

3.50 Frequency response curves of a cat's basilar membrane (after Ghitza [13]). 136
3.51 Magnitude of EIH for vowel /o/ showing the time-frequency resolution (after Ghitza [13]). 136
3.52 Operation of the EIH model for a pure sinusoid (after Ghitza [13]). 137
3.53 Comparison of Fourier and EIH log spectra for clean and noisy speech signals (after Ghitza [13]). 138

4.1 Contour of digit recognition accuracy (percent correct) as a function of endpoint perturbation (in ms) in a multispeaker digit-recognition experiment. Both the initial (beginning point) and the final (ending point) boundary of the detected speech signal were varied (after Wilpon et al. [2]). 144
4.2 Example of mouth click preceding a spoken word (after Wilpon et al. [2]). 145
4.3 Example of breathy speech due to heavy breathing while speaking (after Wilpon et al. [2]). 146
4.4 Example of click produced at the end of a spoken word (after Wilpon et al. [2]). 147
4.5 Block diagram of the explicit approach to speech endpoint detection. 147
4.6 Block diagram of the implicit approach to speech-endpoint detection. 148
4.7 Examples of word boundaries as determined by the implicit endpoint detection algorithm. 148
4.8 Block diagram of the hybrid approach to speech endpoint detection. 149
4.9 Block diagram of typical speech activity detection algorithm. 149
4.10 LPC pole frequency JNDs as a function of the pole bandwidth; the blank circles denote positive frequency perturbations, and the solid circles represent negative frequency perturbations; the fitting curves are parabolic (after Erell et al. [7]). 153
4.11 LPC pole bandwidth JNDs, in a logarithmic scale, as a function of the pole bandwidth itself (after Erell et al. [7]). 154
4.12 Two typical FFT power spectra, $S(\omega)$, of the sound /æ/ in a log scale and their difference magnitude $|V(\omega)|$ as a function of frequency. 159
4.13 LPC model spectra corresponding to the FFT spectra in Figure 4.12, plotted also in a log scale, and their difference magnitude $|V(\omega)|$ as a function of frequency. 159
4.14 Two typical FFT power spectra, $S(\omega)$, of the sound /sh/ in a log scale and their difference magnitude $|V(\omega)|$ as a function of frequency. 160



4.15 LPC model spectra corresponding to the FFT spectra in Figure 4.14, plotted also in a log scale, and their difference magnitude $|V(\omega)|$ as a function of frequency. 160
4.16 Typical FFT power spectra of the sounds /æ/ and /i/ respectively and their difference magnitude as a function of frequency. 161
4.17 LPC model spectra corresponding to the FFT spectra in Figure 4.16 and their difference magnitude $|V(\omega)|$ as a function of frequency. 161
4.18 Scatter plot of $d_2^2$, the cepstral distance, versus $2d_c^2(L)$, the truncated cepstral distance (multiplied by 2), for 800 pairs of all-pole model spectra; the truncation is at $L = 20$ (after Gray and Markel [9]). 165
4.19 Scatter plot of $d_2^2$, the cepstral distance, versus $2d_c^2(L)$, the truncated cepstral distance (multiplied by 2), for 800 pairs of all-pole model spectra; the truncation is at $L = 30$ (after Gray and Markel [9]). 165
4.20 Effects of cepstral liftering on a log LPC spectrum, as a function of the lifter length ($L = 8$ to 16) (after Juang et al. [11]). 169
4.21 Comparison of (a) original sequence of LPC log magnitude spectra; (b) liftered LPC log magnitude spectra, and (c) liftered log magnitude spectra (after Juang et al. [11]). 170
4.22 Comparison of the distortion integrands $V^2(\omega)/2$ and $e^{V(\omega)} - V(\omega) - 1$ (after Gray and Markel [9]). 173
4.23 A scatter plot of $d_{IS}(1/|A_p|^2, 1/|A|^2) + 1$ versus $d_{IS}(1/|A|^2, 1/|A_p|^2) + 1$ as measured from 6800 pairs of speech model spectra. 176
4.24 A linear system with transfer function $H(z) = A(z)/B(z)$. 177
4.25 A scatter diagram of the log spectral distance versus the COSH distortion as measured from a database of 6800 pairs of speech spectra. 178
4.26 LPC spectral pair and various spectral weighting functions; $W_1(\omega)$, $W_2(\omega)$, $W_3(\omega)$ and $W_4(\omega)$ are defined in (4.71), (4.72), (4.73), and (4.74), respectively. 180
4.27a An example of the cosh spectral deviation $F_4(\omega)$ and its weighted version using $W_3(\omega) = 1/\left|A(e^{j\omega})\right|^2$ as the weighting function; in this case the two spectra are of comparable power levels. 182
4.27b An example of the cosh spectral deviation $F_4(\omega)$ and its weighted version using $W_3(\omega) = 1/\left|A(e^{j\omega})\right|^2$ as the weighting function; in this case, the two spectra have significantly different power levels. 182

4.28 Subjectively perceived pitch, in mels, of a tone as a function of the frequency, in Hz; the upper curve relates the subjective pitch to frequency in a linear scale and the lower curve shows the subjective pitch as a function of the frequency in a logarithmic scale (after Stevens and Volkmann [13]). 184
4.29 The critical bandwidth phenomenon; the critical bandwidth as a function of the frequency at the center of the band (after Zwicker, Flottorp and Stevens [14]). 185
4.30 Real part of exp $[j\theta(b)k]$ as a function of $b$, the Bark scale, for different values of $k$ (after Nocerino et al. [16]). 188
4.31 A filter-bank design in which each filter has a triangle bandpass frequency response with bandwidth and spacing determined by a constant mel frequency interval (spacing = 150 mels, bandwidth = 300 mels) (after Davis and Mermelstein [17]). 190
4.32 (a) Series of cylindrical sections concatenated as an acoustic tube model of the vocal tract; (b) the area function of the cylindrical sections in (a) (after Markel and Gray [10]). 191
4.33 A critical band spectrum (a) of a typical vowel sound and the corresponding log spectral differences $V_{LM}$ (b) and $V_{GM}$ (c) as functions of the critical band number (after Nocerino et al. [16]). 193
4.34 A trajectory of the (2nd) cepstral coefficient with 2nd-order polynomial $(h_1 + h_2 t + h_3 t^2)$ fitting on short portions of the trajectory; the width for polynomial fitting is 7 points. 197
4.35 Scatter diagram showing the correlation between the "instantaneous" cepstral distance, $d_2$, and the "differential" or "dynamic" cepstral distance, $d_{2\delta(t)}$; the correlation index is 0.6. 199
4.36 Linear time alignment for two sequences of different durations. 202
4.37 An example of time normalization of two sequential patterns to a common time index; the time warping functions $\phi_x$ and $\phi_y$ map the individual time index $i_x$ and $i_y$, respectively, to the common time index $k$. 203
4.38 The optimal path problem—finding the minimum cost path from point 1 to point $i$ in as many moves as needed. 205
4.39 A trellis structure that illustrates the problem of finding the optimal path from point $i$ to point $j$ in $M$ steps. 207
4.40 An example of local continuity constraints expressed in terms of coordinate increments (after Myers et al. [23]). 210
4.41 The effects of global path constraints and range limiting on the allowable regions for time warping functions. 215
4.42 Illustration of the extreme cases, $T_x - 1 = Q_{\max}(T_y - 1)$ or $T_y - 1 = Q_{\max}(T_x - 1)$, where only linear time warping (single straight path) is allowed. 216



4.43 Type III local continuity constraints with four types of slope weighting (after Myers et al. [23]). 217
4.44 Type II local continuity constraints with 4 types of slope weighting and their smoothed version in which the slope weights are uniformly redistributed along paths where abrupt weight changes exist (after Myers et al. [23]). 218
4.45 Set of allowable grid points for dynamic programming implementation of local path expansion and contraction by 2 to 1. 225
4.46 The allowable path region for dynamic time alignment with relaxed endpoint constraints. 230
4.47 Set of allowable grid points when opening up the initial point range to 5 frames and the final point range to 9 frames. 231
4.48 The allowable path region for dynamic time alignment with localized range constraints. 232
4.49 Dynamic programming for finding $K$-best paths implemented in a parallel manner. 234
4.50 The serial dynamic programming algorithm for finding the $K$-best paths. 236
4.51 Example illustrating the need for nonlinear time alignment of two versions of a spoken word. 239
4.52 Illustration of the effectiveness of dynamic time warping alignment of two versions of a spoken word. 240

5.1 A vector-quantizer–based speech-recognition system. 246
5.2 A trellis quantizer as a finite state machine. 255
5.3 Codebook training for segmental vector quantization. 256
5.4 Block diagram of isolated word recognizer incorporating a word-based VQ preprocessor and a DTW-based postprocessor (after Pan et al. [4]). 258
5.5 Plots of the variation of preprocessor performance parameters $P_1$, $E_1$, and $(1 - \gamma)$ as a function of the distortion threshold $D'$ for several codebook sizes for the digits vocabulary (after Pan et al. [4]). 260
5.6 Plots of the variation of preprocessor performance parameters $E_2$ and $\beta$ as a function of the distortion threshold $D'$ for several codebook sizes for the digits vocabulary (after Pan et al. [4]). 260
5.7 Plots of average fraction of decisions made by the preprocessor, $\gamma$, versus preprocessor decision threshold $D''$ for several codebook sizes for the digits vocabulary (after Pan et al. [4]). 262
5.8 Plots of average fraction of candidate words, $\beta$, passed on to the postprocessor, versus preprocessor decision threshold $D''$ for several codebook sizes for the digits vocabulary (after Pan et al. [4]). 262

5.9 Accumulated DTW distortion scores versus test frame based on casual training with two reference patterns per word (after Rabiner et al. [5]). 265
5.10 A flow diagram of the UWA clustering procedure (after Wilpon and Rabiner [7]). 269
5.11 A flow diagram of the MKM clustering procedure (after Wilpon and Rabiner [7]). 272
5.12 Recognition accuracy (percent correct) as a function of the number of templates per word based on template clustering with kNN decisions ($k = 1, 2, 3, 4$); (a) the top candidate is the correct word, (b) the correct word is among the top 5 candidates (after Rabiner et al. [5]). 278
5.13 Average digit error rate as a function of the number of templates per word for a database of 1000 digits with clusters generated by the UWA and the MKM clustering procedures respectively (after Wilpon and Rabiner [7]). 280
5.14 The nonlinearity function, $g$, applied to the log energy difference between two frames for the energy distance calculation. 282
5.15 Plots of word-recognition error rate versus the analysis frame size (in samples) with various frame shift intervals (33, 67, 100 and 200 samples) (after Rabiner and Wilpon [13]). 283
5.16 Plots of word recognition error rate versus LPC analysis order for three LPC-related distortion measures, the likelihood ratio measure (LR), the bandpass liftered cepstral measure (BPCEP), and the cepstral measure with additional delta cepstrum (DCEP) (after Rabiner and Wilpon [13]). 284
5.17 Four different speaker adaptation scenarios. 286
5.18 Hierarchical codebook adaptation algorithm (after Furui [14]). 290
5.19 Cepstral distortion between input speech pattern and reference templates resulted from hierarchical code-word adaptation (NA=no adaptation); — progressive adaptation, --- direct adaptation (after Furui [14]). 290
5.20 Frame-distortion sequences between pairs of speech utterances: (a) utterances of the same word, (b) utterances of acoustically different words, and (c) utterances of acoustically confusing words (after Rabiner and Wilpon [15]). 293
5.21 Examples illustrating "word" alignment based on dynamic "phone" warping for word equivalence class clustering (after Rabiner and Wilpon [15]). 295
5.22 Plots of average distortion sequences (—) and sequences of standard deviations of the frame distortion (- - - -) for various word pairs (after Rabiner and Wilpon [15]). 299
5.23 Weighting curves for discriminatively comparing words "I" and "Y" (after Rabiner and Wilpon [15]). 300



5.24 Speech-recognition performance in noisy conditions; •: training and testing have matched S/N as indicated by the abscissa; △: only clean training reference is used and the abscissa indicates that the test S/N; □: training and testing S/N's are mismatched with test S/N fixed at 18 dB and the abscissa indicates the training S/N (after Dautrich et al. [17]). 306
5.25 Noise spectrum in a typical personal office with a SUN 3/110. 307
5.26 Mean customer-premises-to-customer-premises attenuation distortion relative to 1004 Hz in a telephone channel (after Carey et al. [21]). 308
5.27 Schematic of the two-input noise cancelling approach. 310
5.28 Noisy speech-recognition performance of several distortion measures and methods (after Mansour and Juang [41]). 316

6.1 A Markov chain with five states (labeled 1 to 5) with selected state transitions. 323
6.2 Markov model of the weather. 323
6.3 Three possible Markov models that can account for the results of hidden coin-tossing experiments. (a) one-coin model, (b) two-coins model, (c) three-coins model. 328
6.4 An $N$-state urn-and-ball model illustrating the general case of a discrete symbol HMM. 329
6.5 (a) Illustration of the sequence of operations required for the computation of the forward variable $\alpha_{t+1}(j)$. (b) Implementation of the computation of $\alpha_t(i)$ in terms of a lattice of observations $t$, and states $i$. 336
6.6 Sequence of operations required for the computation of the backward variable $\beta_t(i)$. 338
6.7 Illustration of the sequence of operations required for the computation of the joint event that the system is in state $i$ at time $t$ and state $j$ at time $t + 1$. 342
6.8 Illustration of three distinct types of HMMs. (a) A 4-state ergodic model. (b) A 4-state left-right model. (c) A 6-state parallel path left-right model. 349
6.9 Equivalence of a state with a mixture density to a multistate single-density distribution (after Juang et al. [21]). 351
6.10 Examples of networks incorporating null transitions. (a) Left-right model. (b) Finite state network. (c) Grammar network. 357
6.11 Illustration of general interstate connections of (a) a normal HMM with exponential state duration density, and (b) a variable duration HMM with specified state densities and no self-transitions from a state back to itself. 358
6.12 Example of how the process of deleted interpolation can be represented using a state diagram. 373

6.13 Block diagram of an isolated word HMM recognizer (after Rabiner [38]). 379
6.14 Average word error rate (for a digits vocabulary) versus the number of states $N$ in the HMM (after Rabiner et al. [18]). 380
6.15 Comparison of estimated density (jagged contour) and model density (smooth contour) for each of the nine components of the observation vector (eight cepstral components, one log energy component) for state 1 of the digit zero (after Rabiner et al. [38]). 381
6.16 Average word error rate as a function of the minimum discrete density value $\epsilon$ (after Rabiner et al. [18]). 382
6.17 The segmental $k$-means training procedure used to estimate parameter values for the optimal continuous mixture density fit to a finite number of observation sequences (after Rabiner et al. [38]). 383
6.18 Plots of (a) log energy; (b) accumulated log likelihood; and (c) state assignment for one occurrence of the word "six" (after Rabiner et al. [38]). 384
6.19 Histograms of the normalized duration density for the five states of the digit "six" (after Rabiner et al. [38]). 385

7.1 Illustration of an isolated string of digits (upper panel) and a fluently spoken version of the same digit string (lower panel). 391
7.2 Illustration of the connected word-recognition problem. 392
7.3 Determination of the optimum alignment of super-reference pattern $R^s$ to $T$, along with corresponding word boundary frames. 394
7.4 Computation ranges for matching $R_v$ against portions of $T$. 395
7.5 Use of range limiting to reduce the size of individual time warps. 396
7.6 Series of paths ending at frame $e$. 397
7.7 Illustration of standard DTW alignment of super-reference and test patterns (a), and level building alignment (b) (after Myers and Rabiner [4]). 400
7.8 Implementation of level 1 of the level building algorithm (after Myers and Rabiner [4]). 402
7.9 Implementation of level 2 of the level building algorithm (after Myers and Rabiner [4]). 404
7.10 Simple example illustrating level building on two reference patterns of equal length (after Myers and Rabiner [4]). 405
7.11 Computation region of level building algorithm for a fixed-length reference pattern (after Myers and Rabiner [4]). 406
7.12 Reduced computation region using upper- and lower-level constraints (after Myers and Rabiner [4]). 407
7.13 Overall computation pattern of level building algorithm for variable length reference patterns (after Myers and Rabiner [4]). 408



7.14 Illustration of beginning range reduction (after Myers and Rabiner [4]). 412
7.15 Illustration of global range reduction (after Myers and Rabiner [4]). 413
7.16 Illustration of the use of reference pattern uncertainty regions (after Myers and Rabiner [4]). 413
7.17 Summary of level building implementation of computation reduction methods (after Myers and Rabiner [4]). 414
7.18 Level building of the string "51560" (after Myers and Rabiner [4]). 417
7.19 Level building of the string "99211" (after Myers and Rabiner [4]). 418
7.20 The one-pass connected word recognition algorithm (after Bridle et al. [6]). 418
7.21 Combinatorics for the one-pass algorithm. 419
7.22 Description of procedure for determining multiple candidate scores (after Myers and Rabiner [4]). 421
7.23 Candidate strings for a four-level search (after Myers and Rabiner [4]). 422
7.24 Illustration of level building flaw for determining the second-best candidate string (after Myers and Rabiner [10]). 422
7.25 Use of HMMs in the level building procedure. 424
7.26 A typical grammar node of an FSN grammar network (after Lee and Rabiner [9]). 425
7.27 Block diagram of connected word recognition computation. 425
7.28 Three possible grammar networks for connected digit recognition (after Lee and Rabiner [9]). 426
7.29 The segmental $k$-means training algorithm for connected word strings (after Rabiner et al. [13]). 428
7.30 Block diagram of connected digit recognition method (after Rabiner et al. [13]). 429
7.31 Connected digit HMM (after Rabiner et al. [13]). 430
7.32 Average speaking rate of talkers in the two connected digit databases as a function of the number digits per string (after Rabiner et al. [13]). 431

8.1 HMM representations of a word (a) and a subword unit (b). 439
8.2 Representations of the acoustic space of speech by (a) partitioned VQ cells, (b) sets of continuous mixture Gaussian densities, and (c) a continuous-density codebook (after Lee et al. [7]). 440
8.3 Representation of a sentence, word, and subword unit in terms of FSNs. 442
8.4 Creation of composite FSN for sentence "Show all alerts." 443

8.5 Segmentations of a training utterance resulting from the segmental $k$-means training for the first several iterations (after Lee et al. [7]). 444
8.6 Segmentation of an utterance into PLUs (after Lee et al. [7]). 445
8.7 Overall block diagram of subword unit based continuous speech recognizer. 451
8.8 FSN for the NG syntax. 452
8.9 FSN of the WP syntax. 453
8.10 Word and sentence accuracies versus number of mixtures per state for the training subset using the WP syntax. 455
8.11 Word and sentence accuracies versus number of mixtures per state for the training subset using the NG syntax. 456
8.12 Word and sentence accuracies versus number of mixtures per state for the four test sets using the WP syntax. 457
8.13 Word and sentence accuracies versus number of mixtures per state for the four test sets using the NG syntax. 458
8.14 Plots of the number of intraword units, interword units, and combined units as a function of the count threshold. 462
8.15 Deleted interpolation model for smoothing discrete density models. 463
8.16 FSN representation of words with three or more units (a), two units (b), and one unit (c). 466
8.17 Word accuracy (%) as a function of the number of generalized triphone models for several training set sizes (after Lee [2]). 469
8.18 Histograms of the unit separation distance for (a) combined intraword and interword units (1282 PLUs) and (b) separate intraword and interword units (1769 PLUs) (after Lee et al. [19]). 471
8.19 Splitting of subword unit $p_i$ into three clusters (after Lee et al. [7]). 472
8.20 Histograms of likelihood score for four context-independent units (vowels) (after Lee et al. [7]). 472
8.21 Average likelihood scores for sets of PLU models obtained from model splitting (after Lee et al. [7]). 474
8.22 Word networks based on all combinations of units (a), or selected combinations of units (b) (after Lee et al. [7]). 474
8.23 Word and sentence accuracy improvements in RM after semantic processing (after Pieraccini and Lee [26]). 479

9.1 Block diagram of a task-specific voice control and dialog system. 483
9.2 Market sales for speech-recognition hardware over time, in each of five market segments. (Data estimated for 1990–1992 sales.) 488



9.3 Average LPC distance scores for first- and second-recognition candidates for each of the six test speakers (after Rabiner et al. [1]). 489
9.4 Block diagram of a directory listing retrieval system based on spelled spoken names (after Rabiner et al. [1]). 492

# LIST OF TABLES


2.1 A condensed list of phonetic symbols for American English. 24
2.2 Formant frequencies for typical vowels. 29
2.3 Sound Lexicon of Digits 41

4.1 Measured DLs and JNDs for Synthetic Vowel Sounds 152
4.2 Examples of Critical Bandwidth 186
4.3 Values of the Elements in the Warping Matrix 189
4.4 Summary of Spectral Distortion Measures 195
4.5 Summary of sets of local constraints and the resulting path specifications 211
4.6 Values of $Q_{max}$ and $Q_{min}$ for different types of paths 214
4.7 Summary of sets of local constraints, slope weights, and DP recursion formulas 223

5.1 Comparison of single-frame vowel-recognition error rates with various distortion measures (after Rabiner and Soong [8]). 275
5.2 Comparison of recognition error rates for a 39 alphadigit-word vocabulary with various distortion measures (after Nocerino et al. [9]). 276

5.3 Comparison of recognition error rate pertaining to several distortion measures with and without energy information (after Nocerino et al. [9]). 282
5.4 Performance of Isolated Word-Recognition Systems 285
5.5 Attenuation distortion relative to 1004 Hz: short connections. 308

6.1 Average Digit Error Rates for Several Recognizers and Evaluation Sets 386

7.1 Average String Error Rates (%) for Connected Digit Recognition Tests 432

8.1 Set of basic PLUs for speech. 438
8.2 Typical word pronunciations (word lexicon) based on context-independent PLUs. 438
8.3 PLU statistics on count and average likelihood score. 446
8.4 Number of intra-word CD units as a function of count threshold, $T$. 461
8.5 Word error rates as a function of occurrence threshold for the feb 89 test set using intraword units with a 38 component/vector analysis. 468
8.6 Word error rates as a function of occurrence threshold for the feb 89 test set using both intraword and interword units (independently) with a 38 component/vector analysis. 468
8.7 Recognition performance on 122-word, office correspondence task with both VI and VD models (after Hon & Lee [25]). 478
8.8 Recognition performance on 991-word, RM task, with both VI and VD models (after Hon & Lee [25]). 478

9.1 Performance scores for several types of speech-recognition systems as measured under laboratory conditions. 484
9.2 Projections for speech recognition. 494

# PREFACE

This book is an outgrowth of an association between the authors which started over 10 years ago when one of us (BHJ) was a graduate student at the University of California at Santa Barbara and the other (LRR) was a supervisor at AT&T Bell Laboratories. We began our relationship with a mutual interest in the problem of designing and implementing vector quantization for speech processing. This association turned into a full technical partnership and strong friendship when Fred Juang joined Bell Laboratories, initially in the development area and subsequently in research. The spark that ignited formal work on this book was a series of short courses taught by one of us (LRR) on speech recognition. After several iterations of teaching, it became clear that the area of speech recognition, although still changing and growing, had matured to the point where a book that covered its theoretical underpinnings was warranted.

Once we had decided to write this book, there were several key issues that had to be resolved, including how deep to go into areas like linguistics, natural language processing, and the practical side of the problem; whether to discuss individual systems proposed by various research labs around the world; and how extensively to cover applications. Although there were no simple answers to these questions, it rapidly became obvious to us that the fundamental goal of the book would be to provide a theoretically sound, technically accurate, and reasonably complete description of the basic knowledge and ideas that constitute a modern system for speech recognition by machine. With these basic guiding principles in mind, we were able to decide consistently (and hopefully reasonably) what material had to be included, and what material would be presented in only a cursory

manner. We leave it up to you, the reader, to decide if our choices have been wise ones.

The formal organization of the book is as follows. Chapter 1, called "Fundamentals of Speech Recognition," provides an overview of the entire field with a discussion of the breadth and depth of the various disciplines that are required for a deep understanding of all aspects of speech recognition. The concept of a task-oriented, speech-recognition system is introduced and it is shown that "base level" speech or sound recognition is only one step in a much larger process where higher-level task information, in the form of syntax, semantics, and pragmatics, can often play a major role. After a formal description of the material to be covered in each of the chapters, we give a brief history of speech recognition research in order to put the material presented in this book in its proper perspective.

Chapter 2, entitled the "The Speech Signal: Production, Perception, and Acoustic-Phonetic Characterization," provides a review of the theory of acoustic-phonetics in which we try to characterize basic speech sounds according to both their linguistic properties and the associated acoustic measurements. We show that although there is a solid basis for the linguistic description of sounds and a good understanding of the associated acoustics of sound production, there is, at best, a tenuous relationship between a given linguistic sound and a repeatable, reliable, measureable set of acoustic parameters. As such a wide variety of approaches to speech recognition have been proposed, including those based on the ideas of acoustic-phonetics, statistical pattern-recognition methods, and artificial intelligence (so-called expert system) ideas. We discuss the relative advantages and disadvantages of each of these approaches and show why, on balance, the pattern-recognition approach has become the method of choice for most modern systems.

In Chapter 3, entitled "Signal Processing and Analysis Methods for Speech Recognition," we discuss the fundamental techniques used to provide the speech features used in all recognition systems. In particular we discuss two well-known and widely used methods of spectrum analysis, namely the filter bank approach and the linear prediction method. We also show how the method of vector quantization can be used to code a spectral vector into one of a fixed number of discrete symbols in order to reduce the computation required in a practical system. Finally we discuss an advanced spectral analysis method that is based on processing within the human auditory system—an ear model. The ultimate goal of such a system is to increase the robustness of the signal representation and make the system relatively insensitive to noise and reverberation, in much the same way as the human ear.

Chapter 4, entitled "Pattern-Comparison Techniques," deals with the fundamental problems of defining speech feature vector patterns (from spoken input), and comparing pairs of feature vector patterns both locally (i.e., at some point in time), and globally (i.e., over the entire pattern) so as to derive a measure of similarity between patterns. To solve this pattern-comparison problem requires three types of algorithms, namely a speech-detection method (which essentially separates the speech signal from the background), a spectral vector comparison method (which compares two individual spectral vectors), and a global pattern comparison method which aligns the two patterns locally in time and compares the aligned patterns over the entire duration of the patterns. It is shown that a key issue is the way in which time alignment between patterns is achieved.

Chapter 5, entitled "Speech Recognition System Design and Implementation Issues," discusses the key issues of training a speech recognizer and adapting the recognizer parameters to different speakers, transmission conditions, and speaking environments. A key concept in most modern systems is that of learning, namely improving recognizer performance over time based on additional training provided by the user of the system. Adaptation methods provide a formalism for such learning.



In Chapter 6, "Theory and Implementation of Hidden Markov Models," we discuss a basic set of statistical modeling techniques for characterizing speech. The collection of methods, popularly called Hidden Markov Models, is a powerful set of tools for providing a statistical model of both the static properties of sounds and the dynamical changes that occur across sounds. Methods for time aligning patterns with models are discussed along with different ways of building the statistical models based on the type of representation, the sound being modeled, the class of talkers, and so forth.

Chapters 7 and 8, entitled "Speech Recognition Based on Connected Word Models" and "Large Vocabulary Continuous Speech Recognition," extend the speech-recognition problem from single word sequences to fluent speech. Modeling techniques based on whole word models are discussed in Chapter 7 where we assume that we are interested in recognizing sequences of digits, alphanumerics, and so forth. For this type of system whole-word models are most reasonable since the vocabulary is typically small and highly constrained. Hence the statistical properties of the word models, in all word contexts, can be learned from a reasonably sized training set. Modeling techniques based on subword units are discussed in Chapter 8 where we assume unlimited size vocabulary. Hence a key issue is what units are used, how context dependent the units should be, how unit models are trained reliably (and robustly to different vocabularies and tasks), and how large vocabulary recognition systems based on such units are efficiently implemented.

Finally, in Chapter 9, entitled "Task-Oriented Applications of Automatic Speech Recognition," we come full circle and return to the concept of a task-oriented system. We discuss the basic principles that make some tasks successful while others fail. By way of example we discuss, in fairly general terms, a couple of task-oriented recognizers and show how they perform in practice.

The material in this book is primarily intended for the practicing engineer, scientist, linguist, programmer, and so forth, who wants to learn more about this fascinating field. We assume a basic knowledge of signal processing and linear system theory as provided in an entry level course in digital signal processing. Although not intended as a formal university course, the material in this book is indeed suitable for a one-semester course at the graduate or high undergraduate level. Within almost every chapter we have provided "exercises" for the student to assess how well he or she understands the material. Solutions to the exercises are provided immediately following the exercise. Hence, for maximum effectiveness, each student must exercise self-discipline to work through an answer before comparing it with the published solution.

In order to truly understand the fundamentals of speech recognition, a person needs hands-on experience with the software, hardware, and platforms. Hence we strongly encourage all serious readers of this book to program the algorithms, implement the systems, and literally build applications. Without such practical experience the words in this book will not come alive for most people.

## ACKNOWLEDGMENTS

Although the authors take full responsibility for the material presented in this book, we owe a great debt to our colleagues, both with AT&T Bell Laboratories and outside, for their many technical contributions which underly the material presented. In particular the authors owe a debt of gratitude to Dr. James L. Flanagan (currently director of the CAIP Institute at Rutgers University) for his roles in guiding and shaping both our careers and the field of speech processing. Without Jim's understanding and inspiration, this book would never have existed.

The number of people who have made substantial contributions to speech recognition are too numerous to mention. However there are three individuals who have had a profound influence on the field and they deserve special mention. The first is Professor Raj Reddy of Carnegie-Mellon University who was essentially the first person to realize the vast potential of speech recognition and has devoted over 25 years as a leader, innovator, and educator in this field. The second individual of note is Dr. Jack Ferguson (retired from Institute for Defense Analyses in Princeton) who is the person most responsible for development of the theory of the Hidden Markov Model as applied to speech recognition. Dr. Ferguson, as editor of the key textbook in this area and lecturer, par excellence, has spread the word on Hidden Markov Models so that this technology has rapidly risen from technical obscurity to become the preeminent method of speech recognition today. Finally, the third individual of note is Dr. Fred Jelinek of IBM, who has led the world's largest speech-recognition research group for almost two decades and has been responsible for a large number of major innovations in large vocabulary speech recognition. These three individuals have played major roles in nurturing the technology and enabling it to reach the state of maturity it has achieved today.

Within Bell Laboratories the authors have drawn freely from the research of our former and current colleagues. We would like to acknowledge the direct support and contributions of the following individuals: Prof. Ronald Schafer (currently at Georgia Tech.), Dr. Steve Levinson, Dr. Bishnu Atal, Dr. Esther Levin, Dr. Tali Tishby (currently at the Hebrew University in Jerusalem), Dr. Oded Ghitza, Jay Wilpon, Dr. Frank Soong, Dr. Mohan Sondhi, Dr. Yariv Ephraim, Dr. Cory Myers (currently at Atlantic Aerospace), Dr. Aaron Rosenberg, Dr. Chin Lee and Dr. Roberto Pieraccini. We thank these colleagues for their research contributions and for their friendship and guidance over the years.

Several individuals provided technical comments on a preliminary version of the manuscript. These comments were invariably insightful and provided valuable feedback on ways to improve the book. We would like to acknowledge and thank the following individuals for their help: Dr. Mohan Sondhi, Dr. Yariv Ephraim, Dr. Esther Levin, Dr. Oded Ghitza, Dr. Roberto Pieraccini, Dr. Chin Lee, Dr. Wu Chou and Dr. Sadaoki Furui (NTT Corporation, JAPAN).

The production of a book is an essentially infinite task which is seemingly an endless one. However all good things do come to an end and this one was no exception. The authors owe a great deal to Ms. Martina Sharp of the Bell Labs Word Processing group, who entered all the text material for the book using the LaTeX system. Tina worked on three



generations of the manuscript with a degree of skill that is essentially unmatched among her peers. It was a pleasure working with Tina and we look forward to future projects with her. Most of the art work for the book was produced by Ms. Danuta Sowinska-Khan of the Bell Labs art department. Danuta was a pleasure to work with and is a professional in her trade. The material she produced always met the highest professional standards and did honor to the manuscript in which it appears. A great deal of assistance in the preparation of drafts of the book, getting figures ready for production, and so forth was provided by Irene Morrongiello. We thank her and wish to express our appreciation for a job well done.

Finally, to the crew at Prentice Hall we owe a debt of gratitude for the professional way the book was handled from inception to final production. Karen Gettman provided the incentive to the authors to actually produce the manuscript and make the book a reality. We thank Lisa Garboski, the production editor, for help in all phases of the book production cycle.

Lawrence R. Rabiner
Biing-Hwang Juang

Chapter 1

# FUNDAMENTALS OF SPEECH RECOGNITION

## 1.1 INTRODUCTION

Automatic recognition of speech by machine has been a goal of research for more than four decades and has inspired such science fiction wonders as the computer HAL in Stanley Kubrick's famous movie *2001—A Space Odyssey* and the robot R2D2 in the George Lucas classic *Star Wars* series of movies. However, in spite of the glamour of designing an intelligent machine that can recognize the spoken word and comprehend its meaning, and in spite of the enormous research efforts spent in trying to create such a machine, we are far from achieving the desired goal of a machine that can understand spoken discourse on any subject by all speakers in all environments. Thus, an important question in this book is, What do we mean by "speech recognition by machine." Another important question is, How can we build a series of bridges that will enable us to advance both our knowledge as well as the capabilities of modern speech-recognition systems so that the "holy grail" of conversational speech recognition and understanding by machine is attained?

Because we do not know how to solve the ultimate challenge of speech recognition, our goal in this book is to give a series of presentations on the fundamental principles of most modern, successful speech-recognition systems so as to provide a framework from which researchers can expand the frontier. We will attempt to avoid making absolute judgments on the relative merits of various approaches to particular speech-recognition problems. Instead we will provide the theoretical background and justification for each topic discussed so that the reader is able to understand why the techniques have proved

valuable and how they can be used to advantage in practical situations.

One of the most difficult aspects of performing research in speech recognition by machine is its interdisciplinary nature, and the tendency of most researchers to apply a monolithic approach to individual problems. Consider the disciplines that have been applied to one or more speech-recognition problems:

1. **signal processing**—the process of extracting relevant information from the speech signal in an efficient, robust manner. Included in signal processing is the form of spectral analysis used to characterize the time-varying properties of the speech signal as well as various types of signal preprocessing (and postprocessing) to make the speech signal robust to the recording environment (signal enhancement).
2. **physics (acoustics)**—the science of understanding the relationship between the physical speech signal and the physiological mechanisms (the human vocal tract mechanism) that produced the speech and with which the speech is perceived (the human hearing mechanism).
3. **pattern recognition**—the set of algorithms used to cluster data to create one or more prototypical patterns of a data ensemble, and to match (compare) a pair of patterns on the basis of feature measurements of the patterns.
4. **communication and information theory**—the procedures for estimating parameters of statistical models; the methods for detecting the presence of particular speech patterns, the set of modern coding and decoding algorithms (including dynamic programming, stack algorithms, and Viterbi decoding) used to search a large but finite grid for a best path corresponding to a "best" recognized sequence of words.
5. **linguistics**—the relationships between sounds (phonology), words in a language (syntax), meaning of spoken words (semantics), and sense derived from meaning (pragmatics). Included within this discipline are the methodology of grammar and language parsing.
6. **physiology**—understanding of the higher-order mechanisms within the human central nervous system that account for speech production and perception in human beings. Many modern techniques try to embed this type of knowledge within the framework of artificial neural networks (which depend heavily on several of the above disciplines).
7. **computer science**—the study of efficient algorithms for implementing, in software or hardware, the various methods used in a practical speech-recognition system.
8. **psychology**—the science of understanding the factors that enable a technology to be used by human beings in practical tasks.

Successful speech-recognition systems require knowledge and expertise from a wide range of disciplines, a range far larger than any single person can possess. Therefore, it is especially important for a researcher to have a good understanding of the fundamentals of speech recognition (so that a range of techniques can be applied to a variety of problems), without necessarily having to be an expert in each aspect of the problem. It is the purpose of this book to provide this expertise by giving in-depth discussions of a number of



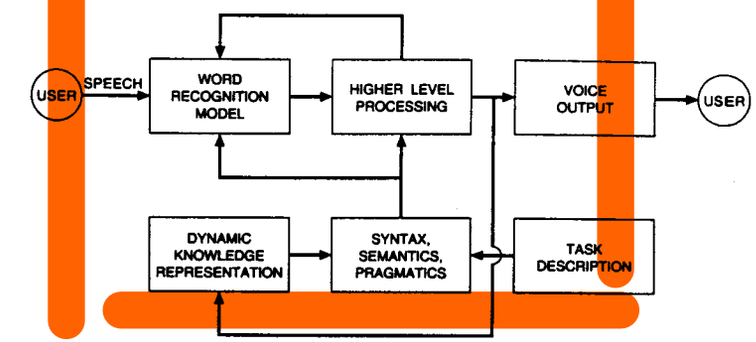



**Figure 1.1** General block diagram of a task-oriented speech-recognition system.

fundamental topics in speech-recognition research.

## 1.2 THE PARADIGM FOR SPEECH RECOGNITION

A general model for speech recognition, shown in Figure 1.1, is used throughout this book. The model begins with a user creating a speech signal (speaking) to accomplish a given task. The spoken output is first recognized in that the speech signal is decoded into a series of words that are meaningful according to the syntax, semantics, and pragmatics of the recognition task. The meaning of the recognized words is obtained by a higher-level processor that uses a dynamic knowledge representation to modify the syntax, semantics, and pragmatics according to the context of what it has previously recognized. In this manner, things such as non sequitors are omitted from consideration at the risk of misunderstanding, but at the gain of minimizing errors for sequentially meaningful inputs. The feedback from the higher-level processing box reduces the complexity of the recognition model by limiting the search for valid (acceptable) input sentences (speech) from the user. The recognition system responds to the user in the form of a voice output, or equivalently, in the form of the requested action being performed, with the user being prompted for more input.

## 1.3 OUTLINE

The material in this book is organized into nine chapters. Chapters 2 through 9 each deals with a basic concept or a fundamental technique used in various speech-recognition systems. The material discussed in these chapters is as follows.

**Chapter 2—The Speech Signal: Production, Perception, and Acoustic-Phonetic Characterization.** In this chapter we review the speech production/perception process in human beings. We show how different speech sounds can be characterized by a set of

spectral and temporal properties that depend on the acoustic-phonetic features of the sound and are manifest in the waveform, the sound spectrogram, or both. Included in the chapter is an overview of the three most common approaches to speech recognition, namely the acoustic-phonetic approach (which tries to directly exploit individual sound properties), the pattern recognition approach (which relies only on gross spectral and temporal properties of speech sounds and uses conventional as well as neural network pattern recognition technology to classify sounds), and the artificial intelligence (AI) approach in which an expert system or a self-organizing (learning) system, as implemented by neural networks, is used to classify sounds. We discuss the strengths and weaknesses of each approach and explain why the pattern-recognition approach is the one most heavily relied on in practical systems. We conclude the chapter with a discussion of the fundamental issues in speech recognition (i.e., those factors that most influence overall system performance), and with a brief overview of current applications.

**Chapter 3—Signal Processing and Analysis Methods for Speech Recognition.** In this chapter we present the two fundamental signal-processing approaches to speech spectral analysis: filter bank and linear predictive methods. We specialize the presentation of these two fundamental techniques to aspects related to speech analysis and compare and contrast the two methods in terms of robustness to speech sounds and required computation. For completeness we also discuss the popular source-coding technique referred to as vector quantization (VQ). Here, a codebook is created to represent the anticipated range of spectral vectors. This enables us to code an arbitrary continuous speech spectral vector into one of a fixed number of discrete codebook symbols at the cost of increased error in signal representation but with the benefit of significantly reduced computation in the recognition process. We conclude this chapter with a discussion of a spectral analysis model that attempts to mimic the processing in the human auditory system—the so-called ear model. Although our knowledge of the higher-order processing in the central nervous system is rudimentary, the importance of ear models is related to their robustness to noise, reverberation, and other environmental factors that often seriously degrade performance of current speech recognizers.

**Chapter 4—Pattern-Comparison Techniques.** In this chapter we discuss three fundamental aspects of comparing a pair of speech patterns. These are the basic concept of detecting speech (i.e., finding the speech signal in a background of noise or other acoustic interference), the idea of computing a measure of the local distance (or similarity) of a pair of spectral representations of a short-time piece of speech signal (a distance or distortion measure), and the concept of temporally aligning and globally comparing a pair of speech patterns corresponding to different speech utterances that may or may not be the same sequence of sounds or words (dynamic time warping algorithms). We show in this chapter how the basic pattern-comparison techniques can be combined in a uniform framework for speech-recognition applications.

**Chapter 5—Speech-Recognition System Design and Implementation Issues.** In this chapter we discuss the remaining pieces (after signal processing and pattern comparison) that enable us to build and study performance of a practical speech-recognition system.



In particular we discuss how speech recognizers are trained and how we can enhance the basic recognition procedure by adding features, by exploiting a preprocessor, by the use of methods of adaptation or by postprocessing the recognizer outputs using a set of pattern discriminators (as opposed to the pattern classifiers used in a conventional implementation). We conclude the chapter with a discussion of various ways of recognizing speech in adverse environments (e.g., noise, stress conditions, or mismatch between training and testing).

**Chapter 6—Theory and Implementation of Hidden Markov Models.** In this chapter we discuss aspects of the theory and implementation of the set of statistical modeling techniques collectively referred to as hidden Markov modeling. Included within these techniques are the algorithms for scoring a statistical (Markovian) model against a speech pattern, the techniques for aligning the model with the speech pattern so as to recover an estimate of the alignment path between different speech sounds and different model states, and the techniques for estimating parameters of the statistical models from a training set of utterances of the sounds being modeled. Also included is a discussion of the practical aspects of building hidden Markov models, including the issues of scaling of data, handling of multiple observation sequences, providing initial estimates of model parameters, and combating the problems of insufficient training data. We conclude the chapter with a practical example illustrating how a simple, isolated word recognizer would be implemented using hidden Markov models.

**Chapter 7—Speech Recognition Based on Connected Word Models.** In this chapter we show how the basic set of techniques developed for recognizing an isolated word or phrase can be readily extended to recognizing a sequence of words (e.g., a string of digits of a credit card number) spoken in a fluent or connected manner. We make the simplifying assumption that the connected word string is recognized by finding the optimal sequence of word models that best matches the spoken string. Hence we assume that the word is the basic recognition unit for these systems, and therefore the training problem is one of estimating the optimal parameters of word models on the basis of training data, which need not contain isolated versions of the words. We describe three "optimal" approaches to solving the recognition part of connected word-recognition problems: (1) the two-level dynamic programming method, (2) the level building method, and (3) the time synchronous level building (or the one-pass) method and discuss the properties, and the relative strengths and weaknesses of each method. We then show how we can optimally train connected word systems, even if isolated versions of the vocabulary words are not available. We conclude the chapter with a discussion of a connected digit recognizer implemented using the methods described in the chapter.

**Chapter 8—Large Vocabulary Continuous Speech Recognition.** In this chapter we discuss the issues in applying speech-recognition technology to the problem of recognizing fluently spoken speech with vocabulary sizes of 1000 or more words (with unlimited vocabularies as the ultimate goal). It is shown that a number of fundamental problems must be solved to implement such a system, including the choice of a basic subword speech unit (from which words, phrases, and sentences can be built up), an effective way

of modeling the basic speech unit, a way of deriving models of the unit, a way of designing and implementing a word lexicon (which provides a mapping between words and subword units), a way of implementing task syntax (the system grammar), a way of implementing the overall recognition part of the system (via some type of network search), and a way of imposing task semantics onto the solution. We concentrate primarily on the issues involved in building large vocabulary recognition systems. For illustrative purposes we describe one reasonable way of building such a system and discuss the resulting performance on a standard database management task.

**Chapter 9—Task Oriented Applications of Automatic Speech Recognition.** The final chapter of the book provides a brief overview of how one might apply the ideas discussed in the book to building a real, task-oriented, speech recognition system. It includes discussions of how one would evaluate recognizer performance and how one might decide whether a proposed task is viable for speech recognition. We also discuss a set of broad classes of applications, which appear to be the most promising ones at this time, along with typical examples of how recognizers have been successfully employed within these broad classes. The chapter concludes with some broad performance projections through the year 2000.

## 1.4 A BRIEF HISTORY OF SPEECH-RECOGNITION RESEARCH

Research in automatic speech recognition by machine has been done for almost four decades. To gain an appreciation for the amount of progress achieved over this period, it is worthwhile to briefly review some research highlights. The reader is cautioned that such a review is cursory, at best, and must therefore suffer from errors of judgment as well as omission.

The earliest attempts to devise systems for automatic speech recognition by machine were made in the 1950s, when various researchers tried to exploit the fundamental ideas of acoustic-phonetics. In 1952, at Bell Laboratories, Davis, Biddulph, and Balashek built a system for isolated digit recognition for a single speaker [1]. The system relied heavily on measuring spectral resonances during the vowel region of each digit. In an independent effort at RCA Laboratories in 1956, Olson and Belar tried to recognize 10 distinct syllables of a single talker, as embodied in 10 monosyllabic words [2]. The system again relied on spectral measurements (as provided by an analog filter bank) primarily during vowel regions. In 1959, at University College in England, Fry and Denes tried to build a phoneme recognizer to recognize four vowels and nine consonants [3]. They used a spectrum analyzer and a pattern matcher to make the recognition decision. A novel aspect of this research was the use of statistical information about allowable sequences of phonemes in English (a rudimentary form of language syntax) to improve overall phoneme accuracy for words consisting of two or more phonemes. Another effort of note in this period was the vowel recognizer of Forgie and Forgie, constructed at MIT Lincoln Laboratories in 1959, in which 10 vowels embedded in a /b/-vowel-/t/ format were recognized in a speaker-independent manner [4]. Again a filter bank analyzer was used to provide spectral information, and a



time varying estimate of the vocal tract resonances was made to decide which vowel was spoken.

In the 1960s several fundamental ideas in speech recognition surfaced and were published. However, the decade started with several Japanese laboratories entering the recognition arena and building special-purpose hardware as part of their systems. One early Japanese system, described by Suzuki and Nakata of the Radio Research Lab in Tokyo [5], was a hardware vowel recognizer. An elaborate filter bank spectrum analyzer was used along with logic that connected the outputs of each channel of the spectrum analyzer (in a weighted manner) to a vowel-decision circuit, and a majority decision logic scheme was used to choose the spoken vowel. Another hardware effort in Japan was the work of Sakai and Doshita of Kyoto University in 1962, who built a hardware phoneme recognizer [6]. A hardware speech segmenter was used along with a zero-crossing analysis of different regions of the spoken input to provide the recognition output. A third Japanese effort was the digit recognizer hardware of Nagata and coworkers at NEC Laboratories in 1963 [7]. This effort was perhaps most notable as the initial attempt at speech recognition at NEC and led to a long and highly productive research program.

In the 1960s three key research projects were initiated that have had major implications on the research and development of speech recognition for the past 20 years. The first of these projects was the efforts of Martin and his colleagues at RCA Laboratories, beginning in the late 1960s, to develop realistic solutions to the problems associated with nonuniformity of time scales in speech events. Martin developed a set cf elementary time-normalization methods, based on the ability to reliably detect speech starts and ends, that significantly reduced the variability of the recognition scores [8]. Martin ultimately developed the method and founded one of the first companies, Threshold Technology, which built, marketed, and sold speech-recognition products. At about the same time, in the Soviet Union, Vintsyuk proposed the use of dynamic programming methods for time aligning a pair of speech utterances [9]. Although the essence of the concepts of dynamic time warping, as well as rudimentary versions of the algorithms for connected word recognition, were embodied in Vintsyuk's work, it was largely unknown in the West and did not come to light until the early 1980s; this was long after the more formal methods were proposed and implemented by others.

A final achievement of note in the 1960s was the pioneering research of Reddy in the field of continuous speech recognition by dynamic tracking of phonemes [10]. Reddy's research eventually spawned a long and highly successful speech-recognition research program at Carnegie Mellon University (to which Reddy moved in the late 1960s) which, to this day, remains a world leader in continuous-speech-recognition systems.

In the 1970s speech-recognition research achieved a number of significant milestones. First the area of isolated word or discrete utterance recognition became a viable and usable technology based on fundamental studies by Velichko and Zagoruyko in Russia [11], Sakoe and Chiba in Japan [12], and Itakura in the United States [13]. The Russian studies helped advance the use of pattern-recognition ideas in speech recognition; the Japanese research showed how dynamic programming methods could be successfully applied; and Itakura's research showed how the ideas of linear predictive coding (LPC), which had already been successfully used in low-bit-rate speech coding, could be extended to speech-

recognition systems through the use of an appropriate distance measure based on LPC spectral parameters.

Another milestone of the 1970s was the beginning of a longstanding, highly successful group effort in large vocabulary speech recognition at IBM in which researchers studied three distinct tasks over a period of almost two decades, namely the New Raleigh language [14] for simple database queries, the laser patent text language [15] for transcribing laser patents, and the office correspondence task, called Tangora [16], for dictation of simple memos.

Finally, at AT&T Bell Labs, researchers began a series of experiments aimed at making speech-recognition systems that were truly speaker independent [17]. To achieve this goal a wide range of sophisticated clustering algorithms were used to determine the number of distinct patterns required to represent all variations of different words across a wide user population. This research has been refined over a decade so that the techniques for creating speaker-independent patterns are now well understood and widely used.

Just as isolated word recognition was a key focus of research in the 1970s, the problem of connected word recognition was a focus of research in the 1980s. Here the goal was to create a robust system capable of recognizing a fluently spoken string of words (e.g., digits) based on matching a concatenated pattern of individual words. A wide variety of connected word-recognition algorithms were formulated and implemented, including the two-level dynamic programming approach of Sakoe at Nippon Electric Corporation (NEC) [18], the one-pass method of Bridle and Brown at Joint Speech Research Unit (JSRU) in England [19], the level building approach of Myers and Rabiner at Bell Labs [20], and the frame synchronous level building approach of Lee and Rabiner at Bell Labs [21]. Each of these "optimal" matching procedures had its own implementational advantages, which were exploited for a wide range of tasks.

Speech research in the 1980s was characterized by a shift in technology from template-based approaches to statistical modeling methods—especially the hidden Markov model approach [22, 23]. Although the methodology of hidden Markov modeling (HMM) was well known and understood in a few laboratories (primarily IBM, Institute for Defense Analyses (IDA), and Dragon Systems), it was not until widespread publication of the methods and theory of HMMs, in the mid-1980s, that the technique became widely applied in virtually every speech-recognition research laboratory in the world.

Another "new" technology that was reintroduced in the late 1980s was the idea of applying neural networks to problems in speech recognition. Neural networks were first introduced in the 1950s, but they did not prove useful initially because they had many practical problems. In the 1980s, however, a deeper understanding of the strengths and limitations of the technology was obtained, as well as the relationships of the technology to classical signal classification methods. Several new ways of implementing systems were also proposed [24, 25].

Finally, the 1980s was a decade in which a major impetus was given to large vocabulary, continuous-speech-recognition systems by the Defense Advanced Research Projects Agency (DARPA) community, which sponsored a large research program aimed at achieving high word accuracy for a 1000-word, continuous-speech-recognition, database management task. Major research contributions resulted from efforts at CMU (notably the well-known SPHINX system) [26], BBN with the BYBLOS system [27], Lincoln Labs [28], SRI [29], MIT [30], and AT&T Bell Labs [31]. The DARPA program has continued into the 1990s, with emphasis shifting to natural language front ends to the recognizer, and the task shifting to retrieval of air travel information. At the same time, speech-recognition technology has been increasingly used within telephone networks to automate as well as enhance operator services.



## REFERENCES


[1] K. H. Davis, R. Biddulph, and S. Balashek, "Automatic Recognition of Spoken Digits," *J. Acoust. Soc. Am.*, 24 (6): 637–642, 1952.

[2] H. F. Olson and H. Belar, "Phonetic Typewriter," *J. Acoust. Soc. Am.*, 28 (6): 1072–1081, 1956.

[3] D. B. Fry, "Theoretical Aspects of Mechanical Speech Recognition"; and P. Denes, "The Design and Operation of the Mechanical Speech Recognizer at University College London," *J. British Inst. Radio Engr.*, 19: 4, 211–229, 1959.

[4] J. W. Forgie and C. D. Forgie, "Results Obtained From a Vowel Recognition Computer Program," *J. Acoust. Soc. Am.*, 31 (11): 1480–1489, 1959.

[5] J. Suzuki and K. Nakata, "Recognition of Japanese Vowels—Preliminary to the Recognition of Speech," *J. Radio Res. Lab*, 37 (8): 193–212, 1961.

[6] T. Sakai and S. Doshita, "The Phonetic Typewriter, Information Processing 1962," *Proc. IFIP Congress*, Munich, 1962.

[7] K. Nagata, Y. Kato, and S. Chiba, "Spoken Digit Recognizer for Japanese Language," *NEC Res. Develop.*, No. 6, 1963.

[8] T. B. Martin, A. L. Nelson, and H. J. Zadell, "Speech Recognition by Feature Abstraction Techniques," Tech. Report AL-TDR-64-176, Air Force Avionics Lab, 1964.

[9] T. K. Vintsyuk, "Speech Discrimination by Dynamic Programming," *Kibernetika*, 4 (2): 81–88, Jan.–Feb. 1968.

[10] D. R. Reddy, "An Approach to Computer Speech Recognition by Direct Analysis of the Speech Wave," Tech. Report No. C549, Computer Science Dept., Stanford Univ., September 1966.

[11] V. M. Velichko and N. G. Zagoruyko, "Automatic Recognition of 200 Words," *Int. J. Man-Machine Studies*, 2: 223, June 1970.

[12] H. Sakoe and S. Chiba, "Dynamic Programming Algorithm Optimization for Spoken Word Recognition," *IEEE Trans. Acoustics, Speech, Signal Proc.*, ASSP-26 (1): 43-49, February 1978.

[13] F. Itakura, "Minimum Prediction Residual Applied to Speech Recognition," *IEEE Trans. Acoustics, Speech, Signal Proc.*, ASSP-23 (1): 67-72, February 1975.

[14] C. C. Tappert, N. R. Dixon, A. S. Rabinowitz, and W. D. Chapman, "Automatic Recognition of Continuous Speech Utilizing Dynamic Segmentation, Dual Classification, Sequential Decoding and Error Recovery," Rome Air Dev. Cen, Rome, NY, Tech. Report TR-71-146, 1971.

[15] F. Jelinek, L. R. Bahl, and R. L. Mercer, "Design of a Linguistic Statistical Decoder for the

Recognition of Continuous Speech," *IEEE Trans. Information Theory*, IT-21: 250–256, 1975.

[16] F. Jelinek, "The Development of an Experimental Discrete Dictation Recognizer," *Proc. IEEE*, 73 (11): 1616–1624, 1985.

[17] L. R. Rabiner, S. E. Levinson, A. E. Rosenberg, and J. G. Wilpon, "Speaker Independent Recognition of Isolated Words Using Clustering Techniques," *IEEE Trans. Acoustics, Speech, Signal Proc.*, ASSP-27: 336–349, August 1979.

[18] H. Sakoe, "Two Level DP Matching—A Dynamic Programming Based Pattern Matching Algorithm for Connected Word Recognition," *IEEE Trans. Acoustics, Speech, Signal Proc.*, ASSP-27: 588–595, December 1979.

[19] J. S. Bridle and M. D. Brown, "Connected Word Recognition Using Whole Word Templates," *Proc. Inst. Acoust. Autumn Conf.*, 25–28, November 1979.

[20] C. S. Myers and L. R. Rabiner, "A Level Building Dynamic Time Warping Algorithm for Connected Word Recognition," *IEEE Trans. Acoustics, Speech, Signal Proc.*, ASSP-29: 284–297, April 1981.

[21] C. H. Lee and L. R. Rabiner, "A Frame Synchronous Network Search Algorithm for Connected Word Recognition," *IEEE Trans. Acoustics, Speech, Signal Proc.*, 37 (11): 1649–1658, November 1989.

[22] J. Ferguson, Ed., *Hidden Markov Models for Speech*, IDA, Princeton, NJ, 1980.

[23] L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proc. IEEE*, 77 (2): 257–286, February 1989.

[24] R. P. Lippmann, "An Introduction to Computing with Neural Nets," *IEEE ASSP Mag.*, 4 (2): 4–22, April 1987.

[25] A. Weibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang, "Phoneme Recognition Using Time-Delay Neural Networks," *IEEE Trans. Acoustics, Speech, Signal Proc.*, 37: 393–404, 1989.

[26] K. F. Lee, H. W. Hon, and D. R. Reddy, "An Overview of the SPHINX Speech Recognition System," *IEEE Trans. Acoustics, Speech, Signal Proc.*, 38: 600–610, 1990.

[27] Y. L. Chow, M. O. Dunham, O. A. Kimball, M. A. Krasner, G. F. Kubala, J. Makhoul, S. Roucos, and R. M. Schwartz, "BBYLOS: The BBN Continuous Speech Recognition System," *Proc. ICASSP 87*, 89–92, April 1987.

[28] D. B. Paul, "The Lincoln Robust Continuous Speech Recognizer," *Proc. ICASSP 89*, Glasgow, Scotland, 449–452, May 1989.

[29] M. Weintraub et al., "Linguistic Constraints in Hidden Markov Model Based Speech Recognition," *Proc. ICASSP 89*, Glasgow, Scotland, 699–702, May 1989.

[30] V. Zue, J. Glass, M. Phillips, and S. Seneff, "The MIT Summit Speech Recognition System: A Progress Report," *Proc. DARPA Speech and Natural Language Workshop*, 179–189, February 1989.

[31] C. H. Lee, L. R. Rabiner, R. Pieraccini, and J. G. Wilpon, "Acoustic Modeling for Large Vocabulary Speech Recognition," *Computer Speech and Language*, 4: 127–165, 1990.

Chapter 2

# THE SPEECH SIGNAL: PRODUCTION, PERCEPTION, AND ACOUSTIC-PHONETIC CHARACTERIZATION

## 2.1 INTRODUCTION

In this chapter we discuss the mechanics of producing and perceiving speech in human beings, and we show how an understanding of these processes leads naturally to several different approaches to speech recognition by machine. We begin by showing how the different classes of speech sounds, or phonetics, can each be characterized in terms of broad acoustic features whose properties are relatively invariant across words and speakers. The ideas of acoustic-phonetic characterization of sounds lead naturally to straightforward implementation of a speech-recognition algorithm based on sequential detection of sounds and sound classes. The strengths and weaknesses of such an approach are discussed. An alternative approach to speech recognition is to use standard pattern-recognition techniques in a framework in which all speech knowledge is "learned" via a training phase. We show that such a "blind" approach has some natural advantages for a wide range of speech-recognition systems. Finally we show how aspects of both the acoustic-phonetic approach and the pattern-recognition approach can be integrated into a hybrid method that includes techniques from artificial intelligence as well as neural network methods.

### 2.1.1 The Process of Speech Production and Perception in Human Beings

Figure 2.1 shows a schematic diagram of the speech-production/speech-perception process in human beings. The production (speech-generation) process begins when the talker

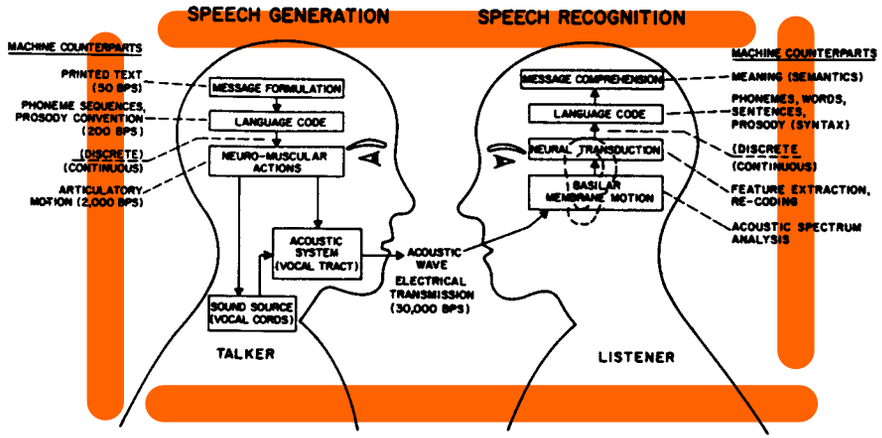



**Figure 2.1** Schematic diagram of speech-production/speech-perception process (after Flanagan [unpublished]).

formulates a message (in his mind) that he wants to transmit to the listener via speech. The machine counterpart to the process of message formulation is the creation of printed text expressing the words of the message. The next step in the process is the conversion of the message into a language code. This roughly corresponds to converting the printed text of the message into a set of phoneme sequences corresponding to the sounds that make up the words, along with prosody markers denoting duration of sounds, loudness of sounds, and pitch accent associated with the sounds. Once the language code is chosen, the talker must execute a series of neuromuscular commands to cause the vocal cords to vibrate when appropriate and to shape the vocal tract such that the proper sequence of speech sounds is created and spoken by the talker, thereby producing an acoustic signal as the final output. The neuromuscular commands must simultaneously control all aspects of articulatory motion including control of the lips, jaw, tongue, and velum (a "trapdoor" controlling the acoustic flow to the nasal mechanism).

Once the speech signal is generated and propagated to the listener, the speech-perception (or speech-recognition) process begins. First the listener processes the acoustic signal along the basilar membrane in the inner ear, which provides a running spectrum analysis of the incoming signal. A neural transduction process converts the spectral signal at the output of the basilar membrane into activity signals on the auditory nerve, corresponding roughly to a feature extraction process. In a manner that is not well understood, the neural activity along the auditory nerve is converted into a language code at the higher centers of processing within the brain, and finally message comprehension (understanding of meaning) is achieved.

A slightly different view of the speech-production/speech-perception process is shown in Figure 2.2. Here we see the steps in the process laid out along a line corresponding to the basic information rate of the signal (or control) at various stages of the



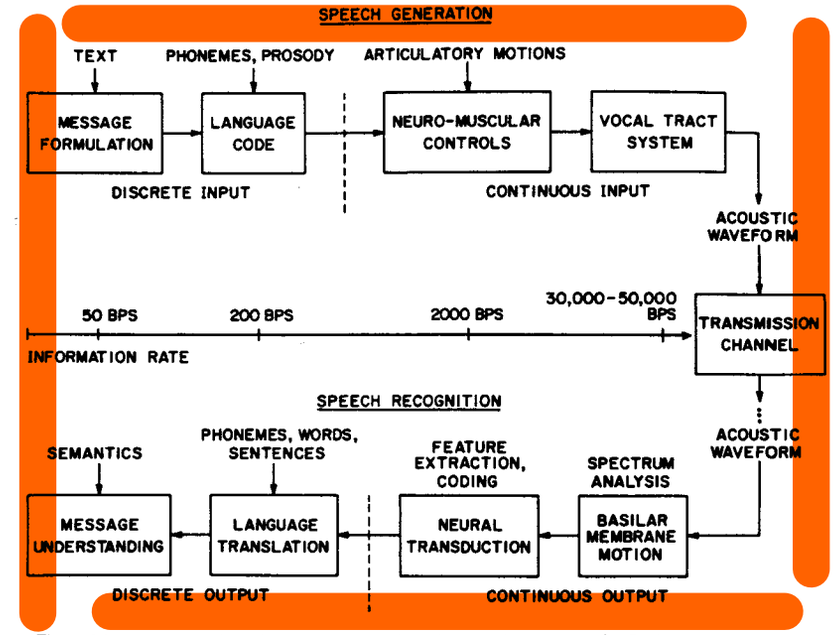



**Figure 2.2** Alternative view of speech-production/speech-perception process (after Rabiner and Levinson [1]).

process. The discrete symbol information rate in the raw message text is rather low (about 50 bps [bits per second] corresponding to about 8 sounds per second, where each sound is one of about 50 distinct symbols). After the language code conversion, with the inclusion of prosody information, the information rate rises to about 200 bps. Somewhere in the next stage the representation of the information in the signal (or the control) becomes continuous with an equivalent rate of about 2000 bps at the neuromuscular control level, and about 30,000–50,000 bps at the acoustic signal level.

A transmission channel is shown in Figure 2.2 [1], indicating that any of several well-known coding techniques could be used to transmit the acoustic waveform from the talker to the listener. The steps in the speech-perception mechanism can also be interpreted in terms of information rate in the signal or its control and follows the inverse pattern of the production process. Thus the continuous information rate at the basilar membrane is in the range of 30,000–50,000 bps, while at the neural transduction stage it is about 2000 bps. The higher-level processing within the brain converts the neural signals to a discrete representation, which ultimately is decoded into a low-bit-rate message.

To illustrate, in a trivial way, how the speech-production/speech-perception process works, consider that the speaker has a goal of finding out whether his office mate has eaten his lunch yet. To express this thought, the speaker formulates the message "Did you eat

yet?" In the process of converting the message to a language code, the text becomes a phonetic sequence of sounds of the form /dId yu it yɛt?/, in which each word is expressed as a sequence of phonemes constituting the ideal pronunciation of the sounds of the word (as spoken in isolation) within the spoken language. However, because the words are not spoken in isolation, and a physical mechanism is used to produce the sounds (the human vocal tract system), and because physical systems obey continuity and smoothness constraints, by the time the message is spoken the sounds become more like the phonetic string /dI jə it jɛt?/. The final d in dId is dropped, the word *you* becomes converted to a word that sounds a lot like "juh," and finally the word *yet* is pronounced as "jet." Remarkably, through the speech-perception process, human beings are usually able to decode this highly stylized version of the text into the correct string; sadly, however, this remains a most difficult task for almost all speech-recognition machines.

## 2.2 THE SPEECH-PRODUCTION PROCESS

Figure 2.3 shows a mid-sagittal plane (longitudinal cross-section) X-ray of the human vocal apparatus [2]. The *vocal tract*, outlined by the dotted lines in Figure 2.3, begins at the opening of the vocal cords, or *glottis*, and ends at the lips. The vocal tract consists of the *pharynx* (the connection from the esophagus to the mouth) and the mouth, or *oral cavity*. In the average male, the total length of the vocal tract is about 17 cm. The cross-sectional area of the vocal tract, determined by the positions of the tongue, lips, jaw, and velum, varies from zero (complete closure) to about 20 $cm^2$. The *nasal tract* begins at the velum and ends at the nostrils. When the *velum*, (a trapdoor-like mechanism at the back of the mouth cavity) is lowered, the nasal tract is acoustically coupled to the vocal tract to produce the nasal sounds of speech.

A schematic diagram of the human vocal mechanism is shown in Figure 2.4 [3]. Air enters the lungs via the normal breathing mechanism. As air is expelled from the lungs through the *trachea* (or windpipe), the tensed vocal cords within the *larynx* are caused to vibrate (in the mode of a relaxation oscillator) by the air flow. The air flow is chopped into quasi-periodic pulses which are then modulated in frequency in passing through the *pharynx* (the throat cavity), the mouth cavity, and possibly the nasal cavity. Depending on the positions of the various articulators (i.e., jaw, tongue, velum, lips, mouth), different sounds are produced.

Figure 2.5 shows plots of the glottal air flow (volume velocity waveform) and the resulting sound pressure at the mouth for a typical vowel sound [4]. The glottal waveform shows a gradual build-up to a quasi-periodic pulse train of air, taking about 15 msec to reach steady state. This build-up is also reflected in the acoustic waveform shown at the bottom of the figure.

A simplified representation of the complete physiological mechanism for creating speech is shown in Figure 2.6 [3]. The lungs and the associated muscles act as the source of air for exciting the vocal mechanism. The muscle force pushes air out of the lungs (shown schematically as a piston pushing up within a cylinder) and through the bronchi and trachea. When the vocal cords are tensed, the air flow causes them to vibrate, producing



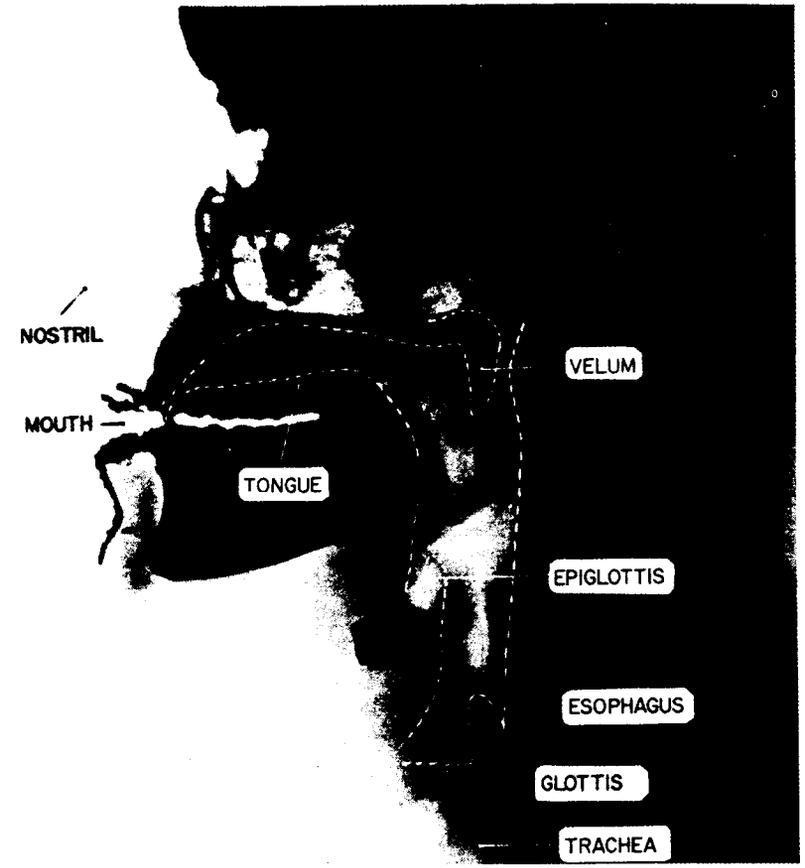



**Figure 2.3** Mid-sagittal plane X-ray of the human vocal apparatus (after Flanagan et al. [2]).

so-called voiced speech sounds. When the vocal cords are relaxed, in order to produce a sound, the air flow either must pass through a constriction in the vocal tract and thereby become turbulent, producing so-called unvoiced sounds, or it can build up pressure behind a point of total closure within the vocal tract, and when the closure is opened, the pressure is suddenly and abruptly released, causing a brief transient sound.

Speech is produced as a sequence of sounds. Hence the state of the vocal cords, as well as the positions, shapes, and sizes of the various articulators, changes over time to reflect the sound being produced. The manner in which different sounds are created will be described later in this chapter. First we divert to a brief discussion of the speech waveform

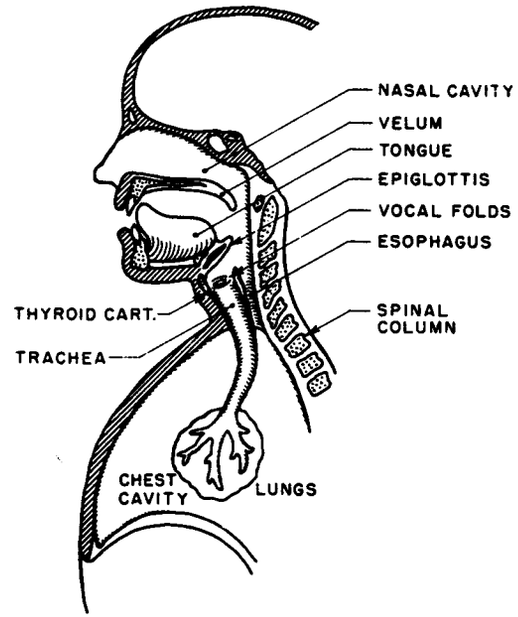



**Figure 2.4** Schematic view of the human vocal mechanism (after Flanagan [3]).

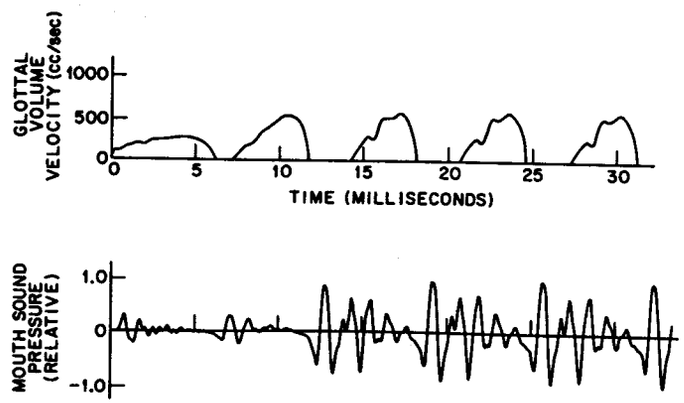



**Figure 2.5** Glottal volume velocity and resulting sound pressure at the start of a voiced sound (after Ishizaka and Flanagan [4]).



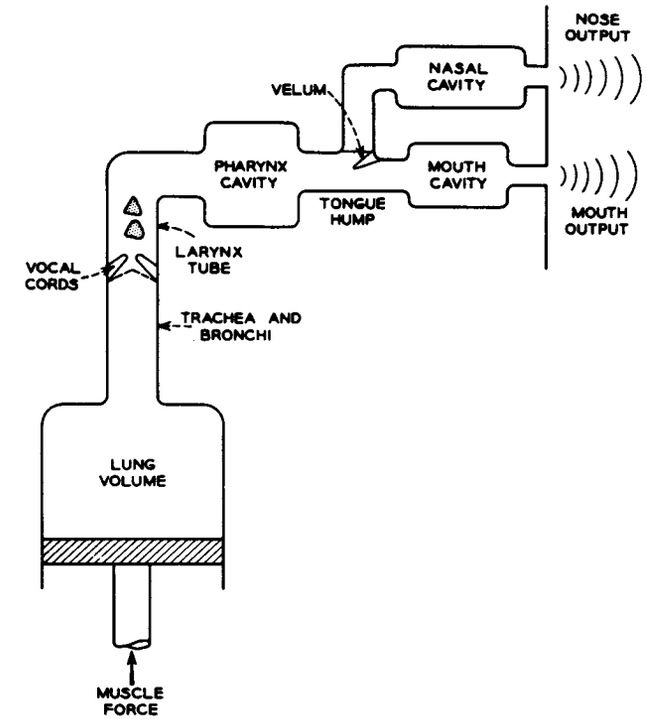



**Figure 2.6** Schematic representation of the complete physiological mechanism of speech production (after Flanagan [3]).

and its spectral representation.

## 2.3 REPRESENTING SPEECH IN THE TIME AND FREQUENCY DOMAINS

The speech signal is a slowly time varying signal in the sense that, when examined over a sufficiently short period of time (between 5 and 100 msec), its characteristics are fairly stationary; however, over long periods of time (on the order of 1/5 seconds or more) the signal characteristics change to reflect the different speech sounds being spoken. An illustration of this effect is given in Figure 2.7, which shows the time waveform corresponding to the initial sounds in the phrase, "It's time . . ." as spoken by a male speaker. Each line of the waveform corresponds to 100 msec (1/10 second) of signal; hence the entire plot encompasses about 0.5 sec.

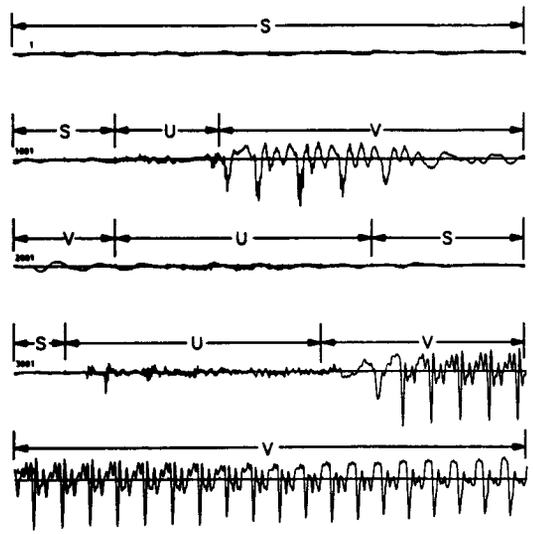



**Figure 2.7** Waveform plot of the beginning of the utterance "It's time."

The slowly time varying nature of the signal can be seen by contrasting the first 100 msec of the waveform (the first line), which corresponds to background silence and is therefore low in amplitude, to the next 100 msec of the waveform (the second line), which first shows a small increase in level, and then a sharp increase in level and a gross change in waveform shape and regularity (it becomes almost periodic).

There are several ways of classifying (labeling) events in speech. Perhaps the simplest and most straightforward is via the state of the speech-production source—the vocal cords. It is accepted convention to use a three-state representation in which the states are (1) silence (S), where no speech is produced; (2) unvoiced (U), in which the vocal cords are not vibrating, so the resulting speech waveform is aperiodic or random in nature; and (3) voiced (V), in which the vocal cords are tensed and therefore vibrate periodically when air flows from the lungs, so the resulting speech waveform is quasi-periodic. The result of applying this type of classification to the waveform of Figure 2.7 is shown in the figure. Initially, before speaking begins, the waveform is classified as silence (S). A brief period of unvoiced (U) sound (whisper or aspiration) is seen prior to the voicing (V) corresponding to the initial vowel in the word *It's*. Following the voicing region, there is a brief, unvoiced aspiration (devoicing of the vowel), followed by a silence region (prior to the /t/ in *It's*), and then a relatively long, unvoiced (U) region corresponding to the /t/ release, followed by the /s/, followed by the /t/ in *time*. Finally there is a long voicing (V) region corresponding to the diphthong /a$^y$/ in *time*.

It should be clear that the segmentation of the waveform into well-defined regions of



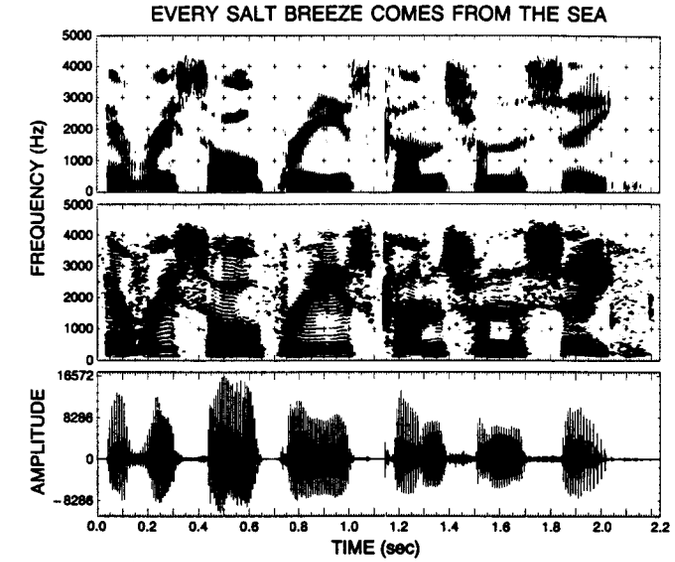



**Figure 2.8** Wideband and narrowband spectrograms and speech amplitude for the utterance "Every salt breeze comes from the sea."

silence, unvoiced, and voiced signals is not exact; it is often difficult to distinguish a weak, unvoiced sound (like /f/ or /th/) from silence, or a weak, voiced sound (like /v/ or /m/) from unvoiced sounds or even silence. However, it is usually not critical to segment the signal to a precision much less than several milliseconds; hence, small errors in boundary locations usually have no consequence for most applications.

An alternative way of characterizing the speech signal and representing the information associated with the sounds is via a spectral representation. Perhaps the most popular representation of this type is the sound spectrogram in which a three-dimensional representation of the speech intensity, in different frequency bands, over time is portrayed. An example of this type of speech representation is given in Figure 2.8, which shows a *wideband spectrogram* in the first panel, a *narrowband spectrogram* in the second panel, and a waveform amplitude plot in the third panel, of a spoken version of the utterance "Every salt breeze comes from the sea" by a male speaker. The wideband spectrogram corresponds to performing a spectral analysis on 15-msec sections of waveform using a broad analysis filter (125 Hz bandwidth) with the analysis advancing in intervals of 1 msec. The spectral intensity at each point in time is indicated by the intensity (darkness) of the plot at a particular analysis frequency. Because of the relatively broad bandwidth of the analysis filters, hence the relatively short duration of the analysis window, the spectral envelope of individual periods of the speech waveform during voiced sections are resolved

and are seen as vertical striations in the spectrogram.

The narrowband spectrogram (shown in the second panel of Figure 2.8) corresponds to performing a spectral analysis on 50-msec sections of waveform using a narrow analysis filter (40 Hz bandwidth), with the analysis again advancing in intervals of 1 msec. Because of the relatively narrow bandwidth of the analysis filters, individual spectral harmonics corresponding to the pitch of the speech waveform, during voiced regions, are resolved and are seen as almost-horizontal lines in the spectrogram. During periods of unvoiced speech, we see primarily high-frequency energy in the spectrograms; during periods of silence we essentially see no spectral activity (because of the reduced signal level).

A third way of representing the time-varying signal characteristics of speech is via a parameterization of the spectral activity based on the model of speech production. Because the human vocal tract is essentially a tube, or concatenation of tubes, of varying cross-sectional area that is excited either at one end (by the vocal cord puffs of air) or at a point along the tube (corresponding to turbulent air at a constriction), acoustic theory tells us that the transfer function of energy from the excitation source to the output can be described in terms of the natural frequencies or resonances of the tube. Such resonances are called *formants* for speech, and they represent the frequencies that pass the most acoustic energy from the source to the output. Typically there are about three resonances of significance, for a human vocal tract, below about 3500 Hz. Figure 2.9 [5] shows a wideband spectrogram, along with the computed formant frequency estimates, for the utterance "Why do I owe you a letter," spoken by a male speaker. There is a good correspondence between the estimated formant frequencies and the points of high spectral energy in the spectrogram. The formant frequency representation is a highly efficient, compact representation of the time-varying characteristics of speech. The major problem, however, is the difficulty of reliably estimating the formant frequencies for low-level voiced sounds, and the difficulty of defining the formants for unvoiced or silence regions. As such, this representation is more of theoretical than of practical interest.

Figures 2.10 and 2.11 show spectral and temporal representations of the phrase "Should we chase," spoken by a male speaker, along with a detailed segmentation of the waveform into individual sounds. The ultimate goal of speech recognition is to uniquely and automatically provide such a segmentation and labeling of speech into constituent sounds or sound groups such as words, then sentences. To understand the limitations on this approach, we will next discuss, in detail, the general sounds of English and the relevant acoustic and phonetic features of the sounds.

## 2.4 SPEECH SOUNDS AND FEATURES

The number of linguistically distinct speech sounds (phonemes) in a language is often a matter of judgment and is not invariant to different linguists. Table 2.1 shows a condensed list of phonetic symbols of American English, their ARPABET representation [6], and an example word in which the sound occurs. Shown in this table are 48 sounds, including 18 vowels or vowel combinations (called diphthongs), 4 vowel-like consonants, 21 standard consonants, 4 syllabic sounds, and a phoneme referred to as a glottal stop (literally a symbol



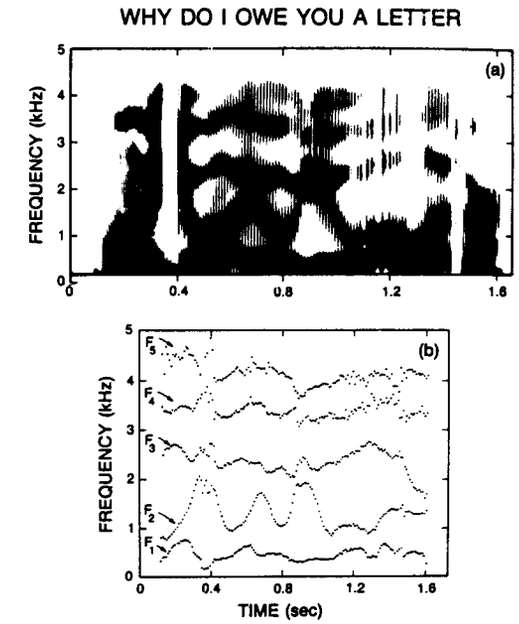



**Figure 2.9** Wideband spectrogram and formant frequency representation of the utterance "Why do I owe you a letter" (after Atal and Hanauer [5]).

for a sound corresponding to a break in voicing within a sound).

Many of the sounds or phonemes shown in Table 2.1 are not considered standard; they represent specialized cases such as the so-called barred I (/ɨ/) in the word *roses*. As such, a more standard representation of the basic sounds and sound classes of American English is shown in Figure 2.12. Here we see the conventional set of 11 vowels, classified as front, mid, or back, corresponding to the position of the tongue hump in producing the vowel; 4 vowel combinations or diphthongs; the 4 semivowels broken down into 2 liquids and 2 glides; the nasal consonants, the voiced and unvoiced stop consonants; the voiced and unvoiced fricatives; whisper; and the affricates. There are a total of 39 of the 48 sounds of Table 2.1 represented in Figure 2.12.

### 2.4.1 The Vowels

The vowel sounds are perhaps the most interesting class of sounds in English. Their importance to the classification and representation of written text is very low; however, most practical speech-recognition systems rely heavily on vowel recognition to achieve

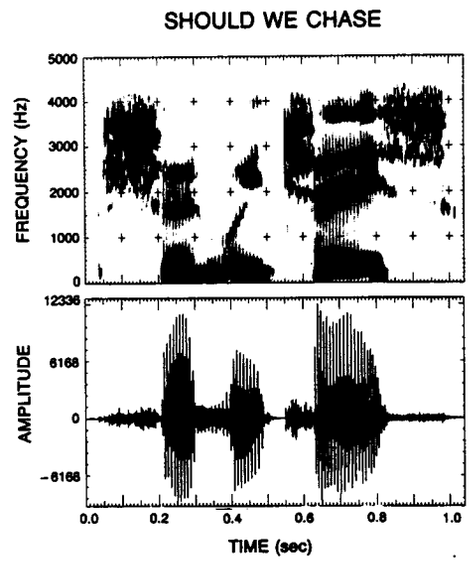



**Figure 2.10** Wideband spectrogram and intensity contour of the phrase "Should we chase."

high performance. To partially illustrate this point, consider the following sections of text:

### Section I

Th_y n_t_d s_gn_f_c_nt _mpr_v_m_nts i_ th_ c_mp_ny's _m_g_, s_p_rv_s__n, th__r w_rk_ng c_nd_t__ns, b_n_f_ts _nd _pp_rt_n_t__s f_r gr_wth.

### Section II

A__i_u_e_ _o_a___a___a_e_ e__e__ia___ __e _a_e, _i__ __e __o_e_ o_ o__u_a_io_a_ e___o_ee_ __i_____ _e__ea_i__.

In Section I we have omitted the conventional vowel letters (a,e,i,o,u); however, with a little effort the average reader can "fill in" the missing vowels and decode the section so that it reads

> They noted significant improvements in the company's image, supervision, their working conditions, benefits and opportunities for growth.

In Section II we have omitted the conventional consonant letters; the resulting text is essentially not decodable. The actual text is



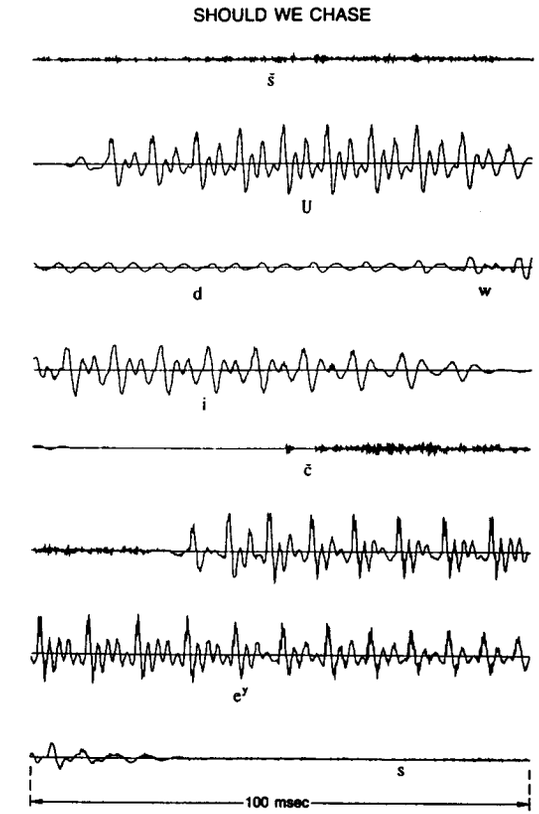



**Figure 2.11** The speech waveform and a segmentation and labeling of the constituent sounds of the phrase "Should we chase."

> Attitudes toward pay stayed essentially the same, with the scores of occupational employees slightly decreasing.

In speaking, vowels are produced by exciting an essentially fixed vocal tract shape with quasi-periodic pulses of air caused by the vibration of the vocal cords. The way in which the cross-sectional area varies along the vocal tract determines the resonance frequencies of the tract (the formants) and thereby the sound that is produced. The vowel sound produced is determined primarily by the position of the tongue, but the positions of the jaw, lips, and to a small extent, the velum, also influence the resulting sound.

**TABLE 2.1.** A condensed list of phonetic symbols for American English.

| Phoneme | ARPABET | Example | Phoneme | ARPABET | Example |
|---|---|---|---|---|---|
| /i/ | IY | beat | /ŋ/ | NX | sing |
| /ɪ/ | IH | bit | /p/ | P | pet |
| /e/ ($e^y$) | EY | bait | /t/ | T | ten |
| /ɛ/ | EH | bet | /k/ | K | kit |
| /æ/ | AE | bat | /b/ | B | bet |
| /a/ | AA | Bob | /d/ | D | debt |
| /ʌ/ | AH | but | /g/ | H | get |
| /ɔ/ | AO | bought | /h/ | HH | hat |
| /o/ ($o^w$) | OW | boat | /f/ | F | fat |
| /U/ | UH | book | /θ/ | TH | thing |
| /u/ | UW | boot | /s/ | S | sat |
| /ə/ | AX | about | /š/ (sh) | SH | shut |
| /ɨ/ | IX | roses | /v/ | V | vat |
| /ɝ/ | ER | bird | /ð/ | DH | that |
| /ɚ/ | AXR | butter | /z/ | Z | zoo |
| /a$^w$/ | AW | down | /ž/ (zh) | ZH | azure |
| /a$^y$/ | AY | buy | /č/ (tsh) | CH | church |
| /ɔ$^y$/ | OY | boy | /ǰ/ (dzh, j) | JH | judge |
| /y/ | Y | you | /ʍ/ | WH | which |
| /w/ | W | wit | /l̩/ | EL | battle |
| /r/ | R | rent | /m̩/ | EM | bottom |
| /l/ | L | let | /n̩/ | EN | button |
| /m/ | M | met | /ɾ/ | DX | batter |
| /n/ | N | net | /ʔ/ | Q | (glottal stop) |

The vowels are generally long in duration (as compared to consonant sounds) and are spectrally well defined. As such they are usually easily and reliably recognized and therefore contribute significantly to our ability to recognize speech, both by human beings and by machine.

There are several ways to characterize and classify vowels, including the typical articulatory configuration required to produce the sounds, typical waveform plots, and typical spectrogram plots. Figures 2.13–2.15 show typical articulatory configurations of the vowels (2.13), examples of vowel waveforms (2.14), and examples of vowel spectrograms (2.15). A convenient and simplified way of classifying vowel articulatory configurations is in terms of the tongue hump position (i.e., front, mid, back), and tongue hump height (high, mid, low), where the tongue hump is the mass of the tongue at its narrowest constriction within the vocal tract. According to this classification the vowels /i/, /ɪ/, /æ/, and /ɛ/ are front vowels, (with different tongue heights) /a/, /ʌ/, and /ɔ/ are mid vowels, and /U/, /u/, and /o/ are back vowels (see also Figure 2.12).

As shown in the acoustic waveform plots of the vowels, in Figure 2.14, the front vowels show a pronounced, high-frequency resonance, the mid vowels show a balance of energy over a broad frequency range, and the back vowels show a predominance of low-frequency spectral information. This behavior is evidenced in the vowel spectrogram



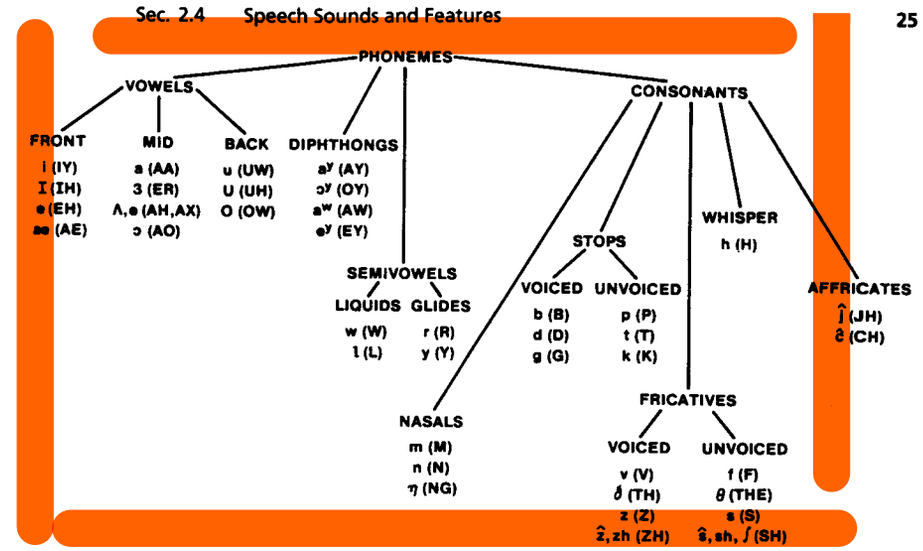



**Figure 2.12** Chart of the classification of the standard phonemes of American English into broad sound classes.

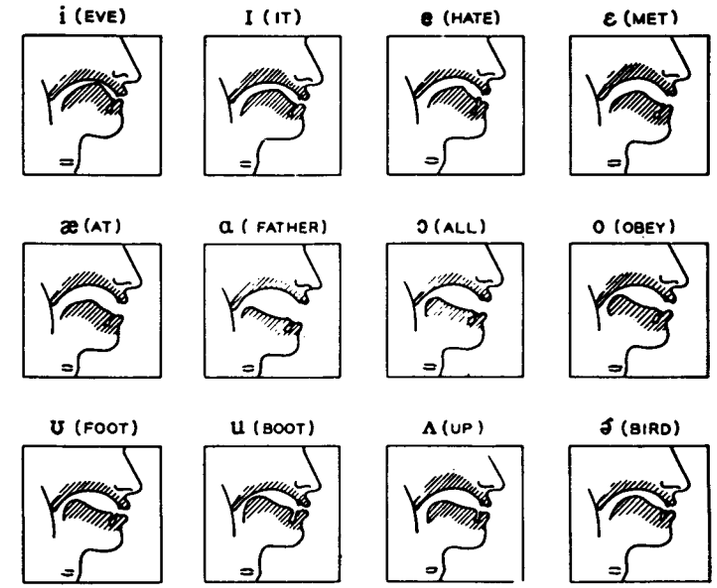



**Figure 2.13** Articulatory configurations for typical vowel sounds (after Flanagan [3]).

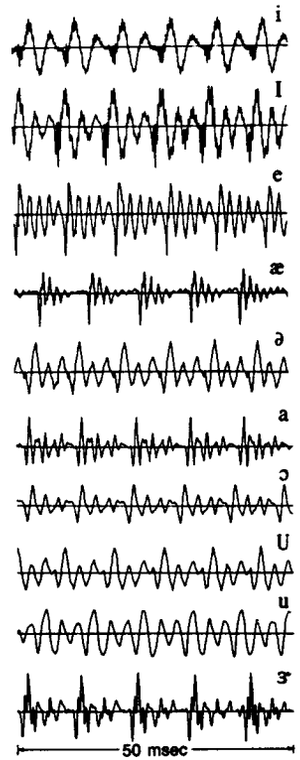



**Figure 2.14** Acoustic waveform plots of typical vowel sounds.

plots of Figure 2.15, in which the front vowels show a relatively high second and third formant frequency (resonance), whereas the mid vowels show well-separated and balanced locations of the formants, and the back vowels (especially /u/) show almost no energy beyond the low-frequency region with low first and second formant frequencies.

The concept of a "typical" vowel sound is, of course, unreasonable in light of the variability of vowel pronunciation among men, women and children with different regional accents and other variable characteristics. To illustrate this point, Figure 2.16 shows a classic plot, made by Gordon Peterson and Harold Barney, of measured values of the first and second formant for 10 vowels spoken by a wide range of male and female talkers who attended the 1939 World's Fair in New York City [7].A wide range of variability can be seen in the measured formant frequencies for a given vowel sound, and also there is



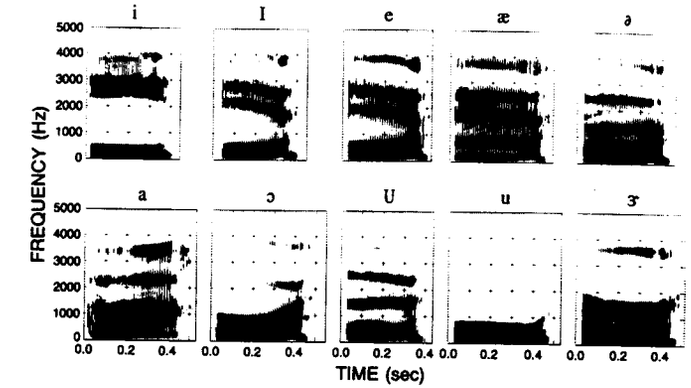



**Figure 2.15** Spectrograms of the vowel sounds.

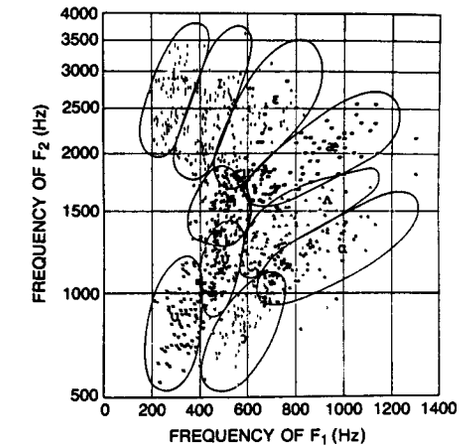



**Figure 2.16** Measured frequencies of first and second formants for a wide range of talkers for several vowels (after Peterson & Barney [7]).

overlap between the formant frequencies for *different* vowel sounds by different talkers. The ellipses drawn in this figure represent gross characterizations of the regions in which most of the tokens of the different vowels lie. The message of Figure 2.16, for speech recognition by machine, is fairly clear; that is, it is not just a simple matter of measuring formant frequencies or spectral peaks accurately to accurately classify vowel sounds; one

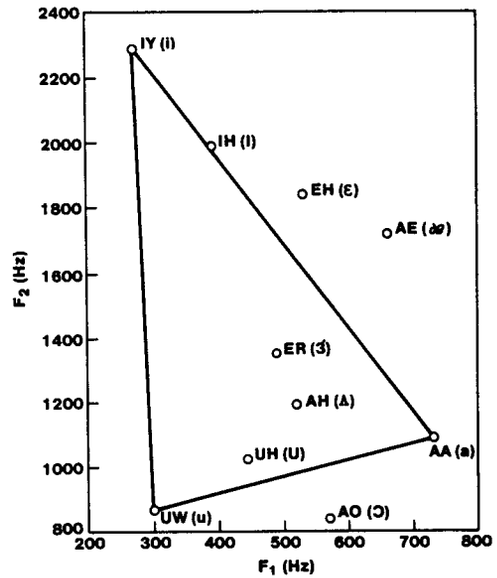



**Figure 2.17** The vowel triangle with centroid positions of the common vowels.

must do some type of talker (accent) normalization to account for the variability in formants and overlap between vowels.

A common way of exploiting the information embodied in Figures 2.15 and 2.16 is to represent each vowel by a centroid in the formant space with the realization that the centroid, at best, represents average behavior and does not represent variability across talkers. Such a representation leads to the classic vowel triangle shown in Figure 2.17 and represented in terms of formant positions by the data given in Table 2.2. The vowel triangle represents the extremes of formant locations in the $F_1$-$F_2$ plane, as represented by /i/ (low $F_1$, high $F_2$), /u/ (low $F_1$, low $F_2$), and /a/ (high $F_1$, low $F_2$), with other vowels appropriately placed with respect to the triangle vertices. The utility of the formant frequencies of Table 2.2 has been demonstrated in text-to-speech synthesis in which high-quality vowel sounds have been synthesized using these positions for the resonances [8].

### 2.4.2 Diphthongs

Although there is some ambiguity and disagreement as to what is and what is not a diphthong, a reasonable definition is that a diphthong is a gliding monosyllabic speech



**TABLE 2.2.** Formant frequencies for typical vowels.

| ARPABET Symbol for Vowel | IPA Symbol | Typical Word | $F_1$ | $F_2$ | $F_3$ |
|---|---|---|---|---|---|
| IY | /i/ | beet | 270 | 2290 | 3010 |
| IH | /I/ | bit | 390 | 1990 | 2550 |
| EH | /ɛ/ | bet | 530 | 1840 | 2480 |
| AE | /æ/ | bat | 660 | 1720 | 2410 |
| AH | /ʌ/ | but | 520 | 1190 | 2390 |
| AA | /a/ | hot | 730 | 1090 | 2440 |
| AO | /ɔ/ | bought | 570 | 840 | 2410 |
| UH | /U/ | foot | 440 | 1020 | 2240 |
| UW | /u/ | boot | 300 | 870 | 2240 |
| ER | /ɝ/ | bird | 490 | 1350 | 1690 |

sound that starts at or near the articulatory position for one vowel and moves to or toward the position for another. According to this definition, there are six diphthongs in American English, namely /a$^y$/ (as in buy), /a$^w$/ (as in down), /e$^y$/ (as in bait), and /ɔ$^y$/ (as in boy), /o/ (as in boat), and /ju/ (as in you).

The diphthongs are produced by varying the vocal tract smoothly between vowel configurations appropriate to the diphthong. Figure 2.18 shows spectrogram plots of four of the diphthongs spoken by a male talker. The gliding motions of the formants are especially prominent for the sounds /a$^y$/, /a$^w$/ and /ɔ$^y$/ and are somewhat weaker for /e$^y$/ because of the closeness (in vowel space) of the two vowel sounds comprising this diphthong.

An alternative way of displaying the time-varying spectral characteristics of diphthongs is via a plot of the values of the second formant versus the first formant (implicitly as a function of time) as shown in Figure 2.19 [9]. The arrows in this figure indicate the direction of motion of the formants (in the ($F_1 - F_2$) plane) as time increases. The dashed circles in this figure indicate average positions of the vowels. Based on these data, and other measurements, the diphthongs can be characterized by a time-varying vocal tract area function that varies between two vowel configurations.

### 2.4.3 Semivowels

The group of sounds consisting of /w/, /l/, /r/, and /y/ is quite difficult to characterize. These sounds are called semivowels because of their vowel-like nature. They are generally characterized by a gliding transition in vocal tract area function between adjacent phonemes. Thus the acoustic characteristics of these sounds are strongly influenced by the context in which they occur. For our purposes, they are best described as transitional, vowel-like sounds, and hence are similar in nature to the vowels and diphthongs.

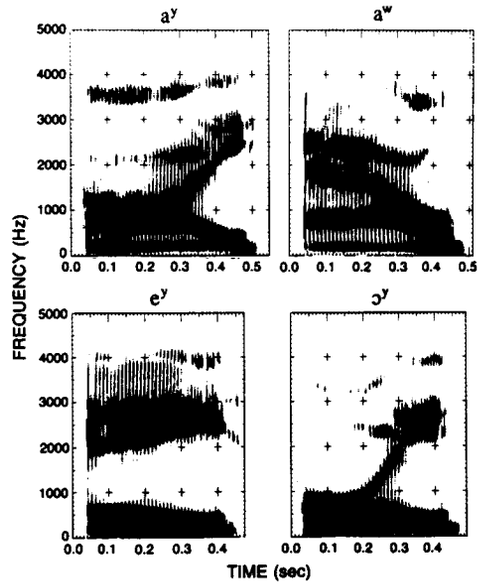



**Figure 2.18** Spectrogram plots of four diphthongs.

### 2.4.4 Nasal Consonants

The nasal consonants /m/, /n/, and /ŋ/ are produced with glottal excitation and the vocal tract totally constricted at some point along the oral passageway. The velum is lowered so that air flows through the nasal tract, with sound being radiated at the nostrils. The oral cavity, although constricted toward the front, is still acoustically coupled to the pharynx. Thus, the mouth serves as a resonant cavity that traps acoustic energy at certain natural frequencies. As far as the radiated sound is concerned, these resonant frequencies of the oral cavity appear as antiresonances, or zeros of the transfer function of sound transmission. Furthermore, nasal consonants and nasalized vowels (i.e., some vowels preceding or following nasal consonants) are characterized by resonances that are spectrally broader, or more highly damped, than those for vowels.

The three nasal consonants are distinguished by the place along the oral tract at which a total constriction is made. For /m/ the constriction is at the lips; for /n/ the constriction is just behind the teeth; and for /ŋ/ the constriction is just forward of the velum itself. Figure 2.20 shows typical speech waveforms and Figure 2.21 spectrograms for two nasal consonants in the context vowel-nasal-vowel. The waveforms of /m/ and /n/ look very similar. The spectrograms show a concentration of low-frequency energy with a midrange of frequencies that contain no prominent peaks. This is because of the particular



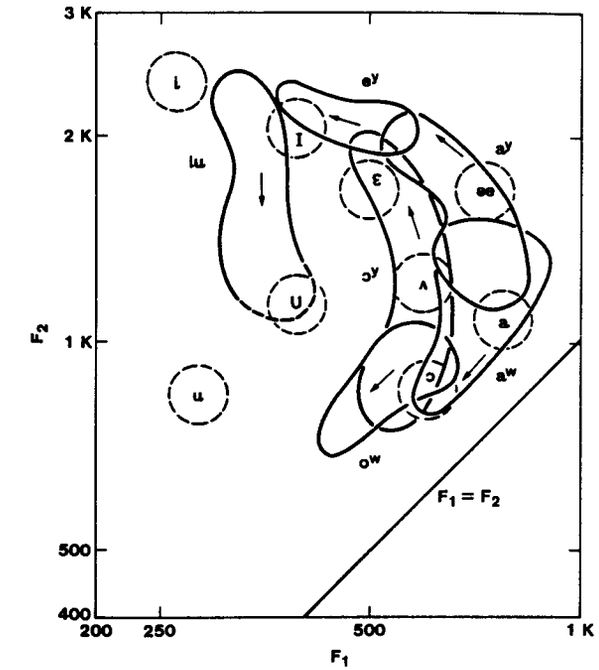



**Figure 2.19** Time variation of the first two formants for the diphthongs (after Holbrook and Fairbanks [9]).

combination of resonances and antiresonances that result from the coupling of the nasal and oral tracts.

### 2.4.5 Unvoiced Fricatives

The unvoiced fricatives /f/, /θ/, /s/, and /sh/ are produced by exciting the vocal tract by a steady air flow, which becomes turbulent in the region of a constriction in the vocal tract. The location of the constriction serves to determine which fricative sound is produced. For the fricative /f/ the constriction is near the lips; for /θ/ it is near the teeth; for /s/ it is near the middle of the oral tract; and for /sh/ it is near the back of the oral tract. Thus the system for producing unvoiced fricatives consists of a source of noise at a constriction, which separates the vocal tract into two cavities. Sound is radiated from the lips—that is, from the front cavity. The back cavity serves, as in the case of nasals, to trap energy and thereby introduce antiresonances into the vocal output. Figure 2.22 shows the waveforms and Figure 2.23 the spectrograms of the fricatives /f/, /s/ and /sh/. The nonperiodic nature

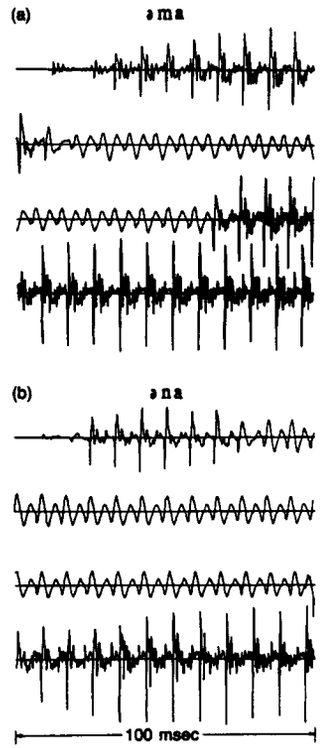



**Figure 2.20** Waveforms for the sequences /ə-m-a/ and /ə-n-a/.

of fricative excitation is obvious in the waveform plots. The spectral differences among the fricatives are readily seen by comparing the three spectrograms.

### 2.4.6 Voiced Fricatives

The voiced fricatives /v/, /th/, /z/ and /zh/ are the counterparts of the unvoiced fricatives /f/, /θ/, /s/, and /sh/, respectively, in that the place of constriction for each of the corresponding phonemes is essentially identical. However, the voiced fricatives differ markedly from their unvoiced counterparts in that two excitation sources are involved in their production. For voiced fricatives the vocal cords are vibrating, and thus one excitation source is at the glottis. However, since the vocal tract is constricted at some point forward of the glottis, the air flow becomes turbulent in the neighborhood of the constriction. Thus the



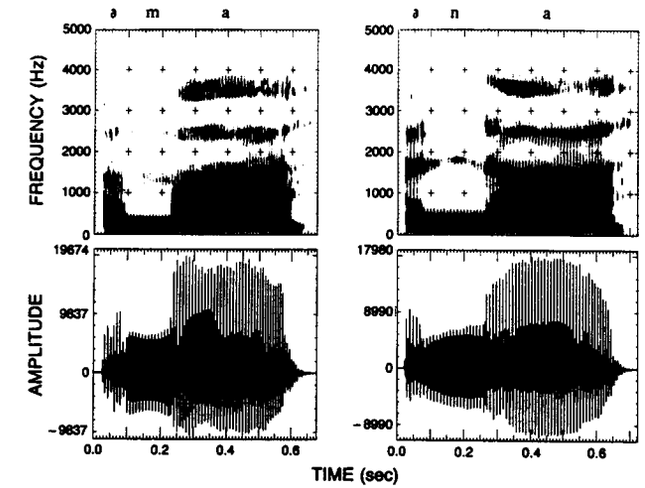



**Figure 2.21** Spectrograms of the sequences /ə-m-a/ and ə-n-a/.

spectra of voiced fricatives can be expected to display two distinct components. These excitation features are readily observable in Figure 2.24, which shows typical waveforms, and in Figure 2.25, which shows spectra for two voiced fricatives. The similarity of the unvoiced fricative /f/ to the voiced fricative /v/ is easily shown in a comparison between corresponding spectrograms in Figures 2.23 and 2.25. Likewise, it is instructive to compare the spectrograms of /sh/ and /zh/.

### 2.4.7 Voiced and Unvoiced Stops

The voiced stop consonants /b/, /d/, and /g/, are transient, noncontinuant sounds produced by building up pressure behind a total constriction somewhere in the oral tract and then suddenly releasing the pressure. For /b/ the constriction is at the lips; for /d/ the constriction is at the back of the teeth; and for /g/ it is near the velum. During the period when there is total constriction in the tract, no sound is radiated from the lips. However, there is often a small amount of low-frequency energy radiated through the walls of the throat (sometimes called a voice bar). This occurs when the vocal cords are able to vibrate even though the vocal tract is closed at some point.

Since the stop sounds are dynamical in nature, their properties are highly influenced by the vowel that follows the stop consonant. As such, the waveforms for stop consonants give little information about the particular stop consonant. Figure 2.26 shows the waveform of the syllable /ə-b-a/. The waveform of /b/ shows few distinguishing features except for the voiced excitation and lack of high-frequency energy.

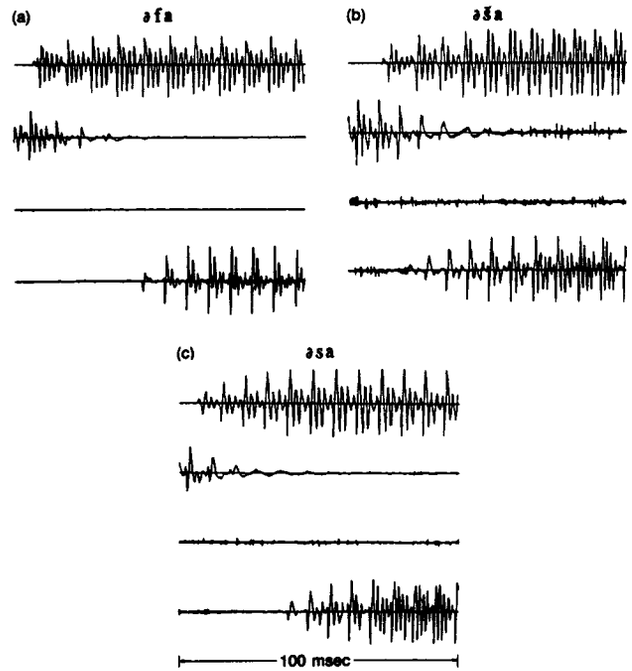



**Figure 2.22** Waveforms for the sounds /f/, /s/ and /sh/ in the context /ə-x-a/ where /x/ is the unvoiced fricative.

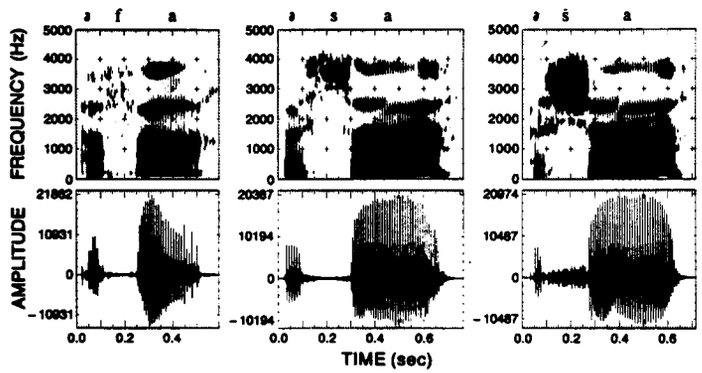



**Figure 2.23** Spectrogram comparisons of the sounds /ə-f-a/, /ə-s-a/ and /ə-sh-a/.



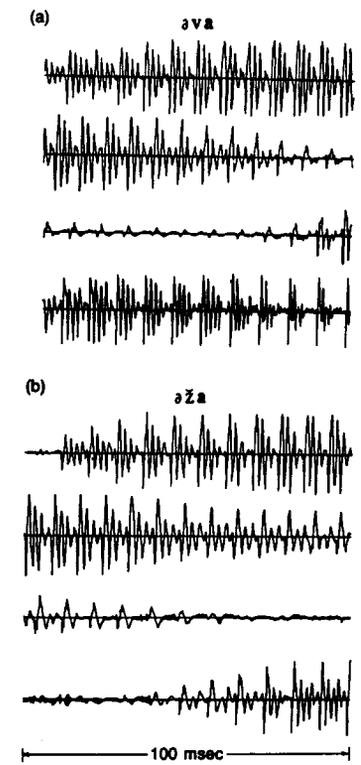



**Figure 2.24** Waveforms for the sequences /ə-v-a/ and /ə-zh-a/.

The unvoiced stop consonants /p/, /t/, and /k/ are similar to their voiced counterparts /b/, /d/, and /g/, with one major exception. During the period of total closure of the tract, as the pressure builds up, the vocal cords do not vibrate. Then, following the period of closure, as the air pressure is released, there is a brief interval of friction (due to sudden turbulence of the escaping air) followed by a period of aspiration (steady air flow from the glottis exciting the resonances of the vocal tract) before voiced excitation begins.

Figure 2.27 shows waveforms and Figure 2.28 shows spectrograms of the voiced stop /b/ and the voiceless stop consonants /p/ and /t/. The "stop gap," or time interval, during which the pressure is built up is clearly in evidence. Also, it can be readily seen that the duration and frequency content of the frication noise and aspiration vary greatly with the stop consonant.

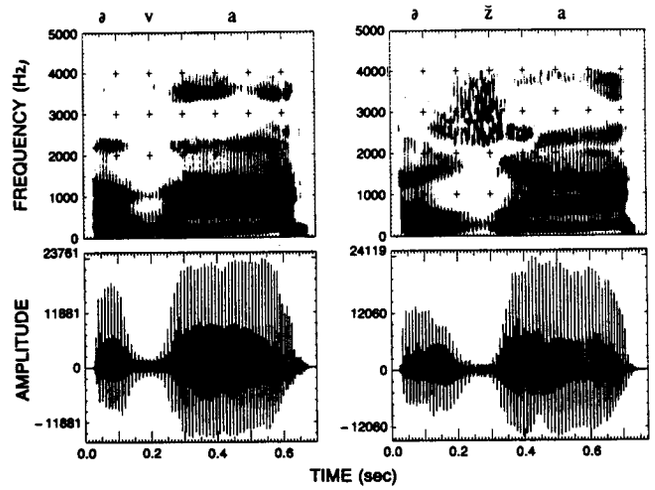



**Figure 2.25** Spectrograms for the sequences /ə-v-a/ and /ə-zh-a/.

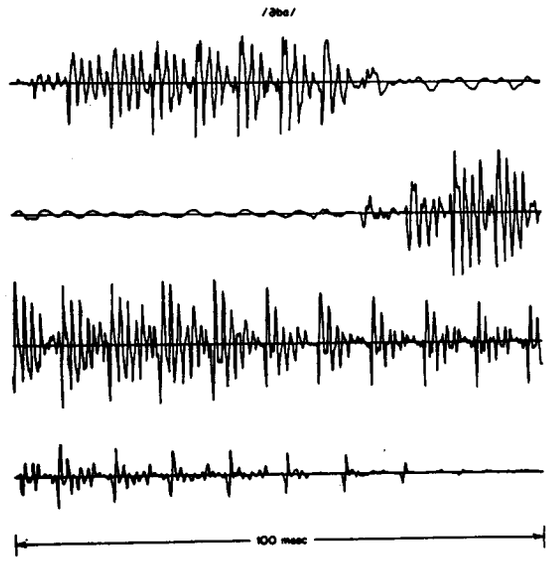



**Figure 2.26** Waveform for the sequence /ə-b-a/.



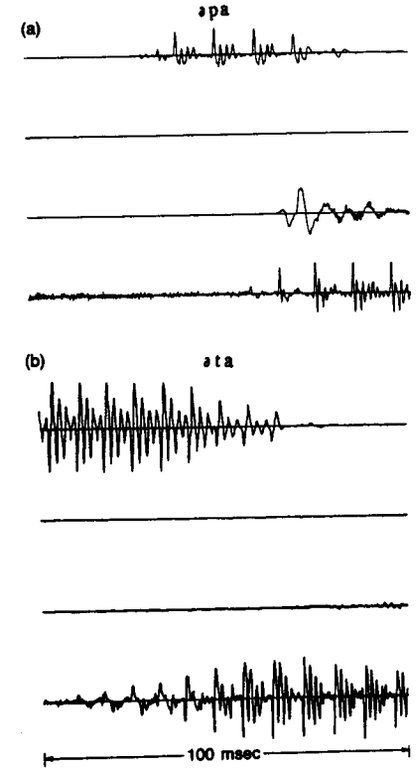



**Figure 2.27** Waveforms for the sequences /ə-p-a/ and /ə-t-a/.

### 2.4.8 Review Exercises

As a self-check on the reader's understanding of the material on speech sounds and their acoustic manifestations, we now digress and present some simple exercises along with the solutions. For maximum effectiveness, the reader is encouraged to think through each exercise before looking at the solution.

**Exercise 2.1**

1. Write out the phonetic transcription for the following words:

   he, eats, several, light, tacos

2. What effect occurs when these five words are spoken in sequence as a sentence? What does this imply about automatic speech recognition?

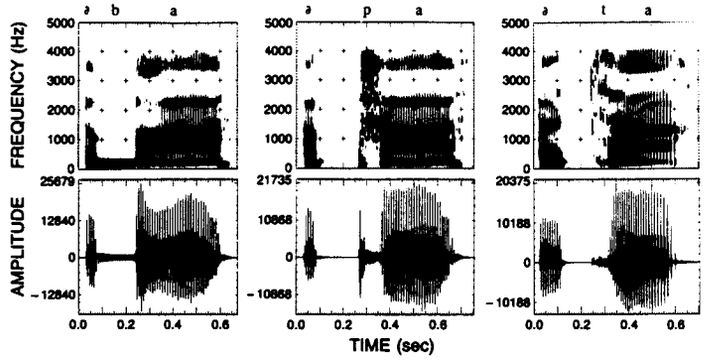



**Figure 2.28** Spectrogram comparisons of the sequences of voiced (/ə-b-a/) and voiceless (/ə-p-a/ and /ə-t-a/) stop consonants.

**Solution 2.1**

1. The phonetic transcriptions of the words are

| Word | Phoneme Sequence | ARPABET |
|---|---|---|
| he | /hi/ | HH–IY |
| eats | /its/ | IY–TS |
| several | /sɛv rʌl/ | S–EH–V–R–AH–L |
| light | /laʸt/ | L–AY–T |
| tacos | /takoz/ | T–AA–K–OW–Z |

2. When the words are spoken together, the last sound of each word merges with the first sound of the succeeding word (since they are the same sound), resulting in strong coarticulation of boundary sounds. The ARPABET transcription for the sentence is:

   HH–IY–T–S–EH–V–R–AH–L–AY–T–AA–K–OW–Z

   All information about word boundaries is totally lost; furthermore, the durations of the common sounds at the boundaries of words are much shorter than what would be predicted from the individual words.

**Exercise 2.2**

Some of the difficulties in large vocabulary speech recognition are related to the irregularities in the way basic speech sounds are combined to produce words. Exercise 2.2 highlights a couple of these difficulties.

1. In word initial position of American English, which phoneme or phonemes can never occur? Which hardly ever occur?
2. There are many word initial consonant clusters of length two, such as *sp*eak, *dr*ank, *pl*ead, and *pr*ess. How many word initial consonant clusters of length three are there in American English? What general rule can you give about the sounds in each of the three positions?



3. A nasal consonant can be combined with a stop consonant (e.g., *camp*, *tend*) in a limited number of ways. What general rule do such combinations obey? There are several notable exceptions to this general rule. Can you give a couple of exceptions? What kind of speaking irregularity often results from these exceptions?

**Solution 2.2**

1. The only phoneme that never occurs in initial word position in English is the /ng/ sound (e.g., *sing*). The only other sound that almost never occurs naturally in English, in initial word position, is /zh/ except some foreign words imported into English, such as gendarme, which does have an initial /zh/.
2. The word initial consonant clusters of length three in English include

   /spl/ — split
   /spr/ — spring
   /skw/ — squirt
   /skr/ — script
   /str/ — string

   The general rule for such clusters is

   /sound s/unvoiced stop/semivowel/

3. The general rule for a nasal-stop combination is that the nasal and stop have the same place of articulation, e.g., front/lips (/mp/), mid/dental (/nt/), back/velar (/ng k/). Exceptions occur in words like summed (/md/) or hanged (/ng d/) or dreamt (/mt/). There is often a tendency to insert an extra stop in such situations (e.g., dreamt → /drempt/).

**Exercise 2.3**

An important speech task is accurate digit recognition. This exercise seeks to exploit knowledge of acoustic phonetics to recognize first isolated digits, and next some simple connected digit strings. We first need a sound lexicon (a dictionary) for the digits. The sound lexicon describes the pronunciations of digits in terms of the basic sounds of English. Such a sound lexicon is given in Table 2.3. A single male adult talker (LRR) spoke each of the 11 digits in random sequence and in isolation, and spectrograms of these spoken utterances are shown in Figure 2.29. Figure 2.30 shows spectrograms of two connected digit sequences spoken by the same talker.

1. Identify each of the 11 digits based on the acoustic properties of the sounds within the digit (as expressed in the sound lexicon). Remember that each digit was spoken exactly once.
2. Try to identify the spoken digits in each of the connected digit strings.

**Solution 2.3**

1. The digits of the top row are 3 and 7:
   a. The digit 3 is cued by the distinctive brief initial fricative (/θ/), followed by the semivowel /ɹ/ where the second and third formants both get very low in frequency, followed by the /i/ where $F_2$ and $F_3$ both become very high in frequency.
   b. The digit 7 is cued by the strong /s/ frication at the beginning, the distinctive /ɛ/, followed by the voiced fricative /v/, a short vowel /ə/ and ending in the strong

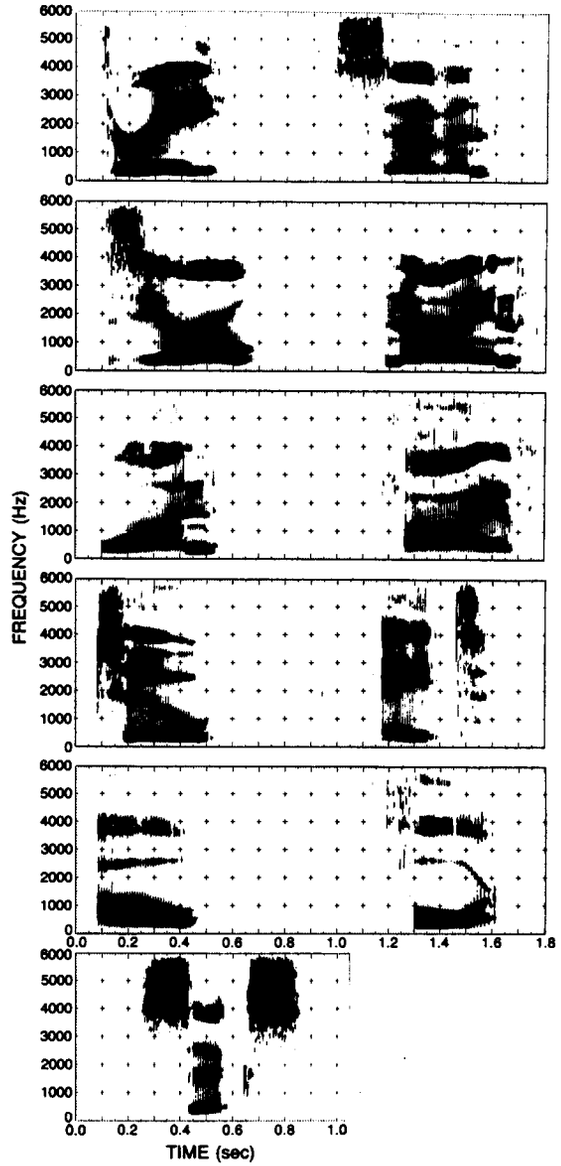



**Figure 2.29** Spectrograms of the 11 isolated digits, 0 through 9 plus oh, in random sequence.



**TABLE 2.3.** Sound Lexicon of Digits

| Word | Sounds | ARPABET |
|---|---|---|
| Zero | /z I r o/ | Z–IH–R–OW |
| One | /w ʌ n/ | W–AH–N |
| Two | /t u/ | T–UW |
| Three | /θ r i/ | TH–R–IY |
| Four | /f o r/ | F–OW–R |
| Five | /f $a^y$ v/ | F–AY–V |
| Six | /s I k s/ | S–IH–K–S |
| Seven | /s ɛ v ə n/ | S–EH–V–AX–N |
| Eight | /$e^y$ t/ | EY–T |
| Nine | /n $a^y$ n/ | N–AY–N |
| Oh | /o/ | OW |

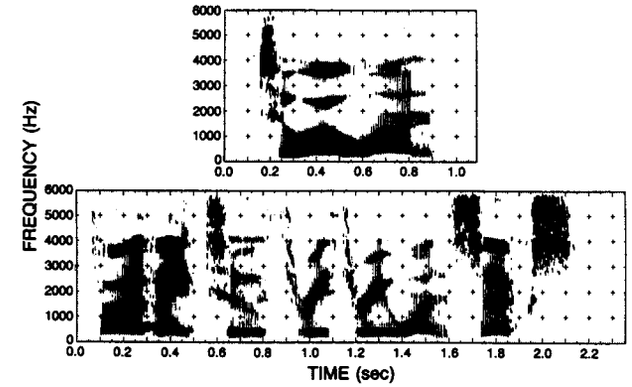



**Figure 2.30** Spectrograms of two connected digit sequences.

nasal /n/.

The digits in the second row are 0 and 9:

**a.** The initial /z/ is cued by the strong frication with the presence of voicing at low frequencies; the following /I/ is seen by the high $F_2$ and $F_3$, the /r/ is signaled by the low $F_2$ and $F_3$, and the diphthong /o/ is signaled by the gliding motion of $F_2$ and $F_3$ toward an /u/-like sound.

**b.** The digit 9 is cued by the distinct initial and final nasals /n/ and by the /$a^y$/ glide between the nasals.

The digits in the third row are 1 and 5:

**a.** The digit 1 is cued by the strong initial semivowel /w/ with very low $F_2$ and by the strong final nasal /n/.

**b.** The digit 5 is cued by the weak initial frication of /f/, followed by the strong diphthong /$a^y$/ and ending in the very weak fricative /v/.

The digits in the fourth row are 2 and 8:

a. The digit 2 is cued by the strong /t/ burst and release followed by the glide to the /u/ sound.

b. The digit 8 is cued by the initial weak diphthong /e$^y$/ followed by a clear stop gap of the /t/ and then the /t/ release.

The digits in the fifth row are "oh" and 4:

a. The digit "oh" is virtually a steady sound with a slight gliding tendency toward /u/ at the end.

b. The digit 4 is cued by the weak initial fricative /f/, followed by the strong /o/ vowel and ending with a classic /r/ where $F_2$ and $F_3$ merge together.

The digit in the last row is 6:

a. The digit 6 is cued by the strong /s/ frication at the beginning and end, and by the steady vowel /I/ followed by the stop gap and release of the /k/.

2. By examining the isolated digit sequences, one can eventually (with a lot of work and some good luck) conclude that the two sequences are

| | | |
|---|---|---|
| Row 1: | 2-oh-1 | (telephone area code) |
| Row 2: | 5-8-2-3-3-1-6 | (7-digit telephone number) |

We will defer any explanation of how any reasonable person, or machine, could perform this task until later in this book when we discuss connected word-recognition techniques. The purpose of this exercise is to convince the reader how difficult a relatively simple recognition task can be.

## 2.5 APPROACHES TO AUTOMATIC SPEECH RECOGNITION BY MACHINE

The material presented in the previous sections leads to a straightforward way of performing speech recognition by machine whereby the machine attempts to decode the speech signal in a sequential manner based on the observed acoustic features of the signal and the known relations between acoustic features and phonetic symbols. This method, appropriately called the acoustic-phonetic approach, is indeed viable and has been studied in great depth for more than 40 years. However, for a variety of reasons, the acoustic-phonetic approach has not achieved the same success in practical systems as have alternative methods. Hence, in this section, we provide an overview of several proposed approaches to automatic speech recognition by machine with the goal of providing some understanding as to the essentials of each proposed method, and the basic strengths and weaknesses of each approach.

Broadly speaking, there are three approaches to speech recognition, namely:

1. the acoustic-phonetic approach
2. the pattern recognition approach
3. the artificial intelligence approach

The acoustic-phonetic approach is based on the theory of acoustic phonetics that postulates that there exist finite, distinctive phonetic units in spoken language and that the phonetic



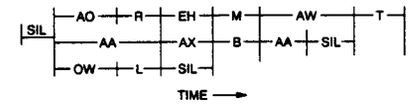



**Figure 2.31** Phoneme lattice for word string.

units are broadly characterized by a set of properties that are manifest in the speech signal, or its spectrum, over time. Even though the acoustic properties of phonetic units are highly variable, both with speakers and with neighboring phonetic units (the so-called coarticulation of sounds), it is assumed that the rules governing the variability are straightforward and can readily be learned and applied in practical situations. Hence the first step in the acoustic-phonetic approach to speech recognition is called a segmentation and labeling phase because it involves segmenting the speech signal into discrete (in time) regions where the acoustic properties of the signal are representative of one (or possibly several) phonetic units (or classes), and then attaching one or more phonetic labels to each segmented region according to the acoustic properties. To actually do speech recognition, a second step is required. This second step attempts to determine a valid word (or string of words) from the sequence of phonetic labels produced in the first step, which is consistent with the constraints of the speech-recognition task (i.e., the words are drawn from a given vocabulary, the word sequence makes syntactic sense and has semantic meaning, etc.).

To illustrate the steps involved in the acoustic-phonetic approach to speech recognition, consider the phoneme lattice shown in Figure 2.31. (A phoneme lattice is the result of the segmentation and labeling step of the recognition process and represents a sequential set of phonemes that are likely matches to the spoken input speech.) The problem is to decode the phoneme lattice into a word string (one or more words) such that every instant of time is included in one of the phonemes in the lattice, and such that the word (or word sequence) is valid according to rules of English syntax. (The symbol SIL stands for silence or a pause between sounds or words; the vertical position in the lattice, at any time, is a measure of the goodness of the acoustic match to the phonetic unit, with the highest unit having the best match.) With a modest amount of searching, one can derive the appropriate phonetic string SIL–AO–L–AX–B–AW–T corresponding to the word string "all about," with the phonemes L, AX, and B having been second or third choices in the lattice and all other phonemes having been first choices. This simple example illustrates well the difficulty in decoding phonetic units into word strings. This is the so-called lexical access problem. Interestingly, as we will see in the next section, the real problem with the acoustic-phonetic approach to speech recognition is the difficulty in getting a reliable phoneme lattice for the lexical access stage.

The pattern-recognition approach to speech recognition is basically one in which the speech patterns are used directly without explicit feature determination (in the acoustic-phonetic sense) and segmentation. As in most pattern-recognition approaches, the method has two steps—namely, training of speech patterns, and recognition of patterns via pattern comparison. Speech "knowledge" is brought into the system via the training procedure. The concept is that if enough versions of a pattern to be recognized (be it a sound, a word, a phrase, etc.) are included in a training set provided to the algorithm, the training procedure

should be able to adequately characterize the acoustic properties of the pattern (with no regard for or knowledge of any other pattern presented to the training procedure). This type of characterization of speech via training is called pattern classification because the machine learns which acoustic properties of the speech class are reliable and repeatable across all training tokens of the pattern. The utility of the method is the pattern-comparison stage, which does a direct comparison of the unknown speech (the speech to be recognized), with each possible pattern learned in the training phase and classifies the unknown speech according to the goodness of match of the patterns.

The pattern-recognition approach to speech recognition is the basis for the remainder of this book. Hence there will be a great deal of discussion and explanation of virtually every aspect of the procedure. However, at this point, suffice it to say that the pattern-recognition approach is the method of choice for speech recognition for three reasons:

1. Simplicity of use. The method is easy to understand, it is rich in mathematical and communication theory justification for individual procedures used in training and decoding, and it is widely used and understood.
2. Robustness and invariance to different speech vocabularies, users, feature sets, pattern comparison algorithms and decision rules. This property makes the algorithm appropriate for a wide range of speech units (ranging from phonemelike units all the way through words, phrases, and sentences), word vocabularies, talker populations, background environments, transmission conditions, etc.
3. Proven high performance. It will be shown that the pattern-recognition approach to speech recognition consistently provides high performance on any task that is reasonable for the technology and provides a clear path for extending the technology in a wide range of directions such that the performance degrades gracefully as the problem becomes more and more difficult.

The so-called artificial intelligence approach to speech recognition is a hybrid of the acoustic-phonetic approach and the pattern-recognition approach in that it exploits ideas and concepts of both methods. The artificial intelligence approach attempts to mechanize the recognition procedure according to the way a person applies its intelligence in visualizing, analyzing, and finally making a decision on the measured acoustic features. In particular, among the techniques used within this class of methods are the use of an expert system for segmentation and labeling so that this crucial and most difficult step can be performed with more than just the acoustic information used by pure acoustic-phonetic methods (in particular, methods that integrate phonemic, lexical, syntactic, semantic, and even pragmatic knowledge into the expert system have been proposed and studied); learning and adapting over time (i.e., the concept that knowledge is often both static and dynamic and that models must adapt to the dynamic component of the data); the use of neural networks for learning the relationships between phonetic events and all known inputs (including acoustic, lexical, syntactic, semantic, etc.) as well as for discrimination between similar sound classes.

The use of neural networks could represent a separate structural approach to speech recognition or be regarded as an implementational architecture that may be incorporated



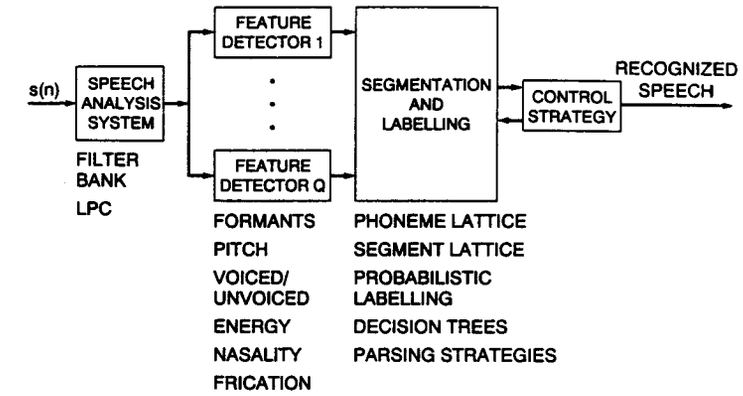



**Figure 2.32** Block diagram of acoustic-phonetic speech-recognition system.

in any of the above three classical approaches. The concepts and ideas of applying neural networks to speech-recognition problems are relatively new; hence we will devote a fair amount of discussion within this chapter to outline the basic ways in which neural networks are used in general, and applied to problems in speech recognition, in particular. In the next several sections we expand on the ideas of these three general approaches to speech recognition by machine.

### 2.5.1 Acoustic-Phonetic Approach to Speech Recognition

Figure 2.32 shows a block diagram of the acoustic-phonetic approach to speech recognition. The first step in the processing (a step common to all approaches to speech recognition) is the speech analysis system (the so-called feature measurement method), which provides an appropriate (spectral) representation of the characteristics of the time-varying speech signal. The most common techniques of spectral analysis are the class of filter bank methods and the class of linear predictive coding (LPC) methods. The properties of these methods will be discussed in great detail in Chapter 3. Broadly speaking, both of these methods provide spectral descriptions of the speech over time.

The next step in the processing is the feature-detection stage. The idea here is to convert the spectral measurements to a set of features that describe the broad acoustic properties of the different phonetic units. Among the features proposed for recognition are nasality (presence or absence of nasal resonance), frication (presence or absence of random excitation in the speech), formant locations (frequencies of the first three resonances), voiced-unvoiced classification (periodic or aperiodic excitation), and ratios of high- and low-frequency energy. Many proposed features are inherently binary (e.g., nasality, frication, voiced-unvoiced); others are continuous (e.g., formant locations, energy ratios). The feature-detection stage usually consists of a set of detectors that operate in parallel and use appropriate processing and logic to make the decision as to presence or absence, or

value, of a feature. The algorithms used for individual feature detectors are sometimes sophisticated ones that do a lot of signal processing, and sometimes they are rather trivial estimation procedures.

The third step in the procedure is the segmentation and labeling phase whereby the system tries to find stable regions (where the features change very little over the region) and then to label the segmented region according to how well the features within that region match those of individual phonetic units. This stage is the heart of the acoustic-phonetic recognizer and is the most difficult one to carry out reliably; hence various control strategies are used to limit the range of segmentation points and label possibilities. For example, for individual word recognition, the constraint that a word contains at least two phonetic units and no more than six phonetic units means that the control strategy need consider solutions with between 1 and 5 internal segmentation points. Furthermore, the labeling strategy can exploit lexical constraints on words to consider only words with $n$ phonetic units whenever the segmentation gives $n - 1$ segmentation points. These constraints are often powerful ones that reduce the search space and significantly increase performance (accuracy of segmentation and labeling) of the system.

The result of the segmentation and labeling step is usually a phoneme lattice (of the type shown in Figure 2.31) from which a lexical access procedure determines the best matching word or sequence of words. Other types of lattices (e.g., syllable, word) can also be derived by integrating vocabulary and syntax constraints into the control strategy as discussed above. The quality of the matching of the features, within a segment, to phonetic units can be used to assign probabilities to the labels, which then can be used in a probabilistic lexical access procedure. The final output of the recognizer is the word or word sequence that best matches, in some well-defined sense, the sequence of phonetic units in the phoneme lattice.

### 2.5.1.1 Acoustic Phonetic Vowel Classifier

To illustrate the labeling procedure on a segment classified as a vowel, consider the flow chart of Figure 2.33. We assume that three features have been detected over the segment—namely, first formant, $F_1$, second formant, $F_2$, and duration of the segment, D. Consider just the set of steady vowels (i.e., we exclude the diphthongs). To classify a vowel segment as one of the 10 steady vowels, several tests can be made to separate groups of vowels. As shown in Figure 2.33 the first test separates vowels with low $F_1$ (called diffuse vowels and including /i/, /I/, /ə/, /U/, /u/) from vowels with high $F_1$ (called compact vowels and including /ɛ/, /æ/, /a/, /ʌ/, /ɔ/). Each of these subsets can be split further on the basis of $F_2$ measurements, with acute vowels having high $F_2$ and grave vowels having low $F_2$. The third test is one based on segment duration, which separates tense vowels (large values of D) from lax vowels (small values of D). Finally, a finer test on formant values separates the remaining unresolved vowels, resolving the vowels into flat vowels (where $F_1 + F_2$ exceeds a threshold T) and plain vowels (where $F_1 + F_2$ falls below the threshold T).

It should be clear that there are several thresholds embedded within the vowel classifier. Such thresholds are often determined experimentally so as to maximize classification accuracy on a given corpus of speech.



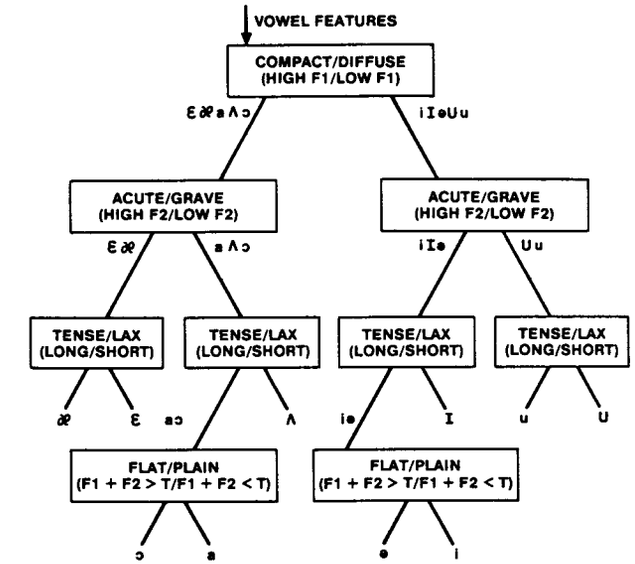



**Figure 2.33** Acoustic-phonetic vowel classifier.

### 2.5.1.2 Speech Sound Classifier

Vowel classification is just a small part of the phonetic labeling procedure of an acoustic-phonetic recognizer. In theory, one needs a method of classifying an arbitrary segment into one (or more) of the 40 plus phonetic units discussed earlier in this chapter. Rather than discussing how to solve this very difficult problem, consider the somewhat simpler problem of classifying a speech segment into one of several broad speech classes—e.g., unvoiced stop, voiced stop, unvoiced fricative. Again there is no simple or generally well-accepted procedure for accomplishing this task; however, we show in Figure 2.34 one simple and straightforward way to accomplish such a classification.

The method uses a binary tree to make decisions as to various broad sound classes. The first decision is a *sound/silence* split in which the speech features (primarily energy in this case) are compared to selected thresholds, and *silence* is split off if the test is negative for speech sounds. The second decision is a *voiced/unvoiced* decision (primarily based on the presence of periodicity within the segment) in which unvoiced sounds are split apart from voiced sounds. A test for unvoiced stop consonants is made (seeing if a stop gap of silence preceded the segment), and this separates the unvoiced stops (/t/, /p/, /k/, /ĉ/) from the unvoiced fricatives (/f/, /θ/, /s/, /š/). A *high-frequency/low-frequency* (energy) test separates voiced fricatives (/v/, /ð/, /z/, /ž/) from other voiced sounds. Voiced stops are separated out by checking to see whether the preceding sound is silence (or silencelike). Finally, a *vowel/sonorant* spectral test (searching for spectral gaps) separates vowels from

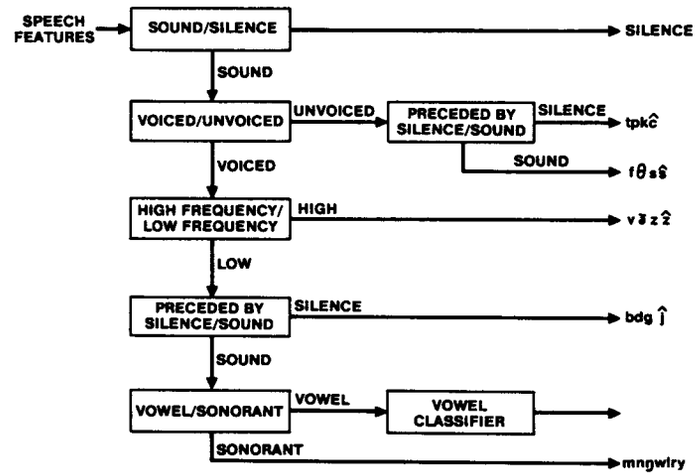



**Figure 2.34** Binary tree speech sound classifier.

sonorants (nasal consonants and /w/, /l/, /r/, and /y/). The vowel classifier of Figure 2.33 can then be used for finer vowel distinctions.

The tests shown in Figure 2.34 are rather crude and are therefore highly prone to error. For example, some voiced stop consonants are *not* preceded by silence or by a silencelike sound. Another problem is that no way of distinguishing diphthongs from vowels is provided. Virtually every decision in the binary tree is subject to scrutiny as to its utility in any practical system.

### 2.5.1.3 Examples of Acoustic Phonetic Labeling

To illustrate some of the difficulties faced by the acoustic-phonetic approach to speech recognition, consider the following example. (Shown in the example is the phonetic labeling of a sentence [only the top-choice phonetic candidate is shown for each segment], along with its decoding into the proper word sequence.) In this example (taken from an actual acoustic-phonetic recognizer) we see that there are inserted phonetic units (Y in "MAY," AX in "BY"), deleted phonetic units (N in "EARN," N in "MONEY"), and phonetic substitutions (J for K in "WORKING," N for NG in "WORKING"). The difficulty of proper decoding of phonetic units into words and sentences grows dramatically with increases in the rates of phoneme insertion, deletion, and substitution.

| | | | | |
|---|---|---|---|---|
| phonemes: | /sil/ –/j/–/e/–/n/ | /m/–/e/–/y/ | /ɝ/–/m/–/ɔ/–/r/ | /m/–/ʌ/–/sil/–/e/ |
| ARPABET: | SIL –JH–EY–N + | M –EY–Y + | ER–M–AO–R + | M–AH–SIL–EY |
| words: | JANE | MAY | EARN MORE | MONEY |



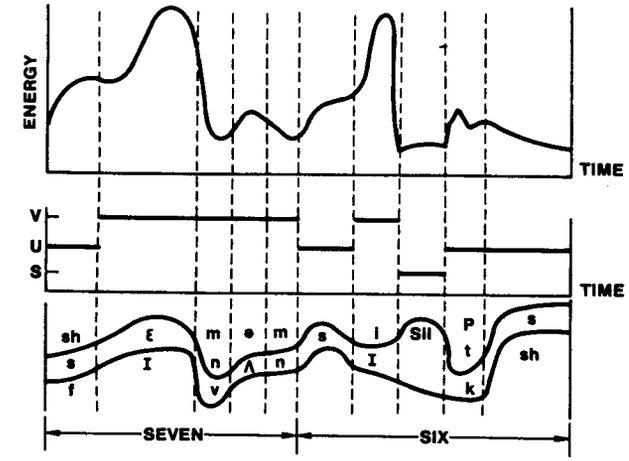



**Figure 2.35** Segmentation and labeling for word sequence "seven–six."

| | | | |
|---|---|---|---|
| phonemes: | /b/–/aʸ/–/ə/ | /w/–/ɝ/–/sil/–/j/–/l/–/n/ | /h/–/a/–/r/–/sil/–/d/ |
| ARPABET: | B –AY–AX + | W –ER–SIL–J –IH–N + | HH–AA–R–SIL–D |
| words: | BY | WORKING | HARD |

Two other examples of acoustic-phonetic segmentation and labeling are given in Figures 2.35 and 2.36. Shown in these figures are the energy contour of the speech signal, the voiced-unvoiced-silence classification over time, the segmentation points, and the lattice of phonetic units. The "proper" decoding of the lattice corresponding to the spoken word is shown as the phonetic units enclosed within the solid heavy lines. For the example of Figure 2.35 (the digit sequence "seven-six"), we see that although most top phoneme candidate errors are within the same sound class (e.g., /sh/ instead of /s/), some errors are between classes (e.g., /m/ instead of /v/). For decoding into digits, such cross-class errors are usually of little significance.

For the example of Figure 2.36 (the word sequence "did you"), the decoding into phonetic units is only the first step in a difficult decoding problem, because the basic speech sounds of the words "did" and "you" are phonologically changed in context from D-IH-D-Y-UW to D-IH-J-UH. This phonological effect exacerbates the problem of acoustic phonetic decoding even further than the insertion/deletion/substitution problems mentioned earlier.

### 2.5.1.4 Issues in Acoustic Phonetic Approach

Many problems are associated with the acoustic-phonetic approach to speech recognition. These problems, in many ways, account for the lack of success in practical speech-recognition systems. Among these are the following:

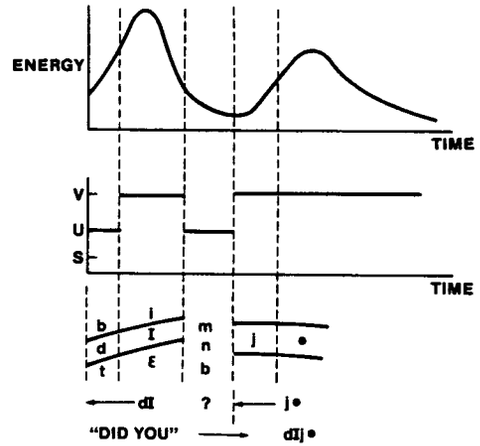



**Figure 2.36** Segmentation and labeling for word sequence "did you."

1. The method requires extensive knowledge of the acoustic properties of phonetic units. (Recall that the existence of phonetic units is assumed a priori in the acoustic-phonetic approach. Knowledge of acoustic properties of these phonetic units often is established in an a posteriori manner.) This knowledge is, at best incomplete, and at worst totally unavailable for all but the simplest of situations (e.g., steady vowels).
2. The choice of features is made mostly based on ad hoc considerations. For most systems the choice of features is based on intuition and is not optimal in a well-defined and meaningful sense.
3. The design of sound classifiers is also not optimal. Ad hoc methods are generally used to construct binary decision trees. More recently classification and regression tree (CART) methods have been used to make the decision trees more robust [10]. However, since the choice of features is most likely to be suboptimal, optimal implementation of CART is rarely achieved.
4. No well-defined, automatic procedure exists for tuning the method (i.e., adjusting decision thresholds, etc.) on real, labeled speech. In fact, there is not even an ideal way of labeling the training speech in a manner consistent and agreed on uniformly by a wide class of linguistic experts.

Because of all these problems, the acoustic-phonetic method of speech recognition remains an interesting idea but one that needs much more research and understanding before it can be used successfully in actual speech-recognition problems.



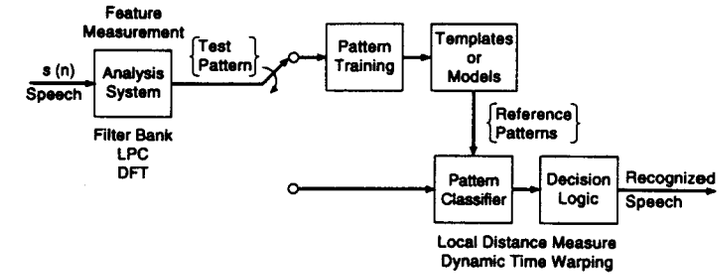



**Figure 2.37** Block diagram of pattern-recognition speech recognizer.

### 2.5.2 Statistical Pattern-Recognition Approach to Speech Recognition

A block diagram of a canonic pattern-recognition approach to speech recognition is shown in Figure 2.37. The pattern-recognition paradigm has four steps, namely:

1. Feature measurement, in which a sequence of measurements is made on the input signal to define the "test pattern." For speech signals the feature measurements are usually the output of some type of spectral analysis technique, such as a filter bank analyzer, a linear predictive coding analysis, or a discrete Fourier transform (DFT) analysis.
2. Pattern training, in which one or more test patterns corresponding to speech sounds of the same class are used to create a pattern representative of the features of that class. The resulting pattern, generally called a reference pattern, can be an exemplar or template, derived from some type of averaging technique, or it can be a model that characterizes the statistics of the features of the reference pattern.
3. Pattern classification, in which the unknown test pattern is compared with each (sound) class reference pattern and a measure of similarity (distance) between the test pattern and each reference pattern is computed. To compare speech patterns (which consist of a sequence of spectral vectors), we require both a local distance measure, in which local distance is defined as the spectral "distance" between two well-defined spectral vectors, and a global time alignment procedure (often called a dynamic time warping algorithm), which compensates for different rates of speaking (time scales) of the two patterns.
4. Decision logic, in which the reference pattern similarity scores are used to decide which reference pattern (or possibly which sequence of reference patterns) best matches the unknown test pattern.

The factors that distinguish different pattern-recognition approaches are the types of feature

measurement, the choice of templates or models for reference patterns, and the method used to create reference patterns and classify unknown test patterns.

The remaining chapters of this book will discuss all aspects of the model shown in Figure 2.37. The general strengths and weaknesses of the pattern recognition model include the following:

1. The performance of the system is sensitive to the amount of training data available for creating sound class reference patterns; generally the more training, the higher the performance of the system for virtually any task.
2. The reference patterns are sensitive to the speaking environment and transmission characteristics of the medium used to create the speech; this is because the speech spectral characteristics are affected by transmission and background noise.
3. No speech-specific knowledge is used explicitly in the system; hence, the method is relatively insensitive to choice of vocabulary words, task, syntax, and task semantics.
4. The computational load for both pattern training and pattern classification is generally linearly proportional to the number of patterns being trained or recognized; hence, computation for a large number of sound classes could and often does become prohibitive.
5. Because the system is insensitive to sound class, the basic techniques are applicable to a wide range of speech sounds, including phrases, whole words, and subword units. Hence we will see how a basic set of techniques developed for one sound class (e.g., words) can generally be directly applied to different sound classes (e.g., subword units) with little or no modifications to the algorithms.
6. It is relatively straightforward to incorporate syntactic (and even semantic) constraints directly into the pattern-recognition structure, thereby improving recognition accuracy and reducing computation.

### 2.5.3 Artificial Intelligence (AI) Approaches to Speech Recognition

The basic idea of the artificial intelligence approach to speech recognition is to compile and incorporate knowledge from a variety of knowledge sources and to bring it to bear on the problem at hand. Thus, for example, the AI approach to segmentation and labeling would be to augment the generally used acoustic knowledge with phonemic knowledge, lexical knowledge, syntactic knowledge, semantic knowledge, and even pragmatic knowledge. To be more specific, we first define these different knowledge sources:

- acoustic knowledge—evidence of which sounds (predefined phonetic units) are spoken on the basis of spectral measurements and presence or absence of features
- lexical knowledge—the combination of acoustic evidence so as to postulate words as specified by a lexicon that maps sounds into words (or equivalently decomposes words into sounds)
- syntactic knowledge—the combination of words to form grammatically correct strings (according to a language model) such as sentences or phrases



- semantic knowledge—understanding of the task domain so as to be able to validate sentences (or phrases) that are consistent with the task being performed, or which are consistent with previously decoded sentences
- pragmatic knowledge—inference ability necessary in resolving ambiguity of meaning based on ways in which words are generally used.

To illustrate the correcting and constraining power of these knowledge sources, consider the following sentences:

1. Go to the refrigerator and get me a book.
2. The bears killed the rams.
3. Power plants colorless happily old.
4. Good ideas often run when least expected.

The first sentence is syntactically meaningful but semantically inconsistent. The second sentence can be interpreted in at least two pragmatically different ways, depending on whether the context is an event in a jungle or the description of a football game between two teams called the "bears" and the "rams." The third sentence is syntactically unacceptable and semantically meaningless. The fourth sentence is semantically inconsistent and can trivially be corrected by changing the word *run* to *come*, a slight phonetic difference.

The word-correcting capability of higher-level knowledge sources is illustrated in Figure 2.38, which shows the word error probability of a recognizer both with and without syntactic constraints, as a function of a "deviation" parameter sigma. As the deviation parameter gets larger, the word error probability increases for both cases; however, without syntax the word error probability rapidly leads to 1.0, but with syntax it increases gradually with increases in the noise parameter.

There are several ways to integrate knowledge sources within a speech recognizer. Perhaps the most standard approach is the "bottom-up" processor (Figure 2.39), in which the lowest-level processes (e.g., feature detection, phonetic decoding) precede higher-level processes (lexical decoding, language model) in a sequential manner so as to constrain each stage of the processing as little as possible. An alternative is the so-called "top-down" processor, in which the language model generates word hypotheses that are matched against the speech signal, and syntactically and semantically meaningful sentences are built up on the basis of the word match scores. Figure 2.40 shows a system that is often implemented in the top-down mode by integrating the unit matching, lexical decoding, and syntactic analyses modules into a consistent framework. (This system will be discussed extensively in the chapter on large-vocabulary continuous-speech recognition.)

A third alternative is the so-called blackboard approach, as illustrated in Figure 2.41. In this approach, all knowledge sources (KS) are considered independent; a hypothesis-and-test paradigm serves as the basic medium of communication among KSs; each KS is data driven, based on the occurrence of patterns on the blackboard that match the templates specified by the KS; the system activity operates asynchronously; assigned cost and utility considerations and an overall ratings policy to combine and propagate ratings across all levels. The blackboard approach was extensively studied at CMU in the 1970s [11].

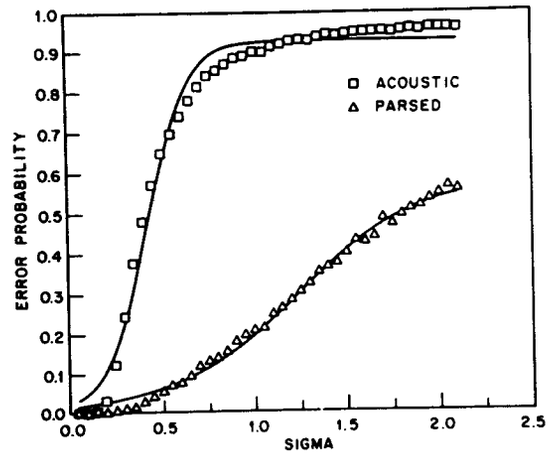



**Figure 2.38** Illustration of the word correction capability of syntax in speech recognition (after Rabiner and Levinson [1]).

### 2.5.4 Neural Networks and Their Application to Speech Recognition

A variety of knowledge sources need to be established in the AI approach to speech recognition. Therefore, two key concepts of artificial intelligence are automatic knowledge acquisition (learning) and adaptation. One way in which these concepts have been implemented is via the neural network approach. In this section, we discuss the motivation for why people have studied neural networks and how they have been applied to speech-recognition systems.

Figure 2.42 shows a conceptual block diagram of a speech understanding system loosely based on a model of speech perception in human beings. The acoustic input signal is analyzed by an "ear model" that provides spectral information (over time) about the signal and stores it in a sensory information store. Other sensory information (e.g., from vision or touch) is available in the sensory information store and is used to provide several "feature-level" descriptions of the speech. Both long-term (static) and short-term (dynamic) memory are available to the various feature detectors. Finally, after several stages of refined feature detection, the final output of the system is an interpretation of the information in the acoustic input.

The system of Figure 2.42 is meant to model the human speech understanding system. The auditory analysis is based loosely on our understanding of the acoustic processing in the ear. The various feature analyses represent processing at various levels in the neural pathways to the brain. The short- and long-term memory provide external control of the neural processes in ways that are not well understood. The overall form of the model is that of a feed forward connectionist network—that is, a neural net. To better explain the strengths and limitations of neural networks, we now give a brief introduction to the issues



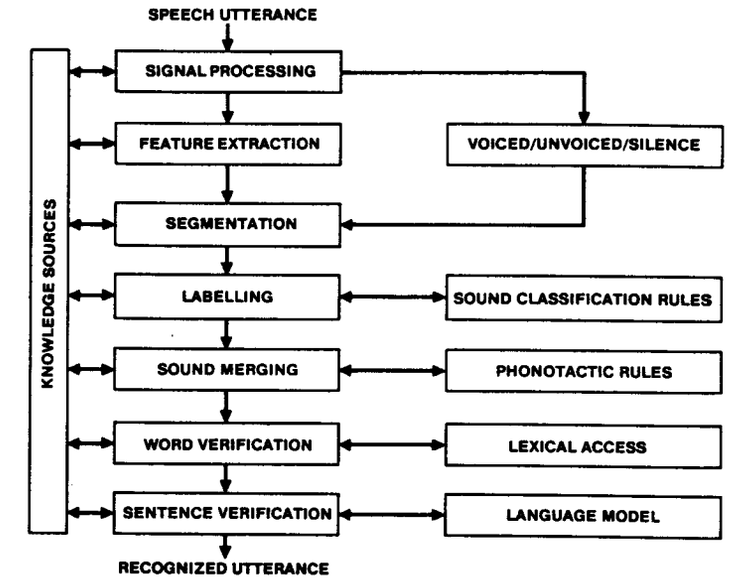



**Figure 2.39** A bottom-up approach to knowledge integration for speech recognition.

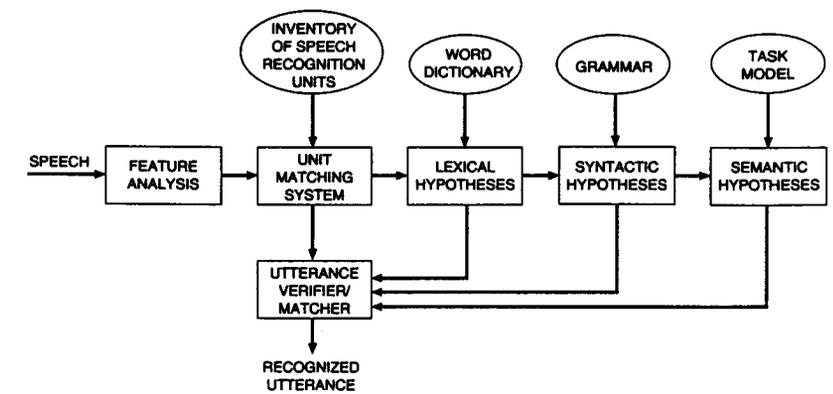



**Figure 2.40** A top-down approach to knowledge integration for speech recognition.

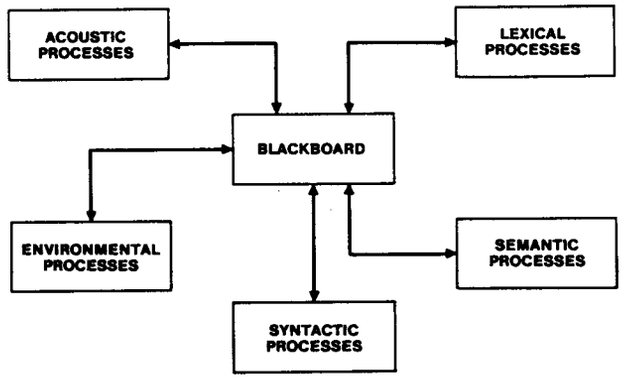



**Figure 2.41** A blackboard approach to knowledge integration for speech recognition (after Lesser et al. [11]).

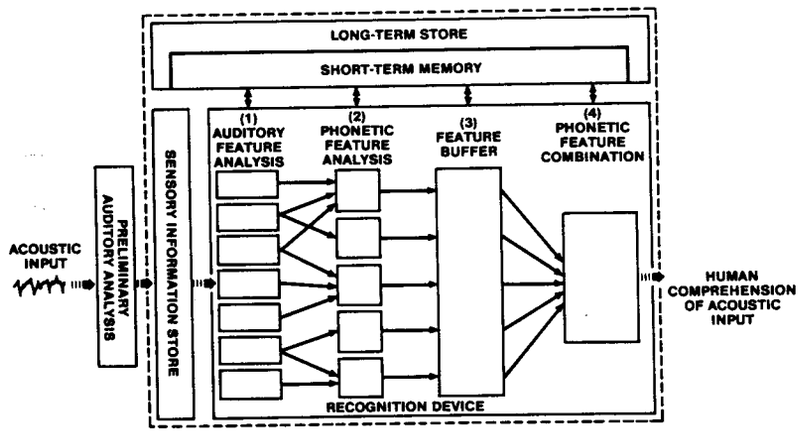



**Figure 2.42** Conceptual block diagram of a human speech understanding system.

in the theory and implementations of neural networks. Then we return to some practical proposals for how neural networks could implement actual speech recognizers.

### 2.5.4.1 Basics of Neural Networks

A neural network, which is also called a connectionist model, a neural net, or a parallel distributed processing (PDP) model, is basically a dense interconnection of simple, nonlinear, computation elements of the type shown in Figure 2.43. It is assumed that there



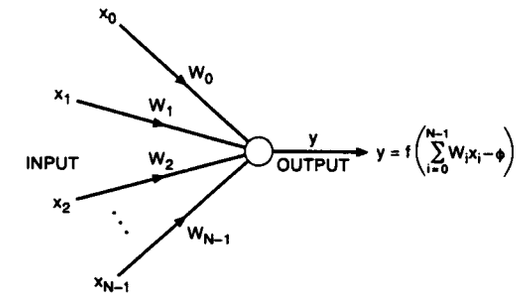



**Figure 2.43** Simple computation element of a neural network.

are $N$ inputs, labeled $x_1, x_2, \ldots, x_N$, which are summed with weights $w_1, w_2, \ldots, w_N$, thresholded, and then nonlinearly compressed to give the output $y$, defined as

$$y = f\left(\sum_{i=1}^{N} w_i x_i - \phi\right), \tag{2.1}$$

where $\phi$ is an internal threshold or offset, and $f$ is a nonlinearity of one of the types given below:

1. hard limiter

$$f(x) = \begin{cases} +1, & x \le 0 \\ -1, & x < 0 \end{cases} \tag{2.2}$$

or

2. sigmoid functions

$$f(x) = \tanh(\beta x), \quad \beta > 0 \tag{2.3}$$

or

$$f(x) = \frac{1}{1 + e^{-\beta x}}, \quad \beta > 0. \tag{2.4}$$

The sigmoid nonlinearities are used most often because they are continuous and differentiable.

The biological basis of the neural network is a model by McCullough and Pitts [12] of neurons in the human nervous system, as illustrated in Figure 2.44. This model exhibits all the properties of the neural element of Figure 2.43, including excitation potential thresholds for neuron firing (below which there is little or no activity) and nonlinear amplification, which compresses strong input signals.

### 2.5.4.2 Neural Network Topologies

There are several issues in the design of so-called artificial neural networks (ANNs), which model various physical phenomena, where we define an ANN as an arbitrary connection of

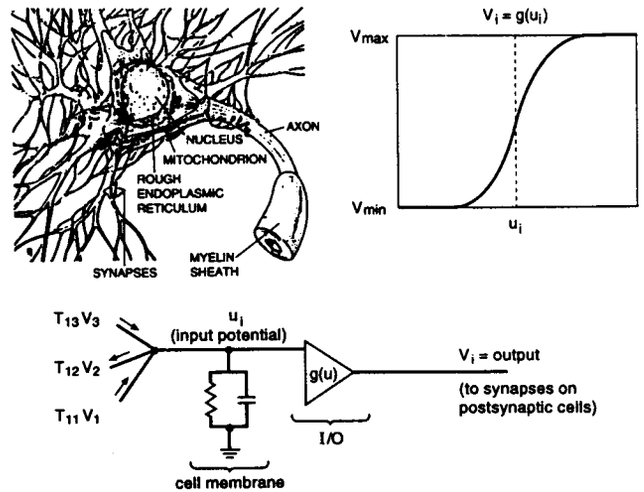



**Figure 2.44** McCullough-Pitts model of neurons (after McCullough and Pitts [12]).

simple computational elements of the type shown in Figure 2.43. One key issue is *network topology*—that is, how the simple computational elements are interconnected. There are three standard and well known topologies:

- single/multilayer perceptrons
- Hopfield or recurrent networks
- Kohonen or self-organizing networks

In the single/multilayer perceptron, the outputs of one or more simple computational elements at one layer form the inputs to a new set of simple computational elements of the next layer. Figure 2.45 shows a single-layer perceptron and a three-layer perceptron. The single-layer perceptron has $N$ inputs connected to $M$ outputs in the output layer. The three-layer perceptron has two hidden layers between the input and output layers. What distinguishes the layers of the multilayer perceptron is the nonlinearity at each layer that enables the mapping between the input and output variables to possess certain particular classification/discrimination properties. For example, it can be proven that a single-layer perceptron, of the type shown in Figure 2.45a, can separate static patterns into classes with class boundaries characterized by hyperplanes in the $(x_1, x_2, \ldots, x_N)$ space. Similarly, a multilayer perceptron, with at least one hidden layer, can realize an arbitrary set of decision regions in the $(x_1, \ldots, x_N)$ space. Thus, for example, if the inputs to a multilayer perceptron are the first two speech resonances ($F_1$ and $F_2$), the network can implement a set of decision regions that partition the $(F_1 - F_2)$ space into the 10 steady state vowels, as shown in Figure 2.46 [13].



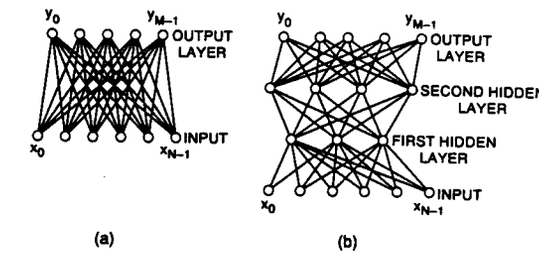



**Figure 2.45** Single-layer and three-layer perceptrons.

- **Perceptrons:**

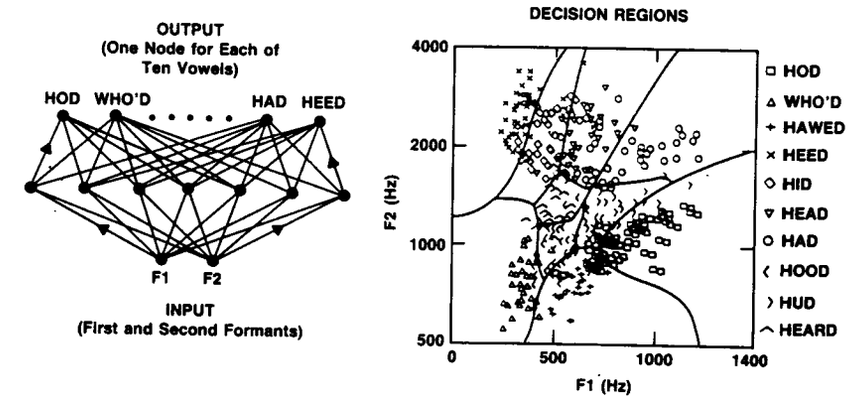



**Figure 2.46** A multilayer perceptron for classifying steady vowels based on $F_1$, $F_2$ measurements (after Lippmann [13]).

The Hopfield network is a *recurrent network* in which the input to each computational element includes both inputs as well as outputs. Thus with the input and output indexed by time, $x_i(t)$ and $y_i(t)$, and the weight connecting the $i^{th}$ node and the $j^{th}$ node denoted by $w_{ij}$, the basic equation for the $i^{th}$ recurrent computational element is

$$y_i(t) = f\left[x_i(t) + \sum_j w_{ij} y_j(t-1) - \phi\right] \tag{2.5}$$

and a recurrent network with $N$ inputs and $N$ outputs would have the form shown in Figure 2.47. The most important property of the Hopfield network is that when $w_{ij} = w_{ji}$ and when the recurrent computation (Eq. (2.5)) is performed asynchronously, for an arbitrary constant input, the network will eventually settle to a fixed point where $y_i(t) = y_i(t-1)$ for all $i$. These fixed relaxation points represent stable configurations of the network and can be used in applications that have a fixed set of patterns to be matched (e.g., printed letters)

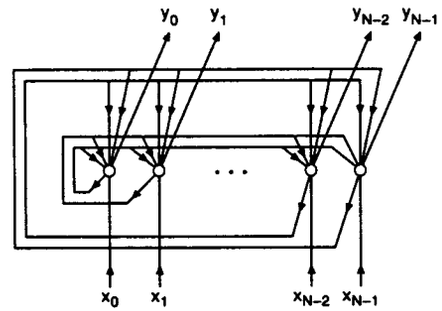



**Figure 2.47** Model of a recurrent neural network.

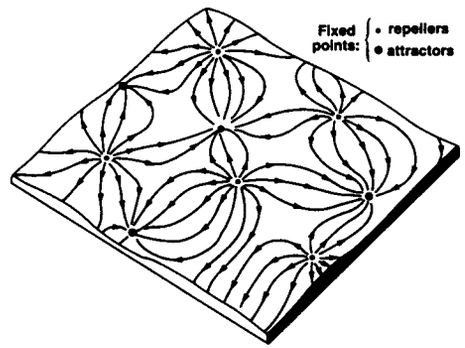



**Figure 2.48** A fixed point interpretation of the Hopfield network.

in the form of a content addressable or associative memory. A simple interpretation of the Hopfield network is shown in Figure 2.48, which shows that the recurrent network has a stable set of attractors and repellers, each forming a fixed point in the input space. Every input vector, $\mathbf{x}$, is either "attracted" to one of the fixed points or "repelled" from another of the fixed points. The strength of this type of network is its ability to correctly classify "noisy" versions of the patterns that form the stable fixed points.

The third popular type of neural network topology is the Kohonen, self-organizing feature map, which is a clustering procedure for providing a codebook of stable patterns in the input space that characterize an arbitrary input vector, by a small number of representative clusters. We defer a discussion of this type of network to the next chapter, where the ideas of vector quantization are presented in detail.

### 2.5.4.3 Network Characteristics

Four model characteristics must be specified to implement an arbitrary neural network:



1. number and type of inputs—The issues involved in the choice of inputs to a neural network are similar to those involved in the choice of features for any pattern-classification system. They must provide the information required to make the decision required of the network.
2. connectivity of the network—This issue involves the size of the network—that is, the number of hidden layers and the number of nodes in each layer between input and output. There is no good rule of thumb as to how large (or small) such hidden layers must be. Intuition says that if the hidden layers are large, then it will be difficult to train the network (i.e., there will be too many parameters to estimate). Similarly, if the hidden layers are too small, the network may not be able to accurately classify all the desired input patterns. Clearly practical systems must balance these two competing effects.
3. choice of offset—The choice of the threshold, $\phi$, for each computational element must be made as part of the training procedure, which chooses values for the interconnection weights ($w_{ij}$) and the offset, $\phi$.
4. choice of nonlinearity—Experience indicates that the exact choice of the nonlinearity, $f$, is not very important in terms of the network performance. However, $f$ must be continuous and differentiable for the training algorithm to be applicable.

### 2.5.4.4 Training of Neural Network Parameters

To completely specify a neural network, values for the weighting coefficients and the offset threshold for each computation element must be determined, based on a labeled set of training data. By a labeled training set of data, we mean an association between a set of $Q$ input vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_Q$ and a set of $Q$ output vectors $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_Q$ where $\mathbf{x}_1 \Rightarrow \mathbf{y}_1, \mathbf{x}_2 \Rightarrow \mathbf{y}_2, \ldots, \mathbf{x}_Q \Rightarrow \mathbf{y}_Q$. For multilayer perceptrons a simple iterative, convergent procedure exists for choosing a set of parameters whose value asymptotically approaches a stationary point with a certain optimality property (e.g., a local minimum of the mean squared error, etc.). This procedure, called back propagation learning, is a simple stochastic gradient technique. For a simple, single-layer network, the training algorithm can be realized via the following convergence steps:

**Perceptron Convergence Procedure**

1. **Initialization**: At time $t = 0$, set $w_{ij}(0)$, $\phi_j$ to small random values (where $w_{ij}$ are the weighting coefficients connecting $i^{\text{th}}$ input node and $j^{\text{th}}$ output node and $\phi_j$ is the offset to a particular computational element, and the $w_{ij}$ are functions of time).
2. **Acquire Input**: At time $t$, obtain a *new* input $\mathbf{x} = \{x_1, x_2, \ldots, x_N\}$ with the desired output, $\mathbf{y}^x = \{y_1^x, y_2^x, \ldots, y_M^x\}$.
3. **Calculate Output**:

$$y_j = f\left(\sum_{i=1}^{N} w_{ij}(t)x_i - \phi_j\right).$$

4. **Adapt Weights**: Update the weights as

$$w_{ij}(t+1) = w_{ij}(t) + \mathcal{T}(t)\,[y_j^x - y_j] \cdot x_i$$

where the "step size" $\mathcal{T}(t)$ satisfies the constraints:

**a.** $$\lim_{T\to\infty} \sum_{t=1}^{T} \mathcal{T}(t) = \infty$$

**b.** $$\lim_{T\to\infty} \sum_{t=1}^{T} \mathcal{T}^2(t) < \infty$$

That is, compute the gradient of the error $\sum_{j=1}^{M}(y_j^x - y_j)^2$ in the direction of the weight $w_{ij}(t)$. (A conventional choice of $\mathcal{T}(t)$ is $1/t$.)

5. **Iteration**: Iterate steps 2–4 until:

$$w_{ij}(t+1) = w_{ij}(t), \qquad \forall i,\ t,\ j.$$

The perceptron convergence procedure is a slow, methodical procedure for estimating the coefficients of a system (a classifier as well as a neural network) based on a mean squared error criterion and has been extensively studied for several decades. The algorithm is simple and is guaranteed to converge, in probability, under a restricted set of conditions on $\mathcal{T}(t)$. However, its speed of convergence in many cases is not sufficiently fast. Alternative procedures for estimating neural network coefficients have been used with varying degrees of success.

#### 2.5.4.5 Advantages of Neural Networks

Neural networks have been given serious consideration for a wide range of problems (including speech recognition) for several reasons. These include the following:

1. They can readily implement a massive degree of parallel computation. Because a neural net is a highly parallel structure of simple, identical, computational elements, it should be clear that they are prime candidates for massively parallel (analog or digital) computation.
2. They intrinsically possess a great deal of robustness or fault tolerance. Since the "information" embedded in the neural network is "spread" to every computational element within the network, this structure is inherently among the least sensitive of networks to noise or defects within the structure.
3. The connection weights of the network need not be constrained to be fixed; they can be adapted in real time to improve performance. This is the basis of the concept of adaptive learning, which is inherent in the neural network structure.
4. Because of the nonlinearity within each computational element, a sufficiently large neural network can approximate (arbitrarily closely) any nonlinearity or nonlinear dynamical system. Hence neural networks provide a convenient way of implementing nonlinear transformations between arbitrary inputs and outputs and are often more efficient than alternative physical implementations of the nonlinearity.



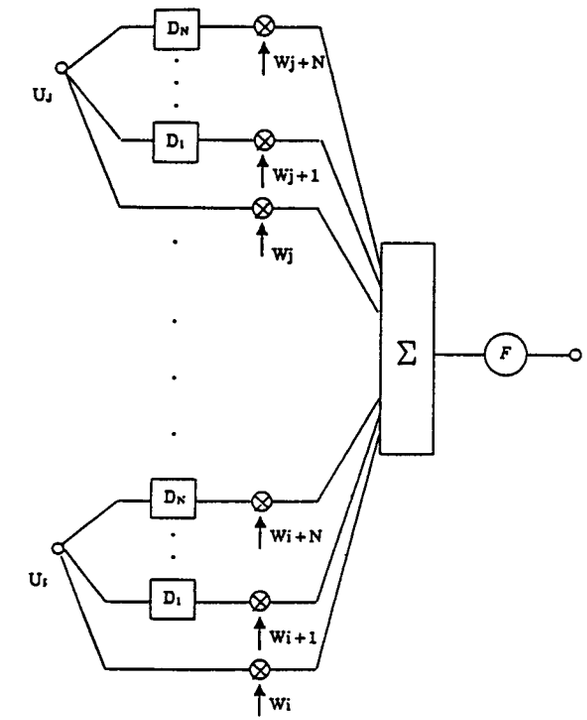



**Figure 2.49** The time delay neural network computational element (after Waibel et al. [14]).

#### 2.5.4.6 Neural Network Structures for Speech Recognition

Conventional artificial neural networks are structured to deal with static patterns. As discussed throughout this chapter, speech is inherently dynamic in nature. Hence, some modifications to the simple structures discussed in the previous sections are required for all but the simplest of problems. There is no known correct or proper way of handling speech dynamics within the framework already discussed; however, several reasonable structures have been proposed and studied and we will point out a few such structures in this section.

Perhaps the simplest neural network structure that incorporates speech pattern dynamics is the time delay neural network (TDNN) computation element shown in Figure 2.49 [14]. This structure extends the input to each computational element to include $N$ speech frames (i.e., spectral vectors that cover a duration of $N\Delta$ seconds, where $\Delta$ is the time separation between adjacent speech spectra). By expanding the input to $N$ frames (where $N$ is on the order of 15), various types of acoustic-phonetic detectors become practical via

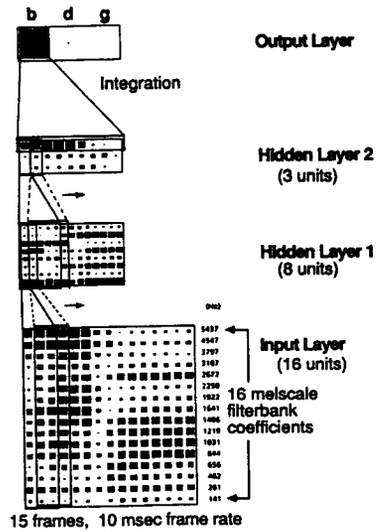



**Figure 2.50** A TDNN architecture for recognizing /b/, /d/ and /g/ (after Waibel et al. [14]).

the TDNN. For example, Figure 2.50 shows a TDNN network with two hidden layers that has been used to distinguish /b/ from /d/ from /g/.

A somewhat different neural network architecture for speech recognition, which combines the concept of a matched filter with a conventional neural network to account for the dynamic within speech, is shown in Figure 2.51 [15]. The "acoustic features" of the speech are estimated via conventional neural network architectures; the pattern classifier takes the detected acoustic feature vectors (delayed appropriately) and convolves them with filters "matched" to the acoustic features and sums up the results over time. At the appropriate time (corresponding to the end of some speech unit to be detected or recognized), the output units indicate the presence of the speech.

To illustrate how the network of Figure 2.51 could be used for speech recognition, consider, as shown in Figure 2.52, a "sound" to be recognized that is characterized (in some type of sound lexicon) as the sequence of acoustic features $(\alpha, \epsilon, \delta, \beta, \gamma)$. Assume that this sound is the input to an appropriately designed network of the type shown in Figure 2.51, and the input is as shown in the first line of Figure 2.52. When the acoustic feature $\alpha$ is detected (as indicated by the line labeled $D_\alpha(t)$), it is delayed and then convolved with a matched filter with a long time spreading function, yielding the signal $D_\alpha(t-\tau) \star P_\alpha(\tau)$ as shown in the next line of the figure. Similarly acoustic features $\varepsilon, \delta, \beta$, and $\gamma$ are detected, delayed appropriately, and convolved with the appropriate matched filter, as shown in the



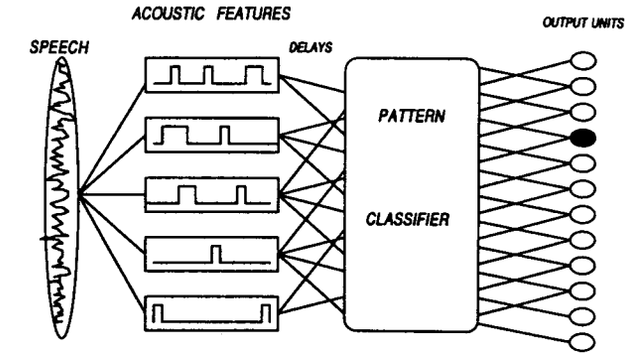



**Figure 2.51** A combination neural network and matched filter for speech recognition (after Tank & Hopfield [15]).

succeeding lines in Figure 2.52. Finally, at the end of the sequence, the convolved outputs are summed up and yield a large value, indicating the recognition of the appropriate sound.

Finally, yet a third way of integrating temporal information into a neural network is shown in Figure 2.53. This network is called a hidden control neural network (HCNN) [16] and uses the time varying control, **c**, as a supplement to the standard input, **x**, to allow the network properties (input-output relations) to change over time in a well-prescribed manner.

## 2.6 SUMMARY

In this chapter we have presented a brief discussion of the basic speech-production/perception mechanism in human beings, and we have illustrated how we can exploit the so-called acoustic-phonetic properties of speech to identify basic sounds. Acoustic-phonetics is the broad underpinning of all speech-recognition work. Differences in approach lie in the degree of reliance on how much acoustic-phonetics can be used in the recognition process. At one extreme is the class of acoustic-phonetic recognition methods that places total reliance on the acoustic-phonetic mapping; at the other extreme is the class of pattern-recognition approaches that do not make a priori assumptions on the phonetic characteristics and instead choose to "relearn" the appropriate acoustic-phonetic mapping for specific word vocabularies and tasks via an appropriately designed training set. Finally, there is the hybrid class of artificial intelligence approaches that exploit, in various degrees, aspects of both extreme views of the speech-recognition process. We also discussed the fundamentals of neural networks, which can be considered a separate structural approach, as well as a new pattern classifier design, with potential to benefit or advance all three classical approaches described in this chapter.

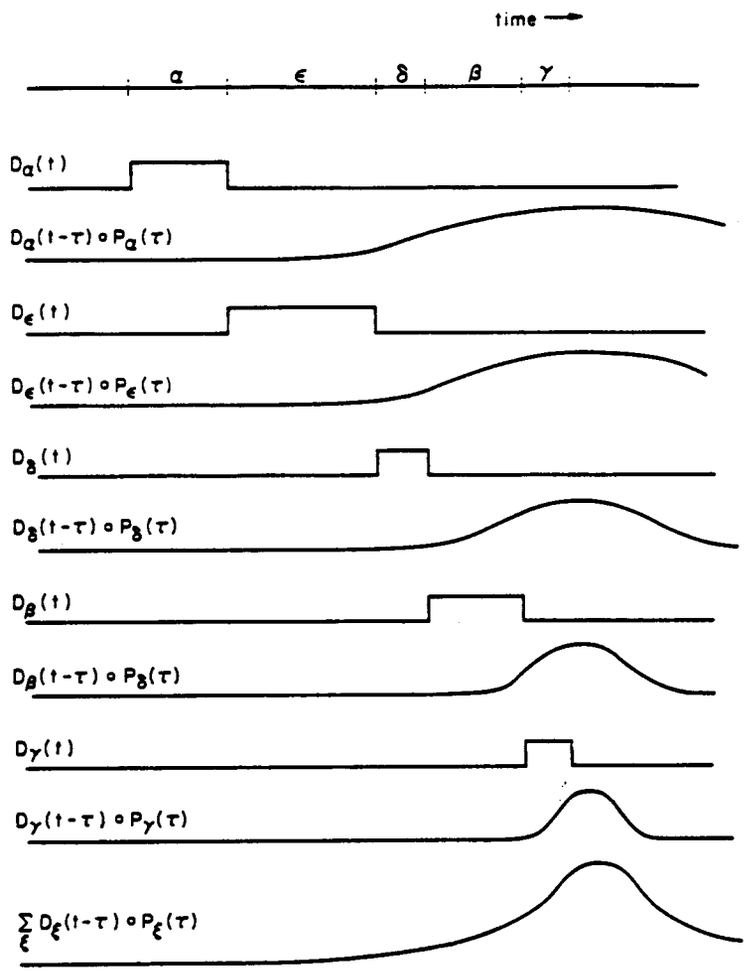



**Figure 2.52** Example illustrating the combination of a neural network and a set of matched filters (after Tank & Hopfield [15]).

In the remainder of this book we will primarily discuss aspects of the pattern-recognition approach to speech recognition. However, the alternative methods will always be lurking just below the surface of our discussion.



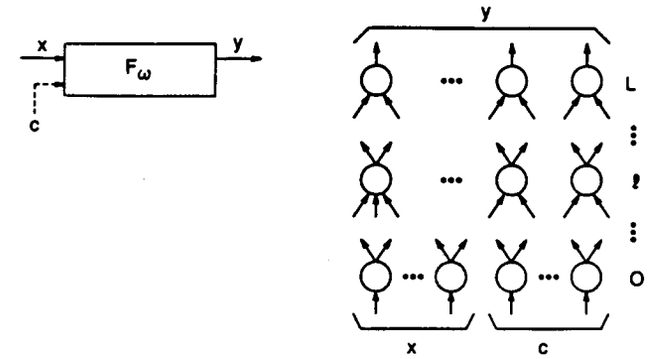



**Figure 2.53** The hidden control neural network (after Levin [16]).

## REFERENCES


[1] L.R. Rabiner and S.E. Levinson, "Isolated and Connected Word Recognition—Theory and Selected Applications," *IEEE Trans. Communications*, COM-29 (5): 621–659, May 1981.

[2] J.L. Flanagan, C.H. Coker, L.R. Rabiner, R.W. Schafer, and N. Umeda, "Synthetic Voices for Computers," *IEEE Spectrum*, 7 (10): 22–45, October 1970.

[3] J.L. Flanagan, *Speech Analysis, Synthesis, and Perception*, 2nd ed., Springer-Verlag, New York, 1972.

[4] K. Ishizaka and J.L. Flanagan, "Synthesis of Voiced Sounds from a Two-Mass Model of the Vocal Cords," *Bell System Tech. J.*, 50 (6): 1233–1268, July–Aug., 1972.

[5] B.S. Atal and S.L. Hanauer, "Speech Analysis and Synthesis by Linear Prediction of the Speech Wave," *J. Acoust. Soc. Am.*, 50 (2): 637-655, August 1971.

[6] J.E. Shoup, "Phonological Aspects of Speech Recognition," 125–138, Ch. 6 in *Trends in Speech Recognition*, W. A. Lea, Ed., Prentice-Hall, Englewood Cliffs, NJ, 1980.

[7] G.E. Peterson and H.L. Barney, "Control Methods Used in a Study of the Vowels," *J. Acoust. Soc. Am.*, 24 (2): 175–194, March 1952.

[8] L.R. Rabiner, "Speech Synthesis by Rule—An Acoustic Domain Approach," PhD. thesis, MIT, Cambridge, MA, June 1967.

[9] A. Holbrook and G. Fairbanks, "Diphthong Formants and Their Movements," *J. Speech Hearing Res.*, 5 (1): 38–58, March 1962.

[10] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone, *Classification and Regression Trees*, Wadsworth & Brooks, Monterey, CA, 1984.

[11] V.R. Lesser, R.D. Fennell, L.D. Erman, and D.R. Reddy, "Organization of the Hearsay–II Speech Understanding System," *IEEE Trans. Acoustics, Speech, Signal Proc.*, ASSP-23 (1): 11–23, 1975.

[12] W.S. McCullough and W.H. Pitts, "A Logical Calculus of Ideas Immanent in Nervous

Activity," *Bull Math Biophysics*, 5: 115–133, 1943.

[13] R. Lippmann, "An Introduction to Computing with Neural Nets," *IEEE ASSP Mag.*, 4 (2): 4–22, April 1987.

[14] A. Weibel, T. Hanazawa, G. Hinton, K. Shikano, and K.J. Lang, "Phoneme Recognition Using Time Delay Neural Networks," *IEEE Trans. Acoustics, Speech, Signal Proc.*, ASSP-37: 328–339, 1989.

[15] D.W. Tank and J.J. Hopfield, "Neural Computation by Concentrating Information in Time," *Proc. Nat. Academy Sciences*, 84: 1896–1900, April 1987.

[16] E. Levin, "Word Recognition Using Hidden Control Neural Architecture," *Proc. ICASSP 90*, 433–436, Albuquerque, NM, April 1990.

Chapter 3

# SIGNAL PROCESSING AND ANALYSIS METHODS FOR SPEECH RECOGNITION

## 3.1 INTRODUCTION

As discussed in Chapter 1, a speech-recognition system, at its most elementary level, comprises a collection of algorithms drawn from a wide variety of disciplines, including statistical pattern recognition, communication theory, signal processing, combinatorial mathematics, and linguistics, among others. Although each of these areas is relied on to varying degrees in different recognizers, perhaps the greatest common denominator of all recognition systems is the signal-processing front end, which converts the speech waveform to some type of parametric representation (generally at a considerably lower information rate) for further analysis and processing. Because of the singular importance of signal-processing techniques to the understanding of how speech recognizers are designed and how they function, we devote this chapter to a discussion of the most commonly used techniques in this area.

A wide range of possibilities exists for parametrically representing the speech signal. These include the short time energy, zero crossing rates, level crossing rates, and other related parameters. Probably the most important parametric representation of speech is the short time spectral envelope, as discussed in Chapter 2. Spectral analysis methods are therefore generally considered as the core of the signal-processing front end in a speech-recognition system. In this chapter we discuss two dominant methods of spectral analysis—namely, the filter-bank spectrum analysis model, and the linear predictive coding (LPC) spectral analysis model. Also discussed in this chapter is the technique called vector

quantization, which is a procedure for encoding a continuous spectral representation by a "typical" spectral shape in a finite codebook (collection) of spectral shapes, thereby reducing the information rate of the signal processing even further. The technique of vector quantization can be applied to any spectral representation, including both the filter bank and LPC models.

Finally, we close with a brief discussion of an auditory signal-processing model that has been proposed as an alternative to both filter banks and LPC models for speech spectral analysis. The argument for such a model is that, because it is based on known properties of the human auditory system (i.e., a model of cochlea mechanics), it is inherently a better representation of the relevant spectral information than either a filter-bank or an LPC model, and furthermore it should be quite robust to noise and reverberation.

### 3.1.1 Spectral Analysis Models

To motivate our discussion and see how the signal-processing techniques fit into our canonic recognition system models, let us review the pattern-recognition model of Figure 3.1a and the acoustic-phonetic model of Figure 3.1b. The three basic steps in the pattern-recognition model are (1) parameter measurement (in which a test pattern is created), (2) pattern comparison, and (3) decision making. The function of the parameter measurement block is to represent the relevant acoustic events in the speech signal in terms of a compact, efficient set of speech parameters. Although the choice of which parameters to use is dictated by other considerations (e.g., computational efficiency, type of implementation, available memory), the way in which the chosen representation is computed is based strictly on signal-processing considerations. In a similar manner, in the acoustic-phonetic model of recognition, the first step in the processing is essentially identical to that used in the pattern-recognition approach—namely, parameter measurement—although the steps that follow are markedly different. Hence, it is clear that a good fundamental understanding of the way in which we use signal-processing techniques to implement the parameter-measurement phase of the recognizer is mandatory for understanding the strengths and shortcomings of the various approaches to speech recognition that have been proposed and studied in the literature.

As mentioned previously, the two most common choices of a signal-processing front end for speech recognition are a bank-of-filters model and an LPC model. The overall structure of the bank-of-filters model is shown in Figure 3.2. The speech signal, $s(n)$, (assumed to be in digital form throughout this book), is passed through a bank of $Q$ bandpass filters whose coverage spans the frequency range of interest in the signal (e.g., 100–3000 Hz for telephone-quality signals, 100–8000 Hz for broadband signals). The individual filters can and generally do overlap in frequency, as shown at the bottom of Figure 3.2. The output of the $i^{th}$ bandpass filter, $X_n(e^{j\omega_i})$ (where $\omega_i$ is the normalized frequency $2\pi f_i/F_s$, with $F_s$ the sampling frequency) is the short-time spectral representation of the signal $s(n)$, at time $n$, as seen through the $i^{th}$ bandpass filter with center frequency $\omega_i$. It can readily be seen that in the bank-of-filters model each bandpass filter processes the speech signal independently to produce the spectral representation $X_n$. The LPC analysis approach, as

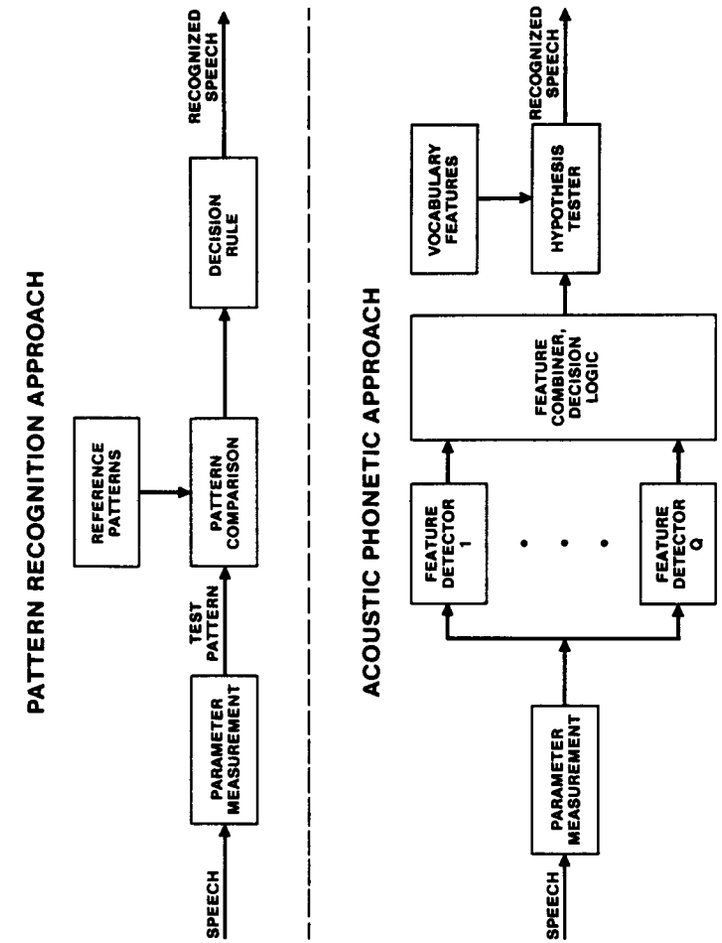



Figure 3.1 (a) Pattern recognition and (b) acoustic phonetic approaches to speech recognition.

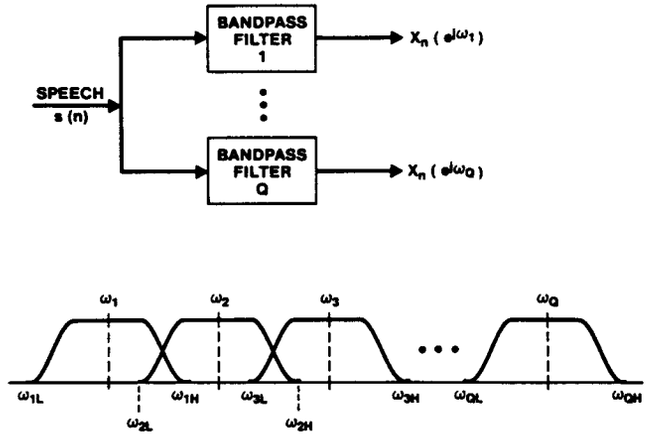



**Figure 3.2** Bank-of-filters analysis model.

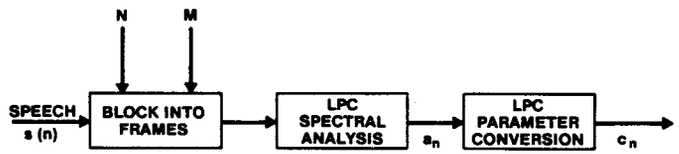



**Figure 3.3** LPC analysis model.

illustrated in Figure 3.3, performs spectral analysis on blocks of speech (speech frames) with an all-pole modeling constraint. This means that the resulting spectral representation $X_n(e^{j\omega})$ is constrained to be of the form $\sigma/A(e^{j\omega})$, where $A(e^{j\omega})$ is a $p^{\text{th}}$ order polynomial with $z$-transform

$$A(z) = 1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_p z^{-p}.$$

The order, $p$, is called the LPC analysis order. Thus the output of the LPC spectral analysis block is a vector of coefficients (LPC parameters) that specify (parametrically) the spectrum of an all-pole model that best matches the signal spectrum over the period of time in which the frame of speech samples was accumulated.

Although alternative signal-processing front-end processors have been proposed for speech-recognition systems, the filter-bank and LPC models have proven themselves to give the highest performance in practical speech-recognition systems. Thus, in this chapter, we will discuss these two analysis approaches in greater detail and show how they fit into the framework of the pattern-recognition and acoustic-phonetic approaches to recognition.



## 3.2 THE BANK-OF-FILTERS FRONT-END PROCESSOR

A block diagram of the canonic structure of a complete filter-bank front-end analyzer is given in Figure 3.4. The sampled speech signal, $s(n)$, is passed through a bank of $Q$ bandpass filters, giving the signals

$$s_i(n) = s(n) * h_i(n), \qquad 1 \le i \le Q \tag{3.1a}$$

$$= \sum_{m=0}^{M_i-1} h_i(m) s(n-m), \tag{3.1b}$$

where we have assumed that the impulse response of the $i^{\text{th}}$ bandpass filter is $h_i(m)$ with a duration of $M_i$ samples; hence, we use the convolution representation of the filtering operation to give an explicit expression for $s_i(n)$, the bandpass-filtered speech signal. Since the purpose of the filter-bank analyzer is to give a measurement of the energy of the speech signal in a given frequency band, each of the bandpass signals, $s_i(n)$, is passed through a nonlinearity, such as a full-wave or half-wave rectifier. The nonlinearity shifts the bandpass signal spectrum to the low-frequency band as well as creates high-frequency images. A lowpass filter is used to eliminate the high-frequency images, giving a set of signals, $u_i(n)$, $1 \le i \le Q$, which represent an estimate of the speech signal energy in each of the $Q$ frequency bands.

To more fully understand the effects of the nonlinearity and the lowpass filter, let us assume that the output of the $i^{\text{th}}$ bandpass filter is a pure sinusoid at frequency $\omega_i$, i.e.

$$s_i(n) = \alpha_i \sin(\omega_i n). \tag{3.2}$$

This assumption is valid for speech in the case of steady state voiced sounds when the bandwidth of the filter is sufficiently narrow so that only a single speech harmonic is passed by the bandpass filter. If we use a full-wave rectifier as the nonlinearity, that is,

$$\begin{aligned} f(s_i(n)) &= s_i(n) \quad \text{for } s_i(n) \ge 0 \\ &= -s_i(n) \quad \text{for } s_i(n) < 0. \end{aligned} \tag{3.3}$$

Then we can represent the nonlinearity output as

$$v_i(n) = f(s_i(n)) = s_i(n) \cdot w(n), \tag{3.4}$$

where

$$w(n) = \begin{cases} +1 & \text{if } s_i(n) \ge 0 \\ -1 & \text{if } s_i(n) < 0 \end{cases} \tag{3.5}$$

as illustrated in Figure 3.5(a)–(c). Since the nonlinearity output can be viewed as a modulation in time, as shown in Eq. (3.4), then in the frequency domain we get the result that

$$V_i(e^{j\omega}) = S_i(e^{j\omega}) \circledast W(e^{j\omega}), \tag{3.6}$$

where $V_i(e^{j\omega})$, $S_i(e^{j\omega})$, and $W(e^{j\omega})$ are the Fourier transforms of the signals $v_i(n)$, $s_i(n)$ and $w(n)$, respectively, and $\circledast$ is circular convolution. The spectrum $S_i(e^{j\omega})$ is a single impulse at $\omega_0 = \omega_i$, whereas the spectrum $W(e^{j\omega})$ is a set of impulses at the odd-harmonic

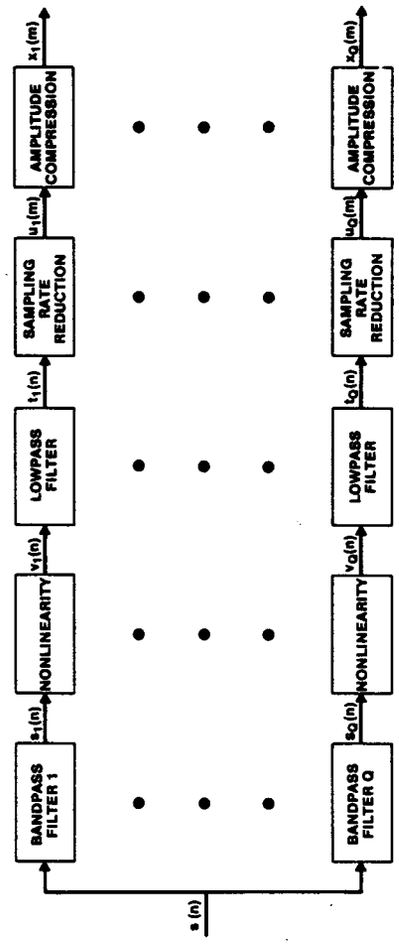



**Figure 3.4** Complete bank-of-filters analysis model.





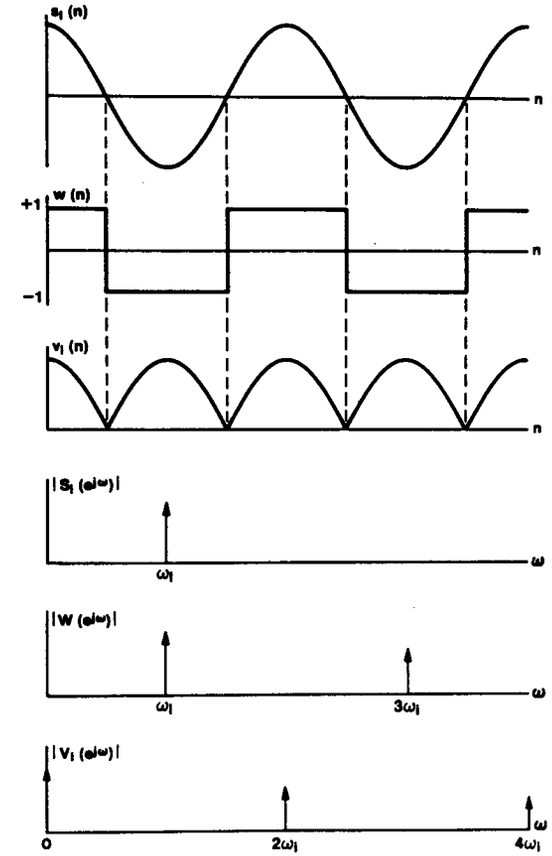



**Figure 3.5** Typical waveforms and spectra for analysis of a pure sinusoid in the filter-bank model.

frequencies $\omega_q = \omega_i q$, $q = 1, 3, \ldots, q_{\max}$. Hence the spectrum of $V_i(e^{j\omega})$ is an impulse at $\omega = 0$ and a set of smaller amplitude impulses at $\omega_q = \omega_i q$, $q = 2, 4, 6, \ldots$, as shown in Figure 3.5 (d)–(f). The effect of the lowpass filter is to retain the DC component of $V_i(e^{j\omega})$ and to filter out the higher frequency components due to the nonlinearity.

The above analysis, although only strictly correct for a pure sinusoid, is a reasonably good model for voiced, quasiperiodic speech sounds so long as the bandpass filter is not so wide that it has two or more strong signal harmonics. Because of the time-varying nature of the speech signal (i.e., the quasiperiodicity), the spectrum of the lowpass signal is not a pure

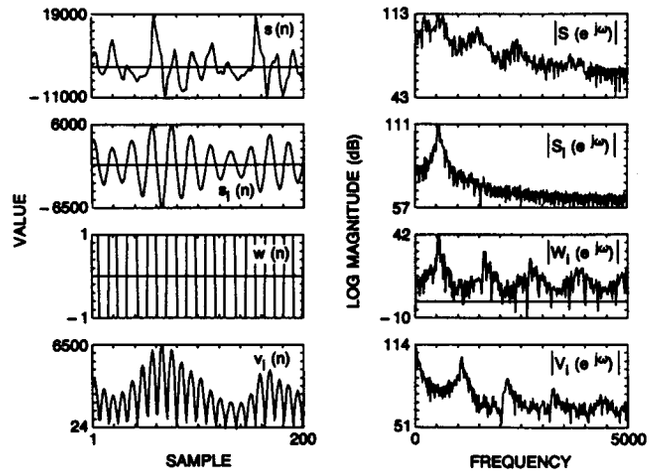



**Figure 3.6** Typical waveforms and spectra of a voice speech signal in the bank-of-filters analysis model.

DC impulse, but instead the information in the signal is contained in a low-frequency band around DC. Figure 3.6 illustrates typical waveforms of $s(n)$, $s_i(n)$, $w(n)$ and $v_i(n)$ for a brief (20 msec) section of voiced speech processed by a narrow bandwidth channel with center frequency of 500 Hz (sampling frequency for this example is 10,000 Hz). Also shown are the resulting spectral magnitudes for the four signals. It can be seen that $|S_i(e^{j\omega})|$ has most of its energy around 500 Hz ($\omega = 1000\pi$), whereas $|W_i(e^{j\omega})|$ (which is quasiperiodic) approximates an odd harmonic signal with peaks at 500, 1500, 2500 Hz. The resulting signal spectrum, $|V_i(e^{j\omega})|$, shows the desired low-frequency concentration of energy as well as the undesired spectral peaks at 1000 Hz, 2000 Hz, etc. The role of the final lowpass filter is to eliminate the undesired spectral peaks.

The bandwidth of the signal, $v_i(n)$, is related to the fastest rate of motion of speech harmonics in a narrow band and is generally acknowledged to be on the order of 20–30 Hz. Hence, the final two blocks of the canonic bank-of-filters model of Figure 3.4 are a sampling rate reduction box in which the lowpass-filtered signals, $t_i(n)$, are resampled at a rate on the order of 40–60 Hz (for economy of representation), and the signal dynamic range is compressed using an amplitude compression scheme (e.g., logarithmic encoding, $\mu$-law encoding).

Consider the design of a $Q = 16$ channel filter bank for a wideband speech signal where the highest frequency of interest is 8 kHz. Assume we use a sampling rate of $F_s = 20$ kHz on the speech data to minimize the effects of aliasing in the analog-to-digital conversion. The information (bit rate) rate of the raw speech signal is on the order of 240 kbits per second (20 k samples per second times 12 bits per sample). At the output of



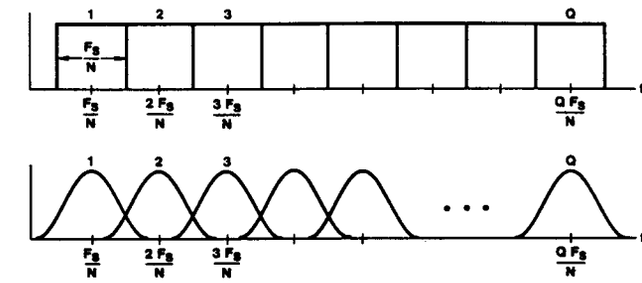



**Figure 3.7** Ideal (a) and realistic (b) set of filter responses of a $Q$-channel filter bank covering the frequency range $F_s/N$ to $(Q + \frac{1}{2})F_s/N$.

the analyzer, if we use a sampling rate of 50 Hz and we use a 7 bit logarithmic amplitude compressor, we get an information rate of 16 channels times 50 samples per second per channel times 7 bits per sample, or 5600 bits per second. Thus, for this simple example we have achieved about a 40-to-1 reduction in bit rate, and hopefully such a data reduction would result in an improved representation of the significant information in the speech signal.

### 3.2.1 Types of Filter Bank Used for Speech Recognition

The most common type of filter bank used for speech recognition is the uniform filter bank for which the center frequency, $f_i$, of the $i^{\text{th}}$ bandpass filter is defined as

$$f_i = \frac{F_s}{N} i, \qquad 1 \le i \le Q, \tag{3.7}$$

where $F_s$ is the sampling rate of the speech signal, and $N$ is the number of uniformly spaced filters required to span the frequency range of the speech. The actual number of filters used in the filter bank, $Q$, satisfies the relation

$$Q \le N/2 \tag{3.8}$$

with equality when the entire frequency range of the speech signal is used in the analysis. The bandwidth, $b_i$, of the $i^{\text{th}}$ filter, generally satisfies the property

$$b_i \ge \frac{F_s}{N} \tag{3.9}$$

with equality meaning that there is no frequency overlap between adjacent filter channels, and with inequality meaning that adjacent filter channels overlap. (If $b_i < F_s/N$, then certain portions of the speech spectrum would be missing from the analysis and the resulting speech spectrum would not be considered very meaningful.) Figure 3.7a shows a set of $Q$ ideal, non-overlapping, bandpass filters covering the range from $F_s/N(\frac{1}{2})$ to $(F_s/N)(Q + \frac{1}{2})$. Similarly Figure 3.7b shows a more realistic set of $Q$ overlapping filters covering approximately the same range.

The alternative to uniform filter banks is nonuniform filter banks designed according to some criterion for how the individual filters should be spaced in frequency. One commonly used criterion is to space the filters uniformly along a logarithmic frequency scale. (A logarithmic frequency scale is often justified from a human auditory perception point of view, as will be discussed in Chapter 4.) Thus for a set of $Q$ bandpass filters with center frequencies, $f_i$, and bandwidths, $b_i$, $1 \le i \le Q$, we set

$$b_1 = C \tag{3.10a}$$

$$b_i = \alpha b_{i-1}, \qquad 2 \le i \le Q \tag{3.10b}$$

$$f_i = f_1 + \sum_{j=1}^{i-1} b_j + \frac{(b_i - b_1)}{2}, \tag{3.11}$$

where $C$ and $f_1$ are the arbitrary bandwidth and center frequency of the first filter, and $\alpha$ is the logarithmic growth factor.

The most commonly used values of $\alpha$ are $\alpha = 2$, which gives an octave band spacing of adjacent filters, and $\alpha = 4/3$ which gives a 1/3 octave filter spacing. Consider the design of a four band, octave-spaced, non-overlapping filter bank covering the frequency band from 200 to 3200 Hz (with a sampling rate of 6.67 kHz). Figure 3.8a shows the ideal filters for this filter bank. Values for $f_1$ and $C$ of 300 Hz and 200 Hz are used, giving the following filter specifications:

Filter 1: $f_1 = 300$ Hz, $b_1 = 200$ Hz
Filter 2: $f_2 = 600$ Hz, $b_2 = 400$ Hz
Filter 3: $f_3 = 1200$ Hz, $b_3 = 800$ Hz
Filter 4: $f_4 = 2400$ Hz, $b_4 = 1600$ Hz

An example of a 12-band, 1/3-octave, ideal filter-bank specifications, covering the band from about 200 to 3200 Hz, is given in Figure 3.8b. For this example, $C = 50$ Hz, and $f_1 \simeq 225$ Hz.

An alternative criterion for designing a nonuniform filter bank is to use the critical band scale directly. The spacing of filters along the critical band is based on perceptual studies and is intended to choose bands that give equal contribution to speech articulation. The general shape of the critical band scale is given in Figure 3.9. The scale is close to linear for frequencies below about 1000 Hz (i.e., the bandwidth is essentially constant as a function $f$), and is close to logarithmic for frequencies above 1000 Hz (i.e., the bandwidth is essentially exponential as a function of $f$). Several variants on the critical band scale have been used, including the mel scale and the bark scale. The differences between these variants are small and are, for the most part, insignificant with regard to design of filter banks for speech-recognition purposes. For example, Figure 3.8c shows a 7-band critical-band filter-bank specification.

Other criteria for designing nonuniform filter banks have been proposed in the literature. For the most part, the uniform and nonuniform designs based on critical band scales have been the most widely used and studied filter-bank methods.



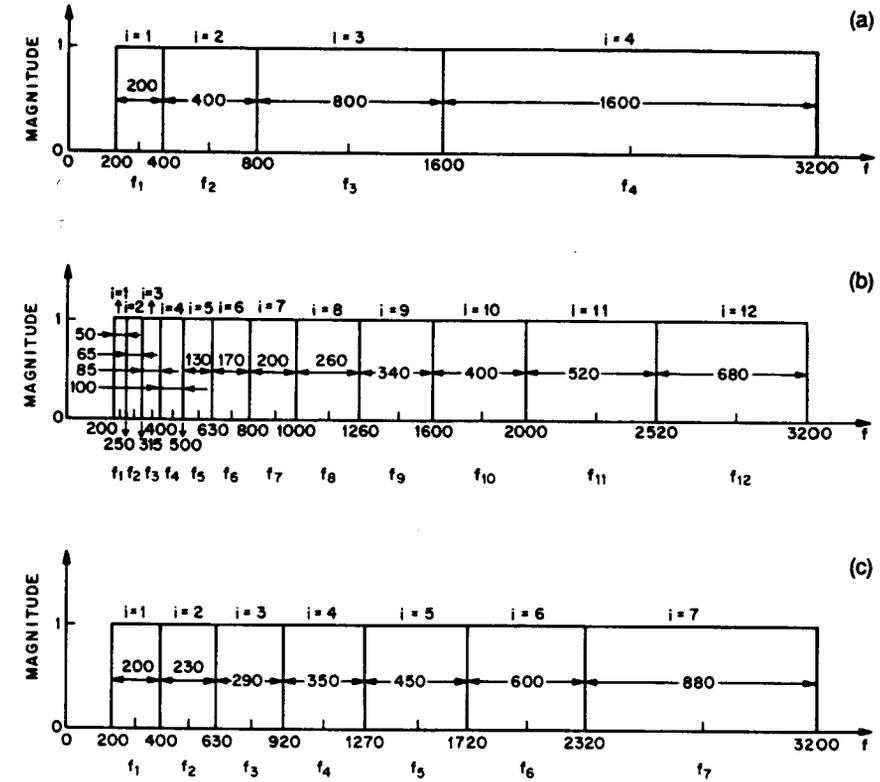



**Figure 3.8** Ideal specifications of a 4-channel octave band-filter bank (a), a 12-channel third-octave band filter bank (b), and a 7-channel critical band scale filter bank (c) covering the telephone bandwidth range (200–3200 Hz).

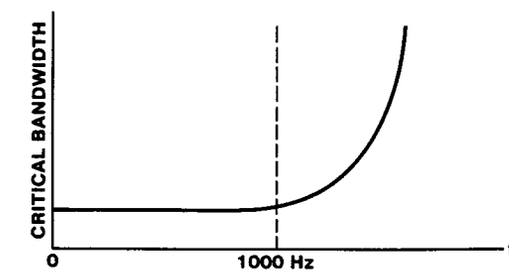



**Figure 3.9** The variation of bandwidth with frequency for the perceptually based critical band scale.

### 3.2.2 Implementations of Filter Banks

A filter bank can be implemented in several ways, depending on the method used to design the individual filters. Design methods for digital filters fall into two broad classes: (1) infinite impulse response (IIR) and (2) finite impulse response (FIR) methods. For IIR filters (also commonly called recursive filters in the literature), the most straightforward, and generally the most efficient implementation is to realize each individual bandpass filter as a cascade or parallel structure. (See Reference [1], pp. 40–46, for a discussion of such structures.)

For FIR filters there are several possible methods of implementing the bandpass filters in the filter bank. The most straightforward and the simplest implementation is the direct form structure. In this case, if we denote the impulse response for the $i^{\text{th}}$ channel as $h_i(n)$, $0 \le n \le L-1$, then the output of the $i^{\text{th}}$ channel, $x_i(n)$, can be expressed as the discrete, finite convolution of the input signal, $s(n)$, with the impulse response, $h_i(n)$, i.e.

$$x_i(n) = s(n) * h_i(n) \tag{3.12a}$$

$$= \sum_{m=0}^{L-1} h_i(m)s(n-m). \tag{3.12b}$$

The computation of Eq. (3.12) is iterated on each channel $i$, for $i = 1, 2, \ldots, Q$. The advantages of the convolutional, direct form structure are its simplicity and that it works for arbitrary $h_i(n)$. The disadvantage of this implementation is the high computational requirement. Thus for a $Q$-channel FIR filter bank, where each bandpass FIR filter has an impulse response of $L$ samples duration, we require

$$C_{\text{DFFIR}} = LQ \quad \cdot, + \qquad \text{(multiplication, addition)} \tag{3.13}$$

for a complete evaluation of $x_i(n)$, $i = 1, 2, \ldots, Q$, at a single value of $n$.

An alternative, less-expensive implementation can be derived for the case in which each bandpass filter impulse response can be represented as a fixed lowpass window, $w(n)$, modulated by the complex exponential, $e^{j\omega_i n}$—that is,

$$h_i(n) = w(n)e^{j\omega_i n}. \tag{3.14}$$

In this case Eq. (2.12b) becomes

$$x_i(n) = \sum_m w(m)e^{j\omega_i m}s(n-m)$$

$$= \sum_m s(m)\, w(n-m)e^{j\omega_i(n-m)}$$

$$= e^{j\omega_i n}\sum_m s(m)w(n-m)e^{-j\omega_i m} \tag{3.15a}$$

$$= e^{j\omega_i n}S_n(e^{j\omega_i}), \tag{3.15b}$$

where $S_n(e^{j\omega_i})$ is the short-time Fourier transform of $s(n)$ at frequency $\omega_i = 2\pi f_i$. The importance of Eq. (3.15) is that efficient procedures often exist for evaluating the short-



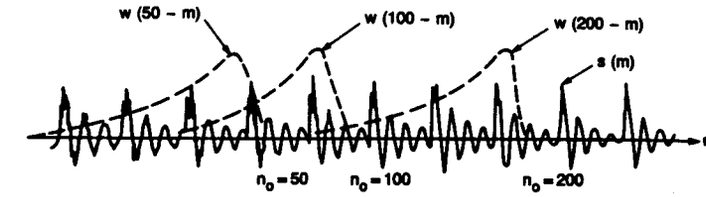



**Figure 3.10** The signals $s(m)$ and $w(n-m)$ used in evaluation of the short-time Fourier transform.

time Fourier transform using FFT methods. We will discuss such procedures shortly; first, however, we briefly review the interpretations of the short-time Fourier transform (see Ref. [2] for a more complete discussion of this fascinating branch of signal processing).

#### 3.2.2.1 Frequency Domain Interpretation of the Short-Time Fourier Transform

The short-time Fourier transform of the sequence $s(m)$ is defined as

$$S_n(e^{j\omega_i}) = \sum_m s(m)w(n-m)e^{-j\omega_i m}. \tag{3.16}$$

If we take the point of view that we are evaluating $S_n(e^{j\omega_i})$ for a fixed $n = n_0$, then we can interpret Eq. (3.16) as

$$S_{n_0}(e^{j\omega_i}) = \text{FT}[s(m)w(n_0-m)]|_{\omega=\omega_i} \tag{3.17}$$

where FT[·] denotes the Fourier Transform. Thus $S_{n_0}(e^{j\omega_i})$ is the conventional Fourier transform of the windowed signal, $s(m)\, w(n_0-m)$, evaluated at the frequency $\omega = \omega_i$. Figure 3.10 illustrates the signals $s(m)$ and $w(n-m)$, at times $n = n_0 = 50$, 100, and 200 to show which parts of $s(m)$ are used in the computation of the short-time Fourier transform. Since $w(m)$ is an FIR filter (i.e., of finite size), if we denote that size by $L$, then using the conventional Fourier transform interpretation of $S_n(e^{j\omega_i})$, we can state the following:

1. If $L$ is large, relative to the signal periodicity (pitch), then $S_n(e^{j\omega_i})$ gives good frequency resolution. That is, we can resolve individual pitch harmonics but only roughly see the overall spectral envelope of the section of speech within the window.
2. If $L$ is small relative to the signal periodicity, then $S_n(e^{j\omega_i})$ gives poor frequency resolution (i.e., no pitch harmonics are resolved), but a good estimate of the gross spectral shape is obtained.

To illustrate these points, Figures 3.11–3.14 show examples of windowed signals, $s(m)w(n-m)$, (part a of each figure) and the resulting log magnitude short time spectra, $20 \log_{10} |S_n(e^{j\omega})|$ (part b of each figure). Figure 3.11 shows results for an $L = 500$-point Hamming window applied to a section of voiced speech. The periodicity of the signal is clearly seen in the windowed time waveform, as well as in the short-time spectrum in which the fundamental frequency and its harmonics show up as narrow peaks at equally spaced

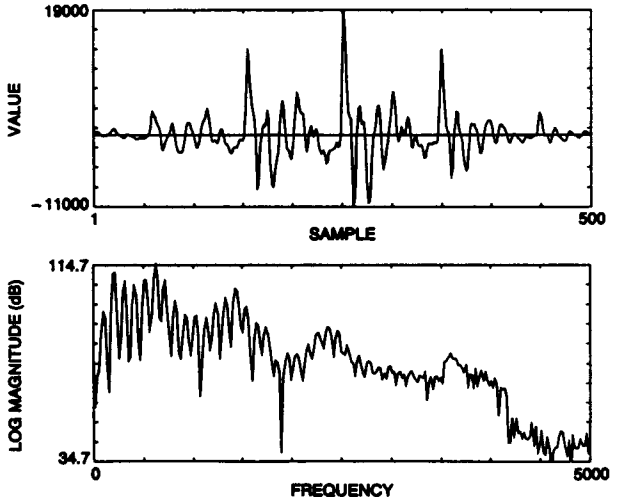



**Figure 3.11** Short-time Fourier transform using a long (500 points or 50 msec) Hamming window on a section of voiced speech.

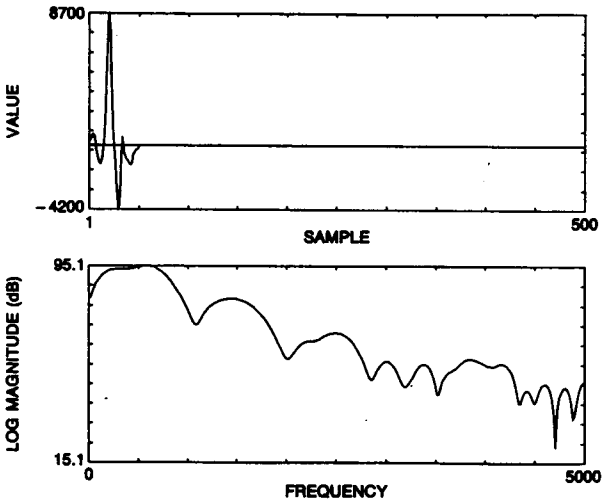



**Figure 3.12** Short-time Fourier transform using a short (50 points or 5 msec) Hamming window on a section of voiced speech.



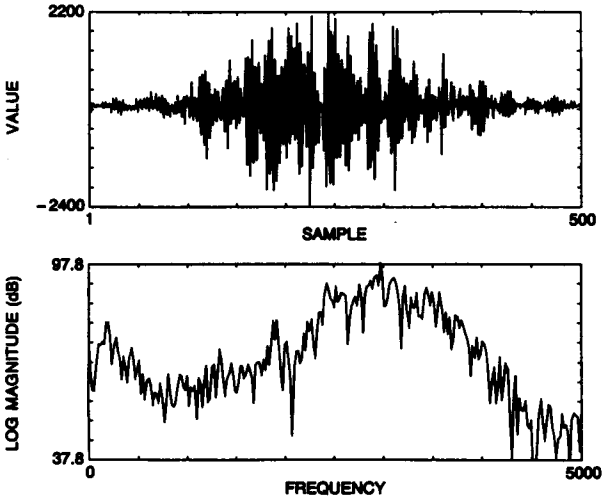



**Figure 3.13** Short-time Fourier transform using a long (500 points or 50 msec) Hamming window on a section of unvoiced speech.

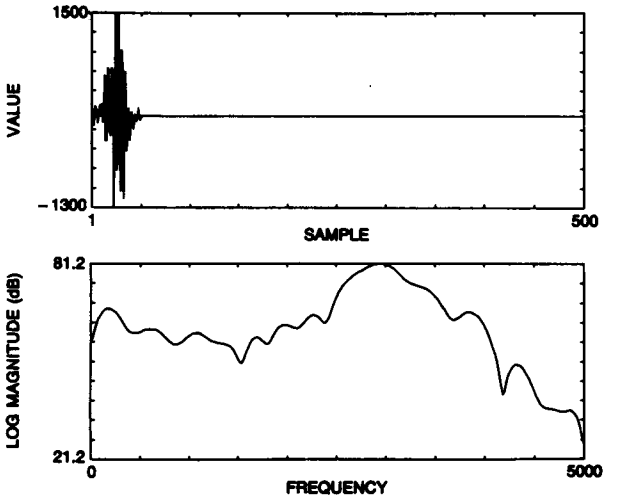



**Figure 3.14** Short-time Fourier transform using a short (50 points or 5 msec) Hamming window on a section of unvoiced speech.

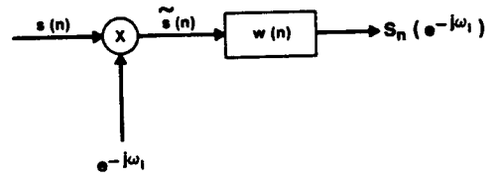



**Figure 3.15** Linear filter interpretation of the short-time Fourier transform.

frequencies. Figure 3.12 shows a similar set of comparisons for an $L = 50$-point Hamming window. For such short windows, the time sequence $s(m)w(n-m)$ does not show the signal periodicity, nor does the signal spectrum. In fact, what we see in the short-time Fourier transform log magnitude is a few rather broad peaks in frequency corresponding roughly to the speech formants.

Figures 3.13 and 3.14 show the effects of using windows on a section of unvoiced speech (corresponding to the fricative /sh/) for an $L = 500$ sample window (Figure 3.13) and $L = 50$ sample window (Figure 3.14). Since there is no periodicity in the signal, the resulting short-time spectral magnitude of Figure 3.13, for the $L = 500$ sample window shows a ragged series of local peaks and valleys due to the random nature of the unvoiced speech. Using the shorter window smoothes out the random fluctuations in the short-time spectral magnitude and again shows the broad spectral envelope very well.

### 3.2.2.2 Linear Filtering Interpretation of the Short-Time Fourier Transform

The linear filtering interpretation of the short-time Fourier transform is derived by considering $S_n(e^{j\omega_i})$, of Eq. (3.16), for fixed values of $\omega_i$, in which case we have

$$S_n(e^{j\omega_i}) = s(n)e^{-j\omega_i n} \circledast w(n). \tag{3.18}$$

That is, $S_n(e^{j\omega_i})$ is a convolution of the lowpass window, $w(n)$, with the speech signal, $s(n)$, modulated to center frequency $\omega_i$. This linear filtering interpretation of $S_n(e^{j\omega_i})$ is illustrated in Figure 3.15.

If we denote the conventional Fourier transforms of $s(n)$ and $w(n)$ by $S(e^{j\omega})$ and $W(e^{j\omega})$, then we see that the Fourier transform of $\tilde{s}(n)$ of Figure 3.15 is just

$$\tilde{S}(e^{j\omega}) = S(e^{j(\omega+\omega_i)}) \tag{3.19}$$

and thus we get

$$\mathrm{FT}(S_n(e^{j\omega_i})) = S(e^{j(\omega+\omega_i)})W(e^{j\omega}). \tag{3.20}$$

Since $W(e^{j\omega})$ approximates 1 over a narrow band, and is 0 everywhere else, we see that, for fixed values, $\omega_i$, the short-time Fourier transform gives a signal representative of the signal spectrum in a band around $\omega_i$. Thus the short-time Fourier transform, $S_n(e^{j\omega_i})$, represents the signal spectral analysis at frequency $\omega_i$ by a filter whose bandwidth is that of $W(e^{j\omega})$.



### 3.2.2.3 Review Exercises

**Exercise 3.1**

A speech signal is sampled at a rate of 20,000 samples per second ($F_s = 20$ kHz). A 20-msec window is used for short-time spectral analysis, and the window is moved by 10 msec in consecutive analysis frames. Assume that a radix-2 FFT is used to compute DFTs.

1. How many speech samples are used in each segment?
2. What is the frame rate of the short-time spectral analysis?
3. What size DFT and FFT are required to guarantee that no time-aliasing will occur?
4. What is the resulting frequency resolution (spacing in Hz) between adjacent spectral samples?

**Solution 3.1**

1. Twenty msec of speech at the rate of 20,000 samples per second gives

$$20 \times 10^{-3} \text{ sec} \times 20{,}000 \text{ samples/sec} = 400 \text{ samples}.$$

Each section of speech is 400 samples in duration.

2. Since the shift between consecutive speech frames is 10 msec (i.e., 200 samples at a 20,000 samples/sec rate), the frame rate is

$$\text{frame rate} = \frac{1}{\text{frame shift}} = \frac{1}{10 \times 10^{-3} \text{ sec}} = 100/\text{sec}.$$

That is, 100 spectral analyses are performed per second of speech.

3. To avoid time aliasing in using the DFT to evaluate the short-time Fourier transform, we require the DFT size to be at least as large as the frame size of the analysis frame. Hence, from part 1, we require at least a 400-point DFT. Since we are using a radix 2 FFT, we require, in theory, a 512-point FFT (the smallest power of 2 greater than 400) to compute the DFT without time aliasing. (We would use the 400 speech samples as the first 400 points of the 512-point array; we pad 112 zero-valued samples to the end of the array to fill in and give a 512-point array.) Since the speech signal is real (as opposed to complex), we can use an FFT size of 256 by appropriate signal preprocessing and postprocessing with a complex FFT algorithm.
4. The frequency resolution of the analysis is defined as

$$\text{frequency resolution} = \frac{\text{sampling rate}}{\text{DFT size}} = \frac{20{,}000 \text{ Hz}}{512} \cong 39 \text{ Hz}.$$

**Exercise 3.2**

If the sequences $s(n)$ and $w(n)$ have normal (long-time) Fourier transforms $S(e^{j\omega})$ and $W(e^{j\omega})$, then show that the short-time Fourier transform

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

can be expressed in the form

$$S_n(e^{j\omega}) = \frac{1}{2\pi}\int_{-\pi}^{\pi} W(e^{j\theta})e^{j\theta n}S(e^{j(\omega+\theta)})d\theta.$$

That is, $S_n(e^{j\omega})$ is a smoothed (by the window spectrum) spectral estimate of $S(e^{j\omega})$ at frequency $\omega$.

**Solution 3.2**

The long-time Fourier transforms of $s(n)$ and $w(n)$ can be expressed as

$$S(e^{j\omega}) = \sum_{n=-\infty}^{\infty} s(n)e^{-j\omega n}$$

$$W(e^{j\omega}) = \sum_{n=-\infty}^{\infty} w(n)e^{-j\omega n}.$$

The window sequence, $w(n)$, can be recovered from its long-time Fourier transform via the integration

$$w(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} W(e^{j\omega})e^{j\omega n}d\omega.$$

Hence, the short-time Fourier transform

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

can be put in the form (by substituting for $w(n-m)$):

$$\begin{aligned} S_n(e^{j\omega}) &= \sum_{m=-\infty}^{\infty} s(m)\left[\frac{1}{2\pi}\int_{-\pi}^{\pi} W(e^{j\theta})e^{j\theta(n-m)}d\theta\right]e^{-j\omega m} \\ &= \frac{1}{2\pi}\int_{-\pi}^{\pi} W(e^{j\theta})e^{j\theta n}\left[\sum_{m=-\infty}^{\infty} s(m)e^{-j(\omega+\theta)m}\right]d\theta \\ &= \frac{1}{2\pi}\int_{-\pi}^{\pi} W(e^{j\theta})e^{j\theta n}S(e^{j(\omega+\theta)})d\theta. \end{aligned}$$

**Exercise 3.3**

If we define the short-time spectrum of a signal in terms of its short-time Fourier transform as

$$X_n(e^{j\omega}) = \left|S_n(e^{j\omega})\right|^2$$

and we define the short-time autocorrelation of the signal as

$$R_n(k) = \sum_{m=-\infty}^{\infty} w(n-m)s(m)w(n-k-m)\,s(m+k)$$

then show that for

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

$R_n(k)$ and $X_n(e^{j\omega})$ are related as a normal (long-time) Fourier transform pair. In other words, show that $X_n(e^{j\omega})$ is the (long-time) Fourier transform of $R_n(k)$, and vice versa.



**Solution 3.3**

Given the definition of $S_n(e^{j\omega})$ we have

$$\begin{aligned} X_n(e^{j\omega}) &= \left|S_n(e^{j\omega})\right|^2 = [S_n(e^{j\omega})][S_n(e^{j\omega})]^* \\ &= \left[\sum_{m=-\infty}^{\infty} s(m)w(n-m)e^{-j\omega m}\right]\left[\sum_{r=-\infty}^{\infty} s(r)w(n-r)e^{j\omega r}\right] \\ &= \sum_{r=-\infty}^{\infty}\sum_{m=-\infty}^{\infty} w(n-m)s(m)w(n-r)s(r)e^{-j\omega(m-r)} \end{aligned}$$

Let $r = k + m$, then:

$$\begin{aligned} X_n(e^{j\omega}) &= \sum_{k=-\infty}^{\infty}\left[\sum_{m=-\infty}^{\infty} s(m)w(n-m)s(m+k)\,w(n-k-m)\right]e^{j\omega k} \\ &= \sum_{k=-\infty}^{\infty} R_n(k)e^{j\omega k} = \sum_{k=-\infty}^{\infty} R_n(k)e^{-j\omega k} \end{aligned}$$

(since $R_n(k) = R_n(-k)$); therefore

$$X_n(e^{j\omega}) = \left|S_n(e^{j\omega})\right|^2 \xleftrightarrow{\text{FT}} R_n(k).$$

### 3.2.2.4 FFT Implementation of Uniform Filter Bank Based on the Short-Time Fourier Transform

We now return to the question of how to efficiently implement the computation of the set of filter-bank outputs (Eq. (3.15)) for the uniform filter bank. If we assume, reasonably, that we are interested in a uniform frequency spacing—that is, if

$$f_i = i(F_s/N), \qquad i = 0, 1, \ldots, N-1 \tag{3.21}$$

then Eq. (3.15a) can be written as

$$x_i(n) = e^{j\left(\frac{2\pi}{N}\right)in}\sum_m s(m)w(n-m)e^{-j\left(\frac{2\pi}{N}\right)im}. \tag{3.22}$$

Now consider breaking up the summation over $m$, into a double summation of $r$ and $k$, in which

$$m = Nr + k, \quad 0 \le k \le N-1, \quad -\infty < r < \infty. \tag{3.23}$$

In other words, we break up the computation over $m$ into pieces of size $N$. If we let

$$s_n(m) = s(m)w(n-m), \tag{3.24}$$

then Eq. (3.22) can be written as

$$x_i(n) = e^{j\left(\frac{2\pi}{N}\right)in}\sum_r\left[\sum_{k=0}^{N-1} s_n(Nr+k)\right]e^{-j\left(\frac{2\pi}{N}\right)i(Nr+k)}. \tag{3.25}$$

Since $e^{-j2\pi ir} = 1$, for all $i, r$, then

$$x_i(n) = e^{j(\frac{2\pi}{N})in} \sum_{k=0}^{N-1} \left[ \sum_r s_n(Nr+k) \right] e^{-j(\frac{2\pi}{N})ik}. \tag{3.26}$$

If we define

$$u_n(k) = \sum_r s_n(Nr+k), \qquad 0 \le k \le N-1 \tag{3.27}$$

we wind up with

$$x_i(n) = e^{j(\frac{2\pi}{N})in} \left[ \sum_{k=0}^{N-1} u_n(k) e^{-j(\frac{2\pi}{N})ik} \right] \tag{3.28}$$

which is the desired result; that is, $x_i(n)$ is a modulated $N$-point DFT of the sequence $u_n(k)$.

Thus the basic steps in the computation of a uniform filter bank via FFT methods are as follows:

1. Form the windowed signal $s_n(m) = s(m)\, w(n-m)$, $m = n-L+1, \ldots, n$, where $w(n)$ is a causal, finite window of duration $L$ samples. Figure 3.16a illustrates this step.
2. Form $u_n(k) = \sum_r s_n(Nr+k)$, $0 \le k \le N-1$. That is, break the signal $s_n(m)$ into pieces of size $N$ samples and add up the pieces (alias them back unto itself) to give a signal of size $N$ samples. Figures 3.16b and c illustrate this step for the case in which $L \gg N$.
3. Take the $N$-point DFT of $u_n(k)$.
4. Modulate the DFT by the sequence $e^{j(\frac{2\pi}{N})in}$.

The modulation step 4 can be avoided by circularly shifting the sequence, $u_n(k)$, by $n \oplus N$ samples (where $\oplus$ is the modulo operation), to give $u_n((k-n))_N$, $0 \le k \le N-1$, prior to the DFT computation.

The computation to implement the uniform filter bank via Eq. (3.28) is essentially

$$C_{\text{FBFFT}} \cong 2N \log N \cdot, +. \tag{3.29}$$

Consider now the ratio, $R$, between the computation for the direct form implementation of a uniform filter bank (Eq. (3.13)), and the FFT implementation (Eq. (3.29)), such that

$$R = \frac{C_{\text{DFFIR}}}{C_{\text{FBFFT}}} = \frac{LQ}{2N \log N}. \tag{3.30}$$

If we assume $N = 32$ (i.e., a 16-channel filter bank), with $L = 128$ (i.e., 12.8 msec impulse response filter at a 10-kHz sampling rate), and $Q = 16$ channels, we get

$$R = \frac{128 \cdot 16}{2 \cdot 32 \cdot 5} = 6.4.$$

The FFT implementation is about 6.4 times more efficient than the direct form structure.



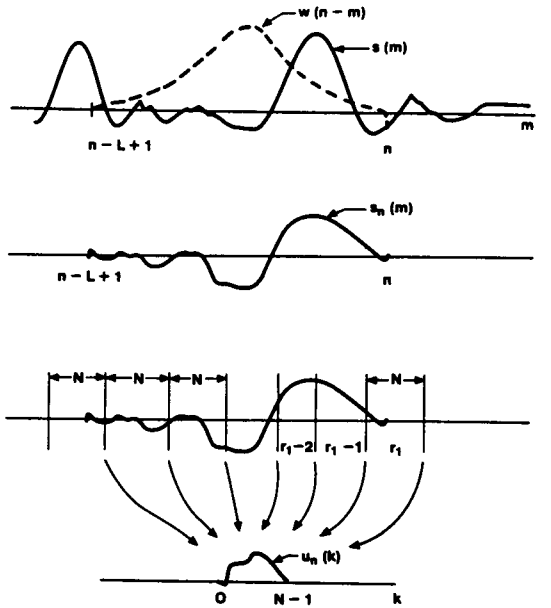



**Figure 3.16** FFT implementation of a uniform filter bank.

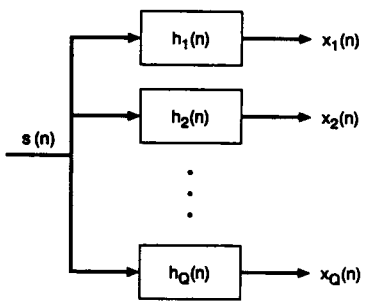



**Figure 3.17** Direct form implementation of an arbitrary nonuniform filter bank.

### 3.2.2.5 Nonuniform FIR Filter Bank Implementations

The most general form of a nonuniform FIR filter bank is shown in Figure 3.17, where the $k^{\text{th}}$ bandpass filter impulse response, $h_k(n)$, represents a filter with center frequency $\omega_k$, and bandwidth $\Delta\omega_k$. The set of $Q$ bandpass filters is intended to cover the frequency range of

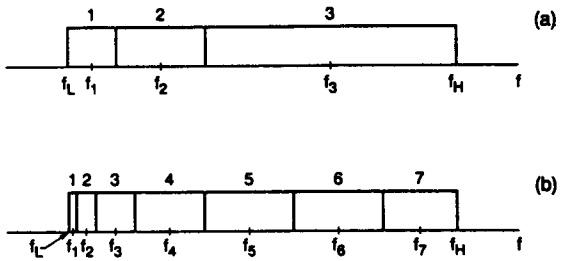



**Figure 3.18** Two arbitrary nonuniform filter-bank ideal filter specifications consisting of either 3 bands (part a) or 7 bands (part b).

interest for the intended speech-processing application.

In its most general form, each bandpass filter is implemented via a direct convolution; that is, no efficient FFT structure can be used. In the case where each bandpass filter is designed via the windowing design method (Ref. [1]), using the same lowpass window, we can show that the composite frequency response of the $Q$-channel filter bank is independent of the number and distribution of the individual filters. Thus a filter bank with the three filters shown in Figure 3.18a has the exact same composite frequency response as the filter bank with the seven filters shown in Figure 3.18b.

To show this we denote the impulse response of the $k^{\text{th}}$ bandpass filter as

$$h_k(n) = w(n)\tilde{h}_k(n), \tag{3.31}$$

where $w(n)$ is the FIR window, and $\tilde{h}_k(n)$ is the impulse response of the ideal bandpass filter being designed. The frequency response of the $k^{\text{th}}$ bandpass filter, $H_k(e^{j\omega})$, can be written as

$$H_k(e^{j\omega}) = W(e^{j\omega}) \circledast \tilde{H}_k(e^{j\omega}). \tag{3.32}$$

Thus the frequency response of the composite filter bank, $H(e^{j\omega})$, can be written as

$$H(e^{j\omega}) = \sum_{k=1}^{Q} H_k(e^{j\omega}) = \sum_{k=1}^{Q} W(e^{j\omega}) \circledast \tilde{H}_k(e^{j\omega}). \tag{3.33}$$

By interchanging the summation and the convolution we get

$$H(e^{j\omega}) = W(e^{j\omega}) \circledast \sum_{k=1}^{Q} \tilde{H}_k(e^{j\omega}). \tag{3.34}$$

By realizing that the summation of Eq. (3.34) is the summation of ideal frequency responses, we see that it is independent of the number and distribution of the individual filters. Thus we can write the summation as

$$\hat{H}(e^{j\omega}) = \sum_{k=1}^{Q} \tilde{H}_k(e^{j\omega}) = \begin{cases} 1, & \omega_{\min} \le \omega \le \omega_{\max} \\ 0, & \text{otherwise} \end{cases}, \tag{3.35}$$



where $\omega_{\min}$ is the lowest frequency in the filter bank, and $\omega_{\max}$ is the highest frequency. Then Eq. (3.34) can be expressed as

$$H(e^{j\omega}) = W(e^{j\omega}) \circledast \hat{H}(e^{j\omega}) \tag{3.36}$$

independent of the number of ideal filters, $Q$, and their distribution in frequency, which is the desired result.

### 3.2.2.6 FFT-Based Nonuniform Filter Banks

One possible way to exploit the FFT structure for implementing uniform filter banks discussed earlier is to design a large uniform filter bank (e.g., $N = 128$ or 256 channels) and then create the nonuniformity by combining two or more uniform channels. This technique of combining channels is readily shown to be equivalent to applying a modified analysis window to the sequence prior to the FFT. To see this, consider taking an $N$-point DFT of the sequence $x(n)$ (derived from the speech signal, $s(n)$, by windowing by $w(n)$). Thus we get

$$X_k = \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi}{N}nk}, \qquad 0 \le k \le N-1 \tag{3.37}$$

as the set of DFT values. If we consider adding DFT outputs $X_k$ and $X_{k+1}$, we get

$$X_k + X_{k+1} = \sum_{n=0}^{N-1} x(n) \left( e^{-j\frac{2\pi}{N}nk} + e^{-j\frac{2\pi}{N}n(k+1)} \right) \tag{3.38}$$

which can be written as

$$X'_k = X_k + X_{k+1} = \sum_{n=0}^{N-1} \left[ x(n) 2 e^{-j\frac{\pi n}{N}} \cos\left(\frac{\pi n}{N}\right) \right] e^{-j\frac{2\pi}{N}nk} \tag{3.39}$$

i.e. the equivalent $k^{\text{th}}$ channel value, $X'_k$, could have been obtained by weighting the sequence, $x(n)$, in time, by the complex sequence $2e^{-j\frac{\pi n}{N}} \cos\left(\frac{\pi n}{N}\right)$. If more than two channels are combined, then a different equivalent weighting sequence results. Thus FFT channel combining is essentially a "quick and dirty" method of designing broader bandpass filters and is a simple and effective way of realizing certain types of nonuniform filter bank analysis structures.

### 3.2.2.7 Tree Structure Realizations of Nonuniform Filter Banks

A third method used to implement certain types of nonuniform filter banks is the tree structure in which the speech signal is filtered in stages, and the sampling rate is successively reduced at each stage for efficiency of implementation. An example of such a realization is given in Figure 3.19a for the 4-band, octave-spaced filter bank shown (ideally) in Figure 3.19b. The original speech signal, $s(n)$, is filtered initially into two bands, a low band and a high band, using quadrature mirror filters (QMFs)—i.e., filters whose frequency responses are complementary. The high band, which covers half the spectrum, is reduced in sampling rate by a factor of 2, and represents the highest octave band ($\pi/2 \le \omega \le \pi$) of

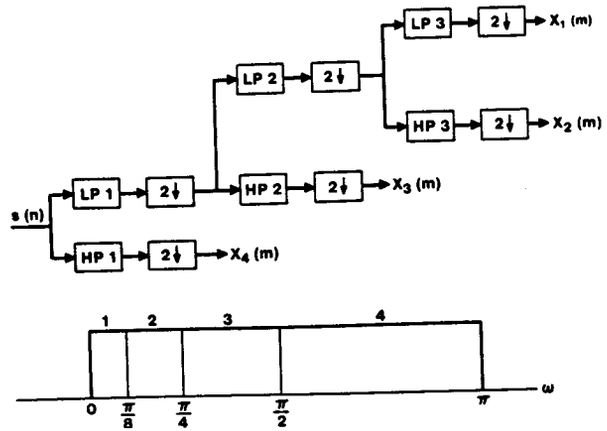



**Figure 3.19** Tree structure implementation of a 4-band, octave-spaced, filter bank.

the filter bank. The low band is similarly reduced in sampling rate by a factor of 2, and is fed into a second filtering stage in which the signal is again split into two equal bands using QMF filters. Again the high band of stage 2 is decimated by a factor of 2 and is used as the next-highest filter bank output; the low band is also decimated by a factor of 2 and fed into a third stage of QMF filters. These third-stage outputs, in this case after decimation by a factor of 2, are used as the two lowest filter bands.

QMF filter bank structures are quite efficient and have been used for a number of speech-processing applications [3]. Their efficiency for arbitrary nonuniform filter bank structures is not as good as for the octave band designs of Figure 3.19.

### 3.2.3 Summary of Considerations for Speech-Recognition Filter Banks

In the previous sections we discussed several methods of implementing filter banks for speech recognition. We have not gone into great detail here because our goal was to make the reader familiar with the issues involved in filter-bank design and implementation, not to make the reader an expert in signal processing. The interested reader is urged to pursue this fascinating area further by studying the material in the References at the end of this chapter. In this section we summarize the considerations that go into choosing the number and types of filters used in the structures discussed earlier in this section.

The first consideration for any filter bank is the type of digital filter used. The choices are IIR (recursive) and FIR (nonrecursive) designs. The IIR designs have the advantage of being implementable in simple, efficient structures. The big disadvantage of IIR filters is that their phase response is nonlinear; hence, to minimize this disadvantage



a trade-off is usually made between the ideal magnitude characteristics that can readily be realized, and the highly nonideal phase characteristics. On the other hand, FIR filters can achieve linear phase without compromising the ability to approximate ideal magnitude characteristics; however, they are usually computationally expensive in implementation. For speech-recognition applications, we have shown how an FFT structure can often be applied to alleviate considerably the computational inefficiency of FIR filter banks; hence, most practical digital filter bank structures use FIR filters (usually in an FFT realization).

Once the type of filter has been decided, the next consideration is the number of filters to be used in the filter bank. For uniform filter banks, the number of filters, $Q$, cannot be too small or else the ability of the filter bank to resolve the speech spectrum is greatly impaired. Thus values of $Q$ less than about 8 are generally avoided. Similarly, the value of $Q$ cannot be too large (unless there is considerable filter overlap), because the filter bandwidths would eventually be too narrow for some talkers (e.g., high-pitch females or children), and there would be a high probability that certain bands would have extremely low speech energy (i.e., no prominent harmonic would fall within the band). Thus, practical systems tend to have values of $Q \leq 32$. Although uniformly spaced filter banks have been widely used for recognition, many practical systems have used nonuniform spacing in an effort to reduce overall computation and to characterize the speech spectrum in a manner considered more consistent with human perception.

A final consideration for practical filter-bank analyzers is the choice of nonlinearity and lowpass filter used at the output of each channel. Typically the nonlinearity has been a full wave rectifier (FWR), a half wave rectifier (HWR), or a center clipper. The resultant spectrum is only weakly sensitive to the nonlinearity. The lowpass filter used in practice varies from a simple integrator to a fairly good quality IIR lowpass filter (typically a Bessel filter).

### 3.2.4 Practical Examples of Speech-Recognition Filter Banks

Figures 3.20–3.25 [4] show examples of a wide range of speech-recognition filter banks, including both uniform and nonuniform designs. Figure 3.20 is for a 15-channel uniform filter bank in which the basic lowpass filter was designed using the windowing technique with a 101-point Kaiser window. Part a of the figure shows the impulse response of the lowpass filter (i.e., an ideal lowpass filter response multiplied by a Kaiser window). Part b of the figure shows the responses of the individual filters in the filter bank (note there is no overlap between adjacent filters), and part c shows the composite frequency response of the overall filter bank. The sidelobe peak ripple of each individual filter is down about 60 dB, and the composite frequency response is essentially ideally flat over the entire frequency range of interest (approximate 100–3000 Hz).

By contrast, Figure 3.21 is for a 15-channel uniform filter bank in which the basic lowpass filter was a Kaiser window (instead of the Kaiser windowed version of the ideal lowpass filter). From parts b and c of this figure, it can be seen that the individual bandpass filters are narrower in bandwidth than those of Figure 3.20; furthermore, the composite

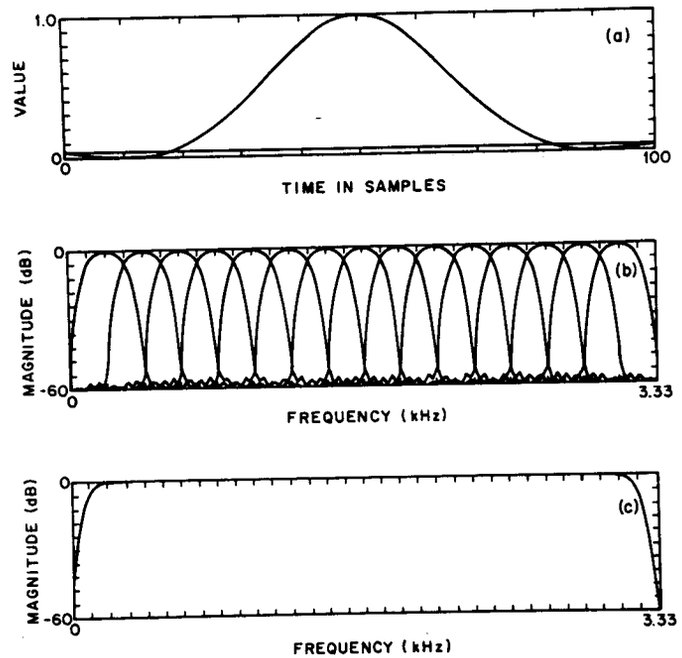



**Figure 3.20** Window sequence, $w(n)$, (part a), the individual filter response (part b), and the composite response (part c) of a $Q = 15$ channel, uniform filter bank, designed using a 101-point Kaiser window smoothed lowpass window (after Dautrich et al. [4]).

filter-bank response shows 18 dB gaps at the boundaries between each filter. Clearly, this filter bank would be unacceptable for speech-recognition applications.

Figures 3.22 and 3.23 show individual filter frequency responses, and the composite frequency response, for a 4-channel, octave-band filter bank, and a 12-channel, 1/3 octave filter bank, frequency, respectively. Each of these nonuniform filter banks was designed to cover the frequency band from 200 to 3200 Hz and used linear-phase FIR filters (101 points for the octave band design, and 201 points for the 1/3 octave band design) for each individual channel. The peak sidelobe ripple was about −40 dB for both filter banks.

Figure 3.24 shows a similar set of responses for a 7-channel critical band filter bank in which each individual filter encompassed two critical bands. Again we used 101-point, linear phase, FIR filters with a peak sidelobe of −54 dB to realize each individual bandpass filter. Finally, Figure 3.25 shows the responses of a 13-channel, critical band filter bank in which the individual channels were highly overlapping. The individual bandpass filter responses are rather poor (e.g., the ratios of center frequency to bandwidth of each filter was about 8). However, this poor frequency resolution characteristic was balanced somewhat by the excellent time resolution of the filters.



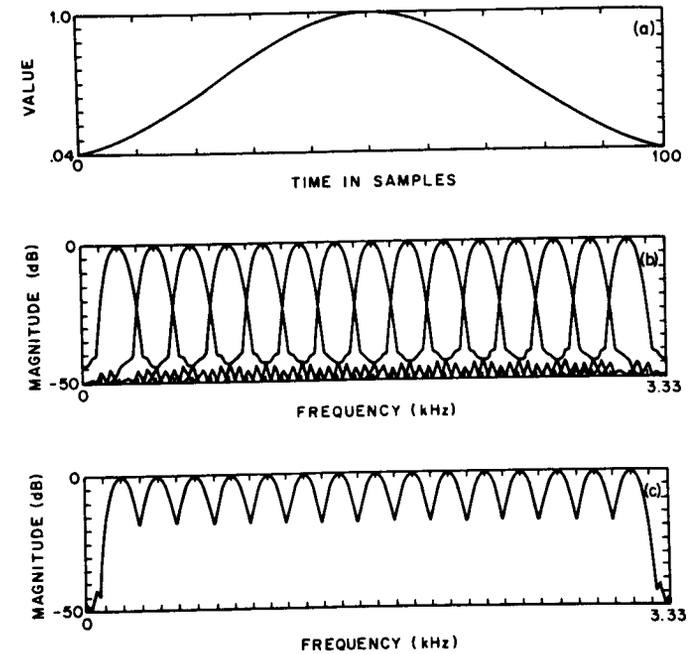



**Figure 3.21** Window sequence, $w(n)$, (part a), the individual filter responses (part b), and the composite response (part c) of a $Q = 15$ channel, uniform filter bank, designed using a 101-point Kaiser window directly as the lowpass window (after Dautrich et al. [4]).

### 3.2.5 Generalizations of Filter-Bank Analyzer

Although we have been concerned primarily with designing and implementing individual channels of a filter-bank analyzer, there is a generalized structure that must be considered as part of the canonic filter-bank analysis method. This generalized structure is shown in Figure 3.26. The generalization includes a signal preprocessor that "conditions" the speech signal, $s(n)$, to a new form, $\hat{s}(n)$, which is "more suitable" for filter-bank analysis, and a postprocessor that operates on the filter-bank output vectors, $x(m)$, to give the processed vectors $\hat{x}(m)$ that are "more suitable" for recognition. Although a wide range of signal-processing operations could go into the preprocessor and postprocessor boxes, perhaps the most reasonable ones include the following.

#### Preprocessor Operations

- signal preemphasis (to equalize the inherent spectral tilt in speech)

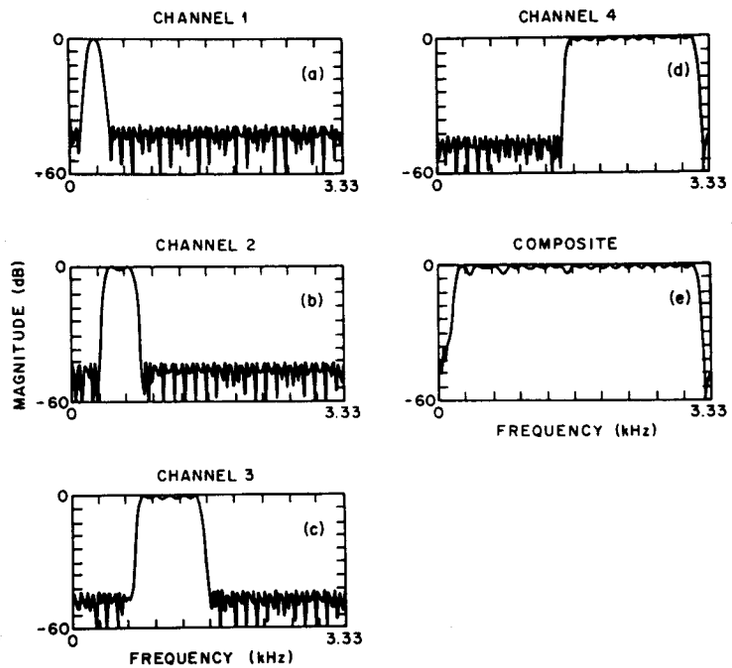



**Figure 3.22** Individual channel responses (parts a to d) and composite filter response (part c) of a $Q = 4$ channel, octave band design, using 101-point FIR filters in each band (after Dautrich et al. [4]).

- noise elimination
- signal enhancement (to make the formant peaks more prominent)

### Postprocessor Operations

- temporal smoothing of sequential filter-bank output vectors
- frequency smoothing of individual filter-bank output vectors
- normalization of each filter-bank output vector
- thresholding and/or quantization of the filter-bank output vectors
- principal components analysis of the filter-bank output vector.

The purpose of the preprocessor is to make the speech signal as clean as possible so far as the filter bank analyzer is concerned; hence, noise is eliminated, long-time spectral trends are removed, and the signal is spectrally flattened to give the best immunity to measurement imperfections. Similarly, the purpose of the postprocessor is to clean up the



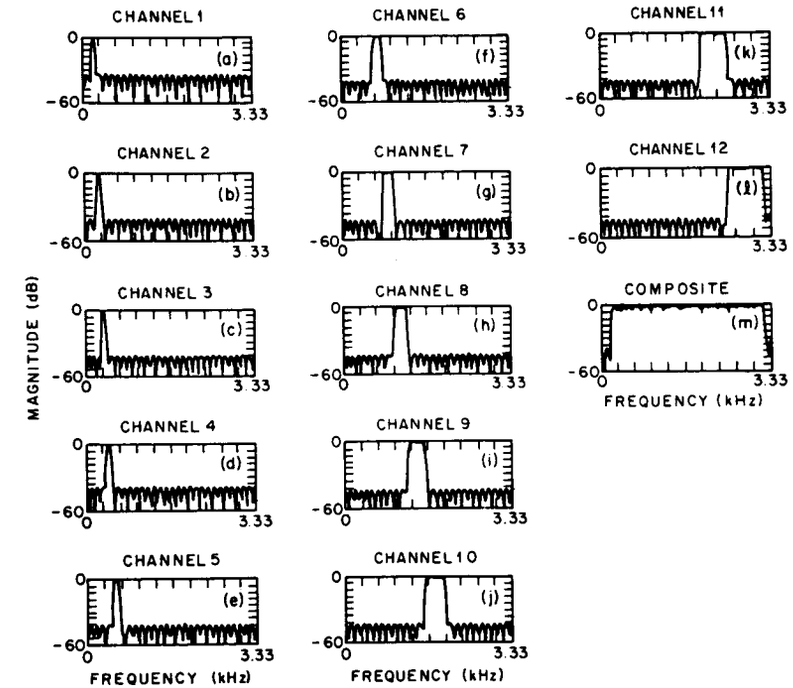



**Figure 3.23** Individual channel responses and composite filter response of a $Q = 12$ channel, 1/3 octave band design, using 201-point FIR filters in each band (after Dautrich et al. [4]).

sequence of feature vectors from the filter-bank analyzer so as to best represent the spectral information in the speech signal and thereby to maximize the chances of successful speech recognition [4,5].

## 3.3 LINEAR PREDICTIVE CODING MODEL FOR SPEECH RECOGNITION

The theory of linear predictive coding (LPC), as applied to speech, has been well understood for many years (see for example Ref. [6]). In this section we describe the basics of how LPC has been applied in speech-recognition systems. The mathematical details and derivations will be omitted here; the interested reader is referred to the references.

Before describing a general LPC front-end processor for speech recognition, it is worthwhile to review the reasons why LPC has been so widely used. These include the following:

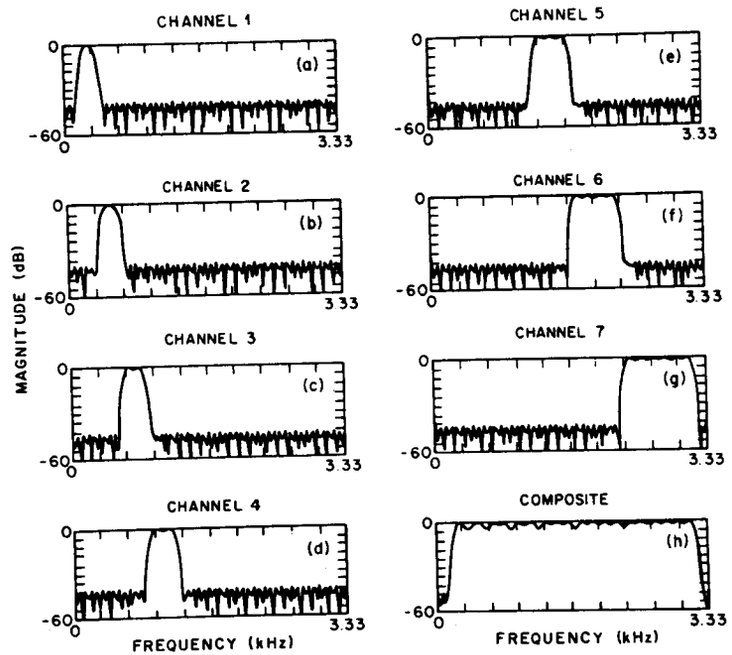



**Figure 3.24** Individual channel responses (parts a to g) and composite filter response (part h) of a $Q = 7$ channel critical band filter bank design (after Dautrich et al. [4]).

1. LPC provides a good model of the speech signal. This is especially true for the quasi steady state voiced regions of speech in which the all-pole model of LPC provides a good approximation to the vocal tract spectral envelope. During unvoiced and transient regions of speech, the LPC model is less effective than for voiced regions, but it still provides an acceptably useful model for speech-recognition purposes.
2. The way in which LPC is applied to the analysis of speech signals leads to a reasonable source-vocal tract separation. As a result, a parsimonious representation of the vocal tract characteristics (which we know are directly related to the speech sound being produced) becomes possible.
3. LPC is an analytically tractable model. The method of LPC is mathematically precise and is simple and straightforward to implement in either software or hardware. The computation involved in LPC processing is considerably less than that required for an all-digital implementation of the bank-of-filters model described in Section 3.2.
4. The LPC model works well in recognition applications. Experience has shown that



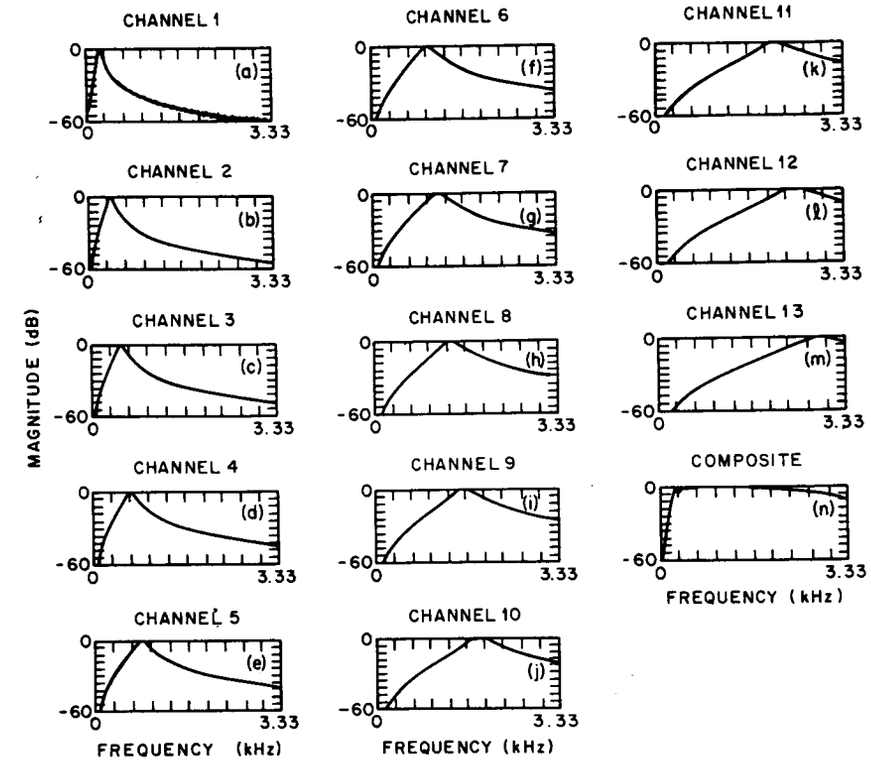



**Figure 3.25** Individual channel responses and composite filter response of a $Q = 13$ channel, critical band spacing filter bank, using highly overlapping filters in frequency (after Dautrich et al. [4]).

**Figure 3.26** Generalization of filter-bank analysis model.

the performance of speech recognizers, based on LPC front ends, is comparable to or better than that of recognizers based on filter-bank front ends (see References [4,5,7]).

Based on the above considerations, LPC front-end processing has been used in a large number of recognizers. In particular, most of the systems to be described in this book are based on this model.

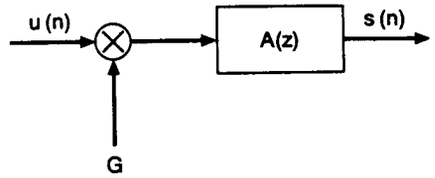



**Figure 3.27** Linear prediction model of speech.

### 3.3.1 The LPC Model

The basic idea behind the LPC model is that a given speech sample at time $n$, $s(n)$, can be approximated as a linear combination of the past $p$ speech samples, such that

$$s(n) \approx a_1 s(n-1) + a_2 s(n-2) + \cdots + a_p s(n-p), \tag{3.40}$$

where the coefficients $a_1, a_2, \ldots, a_p$ are assumed constant over the speech analysis frame. We convert Eq. (3.40) to an equality by including an excitation term, $G\,u(n)$, giving:

$$s(n) = \sum_{i=1}^{p} a_i s(n-i) + G\,u(n), \tag{3.41}$$

where $u(n)$ is a normalized excitation and $G$ is the gain of the excitation. By expressing Eq. (3.41) in the $z$-domain we get the relation

$$S(z) = \sum_{i=1}^{p} a_i z^{-i} S(z) + G\,U(z) \tag{3.42}$$

leading to the transfer function

$$H(z) = \frac{S(z)}{G\,U(z)} = \frac{1}{1 - \sum_{i=1}^{p} a_i z^{-i}} = \frac{1}{A(z)}. \tag{3.43}$$

The interpretation of Eq. (3.43) is given in Figure 3.27, which shows the normalized excitation source, $u(n)$, being scaled by the gain, $G$, and acting as input to the all-pole system, $H(z) = \frac{1}{A(z)}$, to produce the speech signal, $s(n)$. Based on our knowledge that the actual excitation function for speech is essentially either a quasiperiodic pulse train (for voiced speech sounds) or a random noise source (for unvoiced sounds), the appropriate synthesis model for speech, corresponding to the LPC analysis, is as shown in Figure 3.28. Here the normalized excitation source is chosen by a switch whose position is controlled by the voiced/unvoiced character of the speech, which chooses either a quasiperiodic train of pulses as the excitation for voiced sounds, or a random noise sequence for unvoiced sounds. The appropriate gain, $G$, of the source is estimated from the speech signal, and the scaled source is used as input to a digital filter ($H(z)$), which is controlled by the vocal tract



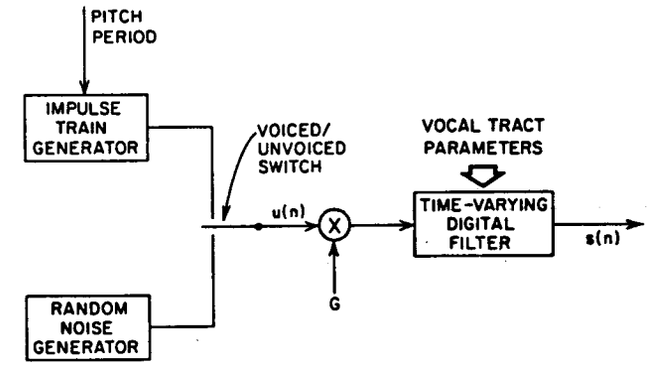



**Figure 3.28** Speech synthesis model based on LPC model.

parameters characteristic of the speech being produced. Thus the parameters of this model are voiced/unvoiced classification, pitch period for voiced sounds, the gain parameter, and the coefficients of the digital filter, $\{a_k\}$. These parameters all vary slowly with time.

### 3.3.2 LPC Analysis Equations

Based on the model of Figure 3.27, the exact relation between $s(n)$ and $u(n)$ is

$$s(n) = \sum_{k=1}^{p} a_k s(n-k) + G\,u(n). \tag{3.44}$$

We consider the linear combination of past speech samples as the estimate $\tilde{s}(n)$, defined as

$$\tilde{s}(n) = \sum_{k=1}^{p} a_k s(n-k). \tag{3.45}$$

We now form the prediction error, $e(n)$, defined as

$$e(n) = s(n) - \tilde{s}(n) = s(n) - \sum_{k=1}^{p} a_k s(n-k) \tag{3.46}$$

with error transfer function

$$A(z) = \frac{E(z)}{S(z)} = 1 - \sum_{k=1}^{p} a_k z^{-k}. \tag{3.47}$$

Clearly, when $s(n)$ is actually generated by a linear system of the type shown in Figure 3.27, then the prediction error, $e(n)$, will equal $G\,u(n)$, the scaled excitation.

The basic problem of linear prediction analysis is to determine the set of predictor

coefficients, $\{a_k\}$, directly from the speech signal so that the spectral properties of the digital filter of Figure 3.28 match those of the speech waveform within the analysis window. Since the spectral characteristics of speech vary over time, the predictor coefficients at a given time, $n$, must be estimated from a *short* segment of the speech signal occurring around time $n$. Thus the basic approach is to find a set of predictor coefficients that minimize the mean-squared prediction error over a short segment of the speech waveform. (Usually this type of short time spectral analysis is performed on successive frames of speech, with frame spacing on the order of 10 msec.)

To set up the equations that must be solved to determine the predictor coefficients, we define short-term speech and error segments at time $n$ as

$$s_n(m) = s(n+m) \tag{3.48a}$$

$$e_n(m) = e(n+m) \tag{3.48b}$$

and we seek to minimize the mean squared error signal at time $n$

$$E_n = \sum_m e_n^2(m) \tag{3.49}$$

which, using the definition of $e_n(m)$ in terms of $s_n(m)$, can be written as

$$E_n = \sum_m \left[ s_n(m) - \sum_{k=1}^{p} a_k s_n(m-k) \right]^2. \tag{3.50}$$

To solve Eq. (3.50), for the predictor coefficients, we differentiate $E_n$ with respect to each $a_k$ and set the result to zero,

$$\frac{\partial E_n}{\partial a_k} = 0, \qquad k = 1, 2, \ldots, p \tag{3.51}$$

giving

$$\sum_m s_n(m-i)s_n(m) = \sum_{k=1}^{p} \hat{a}_k \sum_m s_n(m-i)s_n(m-k). \tag{3.52}$$

By recognizing that terms of the form $\sum s_n(m-i)\, s_n(m-k)$ are terms of the short-term covariance of $s_n(m)$, i.e.,

$$\phi_n(i,k) = \sum_m s_n(m-i)s_n(m-k) \tag{3.53}$$

we can express Eq. (3.52) in the compact notation

$$\boxed{\phi_n(i,0) = \sum_{k=1}^{p} \hat{a}_k \phi_n(i,k)} \tag{3.54}$$

which describes a set of $p$ equations in $p$ unknowns. It is readily shown that the minimum mean-squared error, $\hat{E}_n$, can be expressed as



$$\hat{E}_n = \sum_m s_n^2(m) - \sum_{k=1}^{p} \hat{a}_k \sum_m s_n(m)s_n(m-k) \tag{3.55}$$

$$= \phi_n(0,0) - \sum_{k=1}^{p} \hat{a}_k \phi_n(0,k). \tag{3.56}$$

Thus the minimum mean-squared error consists of a fixed term ($\phi_n(0,0)$) and terms that depend on the predictor coefficients.

To solve Eq. (3.54) for the optimum predictor coefficients (the $\hat{a}_k$s) we have to compute $\phi_n(i,k)$ for $1 \le i \le p$ and $0 \le k \le p$, and then solve the resulting set of $p$ simultaneous equations. In practice, the method of solving the equations (as well as the method of computing the $\phi$s) is a strong function of the range of $m$ used in defining both the section of speech for analysis and the region over which the mean-squared error is computed. We now discuss two standard methods of defining this range for speech.

### 3.3.3 The Autocorrelation Method

A fairly simple and straightforward way of defining the limits on $m$ in the summations is to assume that the speech segment, $s_n(m)$, is identically zero outside the interval $0 \le m \le N-1$. This is equivalent to assuming that the speech signal, $s(m+n)$, is multiplied by a finite length window, $w(m)$, which is identically zero outside the range $0 \le m \le N-1$. Thus the speech sample for minimization can be expressed as

$$s_n(m) = \begin{cases} s(m+n)\cdot w(m), & 0 \le m \le N-1 \\ 0, & \text{otherwise.} \end{cases} \tag{3.57}$$

The effect of weighting of the speech by a window is illustrated in Figures 3.29–3.31. In each of these figures, the upper panel shows the running speech waveform, $s(m)$, the middle panel shows the weighted section of speech (using a Hamming window for $w(m)$), and the bottom panel shows the resulting error signal, $e_n(m)$, based on optimum selection of the predictor parameters.

Based on Eq. (3.57), for $m < 0$, the error signal $e_n(m)$ is exactly zero since $s_n(m) = 0$ for all $m < 0$ and therefore there is no prediction error. Furthermore, for $m > N-1+p$ there is again no prediction error because $s_n(m) = 0$ for all $m > N-1$. However, in the region of $m = 0$ (i.e., from $m = 0$ to $m = p-1$) the windowed speech signal $s_n(m)$ is being predicted from previous samples, some of which are arbitrarily zero. Hence the potential for relatively large prediction errors exists in this region and can actually be seen to exist in the bottom panel of Figure 3.29. Furthermore, in the region of $m = N-1$ (i.e., from $m = N-1$ to $m = N-1+p$) the potential of large prediction errors again exists because the zero-valued (weighted) speech signal is being predicted from at least some nonzero previous speech samples. In the bottom panel of Figure 3.30 we see this effect at the end of the prediction error waveform. These two effects are especially prominent for voiced speech when the beginning of a pitch period occurs at or very close to the $m = 0$ or $m = N-1$ points of the sample. For unvoiced speech, these problems are essentially eliminated because no part of the waveform is position sensitive. Hence we see

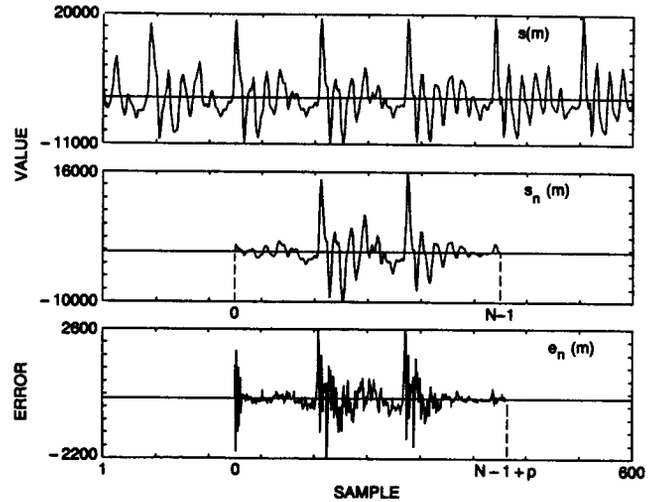



**Figure 3.29** Illustration of speech sample, weighted speech section, and prediction error for voiced speech where the prediction error is large at the beginning of the section.

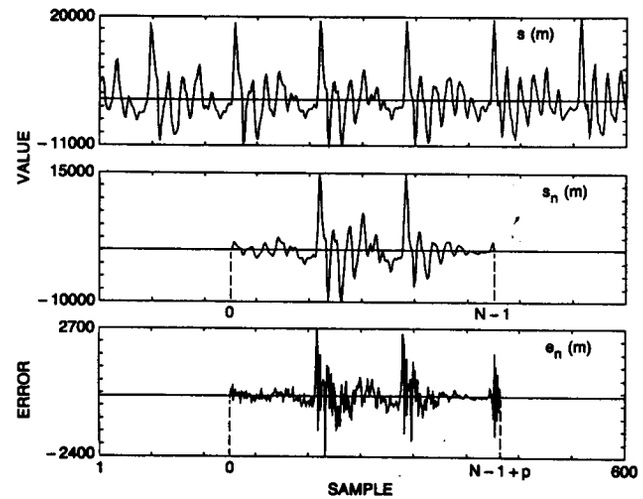



**Figure 3.30** Illustration of speech sample, weighted speech section, and prediction error for voiced speech where the prediction error is large at the end of the section.



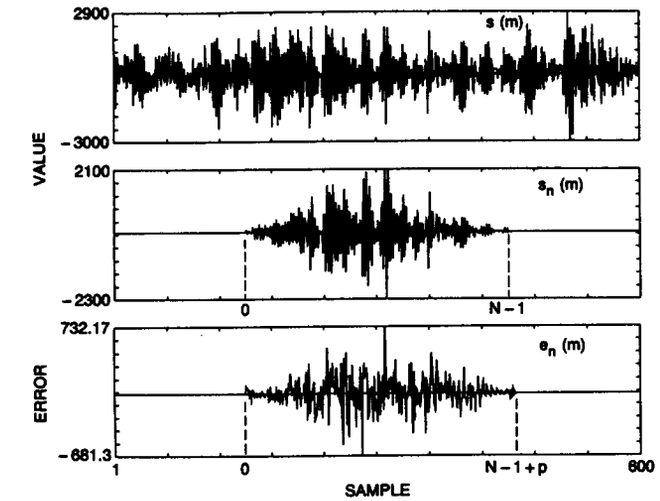



**Figure 3.31** Illustration of speech sample, weighted speech section, and prediction error for unvoiced speech where there are almost no artifacts at the boundaries of the section.

neither effect occurring in the bottom panel of Figure 3.3.1. The purpose of the window of Eq. (3.57) is to taper the signal near $m = 0$ and near $m = N - 1$ so as to minimize the errors at section boundaries.

Based on using the weighted signal of Eq. (3.57) the mean-squared error becomes

$$E_n = \sum_{m=0}^{N-1+p} e_n^2(m) \tag{3.58}$$

and $\phi_n(i,k)$ can be expressed as

$$\phi_n(i,k) = \sum_{m=0}^{N-1+p} s_n(m-i)s_n(m-k), \qquad \begin{matrix} 1 \le i \le p \\ 0 \le k \le p \end{matrix} \tag{3.59}$$

or

$$\phi_n(i,k) = \sum_{m=0}^{N-1-(i-k)} s_n(m)s_n(m+i-k), \qquad \begin{matrix} 1 \le i \le p \\ 0 \le k \le p \end{matrix} . \tag{3.60}$$

Since Eq. (3.60) is only a function of $i - k$ (rather than the two independent variables $i$ and $k$), the covariance function, $\phi_n(i,k)$, reduces to the simple autocorrelation function, i.e.,

$$\phi_n(i,k) = r_n(i-k) = \sum_{m=0}^{N-1-(i-k)} s_n(m)s_n(m+i-k). \tag{3.61}$$

Since the autocorrelation function is symmetric, i.e. $r_n(-k) = r_n(k)$, the LPC equations can be expressed as

$$\boxed{\sum_{k=1}^{p} r_n(|i-k|)\hat{a}_k = r_n(i), \qquad 1 \le i \le p} \tag{3.62}$$

and can be expressed in matrix form as

$$\begin{bmatrix} r_n(0) & r_n(1) & r_n(2) & \cdots & r_n(p-1) \\ r_n(1) & r_n(0) & r_n(1) & \cdots & r_n(p-2) \\ r_n(2) & r_n(1) & r_n(0) & \cdots & r_n(p-3) \\ \vdots & \vdots & \vdots & & \vdots \\ r_n(p-1) & r_n(p-2) & r_n(p-3) & \cdots & r_n(0) \end{bmatrix} \begin{bmatrix} \hat{a}_1 \\ \hat{a}_2 \\ \hat{a}_3 \\ \vdots \\ \hat{a}_p \end{bmatrix} = \begin{bmatrix} r_n(1) \\ r_n(2) \\ r_n(3) \\ \vdots \\ r_n(p) \end{bmatrix}. \tag{3.63}$$

The $p \times p$ matrix of autocorrelation values is a Toeplitz matrix (symmetric with all diagonal elements equal) and hence can be solved efficiently through several well-known procedures. (We will discuss one such procedure, the Durbin algorithm, later in this chapter.)

### 3.3.4 The Covariance Method

An alternative to using a weighting function or window for defining $s_n(m)$ is to fix the interval over which the mean-squared error is computed to the range $0 \le m \le N-1$ and to use the unweighted speech directly—that is,

$$E_n = \sum_{m=0}^{N-1} e_n^2(m) \tag{3.64}$$

with $\phi_n(i,k)$ defined as

$$\phi_n(i,k) = \sum_{m=0}^{N-1} s_n(m-i)s_n(m-k), \qquad \begin{matrix} 1 \le i \le p \\ 0 \le k \le p \end{matrix} \tag{3.65}$$

or, by a change of variables,

$$\phi_n(i,k) = \sum_{m=-i}^{N-i-1} s_n(m)s_n(m+i-k), \qquad \begin{matrix} 1 \le i \le p \\ 0 \le k \le p \end{matrix}. \tag{3.66}$$

If we consider when $i = p$ we see that the computation of Eq. (3.66) involves speech samples $s_n(m)$ defined from $m = -p$ up to $m = N-1-p$ and, when $k = 0$, $s_n(m+i-k)$ involves samples from 0 to $N-1$. Hence the range of speech required for the complete computation is from $s_n(-p)$ to $s_n(N-1)$—that is, the samples $s_n(-p), s_n(-p+1), \ldots, s_n(-1)$, *outside* the error minimization interval, are required.



Using the extended speech interval to define the covariance values, $\phi_n(i,k)$, the matrix form of the LPC analysis equations becomes

$$\begin{bmatrix} \phi_n(1,1) & \phi_n(1,2) & \phi_n(1,3) & \cdots & \phi_n(1,p) \\ \phi_n(2,1) & \phi_n(2,2) & \phi_n(2,3) & \cdots & \phi_n(2,p) \\ \phi_n(3,1) & \phi_n(3,2) & \phi_n(3,3) & \cdots & \phi_n(3,p) \\ \vdots & \vdots & \vdots & & \vdots \\ \phi_n(p,1) & \phi_n(p,2) & \phi_n(p,3) & \cdots & \phi_n(p,p) \end{bmatrix} \begin{bmatrix} \hat{a}_1 \\ \hat{a}_2 \\ \hat{a}_3 \\ \vdots \\ \hat{a}_p \end{bmatrix} = \begin{bmatrix} \phi_n(1,0) \\ \phi_n(2,0) \\ \phi_n(3,0) \\ \vdots \\ \phi_n(p,0) \end{bmatrix}. \tag{3.67}$$

The resulting covariance matrix is symmetric (since $\phi_n(i,k) = \phi_n(k,i)$) but not Toeplitz, and can be solved efficiently by a set of techniques called the Cholesky decomposition method [6]. Since the full covariance form of the LPC analysis equations is generally *not* used for speech-recognition systems, we will not discuss this method further but instead will concentrate on the autocorrelation method of LPC analysis for the remainder of this chapter.

### 3.3.5 Review Exercise

**Exercise 3.4**

Given an LPC system of the form

$$H(z) = \frac{G}{1 - \sum_{k=1}^{p} a_k z^{-k}}$$

how would you evaluate $H(e^{j\omega})$ using FFT techniques?

**Solution 3.4**

Define the LPC polynomial as

$$A(z) = \frac{G}{H(z)} = 1 - \sum_{k=1}^{p} a_k z^{-k}.$$

This finite polynomial in $z$ has a time domain response, $f(n)$, which is an FIR sequence of the form

$$f(n) = \begin{cases} 1, & n = 0 \\ -a_n, & 1 \le n \le p \\ 0, & \text{otherwise} \end{cases}.$$

Hence we can evaluate $A(e^{j\omega})$, using FFTs, by supplementing $f(n)$ with sufficient zero-valued samples to form an $N$-point sequence (e.g., $N = 256$, or $N = 512$), and taking the DFT of that sequence giving $A(e^{j\frac{2\pi}{N}k})$, $0 \le k \le N-1$, i.e. $A(e^{j\omega})|_{\omega=\frac{2\pi k}{N}}$. We can then evaluate $H(e^{j\omega})$ for $\omega = \frac{2\pi k}{N}$, $k = 0, 1, \ldots, N-1$ as $G/A(e^{j\omega})|_{\omega=\frac{2\pi k}{N}}$.

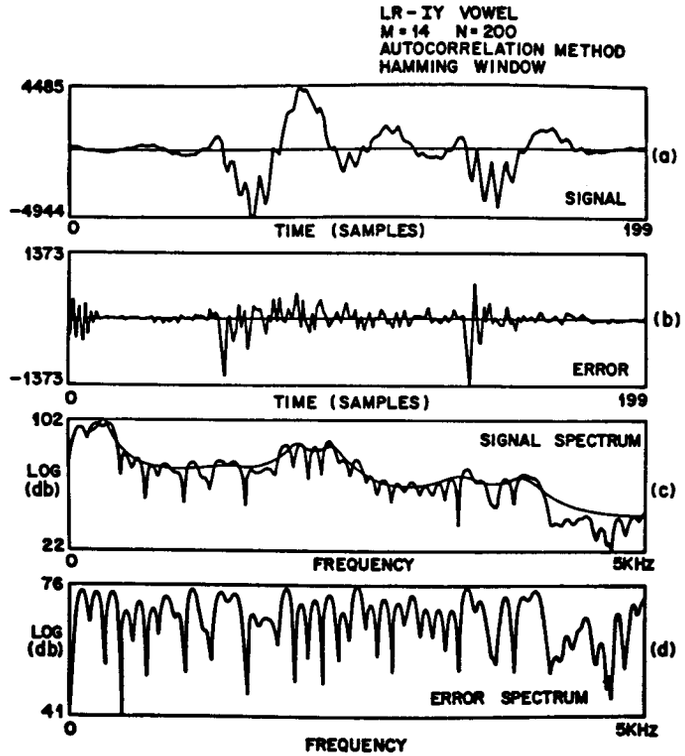



**Figure 3.32** Typical signals and spectra for LPC autocorrelation method for a segment of speech spoken by a male speaker (after Rabiner et al. [8]).

### 3.3.6 Examples of LPC Analysis

To illustrate some of the properties of the signals involved in LPC analysis, Figures 3.32 and 3.33 show series of waveform and spectral plots of the windowed speech signal (part a), the prediction error signal (part b), the signal log spectrum (FFT-based) fitted by an LPC log spectrum (as defined from Exercises 3.4, part c), and the log spectrum of the prediction error signal (part d). The results in Figure 3.32 are for the IY vowel spoken by a male speaker; those of Figure 3.33 are for the AH vowel spoken by a female speaker. For both examples the speech sample size was 20 msec (200 samples at a 10-kHz rate) and the analysis was performed using a $p = 14^{\text{th}}$ order LPC analysis. For the male speaker, about two periods of signal were used in the analysis frame. The error signal is a factor of almost 4 smaller in magnitude than the speech signal and has a much flatter spectral trend than



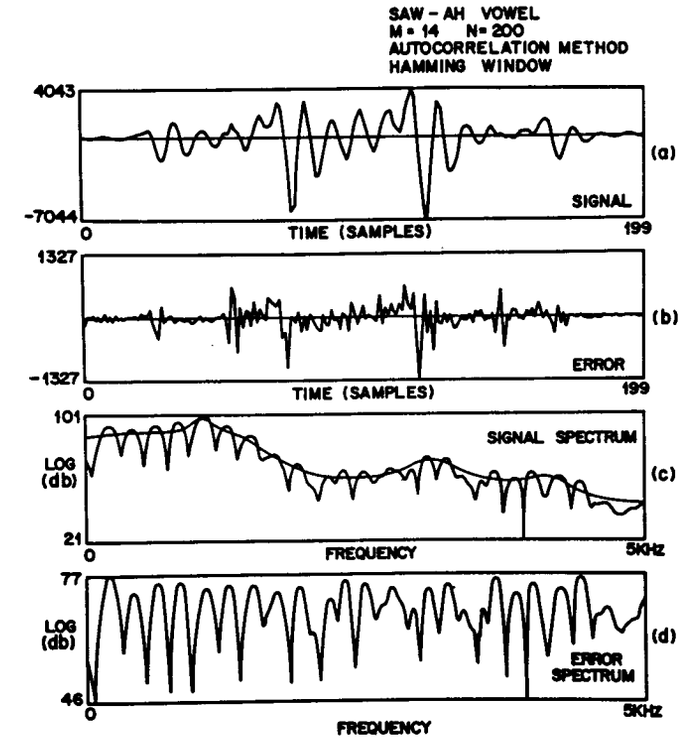



**Figure 3.33** Typical signals and spectra for LPC autocorrelation method for a segment of speech spoken by a female speaker (after Rabiner et al. [8]).

the speech signal. This is the important "whitening" characteristics of the LPC analysis whereby the error signal spectrum is approximately a flat spectrum signal representing the source characteristics rather than those of the vocal tract. Similar behavior is seen in the plots of the vowel from the female talker. Finally, it can be seen that fairly close matches exist between the peaks of the FFT-based signal spectrum and the LPC spectrum.

Figures 3.34–3.36 illustrate some additional properties of the LPC analysis method. Figure 3.34 shows a series of sections of the waveforms (differentiated for preemphasis) for several vowels, and the corresponding prediction error signals. (The prediction error signals have been scaled up in value so as to make their amplitudes comparable to those of the signal; hence, gains of about 4 to 1 were used.) The high-frequency nature of the prediction error signal is seen in all these examples. What can also be seen is that, for many cases, the prediction error signal exhibits sharp pulses at intervals corresponding to the pitch periods of these vowels. This characteristic behavior has been used as the basis

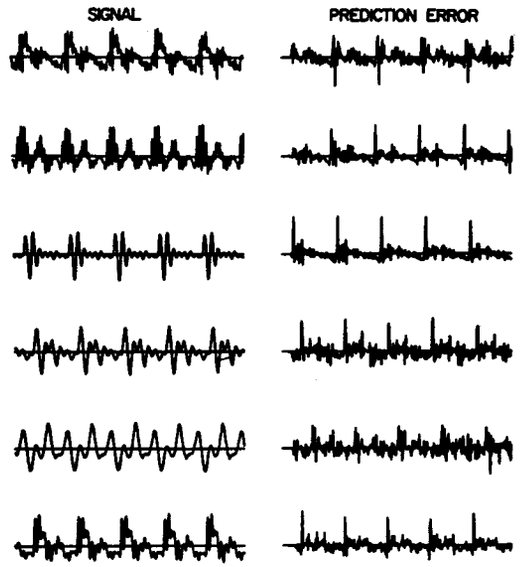



**Figure 3.34** Examples of signal (differentiated) and prediction error for several vowels (after Strube [9]).

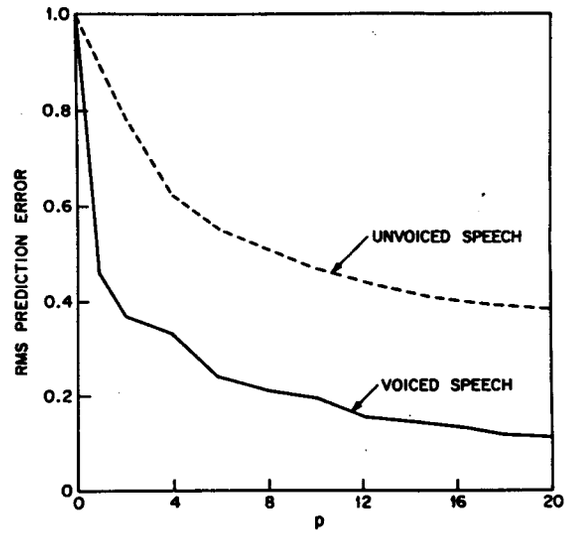



**Figure 3.35** Variation of the RMS prediction error with the number of predictor coefficients, $p$ (after Atal and Hanauer [10]).



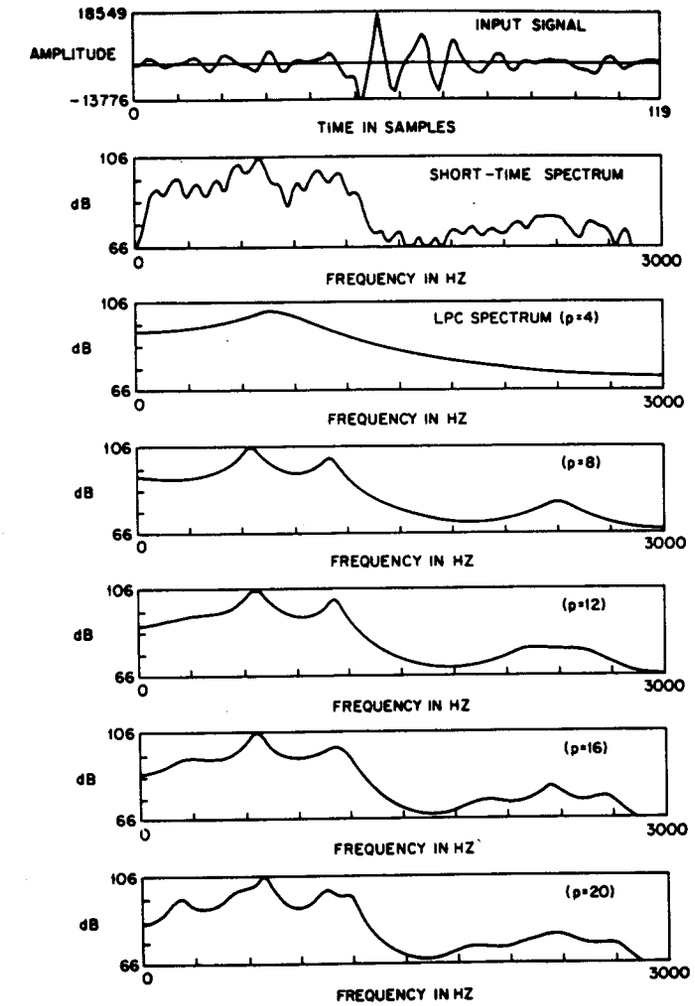



**Figure 3.36** Spectra for a vowel sound for several values of predictor order, $p$.

for several LPC-based pitch period estimation methods.

Figure 3.35 shows the effect of LPC prediction order, $p$, on the RMS prediction error, $E_n$, for both sections of voiced speech (solid curve) and unvoiced speech (dashed curve). The prediction error in the curves is normalized by the signal energy such that at $p = 0$ (i.e., no prediction) $E_n = R_n(0)$. A sharp decrease in normalized prediction error occurs for small values of $p$ (e.g., 1–4); however, beyond this value of $p$ the normalized prediction error decreases much more slowly. It is also seen that the normalized prediction error for unvoiced speech, for a given value of $p$, is significantly higher than for voiced speech. The interpretation of this result is that unvoiced speech is less linearly predictable than voiced speech, a result one would anticipate based on our understanding of the speech-production mechanisms.

Finally, Figure 3.36 shows the effect of prediction order, $p$, on the all-pole spectrum and its ability to match details in the FFT spectrum of the speech segment. Shown in this figure are the input speech segment, the Fourier transform of that segment, and linear predictive spectra for values of $p$ from 4 to 20. It is clear that as $p$ increases, more of the detailed properties of the signal spectrum are preserved in the LPC spectrum. It is equally clear that beyond some value of $p$, the details of the signal spectrum that are preserved are generally irrelevant ones; that is, they do not reflect the relevant spectral resonances or antiresonances of the inherent sound. When the analysis order, $p$, becomes large, the LPC spectrum often tries to fit individual pitch harmonics of the speech signal, thereby resulting in a less parsimonious representation of the sound. On the basis of extensive experimental evaluations, it is generally acknowledged that values of $p$ on the order of 8–10 are reasonable for most speech-recognition applications.

### 3.3.7 LPC Processor for Speech Recognition

At this point, rather than spending more time discussing general properties of LPC methods, we describe the details of the LPC front-end processor that has been widely used in speech-recognition systems. Figure 3.37 shows a block diagram of the LPC processor. The basic steps in the processing include the following:

1. **Preemphasis**—The digitized speech signal, $s(n)$, is put through a low-order digital system (typically a first-order FIR filter), to spectrally flatten the signal and to make it less susceptible to finite precision effects later in the signal processing. The digital system used in the preemphasizer is either fixed or slowly adaptive (e.g., to average transmission conditions, noise backgrounds, or even to average signal spectrum). Perhaps the most widely used preemphasis network is the fixed first-order system:

$$H(z) = 1 - \tilde{a}z^{-1}, \qquad 0.9 \le a \le 1.0. \tag{3.68}$$

In this case, the output of the preemphasis network, $\tilde{s}(n)$, is related to the input to the network, $s(n)$, by the difference equation

$$\tilde{s}(n) = s(n) - \tilde{a}s(n-1). \tag{3.69}$$



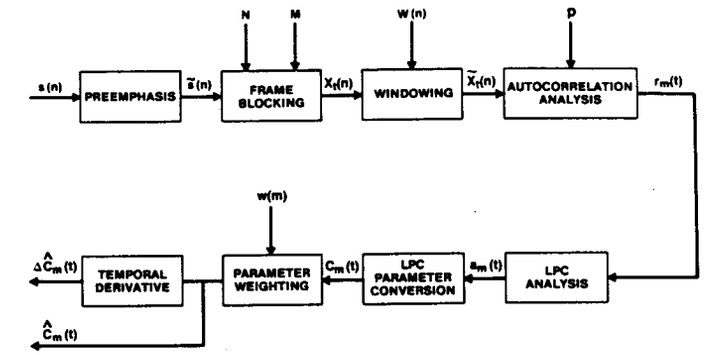



**Figure 3.37** Block diagram of LPC processor for speech recognition.

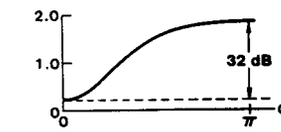



**Figure 3.38** Magnitude spectrum of LPC preemphasis network for $\tilde{a} = 0.95$.

The most common value for $\tilde{a}$ is around 0.95. (For fixed-point implementations a value of $\tilde{a} = 15/16 = 0.9375$ is often used.) A simple example of a first-order *adaptive* preemphasizer is the transfer function

$$H(z) = 1 - \tilde{a}_n z^{-1}, \tag{3.70}$$

where $\tilde{a}_n$ changes with time ($n$) according to the chosen adaptation criterion. One possibility is to choose $\tilde{a}_n = r_n(1)/r_n(0)$. Figure 3.38 shows the magnitude characteristics of $H(e^{j\omega})$ for the value $\tilde{a} = 0.95$. It can be seen that at $\omega = \pi$ (half the sampling rate) there is a 32 dB boost in the magnitude over that at $\omega = 0$.

2. **Frame Blocking**—In this step the preemphasized speech signal, $\tilde{s}(n)$, is blocked into frames of $N$ samples, with adjacent frames being separated by $M$ samples. Figure 3.39 illustrates the blocking into frames for the case in which $M = (1/3)N$. The first illustrated frame consists of the first $N$ speech samples. The second frame begins $M$ samples after the first frame, and overlaps it by $N - M$ samples. Similarly, the third frame begins $2M$ samples after the first frame (or $M$ samples after the second frame) and overlaps it by $N - 2M$ samples. This process continues until all the speech is accounted for within one or more frames. It is easy to see that if $M \le N$, then adjacent frames overlap (as in Figure 3.39), and the resulting LPC spectral estimates will be correlated from frame to frame; if $M \ll N$, then LPC spectral estimates from frame to frame will be quite smooth. On the other hand, if $M > N$, there will be no

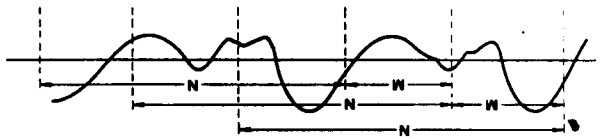



**Figure 3.39** Blocking of speech into overlapping frames.

overlap between adjacent frames; in fact, some of the speech signal will be totally lost (i.e., never appear in any analysis frame), and the correlation between the resulting LPC spectral estimates of adjacent frames will contain a noisy component whose magnitude increases as $M$ increases (i.e., as more speech is omitted from analysis). This situation is intolerable in any practical LPC analysis for speech recognition. If we denote the $\ell^{\text{th}}$ frame of speech by $x_\ell(n)$, and there are $L$ frames within the entire speech signal, then

$$x_\ell(n) = \tilde{s}(M\ell + n), \qquad n = 0, 1, \ldots, N-1, \quad \ell = 0, 1, \ldots, L-1. \tag{3.71}$$

That is, the first frame of speech, $x_0(n)$, encompasses speech samples $\tilde{s}(0), \tilde{s}(1), \ldots, \tilde{s}(N-1)$, the second frame of speech, $x_1(n)$, encompasses speech samples $\tilde{s}(M), \tilde{s}(M+1), \ldots, \tilde{s}(M+N-1)$, and the $L^{\text{th}}$ frame of speech, $x_{L-1}(n)$, encompasses speech samples $\tilde{s}(M(L-1)), \tilde{s}(M(L-1)+1), \ldots, \tilde{s}(M(L-1)+N-1)$. Typical values for $N$ and $M$ are 300 and 100 when the sampling rate of the speech is 6.67 kHz. These correspond to 45-msec frames, separated by 15 msec, or a 66.7-Hz frame rate.

3. **Windowing**—The next step in the processing is to window each individual frame so as to minimize the signal discontinuities at the beginning and end of each frame. The concept here is identical to the one discussed with regard to the frequency domain interpretation of the short-time spectrum in Section 3.2—namely, to use the window to taper the signal to zero at the beginning and end of each frame. If we define the window as $w(n)$, $0 \le n \le N-1$, then the result of windowing is the signal

$$\tilde{x}_\ell(n) = x_\ell(n)w(n), \qquad 0 \le n \le N-1. \tag{3.72}$$

A "typical" window used for the autocorrelation method of LPC (the method most widely used for recognition systems) is the Hamming window, which has the form

$$w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right), \qquad 0 \le n \le N-1. \tag{3.73}$$

4. **Autocorrelation Analysis**—Each frame of windowed signal is next autocorrelated to give

$$r_\ell(m) = \sum_{n=0}^{N-1-m} \tilde{x}_\ell(n)\tilde{x}_\ell(n+m), \qquad m = 0, 1, \ldots, p, \tag{3.74}$$

where the highest autocorrelation value, $p$, is the order of the LPC analysis. Typically, values of $p$ from 8 to 16 have been used, with $p = 8$ being the value used for most systems to be described in this book. A side benefit of the autocorrelation analysis



is that the zeroth autocorrelation, $R_\ell(0)$, is the energy of the $\ell^{\text{th}}$ frame. The frame energy is an important parameter for speech-detection systems and will be discussed further in the next chapter.

5. **LPC Analysis**—The next processing step is the LPC analysis, which converts each frame of $p+1$ autocorrelations into an "LPC parameter set," in which the set might be the LPC coefficients, the reflection (or PARCOR) coefficients, the log area ratio coefficients, the cepstral coefficients, or any desired transformation of the above sets. The formal method for converting from autocorrelation coefficients to an LPC parameter set (for the LPC autocorrelation method) is known as Durbin's method and can formally be given as the following algorithm (for convenience, we will omit the subscript $\ell$ on $r_\ell(m)$):

$$E^{(0)} = r(0) \tag{3.75}$$

$$k_i = \left\{ r(i) - \sum_{j=1}^{L-1} \alpha_j^{(i-1)} r(|i-j|) \right\} \Big/ E^{(i-1)}, \qquad 1 \le i \le p \tag{3.76}$$

$$\alpha_i^{(i)} = k_i \tag{3.77}$$

$$\alpha_j^{(i)} = \alpha_j^{(i-1)} - k_i \alpha_{i-j}^{(i-1)} \tag{3.78}$$

$$E^{(i)} = (1 - k_i^2)E^{(i-1)}, \tag{3.79}$$

where the summation in Eq. (3.76) is omitted for $i = 1$. The set of equations (3.75–3.79) are solved recursively for $i = 1, 2, \ldots, p$, and the final solution is given as

$$a_m = \text{LPC coefficients} = \alpha_m^{(p)}, \qquad 1 \le m \le p \tag{3.80}$$

$$k_m = \text{PARCOR coefficients} \tag{3.81}$$

$$g_m = \text{log area ratio coefficients} = \log\left(\frac{1-k_m}{1+k_m}\right). \tag{3.82}$$

6. **LPC Parameter Conversion to Cepstral Coefficients**—A very important LPC parameter set, which can be derived directly from the LPC coefficient set, is the LPC cepstral coefficients, $c(m)$. The recursion used is

$$c_0 = ln\sigma^2 \tag{3.83a}$$

$$c_m = a_m + \sum_{k=1}^{m-1} \left(\frac{k}{m}\right) c_k a_{m-k}, \qquad 1 \le m \le p \tag{3.83b}$$

$$c_m = \sum_{k=1}^{m-1} \left(\frac{k}{m}\right) c_k a_{m-k}, \qquad m > p, \tag{3.83c}$$

where $\sigma^2$ is the gain term in the LPC model. The cepstral coefficients, which are the coefficients of the Fourier transform representation of the log magnitude spectrum, have been shown to be a more robust, reliable feature set for speech recognition than the LPC coefficients, the PARCOR coefficients, or the log area ratio

coefficients. Generally, a cepstral representation with $Q > p$ coefficients is used, where $Q \simeq \left(\frac{3}{2}\right) p$.

7. **Parameter Weighting**—Because of the sensitivity of the low-order cepstral coefficients to overall spectral slope and the sensitivity of the high-order cepstral coefficients to noise (and other forms of noiselike variability), it has become a standard technique to weight the cepstral coefficients by a tapered window so as to minimize these sensitivities. A formal way of justifying the use of a cepstral window is to consider the Fourier representation of the log magnitude spectrum and the differentiated (in frequency) log magnitude spectrum, such that

$$\log |S(e^{j\omega})| = \sum_{m=-\infty}^{\infty} c_m e^{-j\omega m} \tag{3.84}$$

$$\frac{\partial}{\partial \omega} [\log |S(e^{j\omega})|] = \sum_{m=-\infty}^{\infty} (-jm) c_m e^{-j\omega m}. \tag{3.85}$$

The differential log magnitude spectrum has the property that any fixed spectral slope in the log magnitude spectrum becomes a constant; furthermore, any prominent spectral peak in the log magnitude spectrum (e.g., the formants) is well preserved as a peak in the differentiated log magnitude spectrum. Hence, by considering the multiplication by $(-jm)$ in the representation of the differentiated log magnitude spectrum as a form of weighting, we get

$$\frac{\partial}{\partial \omega} [\log |S(e^{j\omega})|] = \sum_{m=-\infty}^{\infty} \hat{c}_m e^{-j\omega m}, \tag{3.86}$$

where

$$\hat{c}_m = c_m(-jm). \tag{3.87}$$

To achieve the robustness for large values of $m$ (i.e., low weight near $m = Q$) and to truncate the infinite computation of Eq. (3.86), we must consider a more general weighting of the form

$$\hat{c}_m = w_m c_m, \qquad 1 \le m \le Q, \tag{3.88}$$

where an appropriate weighting is the bandpass lifter (filter in the cepstral domain)

$$w_m = \left[1 + \frac{Q}{2} \sin\left(\frac{\pi m}{Q}\right)\right], \qquad 1 \le m \le Q. \tag{3.89}$$

This weighting function truncates the computation and de-emphasizes $c_m$ around $m = 1$ and around $m = Q$.

8. **Temporal Cepstral Derivative**—The cepstral representation of the speech spectrum provides a good representation of the local spectral properties of the signal for the given analysis frame [11]. An improved representation can be obtained by extending the analysis to include information about the temporal cepstral derivative (both first and second derivatives have been investigated and found to improve the performance of speech-recognition systems). To introduce temporal order into the



cepstral representation, we denote the $m^{\text{th}}$ cepstral coefficient at time $t$ by $c_m(t)$. Of course, in practice, the sampling time $t$ refers to the analysis frame rather than an arbitrary time instance. The way in which the cepstral time derivative is approximated is as follows: The time derivative of the log magnitude spectrum has a Fourier series representation of the form

$$\frac{\partial}{\partial t} [\log |S(e^{j\omega}, t)|] = \sum_{m=-\infty}^{\infty} \frac{\partial c_m(t)}{\partial t} e^{-j\omega m}. \tag{3.90}$$

Hence, the temporal cepstral derivative must be determined in an appropriate manner. It is well known that since $c_m(t)$ is a discrete time representation (where $t$ is the frame index), simply using a first- or second-order difference is inappropriate to approximate the derivative (it is very noisy). Hence, a reasonable compromise is to approximate $\partial c_m(t)/\partial t$ by an orthogonal polynomial fit (a least-squares estimate of the derivative) over a finite length window; that is,

$$\frac{\partial c_m(t)}{\partial t} = \Delta c_m(t) \approx \mu \sum_{k=-K}^{K} k c_m(t+k), \tag{3.91}$$

where $\mu$ is an appropriate normalization constant and $(2K+1)$ is the number of frames over which the computation is performed. Typically, a value of $K = 3$ has been found appropriate for computation of the first-order temporal derivative. Based on the computations described above, for each frame $t$, the result of the LPC analysis is a vector of $Q$ weighted cepstral coefficients and an appended vector of $Q$ cepstral time derivatives; that is,

$$\mathbf{o}_t' = (\hat{c}_1(t), \hat{c}_2(t), \ldots, \hat{c}_Q(t), \Delta c_1(t), \Delta c_2(t), \ldots, \Delta c_Q(t)), \tag{3.92}$$

where $\mathbf{o}_t$ is a vector with $2Q$ components and $'$ denotes matrix transpose. Similarly, if second-order temporal derivatives are computed (giving $\Delta^2 c_m(t)$), these are appended to $\mathbf{o}_t$ giving a vector with $3Q$ components (see Section 4.6 for more details).

### 3.3.8 Review Exercises

**Exercise 3.5**

To illustrate LPC analysis via the autocorrelation method, consider a predictor of order $p = 2$. Assume an autocorrelation vector with components $R = (r(0), r(1), r(2))$. Use the Durbin method, described in the previous section, to solve for the LPC coefficients $a_1$ and $a_2$ in terms of the $R$s. Check your answer by solving the matrix equation

$$\begin{bmatrix} r(0) & r(1) \\ r(1) & r(0) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} r(1) \\ r(2) \end{bmatrix}$$

using simple matrix algebra.

**Solution 3.5**

Using the Durbin method, we get the following steps:

$$E^{(0)} = r(0)$$

$$k_1 = r(1)/r(0)$$
$$\alpha_1^{(1)} = r(1)/r(0)$$
$$E^{(1)} = (r^2(0) - r^2(1))/r(0)$$

$$k_2 = (r(2)r(0) - r^2(1))/(r^2(0) - r^2(1))$$
$$\alpha_2^{(2)} = (r(2)r(0) - r^2(1))/(r^2(0) - r^2(1))$$
$$\alpha_1^{(2)} = (r(1)r(0) - r(1)r(2))/(r^2(0) - r^2(1))$$

$$a_1 = \alpha_1^{(2)}$$
$$a_2 = \alpha_2^{(2)}$$

Using matrix algebra we get

$$a_1 r(0) + a_2 r(1) = r(1)$$
$$a_1 r(1) + a_2 r(0) = r(2)$$

Solving directly for $a_1$ and $a_2$ we get

$$a_1 = (r(1)r(0) - r(1)r(2))/(r^2(0) - r^2(1))$$
$$a_2 = (r(2)r(0) - r^2(1))/(r^2(0) - r^2(1))$$

which is the same result as obtained via the Durbin method.

**Exercise 3.6**

Consider two (windowed) speech sequences $x(n)$ and $\hat{x}(n)$ both defined for $0 \le n \le N-1$. (Outside this region both sequences are defined to be 0.) We perform an LPC analysis (using the autocorrelation method) on each frame. Thus, from the autocorrelation sequences

$$r(k) = \sum_{n=0}^{N-1-k} x(n)x(n+k), \qquad 0 \le k \le p$$

$$\hat{r}(k) = \sum_{n=0}^{N-1-k} \hat{x}(n)\hat{x}(n+k), \qquad 0 \le k \le p$$

we solve for the predictor parameter $\mathbf{a}' = (a_0, a_1, \ldots, a_p)$ and $\hat{\mathbf{a}}' = (\hat{a}_0, \hat{a}_1, \ldots, \hat{a}_p)$ ($a_0 = \hat{a}_0 = -1$) where $'$ denotes matrix transpose.

1. Show that the prediction error (residual), defined as

$$E^{(p)} = \sum_{n=0}^{N-1+p} e^2(n) = \sum_{n=0}^{N-1-p} \left[ -\sum_{i=0}^{p} a_i x(n-i) \right]^2$$

can be written in the form

$$E^{(p)} = \mathbf{a}'\mathbf{R}_x\mathbf{a},$$



where $\mathbf{R}_x$ is a $(p+1)$ by $(p+1)$ matrix. Determine $\mathbf{R}_x$.

2. Consider passing the sequence $\hat{x}(n)$ through the inverse LPC system with LPC coefficients $\mathbf{a}$, to give the error signal $\tilde{e}(n)$, defined as

$$\tilde{e}(n) = -\sum_{i=0}^{p} a_i \hat{x}(n-i).$$

Show that the mean-squared error, $\tilde{E}^{(p)}$, defined by

$$\tilde{E}^{(p)} = \sum_{n=0}^{N-1+p} [\tilde{e}(n)]^2$$

can be written in the form

$$\tilde{E}^{(p)} = \mathbf{a}'\mathbf{R}_{\hat{x}}\mathbf{a},$$

where $\mathbf{R}_{\hat{x}}$ is a $(p+1)$ by $(p+1)$ matrix. Determine $\mathbf{R}_{\hat{x}}$.

3. If we form the ratio

$$D = \frac{\tilde{E}^{(p)}}{E^{(p)}}$$

what can be said about the range of $D$?

(This exercise gives an initial appreciation of the concept of distortion measures. Chapter 4 discusses this topic in great detail.)

**Solution 3.6**

1. Since

$$e(n) = -\sum_{i=0}^{p} a_i x(n-i)$$

$$E^{(p)} = \sum_{n=0}^{N-1+p} e^2(n) + \sum_{n=0}^{N-1+p} \left[ -\sum_{i=0}^{p} a_i x(n-i) \right] \left[ -\sum_{j=0}^{p} a_j x(n-j) \right]$$
$$= \sum_{i=0}^{p} a_i \sum_{j=0}^{p} a_j \sum_{n=0}^{N-1+p} x(n-i)x(n-j).$$

But

$$\sum_{n=0}^{N-1+p} x(n-i)x(n-j) = \sum_{n=0}^{N-1+p} x(n)x(n-j+i) = r(|i-j|).$$

Thus

$$E^{(p)} = \sum_{i=0}^{p} a_i \sum_{j=0}^{p} a_j r(|i-j|) = \mathbf{a}'\mathbf{R}_x\mathbf{a},$$

where

$$\mathbf{R}_x = \begin{bmatrix} r(0) & r(1) & \cdots & r(p) \\ r(1) & r(0) & \cdots & r(p-1) \\ \vdots & \vdots & & \vdots \\ r(p) & r(p-1) & \cdots & r(0) \end{bmatrix}$$

2. $\tilde{e}(n) = -\sum_{i=0}^{p} a_i \hat{x}(n-i)$

Repeating the derivation of part 1, we get

$$\tilde{E}^{(p)} = \sum_{i=0}^{p} a_i \sum_{j=0}^{p} a_j \hat{r}(|i-j|) = \mathbf{a}'\mathbf{R}_{\hat{x}}\mathbf{a},$$

where

$$\mathbf{R}_{\hat{x}} = \begin{bmatrix} \hat{r}(0) & \hat{r}(1) & \cdots & \hat{r}(p) \\ \hat{r}(1) & \hat{r}(0) & \cdots & \hat{r}(p-1) \\ \vdots & \vdots & & \vdots \\ \hat{r}(p) & \hat{r}(p-1) & \cdots & \hat{r}(0) \end{bmatrix}$$

3. $D = \dfrac{\tilde{E}^{(p)}}{E^{(p)}} = \dfrac{\mathbf{a}'\mathbf{R}_x\mathbf{a}}{\mathbf{a}'\mathbf{R}_{\hat{x}}\mathbf{a}}$ Since $D$ is a ratio of prediction residuals, and since $E^{(p)}$ is the *minimum* prediction residual for LPC system $\mathbf{a}$, then $\tilde{E}^{(p)}$ must be greater than (or equal to) $E^{(p)}$. Therefore

$$D \geq 1.0$$

**Exercise 3.7**

A proposed measure of spectral distance between two frames of speech represented by LPC coefficient sets $\mathbf{a}$ and $\hat{\mathbf{a}}$, and augmented autocorrelation matrices $\mathbf{R}_x$ and $\mathbf{R}_{\hat{x}}$ (see Exercise 3.6) is:

$$D(\mathbf{a}, \hat{\mathbf{a}}) = \frac{\mathbf{a}'\mathbf{R}_{\hat{x}}\mathbf{a}}{\hat{\mathbf{a}}'\mathbf{R}_{\hat{x}}\hat{\mathbf{a}}}$$

1. Show that the distance function $D(\mathbf{a}, \hat{\mathbf{a}})$ can be written in the computationally efficient form

$$D(\mathbf{a}, \hat{\mathbf{a}}) = \left[ \frac{(r_a(0)\hat{r}(0) + 2\sum_{i=1}^{p} r_a(i)\hat{r}(i))}{\hat{\mathbf{a}}'\mathbf{R}_{\hat{x}}\hat{\mathbf{a}}} \right],$$

where $r_a(i)$ is the autocorrelation of the $\mathbf{a}$ array, i.e.,

$$r_a(i) = \sum_{j=0}^{p-i} a_j a_{j+i}, \qquad 0 \leq i \leq p.$$

2. Assume that the quantities (i.e., vectors, matrices, scalars) $\mathbf{a}, \hat{\mathbf{a}}, \mathbf{R}_x, \mathbf{R}_{\hat{x}}, \hat{\mathbf{a}}'\mathbf{R}_{\hat{x}}\hat{\mathbf{a}}, \mathbf{R}_{\hat{x}}$ and $\mathbf{R}_a$ are precomputed; that is, they are available at the time the distance calculation is required. Contrast the computation required to evaluate $D(\mathbf{a}, \hat{\mathbf{a}})$ using both expressions for $D$ given in this exercise.

**Solution 3.7**

We have that

$$D(\mathbf{a}, \hat{\mathbf{a}}) = \frac{\mathbf{a}'\mathbf{R}_{\hat{x}}, \mathbf{a}}{\hat{\mathbf{a}}'\mathbf{R}_{\hat{x}}\hat{\mathbf{a}}} = \frac{\tilde{E}^{(p)}}{\hat{E}^{(p)}}.$$



From Exercise 3.6 we get

$$\tilde{E}^{(p)} = \sum_{i=0}^{p} a_i \sum_{j=0}^{p} a_j \hat{r}(|j-i|).$$

Letting $k = j - i$ ($j = k + i$) we get

$$\tilde{E}^{(p)} = \sum_{i=0}^{p} a_i \sum_{k=-i}^{p-i} a_{k+i} \hat{r}(|k|).$$

By rearranging the summations on $i$ and $k$ and by recognizing that $a_\ell = 0$, $\ell < 0$ and $a_\ell = 0$, $\ell > p$, we can complete the square by summing on $k$ from $-p$ (the smallest value of $k$) to $+p$ (the largest value of $k$), giving

$$\tilde{E}^{(p)} = \sum_{k=-p}^{p} \left[ \sum_{i=0}^{p-|k|} a_i a_{k+i} \right] \hat{r}(|k|).$$

The inner summation is defined as $r_a(k)$, hence

$$\tilde{E}^{(p)} = \sum_{k=-p}^{p} r_a(k)\hat{r}(|k|).$$

Since $r_a(k) = r_a(-k)$ and $\hat{r}(k) = \hat{r}(-k)$ we can write $\tilde{E}^{(p)}$ as

$$\tilde{E}^{(p)} = r_a(0)\hat{r}(0) + 2\sum_{k=1}^{p} r_a(k)\hat{r}(k).$$

2 Since all the individual quantities are precomputed, to evaluate $D$ as a ratio of residuals; that is,

$$D = \frac{\mathbf{a}'\mathbf{R}_{\hat{x}}\mathbf{a}}{\hat{\mathbf{a}}'\mathbf{R}_{\hat{x}}\hat{\mathbf{a}}}$$

requires

**a.** $(p+1) \times (p+2)$ multiplies and adds to multiply $\mathbf{a}'$ by $\mathbf{R}_{\hat{x}}$ and then multiply the result by $\mathbf{a}$.

**b.** 1 divide to give $D$ since $\hat{\mathbf{a}}'\mathbf{R}_{\hat{x}}\hat{\mathbf{a}}$ is a precomputed scalar.

For the alternative method of evaluating $D$, as discussed in part 1 of this exercise, we require:

**a.** $(p+1)$ multiplies and adds to give the product $\hat{r}(k)r_a(k)$ for $1 \leq k \leq p$ and to give $\hat{r}(0)r_a(0)$.

**b.** 1 divide to give $D$ since $\hat{\mathbf{a}}'\mathbf{R}_{\hat{x}}\hat{\mathbf{a}}$ is a precomputed scalar.

Thus, neglecting the divide, the alternative computation of $D$ requires a factor of $(p+2)$ less computation and therefore is significantly more efficient than direct computation of the ratio of prediction residuals.

### 3.3.9 Typical LPC Analysis Parameters

The computation of the LPC analysis system of Figure 3.37 is specified by a number of variable parameters, including

$N$ number of samples in the analysis frame
$M$ number of samples shift between frames
$p$ LPC analysis order
$Q$ dimension of LPC derived cepstral vector
$K$ number of frames over which cepstral time derivatives are computed.

Although each of these parameters can be varied over a wide range of values, the following table gives typical values for analysis systems at three different sampling rates (6.67 kHz, 8 kHz, 10 kHz).

Typical Values of LPC Analysis Parameters for Speech-Recognition Systems

| parameter | $F_s = 6.67$ kHz | | $F_s = 8$ kHz | | $F_s = 10$ kHz | |
|---|---|---|---|---|---|---|
| $N$ | 300 | (45 msec) | 240 | (30 msec) | 300 | (30 msec) |
| $M$ | 100 | (15 msec) | 80 | (10 msec) | 100 | (10 msec) |
| $p$ | 8 | | 10 | | 10 | |
| $Q$ | 12 | | 12 | | 12 | |
| $K$ | 3 | | 3 | | 3 | |

## 3.4 VECTOR QUANTIZATION

The results of either the filter-bank analysis or the LPC analysis are a series of vectors characteristic of the time-varying spectral characteristics of the speech signal. For convenience, we denote the spectral vectors as $\mathbf{v}_\ell$, $\ell = 1, 2, \ldots, L$, where typically each vector is a $p$-dimensional vector. If we compare the information rate of the vector representation to that of the raw (uncoded) speech waveform, we see that the spectral analysis has significantly reduced the required information rate. Consider, for example, 10-kHz sampled speech with 16-bit speech amplitudes. A raw signal information rate of 160,000 bps is required to store the speech samples in uncompressed format. For the spectral analysis, consider vectors of dimension $p = 10$ using 100 spectral vectors per second. If we again represent each spectral component to 16-bit precision, the required storage is about $100 \times 10 \times 16$ bps, or 16,000 bps—about a 10-to-1 reduction over the uncompressed signal. Such compressions in storage rate are impressive. Based on the concept of ultimately needing only a single spectral representation for each basic speech unit, it may be possible to further reduce the raw spectral representation of speech to those drawn from a small, finite number of "unique" spectral vectors, each corresponding to one of the basic speech units (i.e., the phonemes). This ideal representation is, of course, impractical, because there is so much variability in the spectral properties of each of the basic speech units. However, the concept of building a codebook of "distinct" analysis vectors, albeit with significantly more code words than the basic set of phonemes, remains an attractive idea and is the basis behind a set of techniques commonly called vector quantization (VQ) methods. Based on this line of reasoning, assume that we require a codebook with about 1024 unique spectral vectors



(i.e., about 25 variants for each of the 40 basic speech units). Then to represent an arbitrary spectral vector all we need is a 10-bit number—the index of the codebook vector that best matches the input vector. Assuming a rate of 100 spectral vectors per second, we see that a total bit rate of about 1000 bps is required to represent the spectral vectors of a speech signal. This rate is about $1/16^{\text{th}}$ the rate required by the continuous spectral vectors. Hence the VQ representation is potentially an extremely efficient representation of the spectral information in the speech signal. This is one of the main reasons for the interest in VQ methods.

Before discussing the concepts involved in designing and implementing a practical VQ system, we first discuss the advantages and disadvantages of this type of representation. The key advantages of the VQ representation are

- reduced storage for spectral analysis information. We have already shown that the VQ representation is potentially very efficient. This efficiency can be exploited in a number of ways in practical VQ-based speech-recognition systems.
- reduced computation for determining similarity of spectral analysis vectors. In speech recognition a major component of the computation is the determination of spectral similarity between a pair of vectors. Based on the VQ representation, this spectral similarity computation is often reduced to a table lookup of similarities between pairs of codebook vectors.
- discrete representation of speech sounds. By associating a phonetic label (or possibly a set of phonetic labels or a phonetic class) with each codebook vector, the process of choosing a best codebook vector to represent a given spectral vector becomes equivalent to assigning a phonetic label to each spectral frame of speech. A range of recognition systems exist that exploit these labels so as to recognize speech in an efficient manner.

The disadvantages of the use of a VQ codebook to represent speech spectral vectors are

- an inherent spectral distortion in representing the actual analysis vector. Since there is only a finite number of codebook vectors, the process of choosing the "best" representation of a given spectral vector inherently is equivalent to quantizing the vector and leads, by definition, to a certain level of quantization error. As the size of the codebook increases, the size of the quantization error decreases. However, with any finite codebook there will always be some nonzero level of quantization error.
- the storage required for codebook vectors is often nontrivial. The larger we make the codebook (so as to reduce quantization error), the more storage is required for the codebook entries. For codebook sizes of 1000 or larger, the storage is often nontrivial. Hence an inherent trade-off among quantization error, processing for choosing the codebook vector, and storage of codebook vectors exists, and practical designs balance each of these three factors.

### 3.4.1 Elements of a Vector Quantization Implementation

To build a VQ codebook and implement a VQ analysis procedure, we need the following:

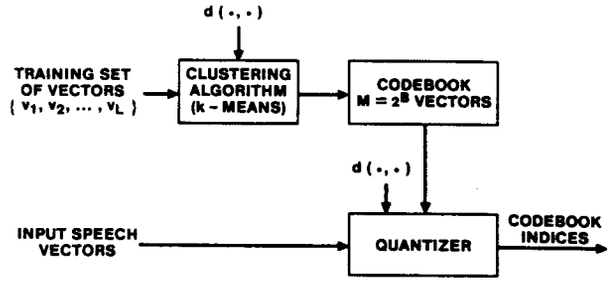



**Figure 3.40** Block diagram of the basic VQ training and classification structure.

1. a large set of spectral analysis vectors, $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_L$, which form a training set. The training set is used to create the "optimal" set of codebook vectors for representing the spectral variability observed in the training set. If we denote the size of the VQ codebook as $M = 2^B$ vectors (we call this a $B$-bit codebook), then we require $L \gg M$ so as to be able to find the best set of $M$ codebook vectors in a robust manner. In practice, it has been found that $L$ should be at least $10M$ in order to train a VQ codebook that works reasonably well.
2. a measure of similarity, or distance, between a pair of spectral analysis vectors so as to be able to cluster the training set vectors as well as to associate or classify arbitrary spectral vectors into unique codebook entries. We denote the spectral distance, $d(\mathbf{v}_i, \mathbf{v}_j)$, between two vectors $\mathbf{v}_i$ and $\mathbf{v}_j$ as $d_{ij}$. We defer a discussion of spectral distance measures to Chapter 4.
3. a centroid computation procedure. On the basis of the partitioning that classifies the $L$ training set vectors into $M$ clusters we choose the $M$ codebook vectors as the centroid of each of the $M$ clusters.
4. a classification procedure for arbitrary speech spectral analysis vectors that chooses the codebook vector closest to the input vector and uses the codebook index as the resulting spectral representation. This is often referred to as the nearest-neighbor labeling or optimal encoding procedure. The classification procedure is essentially a quantizer that accepts, as input, a speech spectral vector and provides, as output, the codebook index of the codebook vector that best matches the input.

Figure 3.40 shows a block diagram of the basic VQ training and classification structure. In the following sections we discuss each element of the VQ structure in more detail.

### 3.4.2 The VQ Training Set

To properly train the VQ codebook, the training set vectors should span the anticipated range of the following:



- talkers, including ranges in age, accent, gender, speaking rate, levels, and other variables.
- speaking conditions, such as quiet room, automobile, and noisy workstation.
- transducers and transmission systems, including wideband microphones, telephone handsets (with both carbon and electret microphones), direct transmission, telephone channel, wideband channel, and other devices.
- speech units including specific-recognition vocabularies (e.g., digits) and conversational speech.

The more narrowly focused the training set (i.e., limited talker populations, quiet room speaking, carbon button telephone over a standard telephone channel, vocabulary of digits) the smaller the quantization error in representing the spectral information with a fixed-size codebook. However, for applicability to a wide range of problems, the training set should be as broad, in each of the above dimensions, as possible.

### 3.4.3 The Similarity or Distance Measure

The spectral distance measure for comparing spectral vectors $\mathbf{v}_i$ and $\mathbf{v}_j$ is of the form

$$d(\mathbf{v}_i, \mathbf{v}_j) = d_{ij} \begin{cases} = 0 & \text{if } \mathbf{v}_i = \mathbf{v}_j \\ > 0 & \text{otherwise} \end{cases} . \tag{3.93}$$

As we will see in Chapter 4, the distance measure commonly used for comparing filter-bank vectors is an $L_1$, $L_2$, or covariance weighted spectral difference, whereas for LPC vectors (and related feature sets such as LPC derived cepstral vectors), measures such as the likelihood and cepstral distance measures are generally used.

### 3.4.4 Clustering the Training Vectors

The way in which a set of $L$ training vectors can be clustered into a set of $M$ codebook vectors is the following (this procedure is known as the generalized Lloyd algorithm or the $K$-means clustering algorithm):

1. Initialization: Arbitrarily choose $M$ vectors (initially out of the training set of $L$ vectors) as the initial set of code words in the codebook.
2. Nearest-Neighbor Search: For each training vector, find the code word in the current codebook that is closest (in terms of spectral distance), and assign that vector to the corresponding cell (associated with the closest code word).
3. Centroid Update: Update the code word in each cell using the centroid of the training vectors assigned to that cell.
4. Iteration: Repeat steps 2 and 3 until the average distance falls below a preset threshold.

Figure 3.41 illustrates the result of designing a VQ codebook by showing the partitioning of a (2-dimensional) spectral vector space into distinct regions, each of which is

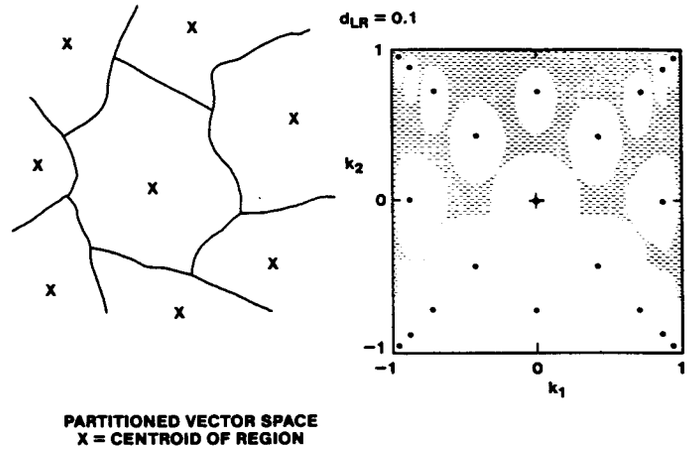



**Figure 3.41** Partitioning of a vector space into VQ cells with each cell represented by a centroid vector.

represented by a centroid vector. The shape of each partitioned cell is highly dependent on the spectral distortion measure and the statistics of the vectors in the training set. (For example, if a Euclidean distance is used, the cell boundaries are hyperplanes.)

Although the above iterative procedure works well, it has been shown that it is advantageous to design an $M$-vector codebook in stages—i.e., by first designing a 1-vector codebook, then using a splitting technique on the code words to initialize the search for a 2-vector codebook, and continuing the splitting process until the desired $M$-vector codebook is obtained. This procedure is called the binary split algorithm and is formally implemented by the following procedure:

1. Design a 1-vector codebook; this is the centroid of the entire set of training vectors (hence, no iteration is required here).
2. Double the size of the codebook by splitting each current codebook $\mathbf{y}_n$ according to the rule

$$\begin{aligned} \mathbf{y}_n^+ &= \mathbf{y}_n(1+\epsilon) \\ \mathbf{y}_n^- &= \mathbf{y}_n(1-\epsilon), \end{aligned} \tag{3.94}$$

   where $n$ varies from 1 to the current size of the codebook, and $\epsilon$ is a splitting parameter (typically $\epsilon$ is chosen in the range $0.01 \le \epsilon \le 0.05$).
3. Use the $K$-means iterative algorithm (as discussed above) to get the best set of centroids for the split codebook (i.e., the codebook of twice the size).
4. Iterate steps 2 and 3 until a codebook of size $M$ is designed.



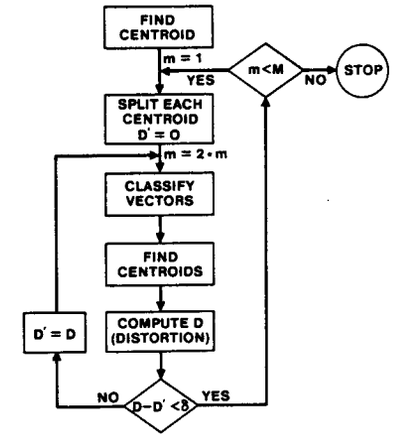



**Figure 3.42** Flow diagram of binary split codebook generation algorithm.

Figure 3.42 shows, in a flow diagram, the detailed steps of the binary split VQ codebook generation technique. The box labeled "Classify Vectors" is the nearest-neighbor search procedure, and the box labeled "Find Centroids" is the centroid update procedure of the $K$-means algorithm. The box labeled "Compute $D$ (Distortion)" sums the distances of all training vectors in the nearest-neighbor search so as to determine whether the procedure has converged (i.e., $D = D'$ of the previous iteration).

To illustrate the effect of codebook size (i.e., number of codebook vectors) on average training set distortion, Figure 3.43 [12] shows experimentally measured values of distortion (in terms of the likelihood ratio measure and the equivalent dB values; see Chapter 4 for more details) versus codebook size (as measured in bits per frame, $B$) for vectors of both voiced and unvoiced speech. It can be seen that very significant reductions in distortion are achieved in going from a codebook size of 1 bit (2 vectors) to about 7 bits (128 vectors) for both voiced and unvoiced speech. Beyond this point, reductions in distortion are much smaller.

One initial motivation for considering the use of a VQ codebook was the assumption that, in the limit, the codebook should ideally have about 40 vectors—i.e., one vector per speech sound. However, since the codebook vectors represent short time spectral measurements, there is inherently a certain degree of variability in specific codebook entries. Figure 3.44 shows a comparison of codebook vector locations in the $F_1 - F_2$ plane for a 32-vector codebook, along with the vowel ellipses discussed in Chapter 2. (The 32 codewords were generated from a training set of conversational speech spoken by a set of male talkers. The training set included both speech and background signals.) It can be seen that the correspondence between codebook vector location and vowel location is weak. Furthermore, there appears to be a tendency to cluster around the neutral vowel /ɜ/.

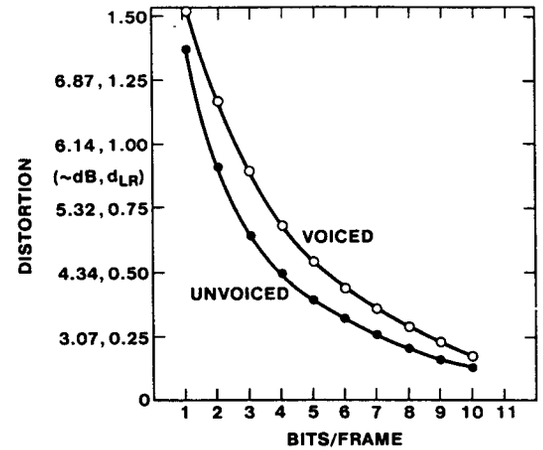



**Figure 3.43** Codebook distortion versus codebook size (measured in bits per frame) for both voiced and unvoiced speech (after Juang et al. [12]).

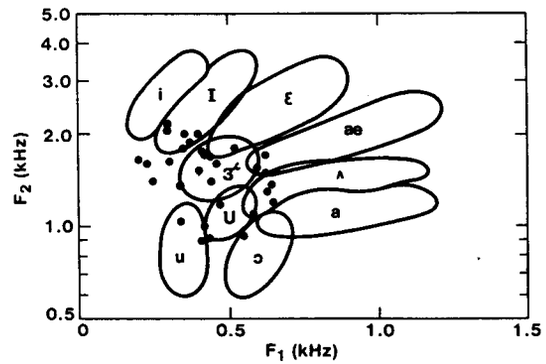



**Figure 3.44** Codebook vector locations in the $F_1 - F_2$ plane (for a 32-vector codebook) superimposed on the vowel ellipses (after Juang et al. [12]).

This can be attributed, in part, to both the distortion measure and to the manner in which spectral centroids are computed.

### 3.4.5 Vector Classification Procedure

The classification procedure for arbitrary spectral vectors is basically a full search through the codebook to find the "best" match. Thus if we denote the codebook vectors of an



$M$-vector codebook as $\mathbf{y}_m$, $1 \le m \le M$, and we denote the spectral vector to be classified (and quantized) as $\mathbf{v}$, then the index, $m^*$, of the best codebook entry is

$$m^* = \arg \min_{1 \le m \le M} d(\mathbf{v}, \mathbf{y}_m). \tag{3.95}$$

For codebooks with large values of $M$ (e.g., $M \ge 1024$), the computation of Eq. (3.95) could be excessive, depending on the exact details of the distance measure; hence, alternative, suboptimal, procedures for designing VQ codebooks have been investigated. We will briefly discuss such methods in a later section of this chapter.

### 3.4.6 Comparison of Vector and Scalar Quantizers

To illustrate the power of the concept of quantizing an entire vector (rather than quantizing individual components of the vector), Figures 3.45 and 3.46 show comparisons of the results of using vector and scalar quantizers on typical speech spectral frames. In Figure 3.45 we see both model (speech) spectra and the resulting quantization error spectrum for 10-bit and 24-bit scalar quantizers and for a 10-bit vector quantizer. It is clear that the quantization error of the 10-bit vector quantizer is comparable to that of the 24-bit scalar quantizer. This implies that the vector quantizer provides a 14-bit reduction in storage (per frame) over a scalar quantizer, i.e., more than a 50% reduction in storage for the same distortion.

Figure 3.46 shows temporal plots of distortion as well as distortion error histograms for the three quantizers of Figure 3.45. It can be seen that even though the average distortion of the 10-bit VQ is comparable to that of the 24-bit scalar quantizer, the peak distortion of the 10-bit VQ is much smaller than the peak distortion of the 24-bit scalar quantizer. This represents another distinct advantage of VQ over scalar quantization.

### 3.4.7 Extensions of Vector Quantization

As mentioned earlier, several straightforward extensions of the ideas of VQ have been proposed and studied, including the following:

1. Use of multiple codebooks in which codebooks are created separately (and independently) for each of several spectral (or temporal) representations of speech. Thus we might consider using a separate codebook for cepstral vectors and a separate codebook for the time derivatives of the cepstral vectors. This method of multiple codebooks has been used extensively in large vocabulary speech-recognition systems.
2. Binary search trees in which a series of suboptimal VQs is used to limit the search space so as to reduce the computation of the overall VQ from $M$ distances to log $(M)$ distances. The training procedure first designs an optimal $M = 2$ VQ and then assigns all training vectors to one of the VQ cells. Next the procedure designs a pair of $M = 2$ VQs, one for each subset of the preceding stage. This process is iterated until the desired size is obtained in log $M$ steps. The suboptimality of the procedure is related to the fact that training vectors initially split along one branch of the VQ cannot join the other branch at a later stage of processing; hence, the overall

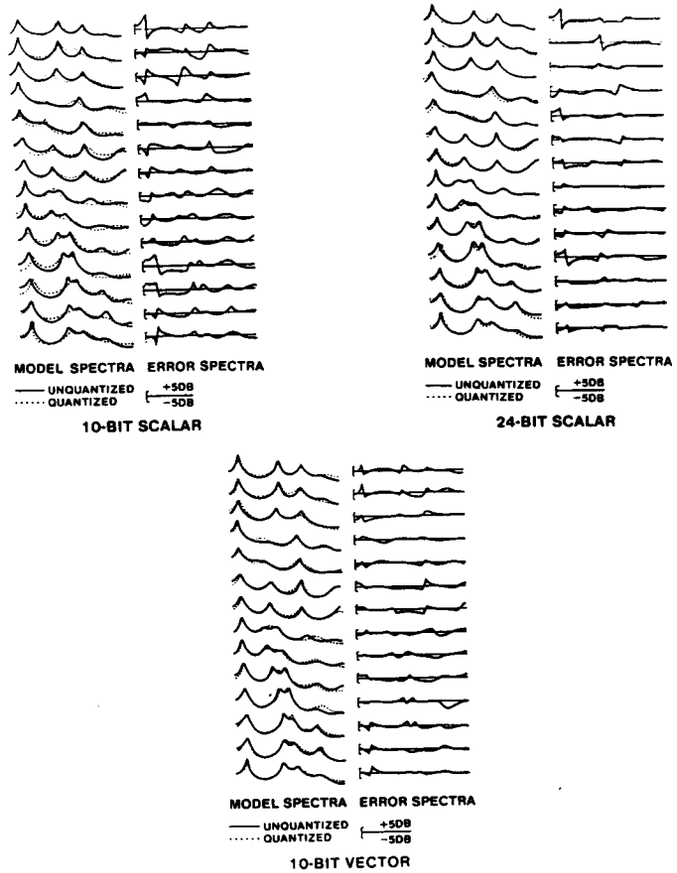



**Figure 3.45** Model and distortion error spectra for scalar and vector quantizers (after Juang et al. [12]).

distortion is not minimal at each branch of the tree.

3. $K$-tuple (fixed-length block) quantizers in which $K$-frames of speech are coded at a time, rather than single frames, as is conventionally the case. The idea is to exploit correlations in time for vowels and vowel-like sounds. The disadvantage occurs for sounds where the correlation along the $K$-tuple is low—i.e., transient sounds and many consonants.
4. Matrix quantization in which a codebook of sounds or words of variable sequence



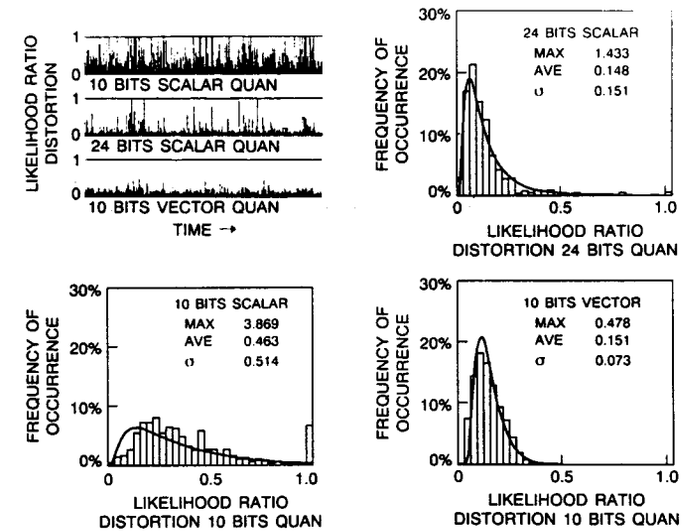



**Figure 3.46** Plots and histograms of temporal distortion for scalar and vector quantizers (after Juang et al. [12]).

length is created. The concept here is to handle time variability via some types of dynamic programming procedure and thereby create a codebook of sequences of vectors that represent typical sounds or words. Such techniques are most applicable to word-recognition systems.

5. Trellis codes in which time sequential dependencies among codebook entries are explicitly determined as part of the training phase. The idea here is that when input vector $\mathbf{v}_n$ is quantized using codeword $\mathbf{y}_\ell$, then input vector $\mathbf{v}_{n+1}$ is quantized using one of a limited subset of codebook entries that are related to $\mathbf{y}_\ell$ via a set of learned sequential constraints, thereby reducing computation of encoding the input, and increasing the ability to interpret the codebook output in terms of basic speech units.
6. Hidden Markov models in which both time and spectral constraints are used to quantize an entire speech utterance in a well-defined and efficient manner. We defer a discussion of hidden Markov models to Chapter 6.

### 3.4.8 Summary of the VQ Method

In later chapters of this book we will see several examples of how VQ concepts can be exploited in speech-recognition systems. Here we have shown that the basic idea of VQ is to reduce the information rate of the speech signal to a low rate through the use of a codebook with a relatively small number of code words. The goal is to be able to

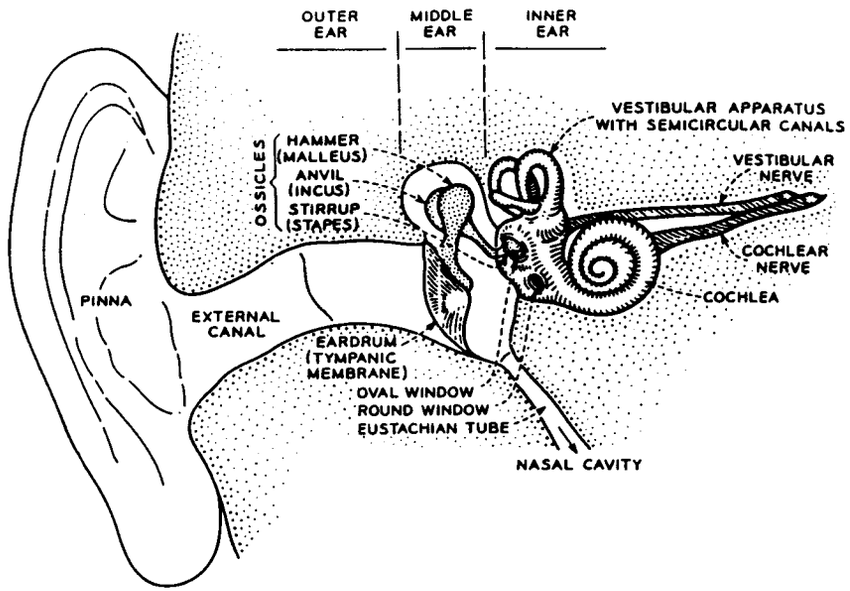



**Figure 3.47** Physiological model of the human ear.

represent the spectral information of the signal in an efficient manner and in a way that direct connections to the acoustic-phonetic framework discussed in Chapter 2 can be made. Various techniques for achieving this efficiency of representation were discussed, and their properties were illustrated on representative examples of speech.

## 3.5 AUDITORY-BASED SPECTRAL ANALYSIS MODELS

The motivation for investigating spectral analysis methods that are physiologically based is to gain an understanding of how the human auditory system processes speech, so as to be able to design and implement robust, efficient methods of analyzing and representing speech. It is generally assumed that the better we understand the signal processing in the human auditory system, the closer we will come to being able to design a system that can truly understand meaning as well as content of speech.

With these considerations in mind, we first examine a physiological model of the human ear. Such a model is given in Figure 3.47 and it shows that the ear has three distinct regions called the outer ear, the middle ear, and the inner ear. The outer ear consists of the pinna (the ear surface surrounding the canal in which sound is funneled), and the external canal. Sound waves reach the ear and are guided through the outer ear to the



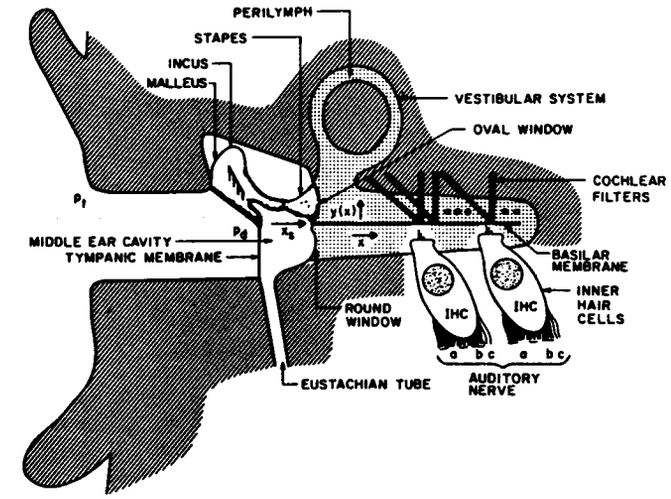



**Figure 3.48** Expanded view of the middle and inner ear mechanics.

middle ear, which consists of the tympanic membrane or eardrum upon which the sound wave impinges and causes to move and a mechanical transducer (the malleus or hammer, the incus or anvil, and the stapes or stirrup), which converts the acoustical sound wave to mechanical vibrations along the inner ear. The inner ear consists of the cochlea, which is a fluid-filled chamber partitioned by the basilar membrane, and the cochlea or auditory nerve. The mechanical vibrations impinging on the oval window at the entrance to the cochlea create standing waves (of the fluid inside the cochlea) that cause the basilar membrane to vibrate at frequencies commensurate with the input acoustic wave frequencies (e.g., the formants of voiced speech) and at a place along the basilar membrane that is associated with these frequencies. (An expanded view of the middle and inner ear mechanics is given in Figure 3.48. The $2\frac{1}{2}$ turn, snail-like shape of the cochlea is shown as a straight tube in this figure for ease of presentation.)

The basilar membrane is characterized by a set of frequency responses at different points along the membrane. Hence, in its simplest form, the cochlea can be modeled as a mechanical realization of a bank of filters (appropriately called cochlea filters). Distributed along the basilar member (in a dense but discrete manner) is a set of sensors called inner hair cells (IHC), which act as mechanical motion to neural activity converters. Mechanical motion at some point along the basilar membrane is sensed by the inner hair cells and causes firing activity at the nerve fibers that innervate the bottom of each IHC. Each IHC is connected to about 10 nerve fibers, each of different diameter and of different synaptic connection. It has been shown experimentally that thin fibers fire (emit neural impulses) only at high motion levels, whereas thick fibers fire at much lower motion levels. A total of about 30,000 nerve fibers link the IHCs to the auditory nerve.

Beyond the auditory nerve, our knowledge of how the information signals (the neural activity along the auditory nerve) are processed and eventually converted to intelligence in the brain is almost primitive. Hence when we attempt to build auditory models for signal processing, we are primarily modeling the middle ear, cochlea, and hair cell systems. The assumption is that the signal produced by such a model exhibits some of the robustness (immunity to noise, reverberation) and efficiency of the human auditory systems. Thus in the remainder of this section we present one such model, called the Ensemble Interval Histogram (EIH) model and show some of the properties of speech signals processed by such a model.

### 3.5.1 The EIH Model

On the basis of the discussion in the preceding section, a model of the cochlea and the hair cell transduction consists of a filter bank that models the frequency selectivity at various points along a simulated basilar membrane, and a nonlinear processor for converting the filter bank output to neural firing patterns along a simulated auditory nerve. Such a model is shown in Figure 3.49 and is called the EIH model [13].

In the EIH model, the mechanical motion of the basilar membrane is sampled using 165 IHC channels, equally spaced, on a log-frequency scale, between 150 and 7000 Hz. The corresponding cochlear filters are based on actual neural tuning curves for cats. The amplitude responses of 28 of these filters (i.e., about 1 in 8 from the model) are shown in Figure 3.50. The phase characteristics of these filters is minimum phase, and the relative gain, measured at the center frequency of the filter, reflects the corresponding value of the cat's middle ear transfer function.

The next stage of processing in the EIH model of Figure 3.49 is an array of level crossing detectors that models the motion-to-neural activity transduction of the hair cell mechanisms. The detection levels of each detector are pseudo-randomly distributed (based on measured distributions of level firings), thereby simulating the variability of fiber diameters and their synaptic connections.

The output of the level-crossing detectors represents the discharge activity of the auditory nerve fibers. Figure 3.51 shows simulated auditory nerve activity, for the first 60 msec of the vowel /o/ in the word "job," as a function of both time and the "characteristic frequency" of the IHC channels. (Note the logarithmic scale of the characteristic frequency which represents the place-to-frequency mapping on the basilar membrane.) In Figure 3.51, a level-crossing occurrence is marked as a single dot, and the output activity of each level-crossing detector is plotted as a separate trace. Each IHC channel contributes seven parallel traces (corresponding to the seven level-crossing detectors for each channel), with the lowest trace representing the lowest-threshold level-crossing detector. If the magnitude of the filter's output is low, only one level will be crossed, as is seen for the very top channels in Figure 3.51. However, for large signal magnitudes, several levels will be activated, creating a "darker" area of activity in the figure.

The level-crossing patterns represent the auditory nerve activity, which, in turn, is the input to a second, more central stage of neural processing, which gives the overall ensemble interval histogram (EIH). Conceptually, the EIH is a measure of the spatial



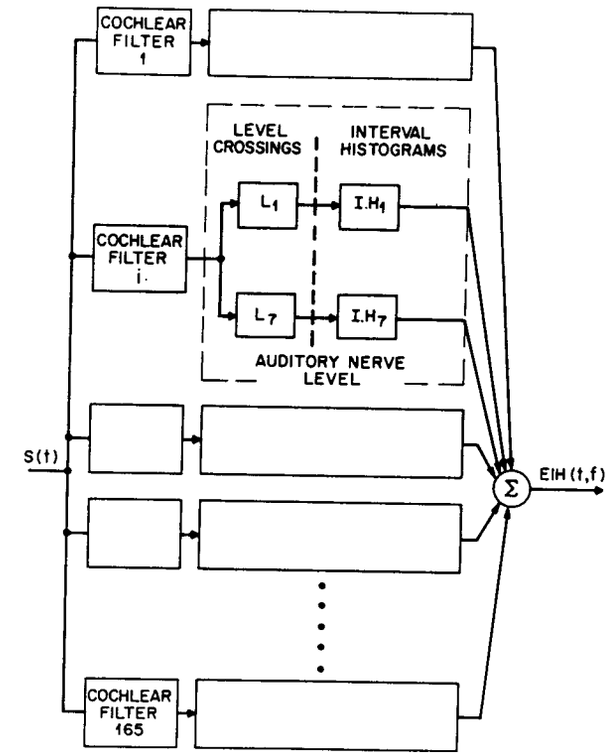



**Figure 3.49** Block diagram of the EIH model (after Ghitza [13]).

extent of coherent neural activity across the simulated auditory nerve. Mathematically, it is the short-term probability density function of the reciprocal of the intervals between successive firings, measured over the entire simulated auditory nerve in a characteristic frequency-dependent time-frequency zone.

As a consequence of the multilevel crossing detectors, the EIH representation preserves information about the signal's overall energy. To illustrate this point, consider the case in which the input signal is a pure sinusoid, i.e. $s(t) = A\sin(2\pi f_0 t)$, and the characteristic frequency of a selected channel is $f_0$, as shown in Figure 3.52a. For a given intensity $A$, the cochlear filter output will activate only some low level-crossing detectors. For a given detector, the time interval between two successive positive-going level crossings is $1/f_0$. Since the histogram is scaled in units of frequency, this interval contributes a count to the $f_0$ bin. For the input signal in Figure 3.52a, all of the intervals are the same, resulting in a histogram in which the magnitude of each bin, save one ($f_0$), is zero. As the signal

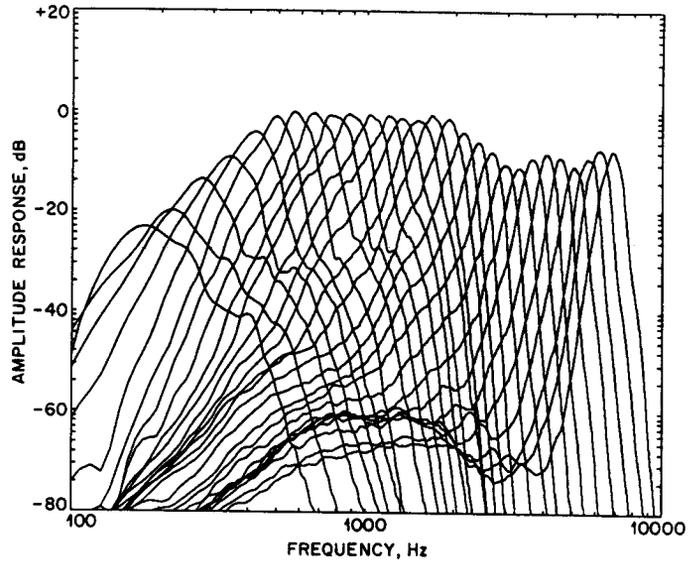



**Figure 3.50** Frequency response curves of a cat's basilar membrane (after Ghitza [13]).

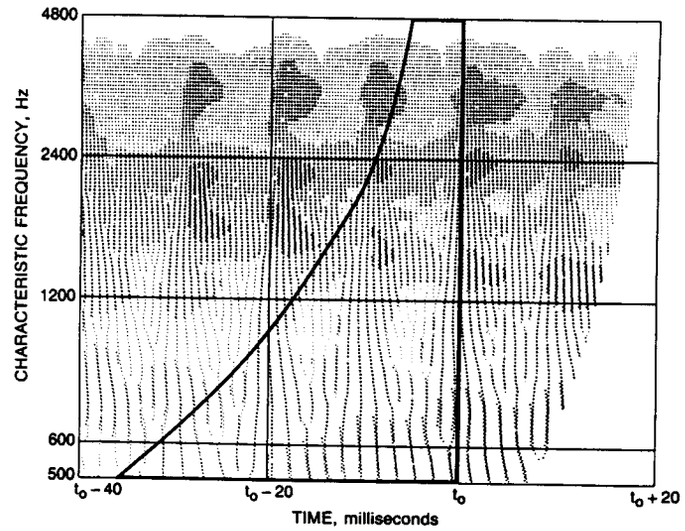



**Figure 3.51** Magnitude of EIH for vowel /o/ showing the time-frequency resolution (after Ghitza [13]).



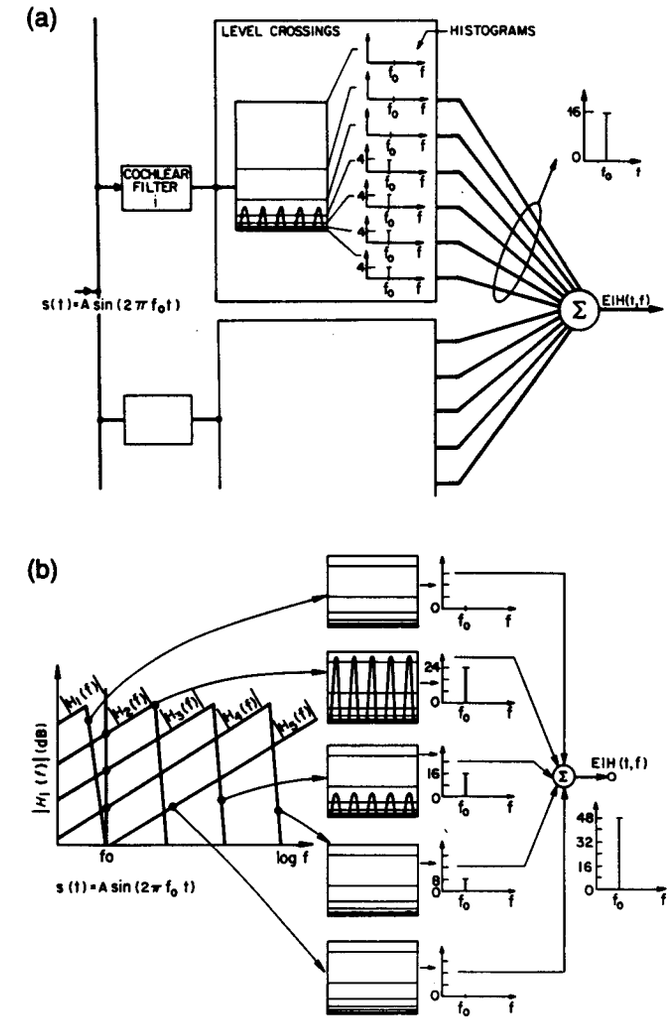



**Figure 3.52** Operation of the EIH model for a pure sinusoid (after Ghitza [13]).

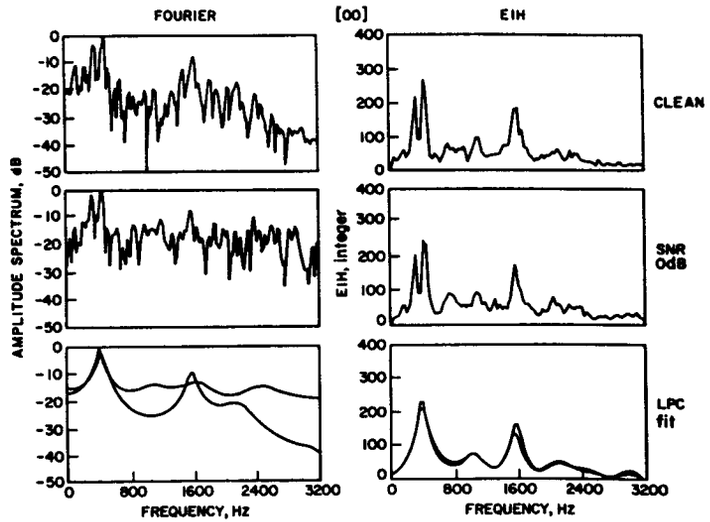



**Figure 3.53** Comparison of Fourier and EIH log spectra for clean and noisy speech signals (after Ghitza [13]).

amplitude $A$ increases, more levels are activated. As a result, this cochlear filter contributes additional counts to the $f_0$ bin of the EIH. Since the crossing levels are equally distributed on a log-amplitude scale, the magnitude of any EIH bin is related, in some fashion, to decibel units. However, this relation is not a straightforward one because there are several sources contributing counts to the $f_0$ bin in a nonlinear manner. Figure 3.52b shows an input signal $s(t) = A\sin(2\pi f_0 t)$ driving five adjacent cochlear filters with an amplitude response $|H_i(f)|$ and a phase response $\phi_i(f)$, $i = 1, 2, \ldots, 5$. Due to the shape of the filters, more than one cochlear channel will contribute to the $f_0$ bin. In fact, all the cochlear filters that produce $s_i(t) = A\,|H_i(f_0)|\,\sin(2\pi f_0 t + \phi_i(f_0))$ will contribute to the $f_0$ bin of the EIH, provided that $A\,|H_i(f_0)|$ exceeds any of the level-crossing thresholds. In Figure 3.52b only cochlear filters 2, 3, and 4 are contributing nonzero histograms to the EIH. The number of counts is different for each filter, depending on the magnitude of $A\,|H_i(f_0)|$.

One goal of auditory-based signal processing is to make the signal more robust to noise and reverberation than alternative spectral analysis procedures such as the filter-bank method or the LPC method. Figure 3.53 illustrates how well the EIH model achieves this goal. Shown in the figure are the log magnitude spectra of a clean (no noise) and a noisy (signal-to-noise ratio of 0 dB) speech signal processed by a standard Fourier filter bank (curves on the left) and by the EIH model (curves on the right). Also shown are LPC polynomial fits to the original signal spectrum (on the left) and to the EIH signal spectrum (on the right) for both the clean signal and the noisy signal. This figure clearly shows a tremendous sensitivity of the Fourier and LPC analyses to noise for the original signals.



(This is especially seen in the LPC polynomial fits.) In the EIH case, the log magnitude spectra are almost unaltered by the noise, and the LPC polynomial fits are extremely close to each other.

The implication of the above results for speech recognition is that the EIH model has potential for use in recognizing speech robustly in noisy and reverberant environments. We will explore this issue in Chapter 5 when we talk about the effects of noise on performance of speech recognizers.

## 3.6 SUMMARY

In this chapter we have discussed several ways of performing spectral analysis of speech, including filter-bank analysis, LPC analysis, vector quantization, and auditory modeling. We have discussed the relative strengths and weaknesses of each approach and given a hint of the advantages and disadvantages for application to actual speech-recognition systems. We will see, in later chapters, how the type of spectral analysis that is used interacts with the processing of other parts of the recognizer. Only through such an understanding can one fully see the trade-offs among the different approaches to speech spectrum analysis.

## REFERENCES

[1] L.R. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing*, Prentice Hall, Englewood Cliffs, NJ, 1975.

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

[3] R.E. Crochiere and L.R. Rabiner, *Multirate Digital Signal Processing*, Prentice Hall, Englewood Cliffs, NJ, 1983.

[4] B.A. Dautrich, L.R. Rabiner, and T.B. Martin, "On the Effects of Varying Filter Bank Parameters on Isolated Word Recognition," *IEEE Trans. Acoustics, Speech, Signal Proc.*, ASSP-31 (4): 793–807, August 1983.

[5] B.A. Dautrich, L.R. Rabiner, and T.B. Martin, "The Effects of Selected Signal Processing Techniques on the Performance of a Filter Bank Based Isolated Word Recognizer," *Bell System Tech. J.*, 62 (5): 1311–1336, May–June 1983.

[6] J.D. Markel and A.H. Gray, Jr., *Linear Prediction of Speech*, Springer-Verlag, 1976.

[7] G.M. White and R.B. Neely, "Speech Recognition Experiments with Linear Prediction, Bandpass Filtering, and Dynamic Programming," *IEEE Trans. Acoustics, Speech, Signal Proc.*, ASSP-24 (2): 183–188, 1976.

[8] L.R. Rabiner, B.S. Atal, and M.R. Sambur, "LPC Prediction Error—Analysis of its Variation with the Position of the Analysis Frame," *IEEE Trans. Acoustics, Speech, Signal Proc.*, ASSP-25 (5): 434–442, October 1977.

[9] H. Strube, "Determination of the Instant of Glottal Closure from the Speech Wave," *J. Acoust. Soc. Am.*, 56 (5): 1625–1629, November 1974.

[10] B.S. Atal and S.L. Hanauer, "Speech Analysis and Synthesis by Linear Prediction of the Speech Wave," *J. Acoust. Soc. Am.*, 50 (2): 637–655, August 1971.

[11] S. Furui, "Speaker Independent Isolated Word Recognition Using Dynamic Features of Speech Spectrum," *IEEE Trans. Acoustics, Speech, Signal Proc.*, ASSP-34 (1): 52-59, February 1986.

[12] J.-H. Juang, D.Y. Wong, and A.H. Gray, Jr., "Distortion Performance of Vector Quantization for LPC Voice Coding," *IEEE Trans. Acoustics, Speech, Signal Proc.*, ASSP-30 (2): 294–304, April 1982.

[13] O. Ghitza, "Auditory Nerve Representation as a Basis for Speech Processing," in *Advances in Speech Signal Processing*, S. Furui and M. Sondhi, Eds., Marcel Dekker, NY, 453–485, 1991.

Chapter 4

# PATTERN-COMPARISON TECHNIQUES

## 4.1 INTRODUCTION

A key question in speech recognition is how speech patterns are compared to determine their similarity (or equivalently, the distance between patterns). Depending on the specifics of the recognition system, pattern comparison can be done in a wide variety of ways.

Consider, for example, the acoustic-phonetic recognizer discussed in Chapter 2, in which every analysis frame is classified according to a defined set of features. Thus the spectral features of compactness, graveness, stress, and flatness can be used to classify sounds as specific vowels (recall Figure 2.33). The decision as to the presence or absence of these vowel features is made based on values of acoustic parameters (e.g., formants, energy in spectral bands, durations). Such decisions are usually based on empirically derived thresholds on the acoustic parameters. The comparison between the vowel features of an arbitrary piece of speech, and those predefined for various vowel classes, can be made via a decision tree (of the type shown in Figure 2.33). An alternative procedure would be to combine the vowel features into a vowel feature vector, and to apply a mathematical rule to make the appropriate vowel classification. The mathematical rule could be in the form of a dissimilarity or distance measure that evaluates the closeness of the input feature vector to a set of predefined vowel feature vectors, or it could be implemented as a Boolean function that maps the binary-valued feature vector directly to a vowel-classification decision (in which case the neural net classifiers of Chapter 2 would potentially be quite useful).

Of greater interest is the pattern-based approach to speech recognition in which the

speech is directly represented by the time sequence of spectral vectors as obtained from the front-end spectral analysis discussed in Chapter 3. Thus we define a test pattern, $\mathcal{T}$, as the concatenation of spectral frames over the duration of the speech, such that

$$\mathcal{T} = \{\mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3, \ldots, \mathbf{t}_I\},$$

where each $\mathbf{t}_i$ is the spectral vector of the input speech at time $i$, and $I$ is the total number of frames of speech. In a similar manner we define a set of reference patterns, $\{\mathcal{R}^1, \mathcal{R}^2, \ldots, \mathcal{R}^V\}$ where each reference pattern, $\mathcal{R}^j$, is also a sequence of spectral frames, such that

$$\mathcal{R}^j = \{\mathbf{r}_1^j, \mathbf{r}_2^j, \ldots, \mathbf{r}_J^j\}.$$

The goal of the pattern-comparison stage is to determine the dissimilarity or distance of $\mathcal{T}$ to each of the $\mathcal{R}^j$, $1 \le j \le V$, in order to identify the reference pattern that has the minimum dissimilarity, and to associate the spoken input with this pattern. (More generally, as will be shown in Chapter 6, we would like to compare the test pattern to a mathematical or statistical characterization of each reference pattern, rather than to a specific reference pattern sequence. In this chapter we will concentrate primarily on comparing two patterns, both of which are sequences of spectral vectors. We call such patterns templates. In Chapter 6 we will extend the pattern comparison concept to the case of statistical models.)

To determine the global similarity of $\mathcal{T}$ to $\mathcal{R}^j$ we need to be concerned with each of the following problems:

1. $\mathcal{T}$ and $\mathcal{R}^j$ generally are of unequal lengths (time durations). This is, of course, because of different speaking rates across different talkers with different dialects.
2. $\mathcal{T}$ and $\mathcal{R}^j$ need not line up in time in any simple or well-prescribed manner. This is because different sounds cannot be varied in duration to the same degree. Thus vowels are easily lengthened or shortened; however, most consonants cannot change dramatically in duration.
3. We need a way to compare pairs of spectral vectors (or equivalently, pairs of spectral representations)—that is, a local dissimilarity or distance measure to evaluate the global similarity and to facilitate a postulated temporal lineup between $\mathcal{T}$ and $\mathcal{R}^j$.

These problems imply the need for both a local dissimilarity measure and a global method of time aligning $\mathcal{T}$ and $\mathcal{R}^j$ so that the accumulated local dissimilarity between spectral frames in the time-aligned patterns is minimized. It will be shown in this chapter that the problem of global time alignment of two patterns can be handled with analytical rigor, and, in fact, can be merged with the problem of determining a good local dissimilarity measure to yield optimal solutions for a wide range of spectral dissimilarity (or, as generally called, spectral distortion) measures.

The concept of a spectral dissimilarity measure is equally applicable to both the feature-based approach and the template-based approach. However, because the acoustic-phonetic feature vector is essentially qualitative (often consisting of simple binary features), the measure so defined has limited numerical sensitivity. For template-based systems, however, the dissimilarity measures that are used have strong physical interpretations in



terms of spectral difference or spectral distortion between the pair of spectra. For this reason, such measures are called spectral-distortion or spectral-distance measures.

One other fundamental problem must be handled to define a global dissimilarity between pairs of patterns. We have implicitly assumed that both the test and reference patterns being compared consist solely of spectral frames corresponding to speech—that is, all background signal has been eliminated from consideration. For this to be the case in practice, we must be able to reliably separate speech from background signal. We call this problem the (speech) endpoint-detection problem and we begin with a discussion of various methods of solving this basic problem in speech processing.

## 4.2 SPEECH (ENDPOINT) DETECTION

The goal of speech detection is to separate acoustic events of interest (e.g., speech to be processed) in a continuously recorded signal from other parts of the signal (e.g., background). The need for speech detection occurs in many applications in telecommunications. For example, in analog multichannel transmission systems, a technique called time-assignment speech interpolation (TASI) is often used to take advantage of the channel idle time by detecting the presence of a talker's speech and assigning an unused channel only when speech is detected in order to allow more customer services than the channels would normally provide [1]. With TASI, a transmission medium with 96 voice channels can serve 235 customers; a factor of 2.5 more users is served because of the long pauses in fluent speech. For automatic speech recognition, endpoint detection is required to isolate the speech of interest so as to be able to create a speech pattern or template.

A key question in speech recognition is how accurately speech must be detected so as to provide the "best" speech patterns for recognition. The definition of "best" here is the pragmatic one—namely, the patterns that provide highest recognition accuracy. To answer this question, a multispeaker digit-recognition experiment was performed to determine the effect of endpoint errors on digit-recognition accuracy [2]. The recorded utterances were first examined by hand (including both examination of the waveform and energy contour, and listening to the hand–endpointed speech) and the endpoints were manually determined. This data set was then used as a standard set from which a set of reference templates for the digits was created and used in subsequent recognition trials. The recognizer employed was one based on a standard LPC front end. For the standard data set, the overall digit recognition accuracy was found to be 93%. With the original reference templates remaining unchanged, the manually determined endpoints of each utterance in the test set were then varied in 15-ms (one speech frame) steps from 150 ms before the standard beginning point to 150 ms after the standard ending point. For each set of changed endpoints, the resulting recognition accuracy was measured using the same utterances as in the base system (the one with hand endpoints). Figure 4.1 shows a contour plot of overall (averaged over all 10 digits) digit recognition accuracy as a function of the perturbation in the endpoint position in ms. As the figure shows, even small endpoint errors often result in relatively significant degradation in digit accuracy. For example, a 3% reduction in digit accuracy occurred if both the endpoints were in error by ±60 ms (4 frames). As the endpoints are moved farther

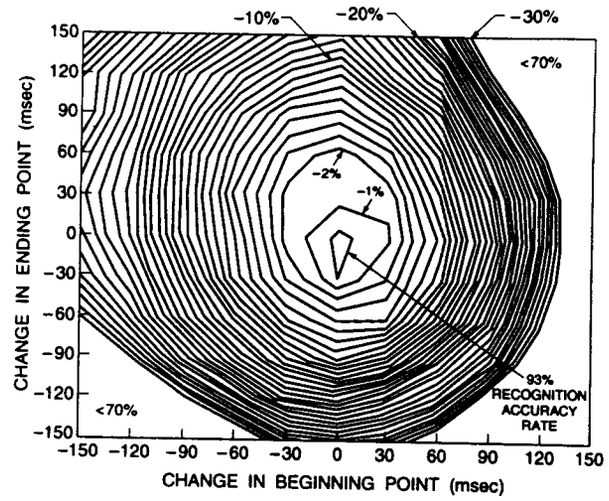



**Figure 4.1** Contour of digit recognition accuracy (percent correct) as a function of endpoint perturbation (in ms) in a multispeaker digit-recognition experiment. Both the initial (beginning point) and the final (ending point) boundary of the detected speech signal were varied (after Wilpon et al. [2]).

away from the reference endpoints, recognition accuracy decreases uniformly. While the results are undoubtedly dependent on the particular recognizer implementation and the way the endpoint information is used in the recognizer, the figure dramatically illustrates the effects of endpoint errors on speech-recognition performance, thereby showing the importance of accurate endpoint detection.

For speech produced in the most benign circumstances—that is, carefully articulated and spoken in a relatively noise-free environment—accurate detection of speech is a simple problem. However, this is not usually the case. In practice, one or more problems usually make accurate endpoint detection difficult. One particular class of problems is those attributed to the speaker and to the manner of producing the speech. For example, during articulation, the talker often produces sound artifacts, including lip smacks, heavy breathing, and mouth clicks and pops. Figures 4.2 through 4.4 show examples of this type of humanly produced sound artifact. The top part of each figure is the energy contour of the utterance on a logarithmic (dB) scale, and the lower part is the time waveform of the corresponding utterance.

Figure 4.2 shows a typical mouth click produced by opening the lips (prior to speaking) when the mouth is relatively dry, thereby causing the lips to pop, (i.e., produce a high-frequency, transient sound). Figure 4.3 shows a high level of breath noise produced at the end of speaking, caused by the speaker's heavy breathing. This artifact typically occurs when a speaker is short of breath (usually from exertion) and combines heavy breathing



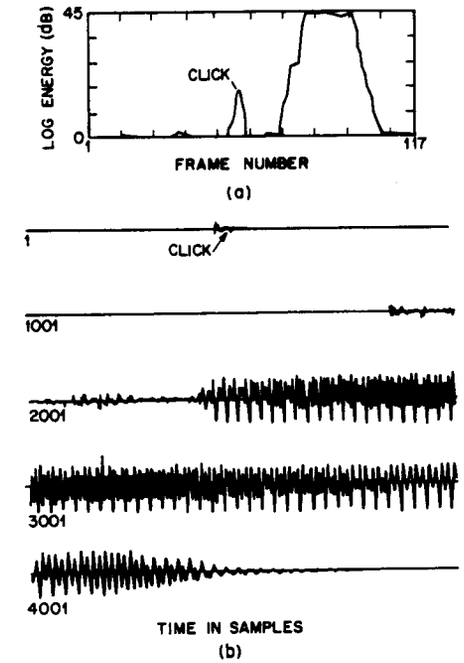



**Figure 4.2** Example of mouth click preceding a spoken word (after Wilpon et al. [2]).

with speaking. Unlike the precursor mouth click, the heavy breathing noise is not separated from the speech and therefore makes accurate endpoint detection quite difficult. Figure 4.4 shows an example of a mouth click produced after speaking. Such clicks are often generated inadvertently from the speaker clicking the lips or snapping the tongue after speaking. In all three examples, we see that the energy level of the artifacts is comparable to speech energy levels.

A second factor making reliable speech endpoint detection difficult is the environmental conditions in which the speech is produced. The ideal environment for talking is a quiet room with no acoustic noise or signal generators other than that produced by the speaker. Such an ideal environment is not always practical; hence, one must consider speech produced in noisy backgrounds (as with fans or machinery running), in nonstationary environments (as in the presence of door slams, irregular road noise, car horns), with speech interference (as from TV, radio, or background conversations), and in hostile circumstances (when the speaker is stressed, such as when navigating an airplane or while moving at high speeds). Some of these interfering signals possess as much speech-

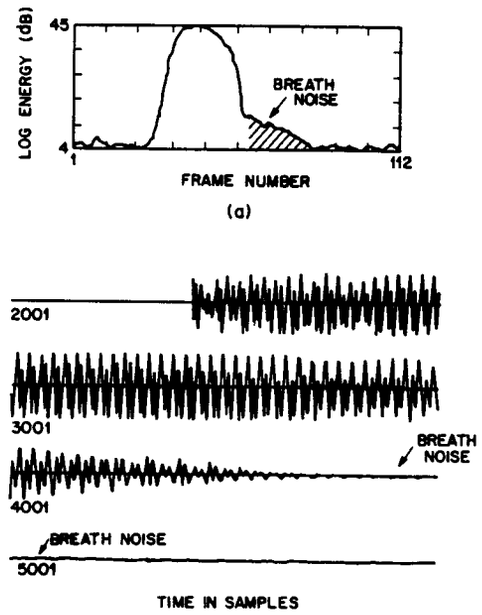



**Figure 4.3** Example of breathy speech due to heavy breathing while speaking (after Wilpon et al. [2]).

like quality as the desired speech signal itself, making accurate endpoint detection quite difficult.

A final source of signal degradation is the distortion introduced by the transmission system over which the speech is sent. Factors like cross-talk, intermodulation distortion, and various types of tonal interference arise to various degrees in the communications channel, again adding to the inherent difficulties in reliably detecting speech endpoints [1].

Many methods have been proposed for speech detection for use in speech recognition systems. These methods can be broadly classified into three approaches according to their interaction with the pattern matching paradigm: (1) the explicit approach, (2) the implicit approach, and (3) the hybrid approach [2].

The explicit approach is based on the premise that speech detection can be accomplished independent of other pattern-matching operations in the later stages of the speech-recognition process. Figure 4.5 shows a block diagram of the explicit approach to endpoint detection. The speech signal is first processed and feature measurements are made. The speech-detection method is then applied to locate and define the speech events.



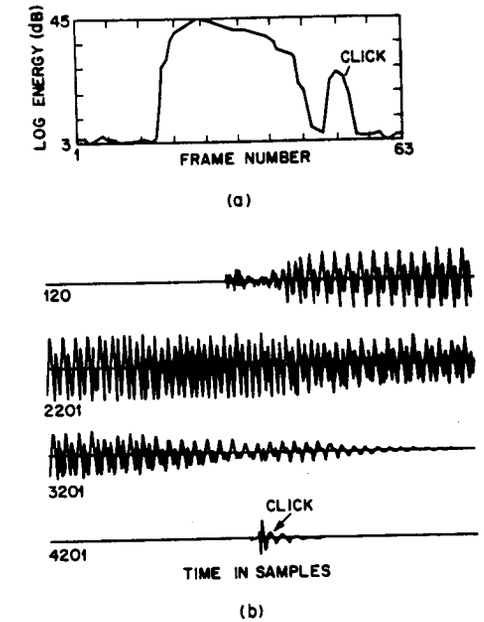



**Figure 4.4** Example of click produced at the end of a spoken word (after Wilpon et al. [2]).

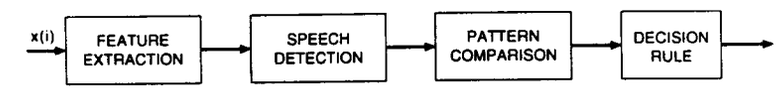



**Figure 4.5** Block diagram of the explicit approach to speech endpoint detection.

The detected speech is sent to the pattern-comparison algorithm, and finally the decision mechanism chooses the recognized word. Although the speech-detection module is likely to use the same measured features as those used for recognition, it is independent of the recognition processing and can use a different set of features, if appropriate. For signals with a stationary and low-level noise (and acoustic) background, the approach produces reasonably good detection accuracy. The approach fails often when the environment is noisy or the interference is nonstationary.

The implicit approach considers the speech-detection problem simultaneously with the pattern-matching and recognition-decision process. It recognizes that speech events are almost always accompanied by a certain acoustic background. With an explicit model of the background signal included in the reference templates, a signal in the form of "background-speech-background" can still be compared and classified. Figure 4.6 shows a block diagram

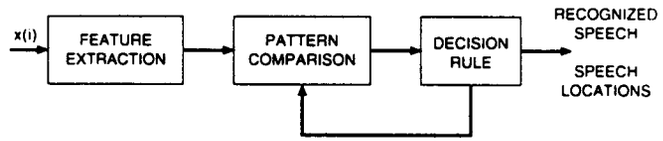



**Figure 4.6** Block diagram of the implicit approach to speech-endpoint detection.

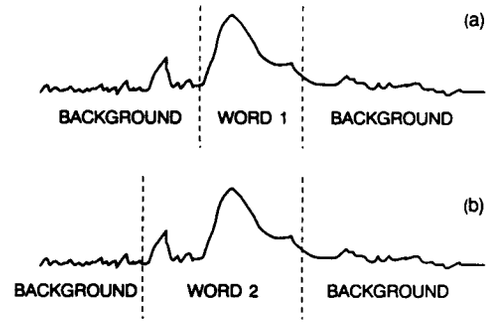



**Figure 4.7** Examples of word boundaries as determined by the implicit endpoint detection algorithm.

of the implicit approach to speech detection. The unmarked signal sequence is processed by the pattern-matching module in which all (or a large set of all) possible endpoint sets are considered and the decision mechanism provides an ordered list of the candidate words as well as the corresponding speech locations. The final result is the best candidate and its associated endpoints. The implicit approach obviously incurs a heavy computational load but offers a potentially higher detection accuracy than the explicit approach.

To illustrate the inherent advantage of the implicit approach to endpoint detection, Figure 4.7 shows an example of a measured log energy contour of an utterance and two possible sets of word boundary locations. The example of Figure 4.7a corresponds to the case in which the initial high-energy transient (labeled "A" in the figure) is classified as part of the background and the word boundaries are as shown. (This case corresponds to a word without an initial stop consonant, such as the word "even.") The example of Figure 4.7b corresponds to the case in which the initial high-energy transient is treated as a stop consonant and classified as part of the word (e.g., the initial part of a word like "acquire"). Hence, depending on the word recognized, the boundary locations could inherently be different with the implicit method, whereas with the explicit method only a single choice of boundary locations is made.

The computational complexity of the implicit approach can be significantly reduced if only a small but reasonable set of estimates of the endpoint set are considered. This gives rise to the hybrid technique, which uses the explicit method to obtain several potential endpoint sets for recognition processing and the implicit method to choose among the



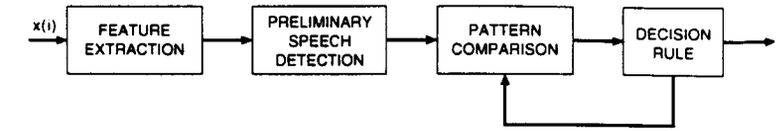



**Figure 4.8** Block diagram of the hybrid approach to speech endpoint detection.

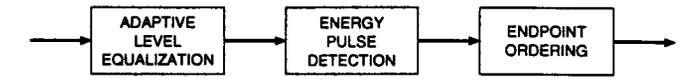



**Figure 4.9** Block diagram of typical speech activity detection algorithm.

alternatives. The most likely candidate word and the corresponding endpoints, as in the implicit approach, are provided by the decision box. Figure 4.8 shows a block diagram of the hybrid method. The hybrid approach has a computational load equivalent to the explicit method, but with an accuracy comparable to the implicit method.

As pointed out, these three approaches to endpoint detection differ in their degree of interaction with the recognition process. The basic speech-detection algorithm used to obtain an estimate of the endpoints often involves feed-forward processing of the measured short-time energy level. A block diagram of such a processing algorithm is shown in Figure 4.9. The adaptive level equalization module estimates the level of the acoustic background and uses the result to equalize the measured energy contour. Preliminary energy pulses, which are speechlike bursts of energy during the recording interval, are then detected from the equalized energy contour. Finally, these potential energy pulse endpoints are ordered, according to their likelihood, to determine the possible sets of word endpoint pairs. Extensive experimentation is necessary in deciding the best values for the required set of thresholds and the ordering logic to provide the most reasonable sets of endpoints.

## 4.3 DISTORTION MEASURES—MATHEMATICAL CONSIDERATIONS

A key component of most pattern-comparison algorithms is a prescribed measurement of dissimilarity between two feature vectors. This measurement of dissimilarity can be handled with mathematical rigor if the patterns are visualized in a vector space. Consider the following framework.

Assume we have two feature vectors, $\mathbf{x}$ and $\mathbf{y}$ defined on a vector space $\chi$. We define a metric or distance function $d$ on the vector space $\chi$ as a real-valued function on the Cartesian product $\chi \times \chi$ such that [3]:

**(a)** $0 \leq d(\mathbf{x}, \mathbf{y}) < \infty$ for $\mathbf{x}, \mathbf{y} \in \chi$ and $d(\mathbf{x}, \mathbf{y}) = 0$ if and only if $\mathbf{x} = \mathbf{y}$;
**(b)** $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$ for $\mathbf{x}, \mathbf{y} \in \chi$;
**(c)** $d(\mathbf{x}, \mathbf{y}) \leq d(\mathbf{x}, \mathbf{z}) + d(\mathbf{y}, \mathbf{z})$ for $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \chi$.

In addition, a distance function is called invariant if

(d) $d(\mathbf{x}+\mathbf{z}, \mathbf{y}+\mathbf{z}) = d(\mathbf{x}, \mathbf{y})$.

Properties (a), (b), and (c) are commonly referred to as the positive definiteness, the symmetry, and the triangle inequality conditions, respectively. A metric with the above properties ensures a high degree of mathematical tractability.

If a measure of difference, $d$, satisfies only the positive definiteness property, we customarily call it a distortion measure, particularly when the vectors are representations of signal spectra.

For speech processing, an important consideration in defining (or choosing) a measure of distance is its subjective meaningfulness. A mathematical measure of distance, to be useful in speech processing, has to have a high correlation between its numerical value and the subjective distance judgment, as evaluated on real speech signals. For speech recognition, the psychophysical consistency that we would like the measure of distance to possess implies that the mathematical measure needs to conform to known linguistic characteristics. This subjective requirement often cannot be satisfied by distance measures with proven mathematical tractability. An example is the traditional squared error distance $d(\mathbf{x}, \mathbf{y}) = (\mathbf{x} - \mathbf{y})^2$. A large difference in the waveform squared error does not always imply large subjective differences; examples are the difference between an original speech signal and an all-pass filtered version of the same signal, or for sounds like /sh/ which is essentially a shaped noise process.

Because it is difficult to simultaneously accommodate the dual objectives of subjective meaningfulness and mathematical tractability, some compromise is inevitable. What we will discuss in this chapter are a number of distortion measures that are considered, to various degrees, subjectively meaningful. The focus of our discussion will be on the mathematical properties of these distortion measures; however, we will try to show that the subjective properties of these measures are reasonably good for recognition. For the most part, we will talk about "distortion measures" instead of "metrics," because of the relaxation of the conditions on symmetry and the triangle inequality. We will not, however, use the term *distance* in a strict sense according to the definition above; we will instead follow the custom in the speech literature that the term *distance* is analogous to *measure of dissimilarity*, a generic descriptive terminology.

## 4.4 DISTORTION MEASURES—PERCEPTUAL CONSIDERATIONS

A key factor in the choice of an appropriate measure of spectral dissimilarity or distance measure is the concept of subjective judgment of sound difference or phonetic relevance. Loosely speaking, a phonetically relevant distance measure has the property that spectral changes that perceptually make the sounds being compared seem to be different should be associated with large distances; similarly spectral changes that keep the sound the same perceptually should be associated with small distances. To illustrate more specifically what we mean by phonetic relevance, consider comparing two spectral representations, $S(\omega)$ and $S'(\omega)$, using a distance measure $d(S, S')$. Spectral changes that do not fundamentally change the perceived sound include



- spectral tilt, i.e., $S'(\omega) = S(\omega) \cdot \omega^{\alpha}$, where $\alpha$ is the spectral tilt factor;
- highpass filtering, i.e., $S'(\omega) = S(\omega) \mid H_{\mathrm{HP}}(e^{j\omega}) \mid^2$, where $H_{\mathrm{HP}}(e^{j\omega})$ is a filter that is unity in value above some cutoff frequency that is typically below the range of the first formant frequency, and attenuates the signal sharply for frequencies below the cutoff frequency;
- lowpass filtering, i.e., $S'(\omega) = S(\omega) \mid H_{\mathrm{LP}}(e^{j\omega}) \mid^2$, where $H_{\mathrm{LP}}(e^{j\omega})$ is a filter that is unity in value below some cutoff frequency that is typically above the range of the third formant frequency, and attenuates the signal sharply for frequencies above the cutoff frequency;
- notch filtering, i.e., $S'(\omega) = S(\omega) \mid H_N(e^{j\omega}) \mid^2$, where $H_N(e^{j\omega})$ is unity except at a narrow range of frequencies where the signal is greatly attenuated (the notch band).

For all the above cases, the differences in spectral content of the two signals being compared is not phonetically relevant (that is, the sounds are basically perceived as being the same phonetically); hence the associated spectral distance measure, $d(S, S')$ should ideally be small.

Spectral changes that perceptually lead to the sounds judged as being phonetically different (large phonetic distance) include

- significant differences in formant locations—that is, the spectral resonances of $S(\omega)$ and $S'(\omega)$ occur at very different frequencies;
- significant differences in formant bandwidths—that is, the frequency widths of the spectral resonances of $S(\omega)$ and $S'(\omega)$ are very different.

For each of these cases the difference in spectral content of the two signals being compared is phonetically relevant (i.e., the sounds are different); hence, the associated spectral distance measure, $d(S, S')$ should ideally be large.

A key question, therefore, is, What do we mean by significant differences in formant locations or bandwidths? To relate a physical measure of difference to a subjectively perceived measure of difference, we need to understand some properties of human auditory discrimination of signals. In particular it is important to understand auditory sensitivity to changes in the speech dimensions or features such as the frequencies and bandwidths of the complex poles and zeros of the speech spectrum, signal intensity, and fundamental frequency. This sensitivity is often presented in the form of just-discriminable change, the change in a physical parameter such that the auditory system can reliably detect the change as measured in standard listening tests. Terms used to describe just-discriminable change include the difference limen (DL), just-noticeable difference (JND), and differential threshold.

Using synthetic vowel sounds generated by a computer synthesizer, the DL for the frequencies of the first and second formants have been found to be on the order of 3 to 5% of the formant frequency [4]. This result was obtained by considering changes in the frequency of only one formant at a time and finding what level of change was reliably detected by listeners. When all formants were changed simultaneously, the measured DL

**TABLE 4.1.** Measured DLs and JNDs for Synthetic Vowel Sounds

| Parameter | DL/JND |
|---|---|
| Fundamental frequency | 0.3–0.5% |
| Formant frequency | 3–5% |
| Formant bandwidth | 20–40% |
| Overall intensity | 1.5 dB |

was shown to be a complicated function of all formant frequencies and was shown to be very sensitive to how close formants were to each other.

The JND in the overall intensity of a synthetic vowel has been shown to be about 1.5 dB [4]. Since the first formant is usually the most intense formant in vowel sounds, this JND of 1.5 dB can also be considered a rough estimate of the intensity DL for the first formant. In terms of the second formant, the intensity DL was found to be about 3 dB for a near-neutral vowel (/æ/).

The DLs for formant bandwidth have not been directly measured. Indirect measurements show that changes on the order of 20 to 40% in formant bandwidth are just-discriminable [5].

Again using synthetic vowel sounds, the fundamental frequency DL was shown to be about 0.3 to 0.5% of the fundamental frequency [6]. An interesting observation is that the formant frequency DL is an order of magnitude smaller than the formant bandwidth DL, and the fundamental frequency DL is an order of magnitude smaller than the formant frequency DL. Table 4.1 shows measured DLs and JNDs for synthetic vowel sounds.

Of more direct interest for speech recognition are the JNDs of the poles of a linear predictive model spectrum. This is of interest because the LPC model spectrum is highly mathematically tractable. The JNDs for the LPC poles were measured by perturbing a single LPC pole of a single frame (22.5 ms long) embedded in a sentential context (2–3 s in duration) [7]. In this case, the measured DLs are not strictly the JNDs for steady-state sounds but are mixed with the dynamics of spectral transitions in the utterance. (In the experiment, a 10th-order LPC analysis was employed. The speech material was in Hebrew.) A complex pole of the LPC model spectrum can be expressed as $z_i = r_i e^{j\omega_i}$, where $r_i$ and $\omega_i$ are the radius and frequency of the complex pole, respectively. The bandwidth $B_i$ of the complex pole $z_i$ is related to the radius $r_i$ by

$$B_i = -\log(r_i) \cdot f_s/\pi \text{ (Hz)}, \tag{4.1}$$

where $f_s$ is the sampling frequency (8 kHz in the experiments). We use $\beta = \log_{10} B$ to denote the bandwidth parameter $B$ on a log scale. Figure 4.10 shows the frequency JNDs (frequency deviation) of an LPC pole as a function of the bandwidth of the corresponding pole. Both positive and negative frequency perturbations are shown in each plot for the four "formant frequencies." Each set of data points is fitted by a parabola of the form $\log(\Delta f) = \log(\Delta f_2)[1 + a(\beta - 2) + b(\beta - 2)^2]$ where $\Delta f_2$ represents the average-frequency JND for the pole with a bandwidth of 100 Hz (i.e., at $\beta = 2$). $\Delta f_2$s were measured to be 62, 158, 355, and 480 Hz for F1, F2, F3, and F4, respectively, displaying a clear trend of



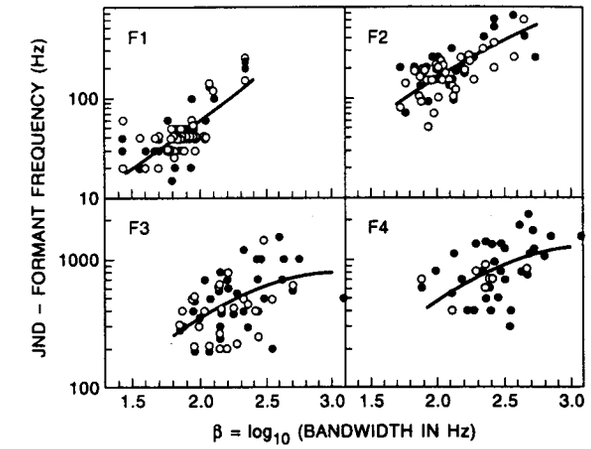



**Figure 4.10** LPC pole frequency JNDs as a function of the pole bandwidth; the blank circles denote positive frequency perturbations, and the solid circles represent negative frequency perturbations; the fitting curves are parabolic (after Erell et al. [7]).

dependence on the pole frequencies. Furthermore, the curves for F1 and F2 indicate, to a first-order approximation, that the frequency JNDs increase as the square root ($a = 0.55$) and the cubic root ($a = 0.36$) of the corresponding bandwidths, respectively.

The bandwidth JNDs (bandwidth deviation) of an LPC pole, as a function of the corresponding pole bandwidth, is shown in Figure 4.11. Unlike the frequency JNDs, the bandwidth JNDs do not show clear dependence on either the bandwidth itself or the "formant" frequency. The average $\beta$-JNDs are 0.48, 0.67, 0.77, and 0.92 for the four "formants," respectively. Note that for sharp formants where the pole radius is close to unity, the JNDs in bandwidth can be used to estimate the JNDs in formant intensity, $I$, according to the relationship

$$\begin{aligned}\Delta I\,[\text{dB}] &\cong -20 \cdot \Delta[\log(1 - r)] \\ &\simeq -20 \cdot \Delta[\log(-\log r)] = -20(\Delta\beta).\end{aligned} \tag{4.2}$$

Therefore, the equivalent average JNDs for the intensity are 9.6, 13.4, 15.4, and 18.4 dB for the four "formants," respectively. These figures are much larger than the measurements obtained by Flanagan [4].

Other measurements of auditory discriminability can be made. All these quantitative characterizations of the auditory percept may be helpful in relating a measurement of spectral difference to a perceived distortion, potentially making comparisons of speech patterns subjectively more meaningful. Perceptual consistency itself, however, is hard to obtain among human listeners. As can be seen from the figures, the variability of the JNDs is large, indicating the difficulty in deriving a perceptually consistent distortion

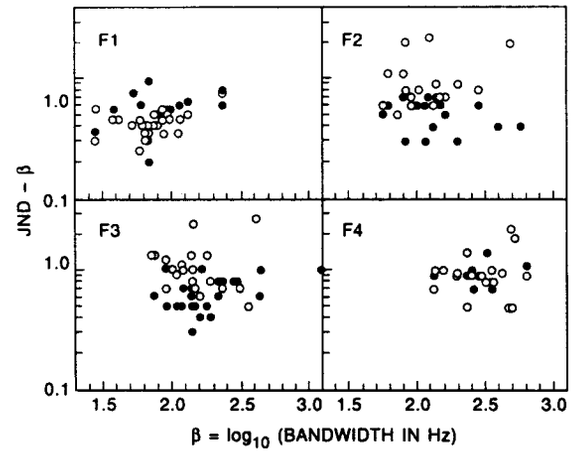



**Figure 4.11** LPC pole bandwidth JNDs, in a logarithmic scale, as a function of the pole bandwidth itself (after Erell et al. [7]).

measure. Even if some measurable speech dimensions can be accurately mapped to human percepts, the technical difficulty that has to be overcome is to find a way to combine these psychoacoustic quantities in a consistent and meaningful manner. It is therefore more practical to approach the problem of defining distortion measures from mathematical signal-processing considerations than from the perceptual point of view, with the anticipation that the resulting distortion measure be somewhat consistent with subjective judgments of distance, or phonetic relevance.

## 4.5 SPECTRAL-DISTORTION MEASURES

Measuring the difference between two speech patterns in terms of average or accumulated spectral distortion appears to be a very reasonable way of comparing patterns, both in terms of its mathematical tractability and its computational efficiency. Also since many psychoacoustic studies of perceived sound differences can be interpreted in terms of differences of spectral features, measurement of spectral distortion can be argued to be reasonable both mathematically and subjectively. Before we begin a discussion of several important spectral distortion measures, some definitions and discussion of basic properties of a speech spectrum are necessary.

Let $S(\omega)$ represent the spectral density of a speech signal $x(i)$, where $\omega$ is normalized frequency ranging from $-\pi$ to $\pi$ ($\pi$ being one-half of the sampling frequency). The spectral density $S(\omega)$ is a nonnegative, even function of $\omega$ with Fourier coefficients $r(n)$ which define an autocorrelation sequence; that is,



$$S(\omega) = \sum_{n=-\infty}^{\infty} r(n)e^{-jn\omega} \tag{4.3}$$

$$r(n) = \int_{-\pi}^{\pi} S(\omega)e^{jn\omega}\frac{d\omega}{2\pi}. \tag{4.4}$$

The autocorrelation function is defined in terms of the speech signal by

$$r_E(n) = E\{x(i)x(i+n)\}, \tag{4.5}$$

if $x(i)$ is a wide-sense stationary process. Taking $r(n) = r_E(n)$ makes $S(\omega)$ a power spectral density.

For computational convenience, when processing a time-varying signal such as speech, we often use the short-term autocorrelation defined over a frame of speech samples $\{x(i),\ i = 0, 1, \ldots, N-1\}$

$$r_N(n) = \sum_{i=0}^{N-|n|-1} x(i)x(i+|n|) \qquad \text{for } n = 0, 1, \ldots, N-1. \tag{4.6}$$

If we define

$$r(n) = \begin{cases} r_N(n), & n = 0, 1, \ldots, N-1 \\ 0, & n \geq N \end{cases} \tag{4.7}$$

$S(\omega)$ is then an energy spectral density.

For an ergodic wide-sense stationary process, the short-term autocorrelation, if normalized by the data length $N$, approaches the autocorrelation of Eq. (4.5) with probability one [8]. We assume this to be the case and define the power spectral density and the autocorrelation as above without ambiguity. Associated with the autocorrelation sequence is a set of autocorrelation matrices, $\mathbf{R}_k$, $k = 1, 2, \ldots, N$, of dimension $(k+1) \times (k+1)$, whose $(i,j)$-th element is given by $r(|i-j|)$. Thus

$$\mathbf{R}_2 = \begin{bmatrix} r(0) & r(1) & r(2) \\ r(1) & r(0) & r(1) \\ r(2) & r(1) & r(0) \end{bmatrix}$$

and

$$\mathbf{R}_N = \begin{bmatrix} r(0) & r(1) & \ldots & r(N) \\ \vdots & & & \\ r(N) & . & \ldots & r(0) \end{bmatrix}.$$

It is often desired in speech processing to represent the speech spectrum by an all-pole model, denoted by $\sigma/A(z)$ where $A(z)$ is a $p$-th order polynomial $A(z) = \sum_{i=0}^{p} a_i z^{-i}$ with $a_0 = 1$. As discussed in Chapter 2, the optimal all-pole model coefficients represented in a vector form $\mathbf{a}_p^t = (a_{p0}, a_{p1}, \ldots, a_{pp})$ are obtained by a residual minimization process. The residual energy resulting from "inverse filtering" the input signal with an

all-zero (discrete time) filter $A(z)$ defined by the coefficient vector $\mathbf{a}^t = (a_0, a_1, \ldots, a_p)$ is

$$\alpha(\mathbf{a}) = \int_{-\pi}^{\pi} \left| \sum_{i=0}^{p} a_i e^{-ji\omega} \right|^2 S(\omega) \frac{d\omega}{2\pi}$$
$$= \mathbf{a}^t \mathbf{R}_p \mathbf{a}. \tag{4.8}$$

Minimization of the residual energy results in

$$\mathbf{a}_p = \arg\min_{\mathbf{a}} \alpha(\mathbf{a}) = \arg\min_{\mathbf{a}} (\mathbf{a}^t \mathbf{R}_p \mathbf{a}). \tag{4.9}$$

The minimum residual is $\sigma_p^2 = \alpha(\mathbf{a}_p)$. Thus, $\sigma_p^2 / \left|A_p(e^{j\omega})\right|^2$, where $A_p(z)$ is defined by the optimal coefficient vector $\mathbf{a}_p$, is the optimal $p$-th order all-pole spectrum of $S(\omega)$.

Other important properties of all-pole modeling of the speech spectrum pertinent to spectral distortion measures that we will discuss in this chapter are the recursive minimization relationship

$$\sigma_p^2 = |\mathbf{R}_p| \,/\, |\mathbf{R}_{p-1}|, \tag{4.10}$$

the notion of one-step predictor error

$$\sigma_\infty^2 \triangleq \lim_{p\to\infty} \sigma_p^2 = \exp\left\{ \int_{-\pi}^{\pi} \log S(\omega) \frac{d\omega}{2\pi} \right\}, \tag{4.11}$$

and the zero mean property of the log spectrum of a minimum phase all-pole model

$$\int_{-\pi}^{\pi} \log \frac{1}{|A(e^{j\omega})|^2} \frac{d\omega}{2\pi} = 0. \tag{4.12}$$

**Exercise 4.1**
In Chapter 3, we showed that minimization of the prediction residual energy is equivalent to solving a set of LPC equations (see Section 3.3). In particular, with the autocorrelation method, the predictor coefficients, $a_i$, satisfy the following set of equations

$$\sum_{k=1}^{p} r(|i-k|) a_k = -r(i), \qquad 1 \le i \le p.$$

(The above equation assumes the convention that $A(z) = \sum_{i=0}^{p} a_i z^{-i}$ rather than $A(z) = 1 - \sum_{i=1}^{p} a_i z^{-i}$.) Verify the result of Eq. (4.10) by actually calculating $\sigma_p^2$ in terms of the autocorrelation coefficients for $p = 1$, and 2.

**Solution 4.1**

1. For $p = 1$,

$$\mathbf{R}_1 = \begin{bmatrix} r(0) & r(1) \\ r(1) & r(0) \end{bmatrix}$$

and $\mathbf{R}_0 = r(0)$. The predictor coefficient $a_1$ satisfies

$$r(0)a_1 = -r(1)$$



and therefore $\mathbf{a}_1^t = [1, -r(1)/r(0)]$. The minimum residual energy $\sigma_1^2$ is thus

$$\sigma_1^2 = \alpha(\mathbf{a}_1) = \mathbf{a}_1^t \mathbf{R}_1 \mathbf{a}_1$$
$$= r(0) - \frac{r^2(1)}{r(0)}$$
$$= \frac{|\mathbf{R}_1|}{|\mathbf{R}_0|}.$$

2. For $p = 2$,

$$\mathbf{R}_2 = \begin{bmatrix} r(0) & r(1) & r(2) \\ r(1) & r(0) & r(1) \\ r(2) & r(1) & r(0) \end{bmatrix}.$$

Also, $|\mathbf{R}_2| = r^3(0) + 2r^2(1)r(2) - r(0)r^2(2) - 2r(0)r^2(1)$. The prediction equations are

$$r(0)a_1 + r(1)a_2 = -r(1)$$
$$r(1)a_1 + r(0)a_2 = -r(2)$$

which have the following solution

$$a_1 = [r(1)r(2) - r(0)r(1)]/[r^2(0) - r^2(1)]$$
$$a_2 = [r^2(1) - r(0)r(2)]/[r^2(0) - r^2(1)].$$

The minimum residual energy $\sigma_2^2$ then becomes

$$\sigma_2^2 = [1 \;\; a_1 \;\; a_2] \begin{bmatrix} r(0) & r(1) & r(2) \\ r(1) & r(0) & r(1) \\ r(2) & r(1) & r(0) \end{bmatrix} \begin{bmatrix} 1 \\ a_1 \\ a_2 \end{bmatrix}$$
$$= \frac{r^3(0) + 2r^2(1)r(2) - r(0)r^2(2) - 2r(0)r^2(1)}{r^2(0) - r^2(1)}$$
$$= \frac{|\mathbf{R}_2|}{|\mathbf{R}_1|}.$$

**Exercise 4.2**
Derive the zero mean property of the log spectrum of a minimum phase all-pole model (Eq. (4.12)).

**Solution 4.2**
Equation (4.12) can be easily derived by recognizing that

$$\int_{-\pi}^{\pi} \log \left|A(e^{j\omega})\right|^2 \frac{d\omega}{2\pi} = \int_{-\pi}^{\pi} \log \left|A(e^{-j\omega})\right|^2 \frac{d\omega}{2\pi}$$
$$= 2\mathrm{Re}\left( \int_{-\pi}^{\pi} \log \left\{A(e^{-j\omega})\right\} \frac{d\omega}{2\pi} \right)$$
$$= 2\mathrm{Re}\left( \oint_{\Gamma} \log \left[A(z^{-1})\right] \frac{dz}{2\pi j z} \right)$$
$$= 2\mathrm{Re}\left(\log \left[A(\infty)\right]\right) = 2\mathrm{Re}[\log(1)] = 0,$$

where $\Gamma$ is a closed contour enclosing all the zeros of $A(z)$ and Re denotes the real part.

### 4.5.1 Log Spectral Distance

Consider two spectra $S(\omega)$ and $S'(\omega)$. The difference between the two spectra on a log magnitude versus frequency scale is defined by

$$V(\omega) = \log S(\omega) - \log S'(\omega). \tag{4.13}$$

One natural choice for a distance or distortion measure between $S$ and $S'$ is the set of $L_p$ norms defined by

$$d(S, S')^p = (d_p)^p = \int_{-\pi}^{\pi} |V(\omega)|^p \frac{d\omega}{2\pi}. \tag{4.14}$$

For $p = 1$, Eq. (4.14) defines the mean absolute log spectral distortion. For $p = 2$, Eq. (4.14) defines the rms log spectral distortion that has found application in many speech-processing systems [9]. When $p$ approaches infinity, Eq. (4.14) reduces to the peak log spectral distortion. Since perceived loudness of a signal is approximately logarithmic, the log spectral distance family appears to be closely tied to the subjective assessment of sound differences; hence, it is a perceptually relevant distortion measure.

The $L_p$ norm of Eq. (4.14) has long been a subject of rigorous mathematical studies. The $L_p$ measures are metrics because they satisfy all the conditions defined for metrics. Since the integrand is expressed in log magnitude, the measures are readily related to dB variations by using the multiplicative factor $10/\log_e(10) = 4.34$.

Several representative examples of log spectral distortions of typical speech spectra are shown in Figures 4.12 through 4.17. There are two groups of plots. One group, including Figures 4.12, 4.14, and 4.16, shows the distortion as calculated from the short-time FFT power spectra and the other, Figures 4.13, 4.15, and 4.17, shows the distortion as calculated from the all-pole smoothed model spectra. The examples chosen show typical variations between different realizations of the vowel /æ/ (Figures 4.12 and 4.13), the fricative /sh/ (Figures 4.14 and 4.15), and between two grossly different sounds, namely /æ/ and /i/ (Figures 4.16 and 4.17). In each figure, the top plot shows an overlay of the two spectra being compared, and the bottom plot shows the magnitude of the log spectral difference $|V(\omega)|$ as a function of the normalized frequency. Also marked along the vertical axis are $d_1$, $d_2$, and $d_\infty$, respectively, for the spectral pairs.

The log spectral difference $|V(\omega)|$, as computed from the FFT of the data sequence, is shown in these figures to be quite irregular. A major part of the variations comes from the difference in the fundamental frequencies, which have little effect on the perceived phonetic content of a steady-state vowel. The log spectral difference between the two all-pole models of the spectra $\left|\log \frac{\sigma^2}{|A(e^{j\omega})|^2} - \log \frac{\sigma'^2}{|A'(e^{j\omega})|^2}\right|$ is much smoother than the FFT counterpart. The smooth spectral difference function allows a closer examination of the properties of the distortion measure. For the same class of sounds (either the sound is slowly changing or they are simply different renditions of the same sound), formant variation produces the largest differences as shown in the curves. Formant variation could be due to a shift in the formant (or pole) frequency or a change in the formant bandwidth that is directly related to the formant intensity. For comparisons of all-pole model spectra, however, variations in a particular formant often cannot be isolated from variations in other formants. As can be



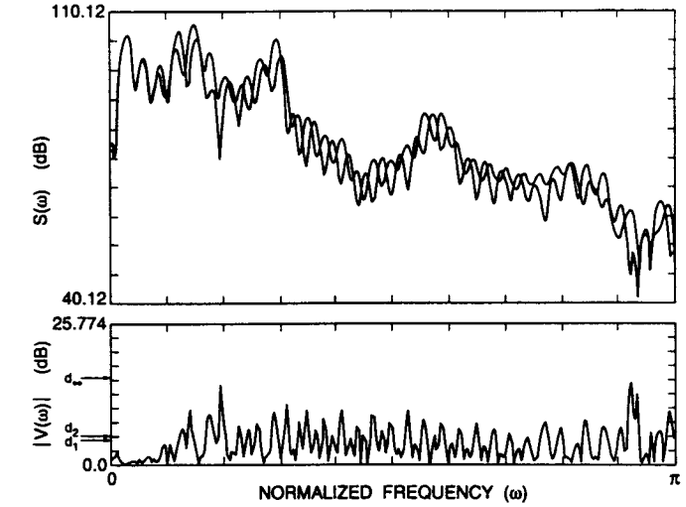



**Figure 4.12** Two typical FFT power spectra, $S(\omega)$, of the sound /æ/ in a log scale and their difference magnitude $|V(\omega)|$ as a function of frequency.

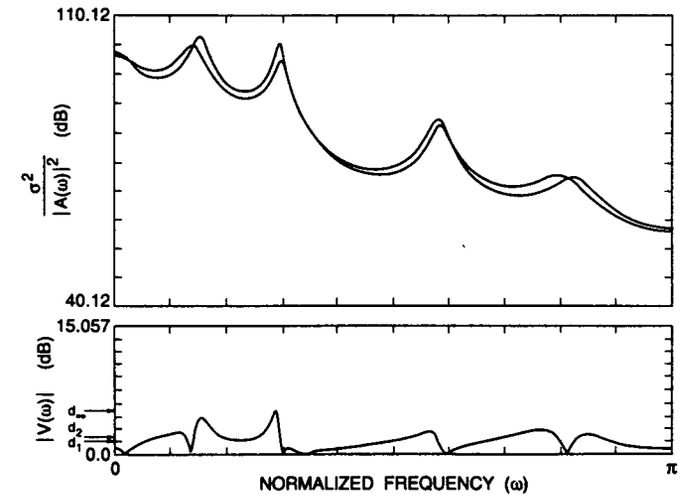



**Figure 4.13** LPC model spectra corresponding to the FFT spectra in Figure 4.12, plotted also in a log scale, and their difference magnitude $|V(\omega)|$ as a function of frequency.

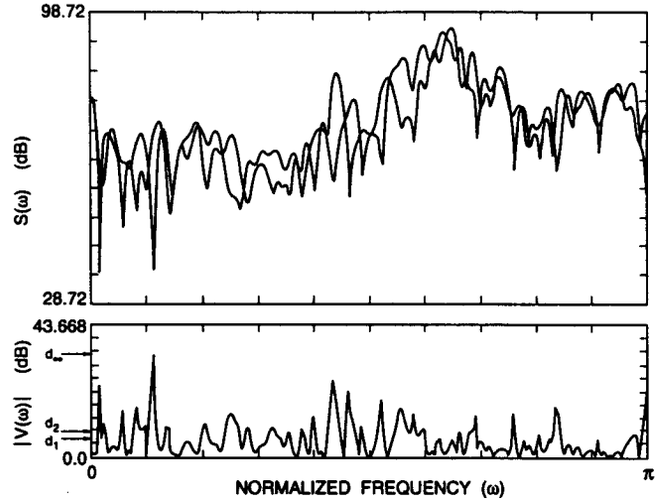



**Figure 4.14** Two typical FFT power spectra, $S(\omega)$, of the sound /sh/ in a log scale and their difference magnitude $|V(\omega)|$ as a function of frequency.

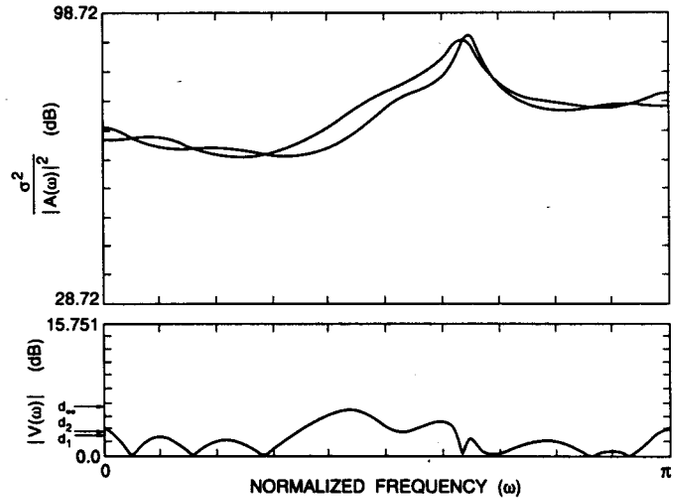



**Figure 4.15** LPC model spectra corresponding to the FFT spectra in Figure 4.14, plotted also in a log scale, and their difference magnitude $|V(\omega)|$ as a function of frequency.



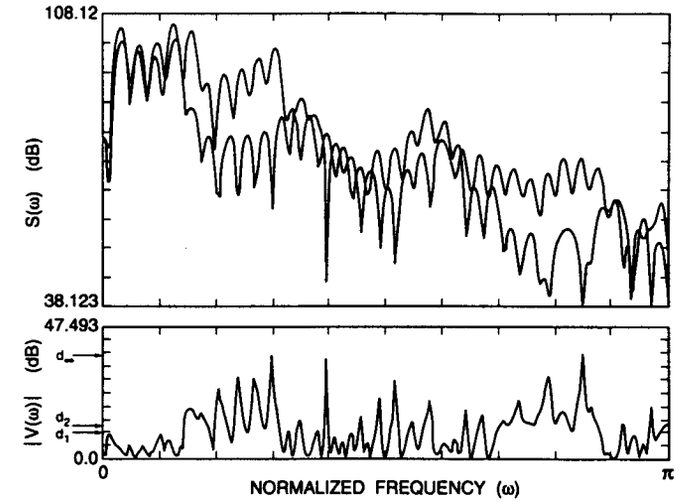



**Figure 4.16** Typical FFT power spectra of the sounds /æ/ and /i/ respectively and their difference magnitude as a function of frequency.

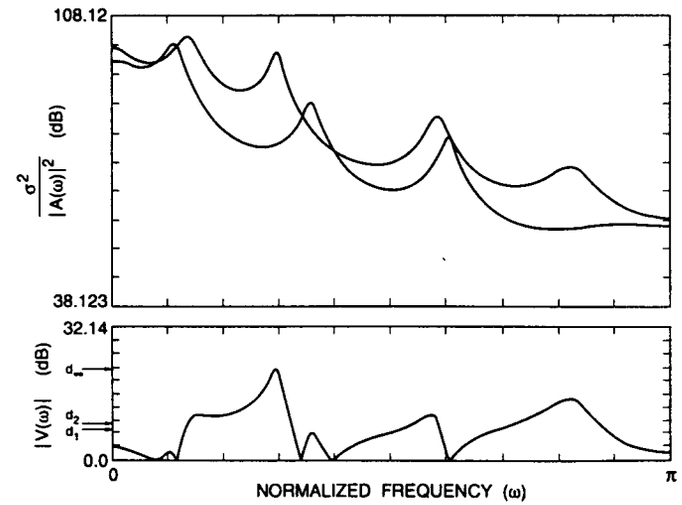



**Figure 4.17** LPC model spectra corresponding to the FFT spectra in Figure 4.16 and their difference magnitude $|V(\omega)|$ as a function of frequency.

seen from the figures, substantial log spectral differences in low power spectrum regions are not unusual. The $L_p$ measures weight these spectral differences equally along the frequency axis regardless of the spectral shapes being compared. Thus the $L_p$ distortion measures do not take the known and well-understood perceptual masking effect into consideration. We will see how alternative spectral distortion measures attempt to use this perceptual weighting later in this chapter.

From the data in Figures 4.12 through 4.17 it can be seen that the $L_p$ distortions, when comparing versions of the same sound, are considerably smaller than when comparing versions of different sounds, especially when using the all-pole model spectra. We exploit these differences in speech recognition by accumulating the spectral distortions over time when comparing speech patterns.

**Exercise 4.3**

Show that the $L_1$ spectral distortion measure

$$(d_1)^1 = \int_0^{2\pi} \left|\log S(\omega) - \log S'(\omega)\right| \frac{d\omega}{2\pi}$$

obeys the mathematical properties of distance metrics, namely:

1. It is positive definite.
2. It is symmetric.
3. It satisfies the triangular inequality.

**Solution 4.3**

For compactness we let $\hat{S} = \log S(\omega)$ and $\tilde{S} = \log S'(\omega)$.

1. Since $\left|\hat{S}(\omega) - \tilde{S}(\omega)\right| \geq 0$, for any $\omega$ then

$$(d_1)^1 = \int_0^{2\pi} \text{(nonnegative quantity)} \frac{d\omega}{2\pi} \geq 0.$$

If $S' = S$ then $(d_1)^1 = 0$.

2. Since $\left|\hat{S} - \tilde{S}\right| = \left|\tilde{S} - \hat{S}\right|$, $(d_1)^1$ is symmetric.
3. Since

$$S_1 - S_3 = S_1 - S_2 + S_2 - S_3,$$

$$|S_1 - S_3| \leq |S_1 - S_2| + |S_2 - S_3|.$$

Therefore,

$$\int_{-\pi}^{\pi} |S_1 - S_3| \frac{d\omega}{2\pi} \leq \int_{-\pi}^{\pi} |S_1 - S_2| \frac{d\omega}{2\pi} + \int_{-\pi}^{\pi} |S_2 - S_3| \frac{d\omega}{2\pi}$$

which leads to

$$d_1(S_1, S_3) \leq d_1(S_1, S_2) + d_1(S_2, S_3).$$

Hence, $(d_1)^1$ satisfies the triangular inequality.



### 4.5.2 Cepstral Distances

The complex cepstrum of a signal is defined as the Fourier transform of the log of the signal spectrum. For a power spectrum (magnitude-squared Fourier transform) $S(\omega)$, which is symmetric with respect to $\omega = 0$ and is periodic for a sampled data sequence, the Fourier series representation of $\log S(\omega)$ can be expressed as

$$\log S(\omega) = \sum_{n=-\infty}^{\infty} c_n e^{-jn\omega}, \tag{4.15}$$

where $c_n = c_{-n}$ are real and often referred to as the cepstral coefficients. Note that

$$c_0 = \int_{-\pi}^{\pi} \log S(\omega) \frac{d\omega}{2\pi}. \tag{4.16}$$

For a pair of spectra $S(\omega)$ and $S'(\omega)$, by applying Parseval's theorem, we can relate the $L_2$ cepstral distance of the spectra to the rms log spectral distance:

$$\begin{aligned} d_2^2 &= \int_{-\pi}^{\pi} |\log S(\omega) - \log S'(\omega)|^2 \frac{d\omega}{2\pi} \\ &= \sum_{n=-\infty}^{\infty} (c_n - c_n')^2, \end{aligned} \tag{4.17}$$

where $c_n$ and $c_n'$ are the cepstral coefficients of $S(\omega)$ and $S'(\omega)$ respectively.

When the speech is modeled by a minimum phase all-pole spectrum, i.e., $S(\omega) \to \sigma^2 / \left|A(e^{j\omega})\right|^2$, the resulting cepstrum has many interesting properties.

For a stable all-pole filter, $\log A(z^{-1})$ is analytic inside the unit circle and can be represented by the Laurent expansion [10]

$$\log\, [\sigma / A(z)] = \log \sigma + \sum_{n=1}^{\infty} c_n z^{-n}. \tag{4.18}$$

Differentiating both sides of the equation with respect to $z^{-1}$ and equating the coefficients of like powers of $z^{-1}$, we derive the following recursion:

$$c_n = -a_n - \frac{1}{n} \sum_{k=1}^{n-1} k\, c_k a_{n-k} \qquad \text{for } n > 0, \tag{4.19}$$

where $a_0 = 1$ and $a_k = 0$ for $k > p$. In terms of the log power spectrum, the Taylor series expansion becomes

$$\log \left[\sigma^2 / \left|A(e^{j\omega})\right|^2\right] = \sum_{n=-\infty}^{\infty} c_n e^{-jn\omega}, \tag{4.20}$$

where $c_0 = \log \sigma^2$ and $c_{-n} = c_n$.

For convenience, we shall refer to the cepstrum derived from a minimum phase all-pole power spectrum as an LPC cepstrum. Equation (4.18) also shows that the gain term

can be easily separated from the rest of the cepstrum that defines the general spectral shape of the speech signal.

The magnitude of the cepstrum is asymptotically bounded. This can be seen by considering the case of a rational $z$-transform $X(z)$ of the signal $x(i)$

$$X(z) = \frac{G z^{r}\left[\prod_{k=1}^{p_1}(1 - u_k z^{-1})\right]\left[\prod_{k=1}^{p_2}(1 - v_k z)\right]}{\left[\prod_{k=1}^{p_3}(1 - z_k z^{-1})\right]\left[\prod_{k=1}^{p_4}(1 - w_k z)\right]}, \tag{4.21}$$

where the magnitudes of the quantities $u_k$, $v_k$, $w_k$, and $z_k$ are all less than one. This is the most general form for $X(z)$, including both poles and zeros, that it is necessary to consider. Again, using the power series expansion, we can show that the complex cepstrum has the form

$$c_n = \begin{cases} \log G & n = 0 \\ \sum_{k=1}^{p_3} \frac{z_k^n}{n} - \sum_{k=1}^{p_1} \frac{u_k^n}{n} & n > 0 \\ \sum_{k=1}^{p_4} \frac{w_k^{-n}}{n} - \sum_{k=1}^{p_2} \frac{v_k^{-n}}{n} & n < 0 \end{cases} \tag{4.22}$$

It is therefore clear that the magnitude of the complex cepstrum is bounded by

$$|c_n| < \gamma \frac{\lambda^{|n|}}{|n|} \qquad \text{for } |n| \to \infty, \tag{4.23}$$

where $\lambda$ is the maximum absolute value of the quantities $z_k$, $w_k$, $u_k$, and $v_k$ and $\gamma$ is a constant scalar. The same result applies to the cepstrum derived from an arbitrary power spectrum—that is, not an all-pole spectrum.

Since the cepstrum is a decaying sequence, the summation in Eq. (4.17) does not require an infinite number of terms. For LPC models that represent highly smoothed envelopes of the speech spectra, the summation in Eq. (4.17) is usually truncated to only a small number of terms, typically 10–30. Since the first $p$ cepstral coefficients (excluding $c_0$) uniquely determine the minimum-phase all-pole filter, the number of terms must be no less than $p$ for the truncated distance to be a legitimate spectral distance for the two all-pole spectra. A truncated cepstral distance is defined by

$$d_c^2(L) = \sum_{n=1}^{L} (c_n - c_n')^2. \tag{4.24}$$

To further justify truncation, the relationship between the distortion measure of Eqs. (4.17) and (4.24), particularly in the case of the LPC cepstrum, can be experimentally examined. Figures 4.18 and 4.19 show scatter plots of $d_2^2$ (without the $c_0$ term) and $2d_c^2(L)$ as obtained from 800 all-pole spectra (compared in a consecutive pair manner) for $L = 20$ and 30 ($p = 10$), respectively. The correlation in each set of data is respectively 0.997 ($L = 20$),



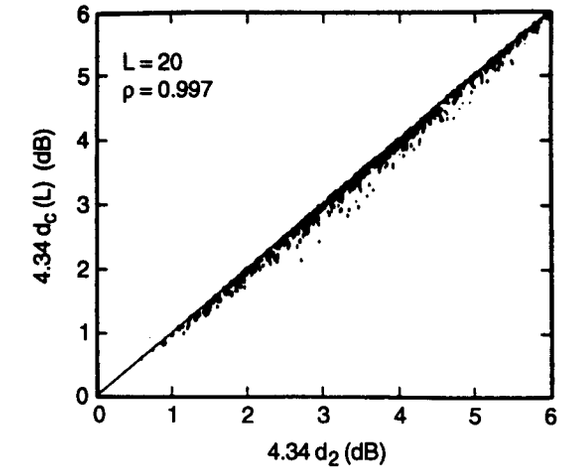



**Figure 4.18** Scatter plot of $d_2^2$, the cepstral distance, versus $2d_c^2(L)$, the truncated cepstral distance (multiplied by 2), for 800 pairs of all-pole model spectra; the truncation is at $L = 20$ (after Gray and Markel [9]).

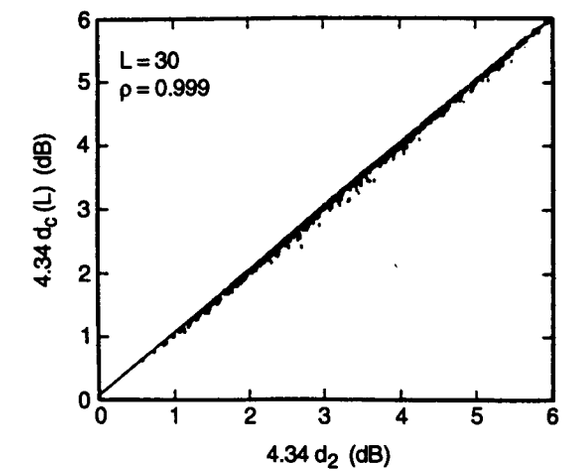



**Figure 4.19** Scatter plot of $d_2^2$, the cepstral distance, versus $2d_c^2(L)$, the truncated cepstral distance (multiplied by 2), for 800 pairs of all-pole model spectra; the truncation is at $L = 30$ (after Gray and Markel [9]).

and 0.999 ($L = 30$). Therefore, the truncated cepstral distance is a very efficient method for estimating the rms log spectral distance, particularly when the spectrum is represented by an all-pole model.

**Exercise 4.4**

Show that Eq. (4.17) is valid.

**Solution 4.4**

$$
\begin{aligned}
d_2^2(S,S') &= \int_{-\pi}^{\pi} \left|\log S(\omega) - \log S'(\omega)\right|^2 \frac{d\omega}{2\pi} \\
&= \int_{-\pi}^{\pi} \left| \sum_{n=-\infty}^{\infty} c_n e^{-jn\omega} - \sum_{n=-\infty}^{\infty} c_n' e^{-jn\omega} \right|^2 \frac{d\omega}{2\pi} \\
&= \int_{-\pi}^{\pi} \left| \sum_{n=-\infty}^{\infty} (c_n - c_n') e^{-jn\omega} \right|^2 \frac{d\omega}{2\pi} \\
&= \int_{-\pi}^{\pi} \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} (c_n - c_n')(c_m^* - c_m'^*) e^{-j\omega(n-m)} \frac{d\omega}{2\pi} \\
&= \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} (c_n - c_n')(c_m^* - c_m'^*) \int_{-\pi}^{\pi} e^{-j\omega(n-m)} \frac{d\omega}{2\pi} \\
&= \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} (c_n - c_n')(c_m^* - c_m'^*) \delta(n-m) \\
&= \sum_{n=-\infty}^{\infty} \left|c_n - c_n'\right|^2 = \sum_{n=-\infty}^{\infty} (c_n - c_n')^2.
\end{aligned}
$$

Note that $S$ and $S'$ are power spectra, which are even functions, and therefore the cepstral coefficients are real.

### 4.5.3 Weighted Cepstral Distances and Liftering

The usefulness of cepstral distances goes far beyond the simple fact that it is an efficient method for estimating the rms log spectral distance. Several other properties of the cepstrum, when properly utilized, are beneficial for speech-recognition applications. We begin with a discussion of the variability of the cepstrum.

It can be shown [11] that under certain regular conditions, the cepstral coefficients, except $c_0$, have: (1) zero means, and (2) variances essentially inversely proportional to the square of the coefficient index, such that

$$E\{c_n^2\} \sim \frac{1}{n^2}. \tag{4.25}$$

If we incorporate the $n^2$ factor into the cepstral distance so as to normalize (by the variance



inverse) the contribution from each cepstral term, the distance of Eq. (4.17) becomes

$$
\begin{aligned}
d_{2W}^2 &= \sum_{n=-\infty}^{\infty} n^2 (c_n - c_n')^2 \\
&= \sum_{n=-\infty}^{\infty} (nc_n - nc_n')^2.
\end{aligned} \tag{4.26}
$$

The distance $d_{2W}^2$ has two interesting interpretations. Notice that from Eq. (4.15)

$$\frac{d}{d\omega}[\log S(\omega)] = \sum_{n=-\infty}^{\infty} -jnc_n e^{-jn\omega}. \tag{4.27}$$

Therefore,

$$d_{2W}^2 = \int_{-\pi}^{\pi} \left| \frac{d \log S(\omega)}{d\omega} - \frac{d \log S'(\omega)}{d\omega} \right|^2 \frac{d\omega}{2\pi} \tag{4.28}$$

which is an $L_2$ distance based upon the differences between the spectral slopes (first-order derivatives) of the corresponding log power spectra pair.

The sequence $nc_n$ is often referred to as the "root power sums" for the LPC cepstrum. Similar to Eqs. (4.21) and (4.22), we can express the predictor polynomial in terms of its roots

$$A(z) = \prod_{i=1}^{p} (1 - z_i z^{-1}) \tag{4.29}$$

which leads to

$$\log\ [1/A(z)] = \sum_{i=1}^{p} \sum_{n=1}^{\infty} \frac{z_i^n}{n} z^{-n}. \tag{4.30}$$

By comparing Eqs. (4.18) and (4.30), we obtain

$$nc_n = \sum_{i=1}^{p} z_i^n, \tag{4.31}$$

which is a sum of the $n^{\text{th}}$ power of the roots of $A(z)$. The distance defined in Eq. (4.26) is thus called the "root power sum" distance.

Equation (4.31) has another interpretation in terms of the group delay spectrum. Note that

$$\log\ [1/A(z)] = \log\ [1/|A(z)|] + j \arg [1/A(z)]. \tag{4.32}$$

Therefore, with $z_i = r_i e^{j\omega_i}$,

$$
\begin{aligned}
\log\ [1/|A(z)|] &= \mathrm{Re}\left\{ \sum_{i=1}^{p} \sum_{n=1}^{\infty} \frac{z_i^n}{n} z^{-n} \right\} \\
&= \sum_{n=1}^{\infty} \left( \sum_{i=1}^{p} \frac{z_i^n}{n} \right) \mathrm{Re}[z^{-n}]
\end{aligned} \tag{4.33}
$$

and

$$\arg[1/A(z)] = \mathrm{Im}\left\{\sum_{i=1}^{p}\sum_{n=1}^{\infty}\frac{z_i^n}{n}z^{-n}\right\}$$
$$= \sum_{n=1}^{\infty}\left(\sum_{i=1}^{p}\frac{z_i^n}{n}\right)\mathrm{Im}[z^{-n}], \tag{4.34}$$

where Re and Im are the real part and imaginary part, respectively. (Since $A(z)$ has real coefficients, the roots are either real or in conjugate pairs such that the sum $\sum_{i=1}^{p} z_i^n/n$ is real.) The group delay, which is the negative derivative of the phase with respect to the frequency $\omega$, is then

$$-\frac{\partial}{\partial\omega}\arg[1/A(e^{j\omega})] = \sum_{n=1}^{\infty}\left(\sum_{i=1}^{p} z_i^n\right)\mathrm{Re}(e^{-jn\omega}). \tag{4.35}$$

The coefficients of the series expansion—the group delay spectrum—are thus seen to be the root power sums, or equivalently the index-weighted cepstral sequence $n\,c_n$ as shown in Eq. (4.31).

While the distance of Eq. (4.26), either motivated by variance normalization or an attempt to emphasize spectral slopes (with the added appeal of a root power sum formulation), indicates the possibility of using a weighted cepstral distance as a distortion measure, the variabilities of cepstral coefficients and their significance need to be critically considered, particularly from a speech-recognition perspective, for the choice of weighting functions.

The variability of various cepstral coefficients can be examined by contrasting the coefficient variances (denoted by $\sigma_s^2(i)$ for the $i^{\text{th}}$ cepstral coefficient) as obtained from a mixed speech source, with rich phonetic content, to those (denoted by $\sigma_f^2(i)$) from an artificial signal, produced by exciting a fixed all-pole system with a Gaussian independent identically distributed random noise. The ratio $\sigma_f^2(i)/\sigma_s^2(i)$ can be shown to increase as a function of the index $i$. At $i = 15$, the variance ratio is close to 0.6, about 5 times that at $i = 1$. This shows that the variability of higher cepstral coefficients are more influenced by the inherent artifacts of the LPC analysis (due to the all-pole constraint, analysis window position, system interaction with the excitation, etc.) than that of lower cepstral coefficients. For speech recognition, therefore, suppression of high cepstral coefficients in the calculation of a cepstral distance should lead to a more reliable measurement of spectral differences than otherwise.

The variability of low cepstral coefficients, while less affected by the analysis mechanism, is primarily due to variations in transmission, speaker characteristics, vocal efforts, and other factors. For example, the effect of differences in frequency response roll off in various telephone channels is usually most prominent in the first few cepstral coefficients. The LPC spectrum also includes components that are strong functions of the speaker's glottal shape and vocal cord duty cycles. These components appear, to a first order, as the overall spectral tilt, which again affects mainly the first few cepstral coefficients. For recognition of speech, independent of the talker's identity, these sources of variability, that are not essential in representing the phonetic content of the sound, need to be de-emphasized.



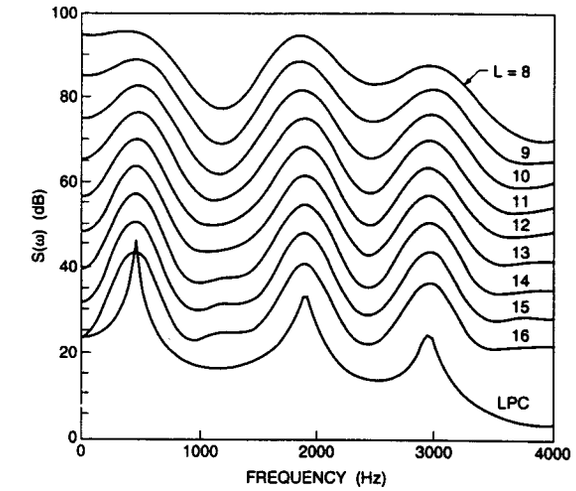



**Figure 4.20** Effects of cepstral liftering on a log LPC spectrum, as a function of the lifter length ($L = 8$ to 16) (after Juang et al. [11]).

A cepstral weighting, or liftering procedure, $w(n)$ can therefore be designed to control the noninformation-bearing cepstral variabilities for *reliable* discrimination of sounds. The index weighting, as used in Eq. (4.26), is one example of a simple form of cepstral weighting. Another, more sophisticated weighting is a raised sine function of the form

$$w(n) = \begin{cases} 1 + h\sin\left(\frac{n\pi}{L}\right) & \text{for } n = 1, 2, \ldots, L \\ 0 & \text{for } n \le 0, n > L, \end{cases} \tag{4.36}$$

where $h$ is usually chosen as $L/2$ and $L$ is typically $10 \sim 16$ for speech of 4 kHz bandwidth. The weighted sequence, $w(n)c_n$, corresponds to a smoothed log power spectrum. Figure 4.20 shows the effect of liftering on the original LPC log spectrum as a function of the lifter length $L$ (from $L = 8$ to 16). As is clearly seen, the sharp spectral peaks typical in the LPC spectrum of a vowel sound are effectively smoothed. The original sharp spectral peaks are highly sensitive to the LPC analysis condition (e.g., window position) and the resulting peakiness creates unnecessary sensitivity in spectral comparison. The liftering process tends to reduce the sensitivity without altering the fundamental "formant" structure. Furthermore, the LPC spectral tilt of approximately 8 dB/octave in the figure is effectively removed.

The effect of the liftering process can be further demonstrated through a sequence of spectral plots. Figure 4.21(a) is a hidden line plot of a series of 30 consecutive LPC log

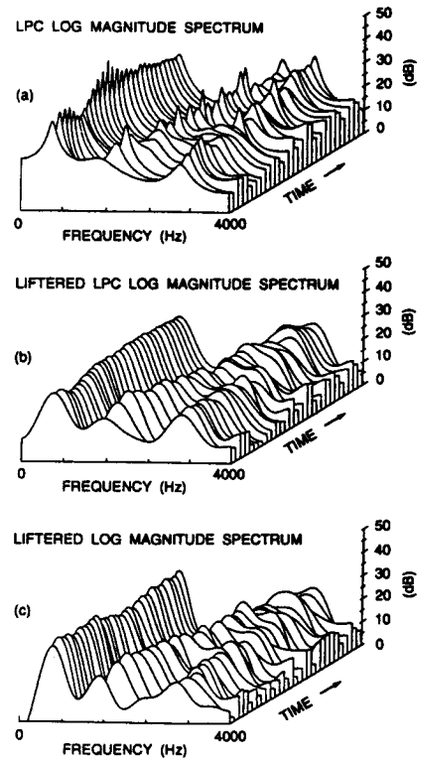



**Figure 4.21** Comparison of (a) original sequence of LPC log magnitude spectra; (b) liftered LPC log magnitude spectra, and (c) liftered log magnitude spectra (after Juang et al. [11]).

spectra, corresponding to a vowel-like sound. The randomness of the spectral components and the sharp spectral peaks that lead to excessive spectral sensitivity are clearly shown. Figure 4.21(b) is the same log spectral plot after the lifter defined in Eq. (4.31) with $L = 12$ is applied. The undesirable (noiselike) components of the LPC spectrum are reduced or removed, while the essential characteristics of the "formant" structure are retained. For comparison, Figure 4.21(c) displays a sequence of liftered (smoothed) log spectra of the same speech signal as in Figure 4.21(a) and (b), obtained from Fourier transforms instead of LPC analysis.



Based on the above, a useful form of weighted cepstral distance is thus

$$d_{cW}^2 = \sum_{n=1}^{L} [w(n)c_n - w(n)c_n']^2, \tag{4.37}$$

where $w(n)$ is any reasonable lifter function such as that of Eq. (4.36). The difference between the index weighting function Eq. (4.26) and the raised sine function Eq. (4.36) is that the latter has a smooth roll off at both low and high indices while the former has a sharp cutoff at $n = L + 1$.

### 4.5.4 Likelihood Distortions

The log spectral difference $V(\omega)$ as defined by Eq. (4.13) is the basis of many speech distortion measures. The distortion measure originally proposed by Itakura and Saito (called the Itakura-Saito distortion measure) in their formulation of linear prediction as an approximate maximum likelihood estimation is [12]

$$\begin{aligned} d_{\mathrm{IS}}(S, S') &= \int_{-\pi}^{\pi} \left[e^{V(\omega)} - V(\omega) - 1\right] \frac{d\omega}{2\pi} \\ &= \int_{-\pi}^{\pi} \frac{S(\omega)}{S'(\omega)} \frac{d\omega}{2\pi} - \log \frac{\sigma_\infty^2}{\sigma_\infty'^2} - 1, \end{aligned} \tag{4.38}$$

where $\sigma_\infty^2$ and $\sigma_\infty'^2$ are the one-step prediction errors of $S(\omega)$ and $S'(\omega)$, respectively, as defined in Eq. (4.11). Besides maximum likelihood interpretations, the connection of the Itakura-Saito distortion measure with many statistical and information theoretic notions is also well established. These include the likelihood ratio test and relative entropy (also called discrimination information, Kullback-Leibler information, directed divergence). Although we have considered the Itakura-Saito measure a likelihood related quantity, our discussion in the following focuses on the speech spectrum comparison.

The Itakura-Saito distortion measure can be used to illustrate the spectral matching properties of linear prediction by replacing $S'(\omega)$ with a $p^{\text{th}}$ order all-pole spectrum $\sigma^2 / \left|A(e^{j\omega})\right|^2$, leading to

$$\begin{aligned} d_{\mathrm{IS}}\left(S, \frac{\sigma^2}{|A|^2}\right) &= \frac{1}{\sigma^2} \int_{-\pi}^{\pi} S(\omega) \left|A(e^{j\omega})\right|^2 \frac{d\omega}{2\pi} - \log \sigma_\infty^2 + \log \sigma^2 - 1 \\ &= \frac{\mathbf{a}^t \mathbf{R}_p \mathbf{a}}{\sigma^2} - \log \sigma_\infty^2 + \log \sigma^2 - 1. \end{aligned} \tag{4.39}$$

The first term in Eq. (4.39) is simply the ratio of the prediction residual energy to the gain term (squared), i.e., $\alpha/\sigma^2$. Minimization of the Itakura-Saito distortion, over the predictor coefficients $\mathbf{a}$, is thus equivalent to minimization of the residual energy; that is

$$\mathbf{a}_p = \arg\min_{\mathbf{a}} d_{\text{IS}}\left(S, \frac{\sigma^2}{|A|^2}\right) = \arg\min_{\mathbf{a}} \frac{\mathbf{a}^t \mathbf{R}_p \mathbf{a}}{\sigma^2}$$
$$= \arg\min_{\mathbf{a}} \mathbf{a}^t \mathbf{R}_p \mathbf{a}, \tag{4.40}$$

as reflected in Eq. (4.9). The distortion $d_{\text{IS}}\left(S, \frac{\sigma^2}{|A_p|^2}\right)$ can be further minimized by letting

$$\sigma^2 = \alpha(\mathbf{a}_p) \triangleq \sigma_p^2, \tag{4.41}$$

resulting in

$$\min_{\sigma/A} d_{\text{IS}}\left(S, \frac{\sigma^2}{|A|^2}\right)$$
$$= d_{\text{IS}}\left(S, \frac{\sigma_p^2}{|A_p|^2}\right)$$
$$= \log\left(\sigma_p^2/\sigma_\infty^2\right). \tag{4.42}$$

Note that the residual energy can be easily evaluated by writing

$$\mathbf{a}^t \mathbf{R}_p \mathbf{a} = r(0)r_a(0) + 2\sum_{n=1}^{p} r(n)r_a(n), \tag{4.43}$$

where

$$r_a(n) \triangleq \sum_{i=0}^{p-n} a_i a_{i+n}, \qquad \text{for } n = 0, 1, \ldots, p, \tag{4.44}$$

which is the autocorrelation of the predictor coefficients. (Recall Exercises 3.6 and 3.7.)

Suppose $S(\omega)$ in Eq. (4.39) is replaced by its optimal $p^{\text{th}}$ order LPC model spectrum $\sigma_p^2/|A_p|^2$; then

$$d_{\text{IS}}\left(\frac{\sigma_p^2}{|A_p|^2}, \frac{\sigma^2}{|A|^2}\right) = \frac{\sigma_p^2}{\sigma^2}\int_{-\pi}^{\pi} \frac{|A|^2}{|A_p|^2}\frac{d\omega}{2\pi} - \log\frac{\sigma_p^2}{\sigma^2} - 1. \tag{4.45}$$

Because the first $(p+1)$ autocorrelations, $r(0), r(1), \ldots, r(p)$, of $S(\omega)$ and $\sigma_p^2/|A_p|^2$ are matched (equal), (recall that the order of $A(e^{j\omega})$ and $A_p(e^{j\omega})$ are the same):

$$\int_{-\pi}^{\pi} S(\omega)\left|A(e^{j\omega})\right|^2 \frac{d\omega}{2\pi} = \int_{-\pi}^{\pi} \sigma_p^2 \frac{\left|A(e^{j\omega})\right|^2}{\left|A_p(e^{j\omega})\right|^2}\frac{d\omega}{2\pi} \tag{4.46}$$

and thus from (4.39)

$$d_{\text{IS}}\left(S, \frac{\sigma^2}{|A|^2}\right) = d_{\text{IS}}\left(\frac{\sigma_p^2}{|A_p|^2}, \frac{\sigma^2}{|A|^2}\right) + \log\frac{\sigma_p^2}{\sigma_\infty^2}$$



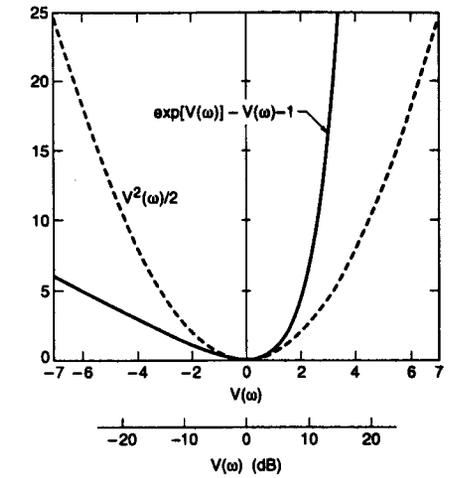



**Figure 4.22** Comparison of the distortion integrands $V^2(\omega)/2$ and $e^{V(\omega)} - V(\omega) - 1$ (after Gray and Markel [9]).

$$= d_{\text{IS}}\left(S, \frac{\sigma_p^2}{|A_p|^2}\right) + d_{\text{IS}}\left(\frac{\sigma_p^2}{|A_p|^2}, \frac{\sigma^2}{|A|^2}\right). \tag{4.47}$$

This additive property is an important characteristic of the Itakura-Saito distortion measure. It shows that the distortion between a signal spectrum and an arbitrary $p^{\text{th}}$ order all-pole model spectrum is the sum of two distortions: one from the signal spectrum to its optimal $p^{\text{th}}$ order all-pole model spectrum and the other from the optimal $p^{\text{th}}$ order all-pole spectrum (for $S(\omega)$) to the all-pole model spectrum under comparison.

It is of interest to compare the Itakura-Saito distortion to the log spectral distance of Eq. (4.17). The integrand of Eq. (4.38) shows an asymmetric weighting with respect to the value of $V(\omega)$. Plotted in Figure 4.22 is the curve $e^V - V - 1$ as a function of $V$, together with the curve $V^2/2$, which is the basis of the log spectral distance. The asymmetry is clearly visible; a positive spectral deviation $V(\omega)$ contributes more heavily toward the distortion than a negative $V(\omega)$ because

$$e^V - V - 1 \simeq e^V \qquad \text{for } V \gg 1$$

and

$$e^V - V - 1 \simeq -V \qquad \text{for } V \ll 1.$$

Furthermore, since

$$e^V - V - 1 = \frac{V^2}{2!} + \frac{V^3}{3!} + \cdots$$

$$\simeq \frac{V^2}{2}, \qquad \text{for } |V| \ll 1, \tag{4.48}$$

it is observed that

$$d_{\mathrm{IS}}(S, S') \simeq \frac{1}{2} d_2^2(S, S') \tag{4.49}$$

for "small" distortion.

It is important to note that the Itakura-Saito distortion is defined for two spectra without specific normalization of the overall spectral level (or gain). The gain matching property of Eq. (4.41) thus gives room for other possible definitions of the distortion. This is desirable because, as in the truncated cepstral distance case, the gain term or the overall spectral level is often treated in a manner separate from the comparison of the spectral shapes (i.e., $1/|A|^2$).

Suppose our emphasis is on the spectral shape. In evaluating $d_{\mathrm{IS}}(\sigma_p^2/|A_p|^2, \sigma^2/|A|^2)$, we let $\sigma^2$ vary such that it matches the residual energy $\alpha$ resulting from inverse filtering $x(i)$ with $A(z)$; that is, we set

$$\sigma^2 = \alpha = \int_{-\pi}^{\pi} S(\omega) \left|A(e^{j\omega})\right|^2 \frac{d\omega}{2\pi}. \tag{4.50}$$

Then, from Eq. (4.45), we have

$$\begin{aligned} d_{\mathrm{IS}}(\sigma_p^2/|A_p|^2, \alpha/|A|^2) &= \log \frac{\alpha}{\sigma_p^2} \\ &= \log \frac{\mathbf{a}^t \mathbf{R}_p \mathbf{a}}{\sigma_p^2} \end{aligned} \tag{4.51}$$

which is often referred to as the Itakura distortion measure in speech-recognition applications. Since $\mathbf{R}_p/\sigma_p^2$, the residual-normalized autocorrelation matrix, is associated with the spectrum $1/|A_p|^2$, another way to write the Itakura distortion measure is

$$\begin{aligned} d_{\mathrm{I}}(1/|A_p|^2, 1/|A|^2) &= \log \left\{ \int_{-\pi}^{\pi} \frac{\left|A(e^{j\omega})\right|^2}{\left|A_p(e^{j\omega})\right|^2} \frac{d\omega}{2\pi} \right\}, \\ &= \log \left( \frac{\alpha}{\sigma_p^2} \right) \end{aligned} \tag{4.52}$$

which is defined for two unity gain all-pole spectra. The role of the gain terms is not explicit in the Itakura distortion. This is considered sensible because the signal level essentially makes no difference in the human understanding of speech so long as it is unambiguously heard.

Another gain-independent distortion measure can be derived directly from the Itakura-Saito distortion measure. Traditionally, it is called the likelihood ratio distortion and is defined by

$$d_{\mathrm{LR}}\left(\frac{1}{|A_p|^2}, \frac{1}{|A|^2}\right) = d_{\mathrm{IS}}\left(\frac{1}{|A_p|^2}, \frac{1}{|A|^2}\right)$$



$$\begin{aligned} &= \int_{-\pi}^{\pi} \frac{\left|A(e^{j\omega})\right|^2}{\left|A_p(e^{j\omega})\right|^2} \frac{d\omega}{2\pi} - 1 \\ &= \frac{\mathbf{a}^t \mathbf{R}_p \mathbf{a}}{\sigma_p^2} - 1. \end{aligned} \tag{4.53}$$

Since

$$u = \exp(\log u) = 1 + \log u + \frac{1}{2!}(\log u)^2 + \frac{1}{3!}(\log u)^2 + \cdots,$$

and $\alpha/\sigma_p^2 \geq 1$,

$$\begin{aligned} d_{\mathrm{I}}(1/|A_p|^2, 1/|A|^2) &= \log \frac{\alpha}{\sigma_p^2} \\ &\simeq \frac{\alpha}{\sigma_p^2} - 1, \qquad \text{for } \frac{\alpha}{\sigma_p^2} \simeq 1, \\ &= d_{\mathrm{LR}}\left(\frac{1}{|A_p|^2}, \frac{1}{|A|^2}\right). \end{aligned} \tag{4.54}$$

That is, when the distortion is small, the Itakura distortion measure is not very different from the likelihood ratio distortion measure.

In the above discussion of the likelihood distortions (or the Itakura-Saito distortion family), we encountered the signal spectrum as well as various forms of the model spectra. When we measure the distortions among various forms of spectral representations in terms of "likelihood," we should be cautious in evaluating arithmetic operations on these distortions. For example, $d_{\mathrm{IS}}\left(\frac{\sigma_p^2}{|A_p|^2}, \frac{\sigma^2}{|A|^2}\right) \neq d_{\mathrm{IS}}\left(\frac{\sigma_p^2}{|A_p|^2}, \frac{\sigma_p^2}{|A|^2}\right) + d_{\mathrm{IS}}\left(\frac{\sigma_p^2}{|A|^2}, \frac{\sigma^2}{|A|^2}\right)$. Another particular property of the likelihood distortion that deserves careful consideration is that they are asymmetric—that is,

$$d_{\mathrm{IS}}(S, S') \neq d_{\mathrm{IS}}(S', S). \tag{4.55}$$

To illustrate this point, consider two signals, one being tonelike and the other being white noise. These two signals are represented in terms of the all-pole models $1/A_t(z)$ and $1/A_w(z)$, respectively, in which

$$A_t(z) = 1 - 1.2726\, z^{-1} + 0.81\, z^{-2}$$

and

$$A_w(z) = 1.$$

The two roots of $A_t(z)$ are $0.9\, e^{\pm j\pi/4}$ which indicate a resonance at $\pi/4$ normalized frequency or at 1000 Hz when the sampling frequency is 8000 Hz. It can be easily shown that

$$d_{\mathrm{IS}}\left(1/|A_t|^2, 1/|A_w|^2\right) = 4.75$$

and

$$d_{\mathrm{IS}}\left(1/|A_w|^2, 1/|A_t|^2\right) = 2.28.$$

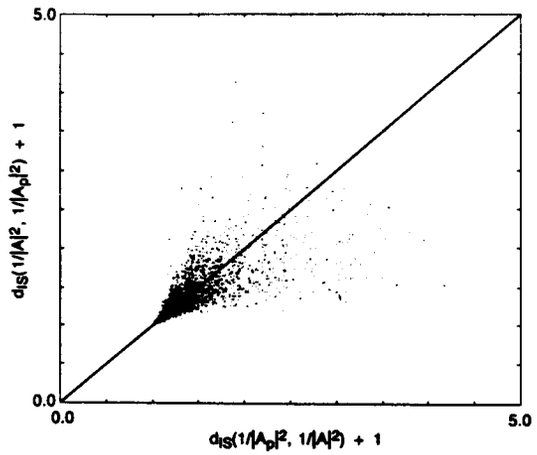



**Figure 4.23** A scatter plot of $d_{IS}(1/|A_p|^2, 1/|A|^2)+1$ versus $d_{IS}(1/|A|^2, 1/|A_p|^2)+1$ as measured from 6800 pairs of speech model spectra.

Therefore, when an input tonelike signal is compared to a white noise, the computed distortion is higher than that in the case in which a white noise signal is compared to a tonelike signal. Previous studies in auditory masking have shown a similar asymmetric behavior of masking between tone and noise; it is easier to perceive noise in a tone than it is to perceive a tone in noise. This can be construed as a strong justification for the Itakura-Saito distortions from strictly a subjective, qualitative viewpoint.

The asymmetry of the distortions is further seen in Figure 4.23 in which a scatter plot of $d_{IS}(1/|A_p|^2, 1/|A|^2)+1$ versus $d_{IS}(1/|A|^2, 1/|A_p|^2)+1$ as measured from real speech spectra is shown. Note that

$$d_{IS}(1/|A_p|^2, 1/|A|^2)+1 = \int_{-\pi}^{\pi} \frac{|A(e^{j\omega})|^2}{|A_p(e^{j\omega})|^2} \frac{d\omega}{2\pi}$$

and

$$d_{IS}(1/|A|^2, 1/|A_p|^2)+1 = \int_{-\pi}^{\pi} \frac{|A_p(e^{j\omega})|^2}{|A(e^{j\omega})|^2} \frac{d\omega}{2\pi}.$$

The asymmetry for large distortions is substantial.

Finally, another interpretation of the Itakura-Saito distortions is of particular interest from a signal-processing and filtering point of view. Consider the Itakura-Saito distortion between the input and output signals of a linear system $H(z)$, as shown in Figure 4.24. Let the input and output power spectra be $S(\omega)$ and $S'(\omega)$, respectively; clearly



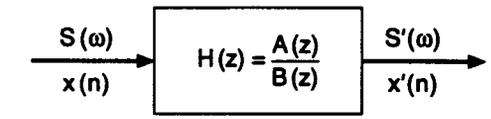



**Figure 4.24** A linear system with transfer function $H(z) = A(z)/B(z)$.

$$S'(\omega) = |H(e^{j\omega})|^2 S(\omega). \tag{4.56}$$

Substituting Eq. (4.56) into (4.13), we obtain

$$V(\omega) = -\log |H(e^{j\omega})|^2$$

and

$$d_{IS}(S, S') = \int_{-\pi}^{\pi} \left[ \frac{1}{|H(e^{j\omega})|^2} + \log |H(e^{j\omega})|^2 - 1 \right] \frac{d\omega}{2\pi}. \tag{4.57}$$

Usually, $H(z)$ has a rational form

$$H(z) = \frac{A(z)}{B(z)},$$

where the polynomials $A(z) = 1 + \sum_{i=1}^{p_1} a_i z^{-1}$ and $B(z) = 1 + \sum_{i=1}^{p_2} b_i z^{-1}$ have all their zeros inside the unit circle. Then

$$\begin{aligned} d_{IS}(S, S') &= \int_{-\pi}^{\pi} \frac{1}{|H(e^{j\omega})|^2} \frac{d\omega}{2\pi} - 1 = \int_{-\pi}^{\pi} \frac{|B(e^{j\omega})|^2}{|A(e^{j\omega})|^2} \frac{d\omega}{2\pi} - 1 \\ &= d_{IS}\left( \frac{1}{|A|^2}, \frac{1}{|B|^2} \right) \end{aligned} \tag{4.58}$$

due to the zero mean property of Eq. (4.12). This interpretation gives us a convenient means of modifying a signal to achieve a prescribed distortion level from the original signal, namely by applying the appropriate level of linear filtering.

### 4.5.5 Variations of Likelihood Distortions

Unlike the cepstral distance, likelihood distortions are generally asymmetric, as explained in the preceding section. When one needs to symmetrize the distortion measure while retaining the notion of likelihood or all-pole spectral matching, there are many possibilities, one of which is to combine the two directed distortion measurements in a form such as

$$d_x^{(m)}(S, S') = \frac{1}{2} \left\{ [d_{IS}(S, S')]^m + [d_{IS}(S', S)]^m \right\}^{1/m}. \tag{4.59}$$

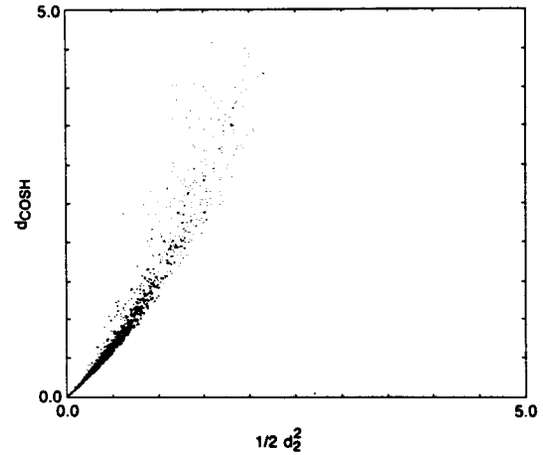



**Figure 4.25** A scatter diagram of the log spectral distance versus the COSH distortion as measured from a database of 6800 pairs of speech spectra.

If $m = 1$,

$$d_x^{(1)}(S,S') = \frac{1}{2}\left[d_{IS}(S,S') + d_{IS}(S',S)\right]. \tag{4.60}$$

This distortion $d_x^{(1)}(S,S')$ is termed a COSH distortion [9] because

$$\begin{aligned} d_x^{(1)}(S,S') &= \frac{1}{2}\int_{-\pi}^{\pi}\left[e^{V(\omega)} - V(\omega) - 1 + e^{-V(\omega)} + V(\omega) - 1\right]\frac{d\omega}{2\pi} \\ &= \int_{-\pi}^{\pi}\{\cosh[V(\omega)] - 1\}\frac{d\omega}{2\pi} \\ &\triangleq d_{COSH}(S,S'). \end{aligned} \tag{4.61}$$

Note that

$$\cosh V = 1 + \frac{V^2}{2!} + \frac{V^4}{4!} + \cdots.$$

Therefore,

$$d_{COSH}(S,S') \geq \frac{1}{2}d_2^2(S,S'). \tag{4.62}$$

Figure 4.25 shows a scatter diagram, obtained from real speech data, with the log spectral distance on the horizontal axis and the COSH distortion on the vertical axis. The COSH distortion is almost identical to twice the log spectral distance for small distortions but spreads out significantly above the equality line for large distortions.

The COSH distortion measure can be interpreted based on the theory of random



processes. It is related to the symmetrized directed divergence (or simply divergence) between $N^{th}$ order probability densities ($N$ being the number of data values in the sequence). It can also be interpreted as a distance, measuring how closely two specific Gaussian processes match each other. Beyond the theoretical appeal, however, the COSH measure has not been extensively used in speech-recognition applications.

Another possible variation on the likelihood distortion is the family of weighted likelihood distortions. The purpose of weighting here is to take the spectral shape directly into account as a weighting function such that different spectral components along the frequency axis can be emphasized or de-emphasized to reflect some of the observed perceptual effects. To see how various weighting schemes have been used, we first unify the distortion measures already presented by defining

$$(1)F_1(\omega) \triangleq |\log S(\omega) - \log S'(\omega)|^2 \tag{4.63}$$

$$(2)F_2(\omega) \triangleq \log\frac{S'(\omega)}{S(\omega)} + \frac{S(\omega)}{S'(\omega)} - 1 \tag{4.64}$$

$$(3)F_3(\omega) \triangleq \log\frac{S(\omega)}{S'(\omega)} + \frac{S'(\omega)}{S(\omega)} - 1 \tag{4.65}$$

$$(4)F_4(\omega) = \frac{1}{2}\left[\frac{S'(\omega)}{S(\omega)} + \frac{S(\omega)}{S'(\omega)}\right] - 1. \tag{4.66}$$

Using this notation we can express the various distortion measures already discussed as

$$d_2^2(S,S') = \int_{-\pi}^{\pi} F_1(\omega)\frac{d\omega}{2\pi} \tag{4.67}$$

$$d_{IS}(S,S') = \int_{-\pi}^{\pi} F_2(\omega)\frac{d\omega}{2\pi} \tag{4.68}$$

$$d_{IS}(S',S) = \int_{-\pi}^{\pi} F_3(\omega)\frac{d\omega}{2\pi} \tag{4.69}$$

$$d_{COSH}(S,S') = \int_{-\pi}^{\pi} F_4(\omega)\frac{d\omega}{2\pi}. \tag{4.70}$$

Again, assume that $\sigma^2/\left|A(e^{j\omega})\right|^2$ and $\sigma'^2/\left|A'(e^{j\omega})\right|^2$ are the optimal $p^{th}$ order all-pole model spectra for $S(\omega)$ and $S'(\omega)$, respectively. We can now examine the following weighting functions:

$$(1) \qquad W_1(\omega) = \frac{1}{2}\left[\frac{1}{|A(e^{j\omega})|^2} + \frac{1}{|A'(e^{j\omega})|^2}\right] \tag{4.71}$$

$$(2) \qquad W_2(\omega) = \frac{1}{\sigma^2 + \sigma'^2}\left[\frac{\sigma^2}{|A(e^{j\omega})|^2} + \frac{\sigma'^2}{|A'(e^{j\omega})|^2}\right] \tag{4.72}$$

$$(3) \qquad W_3(\omega) = \frac{1}{|A(e^{j\omega})|^2} \tag{4.73}$$

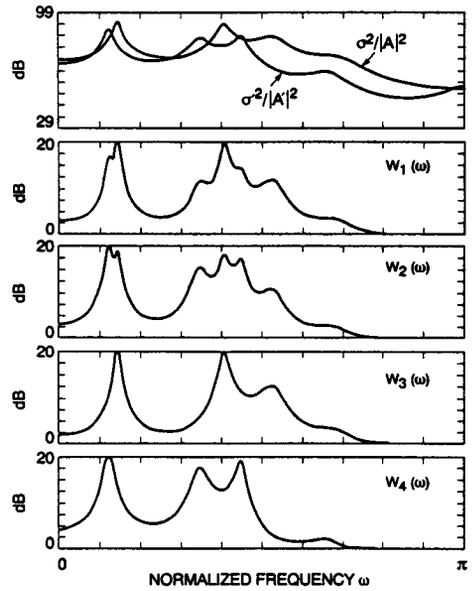



**Figure 4.26** LPC spectral pair and various spectral weighting functions; $W_1(\omega)$, $W_2(\omega)$, $W_3(\omega)$ and $W_4(\omega)$ are defined in (4.71), (4.72), (4.73), and (4.74), respectively.

$$(4) \qquad W_4(\omega) = \frac{1}{|A'(e^{j\omega})|^2}. \tag{4.74}$$

Figure 4.26 shows an example of the use of these weighting functions for a given pair of all-pole spectra. [For visual effect, the weighting functions plotted in Figure 4.26 are in fact $10 \log_{10}\{W(\omega) - \min_{\omega} W(\omega)\}$ on a dB scale.] As can be seen, $W_3$ and $W_4$ attempt to emphasize the spectral peaks in $1/|A(e^{j\omega})|^2$ and $1/|A'(e^{j\omega})|^2$, respectively, while $W_1$ and $W_2$ weight both spectra in a balanced, symmetric fashion. These weighting functions can be incorporated in the integrand of Eq. (4.67)–(4.70) to achieve the desired effects of nonuniform frequency weighting. For example, an asymmetrically weighted COSH distortion can be defined by using $W_3(\omega)$ to weight the spectral deviation $F_4(\omega)$, leading to

$$d_{\mathrm{W\,COSH}}(S, S') = \int_{-\pi}^{\pi} F_4(\omega) W_3(\omega) \frac{d\omega}{2\pi}. \tag{4.75}$$

Again, when $S(\omega)$ and $S'(\omega)$ are replaced by their corresponding optimal $p^{\text{th}}$ order all-pole model spectra $\sigma^2/|A(e^{j\omega})|^2$ and $\sigma'^2/|A'(e^{j\omega})|^2$, evaluation of the asymmetrically weighted



COSH distortion measure can be accomplished by recognizing the fact that

$$\int_{-\pi}^{\pi} \frac{1}{|A(e^{j\omega})|^2} \frac{d\omega}{2\pi} = \frac{r(0)}{\sigma^2} \tag{4.76a}$$

and

$$\int_{-\pi}^{\pi} \frac{1}{|A'(e^{j\omega})|^2} \frac{d\omega}{2\pi} = \frac{r'(0)}{\sigma'^2}, \tag{4.76b}$$

where $r(0)$ and $r'(0)$ are the energy of $S(\omega)$ and $S'(\omega)$ respectively. Specifically,

$$\begin{aligned} d_{\mathrm{WCOSH}}\left(\frac{\sigma^2}{|A|^2}, \frac{\sigma'^2}{|A'|^2}\right) &= \frac{1}{2}\int_{-\pi}^{\pi} \left[\frac{\sigma'^2}{\sigma^2 |A'(e^{j\omega})|^2} + \frac{\sigma^2 |A'(e^{j\omega})|^2}{\sigma'^2 |A(e^{j\omega})|^4}\right] \frac{d\omega}{2\pi} \\ &\quad - \int_{-\pi}^{\pi} \frac{1}{|A(e^{j\omega})|^2} \frac{d\omega}{2\pi} \\ &= \frac{1}{2}\left\{\frac{r'(0)}{\sigma^2} + \frac{\sum_{i=-p}^{p} r'_a(i)\tilde{r}(i)}{\sigma'^2\sigma^2}\right\} - \frac{r(0)}{\sigma^2}, \end{aligned} \tag{4.77}$$

where $r'_a(i)$ are the autocorrelations of the predictor coefficients of $A'(z)$ defined in a way similar to Eq. (4.44), and

$$\tilde{r}(i) = \sum_n \hat{r}(n)\hat{r}(n+i) \tag{4.78}$$

is the autocorrelation of the autocorrelations for $\sigma^2/|A(e^{j\omega})|^2$. The autocorrelations of $\sigma^2/|A(e^{j\omega})|^2$, $\hat{r}(n)$, are defined by

$$\hat{r}(n) = \begin{cases} r(n), & n = 0, 1, 2, \ldots, p \\ \sum_{i=1}^{p} a_i r(n-i), & n \geq p+1 \end{cases} \tag{4.79}$$

which has an extension beyond $n = p$ according to the well known maximum entropy principle. Figure 4.27 shows two examples of the integrands $F_4(\omega)$ and $F_4(\omega)W_3(\omega)$, on a dB scale, as functions of frequency. Figure 4.27a plots two LPC model spectra of similar power level; major spectral deviations of the distortion occur near spectral peaks. The distortion emphasis in different frequency regions is clearly observed. In Figure 4.27b, two LPC model spectra with an overall power level difference of approximately 15 dB are compared. The COSH integrand $F_4(\omega)$ does not show obvious dependence on the general trend of the spectra being compared. After weighting by $1/|A(e^{j\omega})|^2$ which comes from the high-level spectrum, the weighted integrand $F_4(\omega)W_3(\omega)$ essentially has the same shape as $1/|A(e^{j\omega})|^2$, showing to a certain degree the desired effects of masking.

A particular weighted distortion measure that has received considerable attention is the weighted likelihood ratio distortion. By weighting $F_2(\omega)$ and $F_3(\omega)$ with $W_4(\omega)$ and $W_3(\omega)$, respectively, we form a new weighted spectral deviation function

$$F_6(\omega) = \frac{1}{2}\left[F_2(\omega)W_4(\omega) + F_3(\omega)W_3(\omega)\right]. \tag{4.80}$$

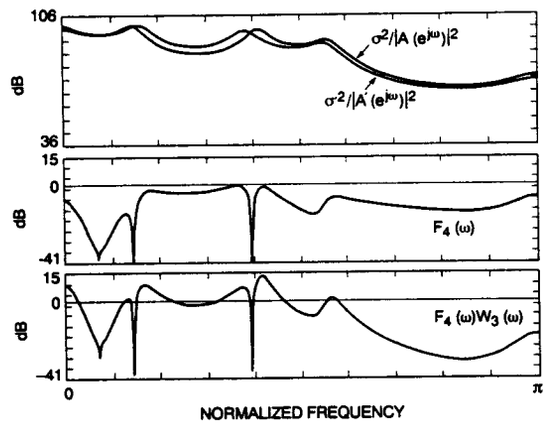



**Figure 4.27a** An example of the cosh spectral deviation $F_4(\omega)$ and its weighted version using $W_3(\omega) = 1/\left|A(e^{j\omega})\right|^2$ as the weighting function; in this case the two spectra are of comparable power levels.

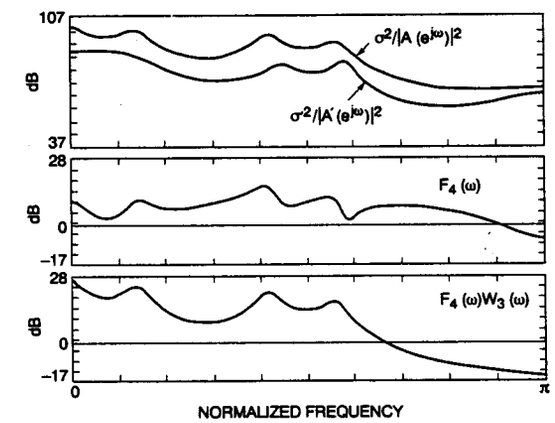



**Figure 4.27b** An example of the cosh spectral deviation $F_4(\omega)$ and its weighted version using $W_3(\omega) = 1/\left|A(e^{j\omega})\right|^2$ as the weighting function; in this case, the two spectra have significantly different power levels.



The weighted likelihood ratio distortion is then defined by integrating $F_6(\omega)$ over the frequency range $[-\pi, \pi]$

$$d_{\mathrm{WLR}}(S, S') = \int_{-\pi}^{\pi} F_6(\omega)\frac{d\omega}{2\pi}. \tag{4.81}$$

If the two spectra being compared are unity gain all-pole spectra, that is, $S \to 1/|A|^2$ and $S' \to 1/|A'|^2$, then

$$F_6 = \frac{1}{2}\left(\frac{1}{|A|^2} - \frac{1}{|A'|^2}\right)\left(\log\frac{1}{|A|^2} - \log\frac{1}{|A'|^2}\right). \tag{4.82}$$

Integration of $F_6(\omega)$ can be evaluated by

$$\frac{1}{2}\int_{-\pi}^{\pi}\left\{\frac{1}{|A(e^{j\omega})|^2} - \frac{1}{|A'(e^{j\omega})|^2}\right\}\left\{\log\frac{1}{|A(e^{j\omega})|^2} - \log\frac{1}{|A'(e^{j\omega})|^2}\right\}\frac{d\omega}{2\pi}$$
$$= \sum_{n=1}^{\infty}\left[\frac{\hat{r}(n)}{\sigma^2} - \frac{\hat{r}'(n)}{\sigma'^2}\right](c_n - c_n') \tag{4.83}$$

where $c_n$ and $c_n'$ are cepstral coefficients of $\log\frac{1}{|A|^2}$ and $\log\frac{1}{|A'|^2}$, respectively, defined by Eq. (4.20); $\sigma^2$ and $\sigma'^2$ are the minimum residual energy of $S(\omega)$ and $S'(\omega)$, respectively; and $\hat{r}(n)$ and $\hat{r}'(n)$ are defined according to Eq. (4.79) for $\sigma^2/|A|^2$ and $\sigma'^2/|A'|^2$ in an identical manner. Note that both $c_0$ and $c_0'$ are not involved in the weighted likelihood ratio measure and the factor $1/2$ in Eq. (4.83) is eliminated because the autocorrelation sequences and the cepstral sequences are all symmetric with respect to $n = 0$.

It is interesting to compare $d_{\mathrm{WLR}}$ in Eq. (4.83) to $d_2^2$ according to Eq. (4.14). In $d_2^2$, the log spectral deviation ($\log 1/|A|^2 - \log 1/|A'|^2$) can be considered to be weighted by itself. In $d_{\mathrm{WLR}}$, this weighting is replaced by the linear deviation ($1/|A|^2 - 1/|A'|^2$) which shows a heavier emphasis in spectral peak areas than the compressed deviation ($\log 1/|A|^2 - \log 1/|A'|^2$). This property may be desirable in applications where extraordinary emphasis on spectral peaks is necessary, such as speech recognition in noisy environments.

### 4.5.6 Spectral Distortion Using a Warped Frequency Scale

Psychophysical studies have shown that human perception of the frequency content of sounds, either for pure tones or for speech signals, does not follow a linear scale. This research has led to the idea of defining subjective pitch of pure tones. Thus for each tone with an actual frequency, $f$, measured in Hz, a subjective pitch is measured on a scale called the "mel" scale. As a reference point, the pitch of a 1 kHz tone, 40 dB above the perceptual hearing threshold, is defined as 1000 mels. Other subjective pitch values are obtained by adjusting the frequency of a tone such that it is half or twice the perceived

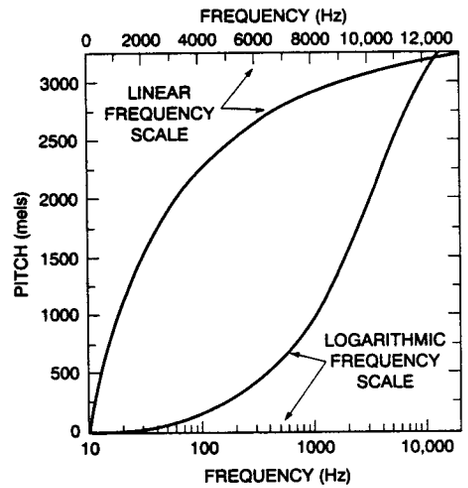



**Figure 4.28** Subjectively perceived pitch, in mels, of a tone as a function of the frequency, in Hz; the upper curve relates the subjective pitch to frequency in a linear scale and the lower curve shows the subjective pitch as a function of the frequency in a logarithmic scale (after Stevens and Volkmann [13]).

pitch of a reference tone (with a known mel frequency). Figure 4.28, taken from the classic paper of Stevens and Volkmann [13], shows subjective pitch as a function of frequency. The upper curve shows the relationship of the subjective pitch to frequency on a linear scale; it shows that subjective pitch in mels increases less and less rapidly as the stimulus frequency is increased linearly. The lower curve, on the other hand, shows the subjective pitch as a function of the logarithmic frequency; it indicates that the subjective pitch is essentially linear with the logarithmic frequency beyond about 1000 Hz.

Another important subjective criterion of the frequency content of a signal is the critical band that refers to the bandwidth at which subjective responses, such as loudness, become significantly different. The loudness of a band of noise at a constant sound pressure remains constant as the noise bandwidth increases up to the width of the critical band; after that increased loudness is perceived. Similarly, a subcritical bandwidth complex sound (multitone) of constant intensity is about as loud as an equally intense pure tone of a frequency lying at the center of the band, regardless of the overall frequency separation of the multiple tones. When the separation exceeds the critical bandwidth, the complex sound is perceived as becoming louder. Figure 4.29 shows plots of the critical bandwidth as a function of the frequency at the center of the band, based on the measurements by Zwicker and his colleagues [14]. Table 4.2 provides a set of approximations to measured critical bandwidths (reliable within about ±15%).



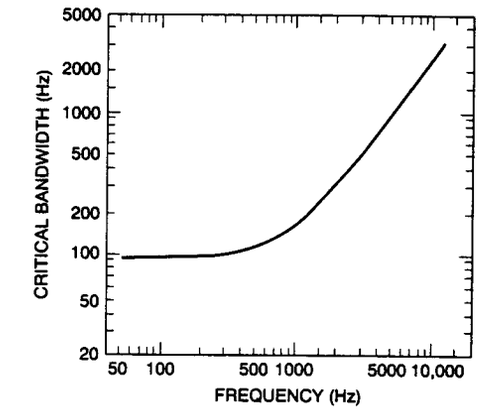



**Figure 4.29** The critical bandwidth phenomenon; the critical bandwidth as a function of the frequency at the center of the band (after Zwicker, Flottorp and Stevens [14]).

As can be seen from Figures 4.28 and 4.29, there is a certain resemblance between the critical bandwidth and the subjective pitch in mels (on the log frequency scale). In fact, it has been conjectured that critical bands as well as equal mel intervals of subjective pitch, and even difference limens, correspond to constant spatial distances along the basilar membrane. If we assume 24 contiguous, nonoverlapping critical bands laid on a 32-mm-long basilar membrane, each critical band is seen to correspond to approximately 1.3 mm in length. Also, in a critical band, the subjective pitch range is fairly constant, at about 150 mels.

The subjective nonlinear perception of frequency has led to an objective computational model that provides a mechanism to convert a physically measured spectrum of a given sound into a psychological, "subjective spectrum." In the model, each frequency component of the spectrum is replaced by a specific loudness level according to an empirical power law over a range of "tonalness" units. A unit of tonalness corresponds in width to a critical band and is called a Bark (after the German acoustician Barkhausen). For example, an ideal pure tone that has a single-line physical spectrum is represented by a "subjective spectrum" with spread over several Barks, as inferred from the corresponding masking audiogram.[1]

When applied to speech perception, however, the significance of critical bands is not as straightforward as the case of tone perception. By measuring the intelligibility of

[1]An audiogram measures a subject's sensitivity to a pure tone at various frequencies with quiet background; a masking audiogram is an audiogram measured with constant presence of a selected pure tone. (A series of masking audiograms can be obtained when the frequency of the selected pure tone is varied over an intended range.)

**TABLE 4.2.** Examples of Critical Bandwidth

| Critical Band Number | Center Frequency (Hz) | Critical Band (Hz) | Lower Cutoff Frequency (Hz) | Upper Cutoff Frequency (Hz) |
|---|---|---|---|---|
| 1 | 50 | – | – | 100 |
| 2 | 150 | 100 | 100 | 200 |
| 3 | 250 | 100 | 200 | 300 |
| 4 | 350 | 100 | 300 | 400 |
| 5 | 450 | 110 | 400 | 510 |
| 6 | 570 | 120 | 510 | 630 |
| 7 | 700 | 140 | 630 | 770 |
| 8 | 840 | 150 | 770 | 920 |
| 9 | 1,000 | 160 | 920 | 1,080 |
| 10 | 1,170 | 190 | 1,080 | 1,270 |
| 11 | 1,370 | 210 | 1,270 | 1,480 |
| 12 | 1,600 | 240 | 1,480 | 1,720 |
| 13 | 1,850 | 280 | 1,720 | 2,000 |
| 14 | 2,150 | 320 | 2,000 | 2,320 |
| 15 | 2,500 | 380 | 2,320 | 2,700 |
| 16 | 2,900 | 450 | 2,700 | 3,150 |
| 17 | 3,400 | 550 | 3,150 | 3,700 |
| 18 | 4,000 | 700 | 3,700 | 4,400 |
| 19 | 4,800 | 900 | 4,400 | 5,300 |
| 20 | 5,800 | 1,100 | 5,300 | 6,400 |
| 21 | 7,000 | 1,300 | 6,400 | 7,700 |
| 22 | 8,500 | 1,800 | 7,700 | 9,500 |
| 23 | 10,500 | 2,500 | 9,500 | 12,000 |
| 24 | 13,500 | 3,500 | 12,000 | 15,500 |

highpass and lowpass filtered speech, French and Steinberg [15] found that 20 adjacent frequency bands, each approximating a critical band, contribute about equally to the intelligibility of the speech, leading to the concept of the articulation index. However, it was also found that above 1500 Hz, the width of these equal intelligibility-bearing bands does not grow as rapidly with frequency as does the critical bandwidth.

The phenomena of the mel pitch scale and the critical band suggest some possible modifications of the spectral distance measure. One such modification is to first warp the frequency scale to the mel or Bark scale in order to make more meaningful distance calculations.

To see how such a frequency scale warping can be achieved in practice, consider the $L_2$ log spectral distance of Eq. (4.14) and a Bark scale warping. The warped cepstral distance, $\tilde{d}_2^2$, can be computed as [16]

$$\tilde{d}_2^2(S,S') = \int_{-B}^{B} |\log S(\theta(b)) - \log S'(\theta(b))|^2 \frac{db}{2B}, \tag{4.84}$$

where $b$ is frequency in Barks, $S(\theta(b))$ is the spectrum on a Bark scale, and $B$ is the Nyquist frequency in Barks. Note that $\theta$ is a nonlinear function that maps the Bark scale to the



linear frequency scale, i.e. $\omega = \theta(b)$. Using an equivalent representation of Eq. (4.15), we can rewrite Eq. (4.84) as

$$\begin{aligned}\tilde{d}_2^2 &= \frac{1}{2B}\sum_{i=-\infty}^{\infty}\sum_{k=-\infty}^{\infty}(c_i - c_i')(c_k - c_k')\int_{-B}^{B} e^{j\theta(b)(i-k)}db \\ &= \sum_{i=-\infty}^{\infty}\sum_{k=-\infty}^{\infty}(c_i - c_i')(c_k - c_k')w_{ik},\end{aligned} \tag{4.85}$$

where the warping function $w_{ik}$ is defined by

$$w_{ik} = \int_{-B}^{B} e^{j\theta(b)(i-k)}\frac{db}{2B}. \tag{4.86}$$

Like the cepstral distance, we can define a truncated warped cepstral distance by imposing a limit, say $\pm L$, to the summation range in Eq. (4.85); i.e.,

$$\tilde{d}_c^2(L) = \sum_{i=-L}^{L}\sum_{k=-L}^{L}(c_i - c_i')(c_k - c_k')w_{ik}. \tag{4.87}$$

(Typically, $L = 16$ for $8 - 10^{\text{th}}$ order LPC smooth model spectral representations.) Because we assume $\theta(-b) = -\theta(b)$ for negative frequencies, the warping function $w_{ij}$ depends only on the absolute difference of the indices, $w_{ik} = w_{|i-k|}$, and therefore the warping matrix $\mathbf{W} = [w_{ik}]$ is of a Toeplitz form. The warped distance can thus be written, with matrix notation, as

$$\tilde{d}_c^2(L) = (\mathbf{c} - \mathbf{c}')^t\,\mathbf{W}(\mathbf{c} - \mathbf{c}'), \tag{4.88}$$

where $\mathbf{c}^t = [c_{-L}, \ldots, c_0, \ldots, c_L]$. When $c_0$ is to be excluded from the distance calculation, we set $c_0 = c_0'$.

The value of $w_{ik}$ can be evaluated by numerical integration along the Bark scale. Figure 4.30 shows the real part of $\exp[j\theta(b)k]$ for different values of $k$. The $x$-axis of the plots spans from 0 to 16.2 Barks, corresponding to a bandwidth of 3333 Hz. Table 4.3 lists the values of $w_{ik}$ for $0 \le |i-k| \le 64$. In obtaining Table 4.3, $\theta(b)$ is defined by

$$\begin{aligned}\theta(b) &= \theta_1(b) = \left(\frac{\pi}{3333}\right)\left(\frac{1000}{0.76}\right)\tan\left(\frac{b}{13}\right) && \text{for } |b| \le 6 \\ \theta(b) &= \theta_2(b) = \left(\frac{\pi}{3333}\right)(1000)\,10^{(b-8.776)/10} && \text{for } |b| \ge 13 \\ \theta(b) &= \frac{1}{2}[\theta_1(b) + \theta_2(b)] && \text{for } 6 < |b| < 13.\end{aligned} \tag{4.89}$$

For efficient calculation of $\tilde{d}_c^2$, the square root of the warping matrix, $\mathbf{W}^{1/2}$ ($\mathbf{W} = \mathbf{W}^{1/2}\mathbf{W}^{1/2}$), can be used to first transform the cepstral coefficients, followed by a direct Euclidean distance calculation. In this manner, $\mathbf{W}^{1/2}$ could be considered part of the front-end processing. In an automatic speech-recognition system, where reference speech patterns

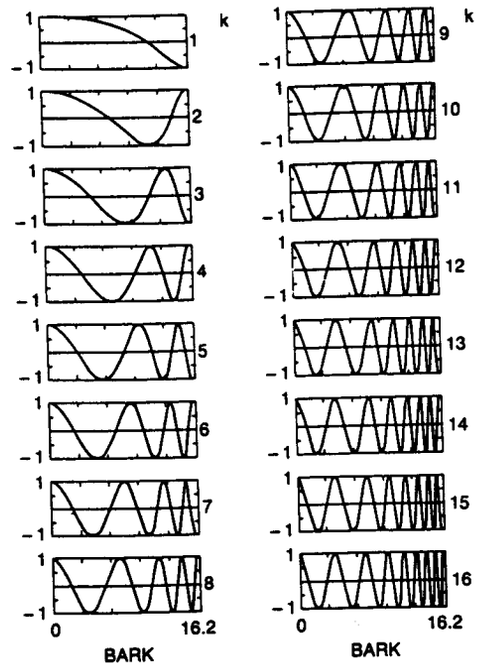



**Figure 4.30** Real part of exp $[j\theta(b)k]$ as a function of $b$, the Bark scale, for different values of $k$ (after Nocerino et al. [16]).

have to be repetitively compared, this leads to a substantial reduction in computational requirements.

The same frequency-warping techniques can be applied to other spectral distortion measures. The warped cepstral distance appears to be the most straightforward in terms of computational efficiency.

Besides nonlinear frequency warping, the notion of critical band can also be incorporated in defining spectral distortion. This is equivalent to considering the spectral deviations of the "subjective spectrum" discussed previously. Note that in a digital implementation of the frequency warping scheme, $S(\theta(b))$ represents an interpolated and then nonuniformly sampled spectrum of $S(\omega)$. The subjective spectrum, nevertheless, is associated with loudness, a perceptual quantity, integrated over possible contributions from all applicable spectral components due to the inclusion of the masking effects implied in the critical bandwidth phenomenon.

One approach to simulating the subjective spectrum is to use a filter bank, spaced uniformly on a nonlinear, warped frequency scale, such as the mel scale or the Bark scale



**TABLE 4.3.** Values of the Elements in the Warping Matrix

| $\lvert i-k\rvert$ | $w_{ik}$ | $\lvert i-k\rvert$ | $w_{ik}$ |
|---|---|---|---|
| 0 | 1.00 | 33 | −1.13E−3 |
| 1 | 0.37 | 34 | −5.75E−5 |
| 2 | 0.12 | 35 | −9.54E−3 |
| 3 | 3.02E−2 | 36 | 3.70E−3 |
| 4 | 2.33E−3 | 37 | −3.24E−3 |
| 5 | −1.43E−2 | 38 | −2.74E−4 |
| 6 | 2.34E−3 | 39 | 1.19E−4 |
| 7 | −1.78E−3 | 40 | 8.14E−3 |
| 8 | 6.70E−3 | 41 | −3.69E−3 |
| 9 | 1.00E−3 | 42 | 2.68E−3 |
| 10 | 1.36E−3 | 43 | 6.53E−4 |
| 11 | −5.25E−3 | 44 | −1.19E−3 |
| 12 | 1.61E−3 | 45 | −7.70E−3 |
| 13 | −5.57E−3 | 46 | 2.68E−3 |
| 14 | 3.13E−4 | 47 | −3.06E−3 |
| 15 | −1.28E−2 | 48 | −1.95E−3 |
| 16 | 5.09E−3 | 49 | 1.30E−3 |
| 17 | −2.65E−3 | 50 | 6.26E−3 |
| 18 | 3.74E−3 | 51 | −2.65E−3 |
| 19 | −1.27E−3 | 52 | 2.45E−3 |
| 20 | 1.15E−2 | 53 | 2.22E−3 |
| 21 | −5.42E−3 | 54 | −2.36E−3 |
| 22 | 2.51E−3 | 55 | −5.80E−3 |
| 23 | −3.10E−3 | 56 | 1.66E−3 |
| 24 | 7.26E−4 | 57 | −2.82E−3 |
| 25 | −1.12E−2 | 58 | −3.37E−3 |
| 26 | 4.59E−3 | 59 | 2.40E−3 |
| 27 | −3.19E−3 | 60 | 4.39E−3 |
| 28 | 1.60E−3 | 61 | −1.68E−3 |
| 29 | −8.97E−4 | 62 | 2.24E−3 |
| 30 | 9.91E−3 | 63 | 3.47E−3 |
| 31 | −4.66E−3 | 64 | −3.35E−3 |
| 32 | 2.77E−3 | | |

[17]. Figure 4.31 shows an example of such a filter bank, in which each filter has a triangular bandpass frequency response, and the spacing as well as the bandwidth is determined by a constant mel frequency interval. (The spacing is approximately 150 mels and the width of the triangle is 300 mels.) The modified spectrum of $S(\omega)$ thus consists of the output power of these filters when $S(\omega)$ is the input. Denoting these power coefficients by $\tilde{S}_k$, $k = 1, 2, \ldots, K$, we can calculate what is called the mel-frequency cepstrum, $\tilde{c}_n$, as

$$\tilde{c}_n = \sum_{k=1}^{K} (\log \tilde{S}_k) \cos\left[n\left(k - \frac{1}{2}\right)\frac{\pi}{K}\right], \qquad n = 1, 2, \ldots, L, \tag{4.90}$$

where $L$ is the desired length of the cepstrum. Using the mel-frequency cepstrum in

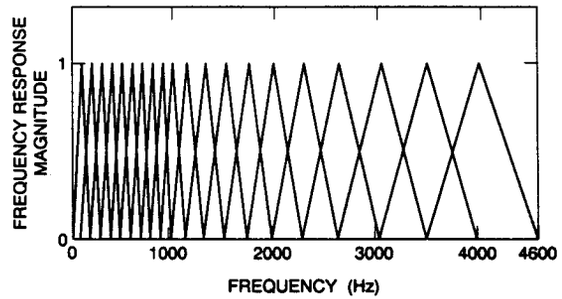



**Figure 4.31** A filter-bank design in which each filter has a triangle bandpass frequency response with bandwidth and spacing determined by a constant mel frequency interval (spacing = 150 mels, bandwidth = 300 mels) (after Davis and Mermelstein [17]).

Eq. (4.24) results in a mel-frequency cepstral distance, $d_{\tilde{c}}^2(L)$,

$$d_{\tilde{c}}^2(L) = \sum_{n=1}^{L} (\tilde{c}_n - \tilde{c}'_n)^2 \tag{4.91}$$

for the two spectra $S(\omega)$ and $S'(\omega)$.

### 4.5.7 Alternative Spectral Representations and Distortion Measures

A speech spectrum can be represented in a number of ways. The cepstrum is just one popular choice. When the speech spectrum is modeled by an all-pole spectrum, many other parametric representations are also possible. An obvious parameter set that defines the all-pole spectral shape is the set of $p$ (nontrivial) coefficients $a_i$ of the predictor $A(z)$. Another important transformation of the predictor coefficients is the set of "partial correlation" (PARCOR) coefficients or reflection coefficients denoted by $\{k_i\}_{i=1}^{p}$. The reflection coefficients are directly obtained in the autocorrelation method for linear prediction analysis as described in Chapter 3.

The reflection coefficients can be related to the cross-sectional areas of a nonuniform acoustic tube, which is used to model the vocal tract of the speech-production apparatus. As shown in Figure 4.32, the vocal tract can be approximated by stacking together $p$ equal length cylindrical sections, each with a constant cross-sectional area $\mathcal{A}_i$, $i = 1, 2, \ldots, p$. ($\mathcal{A}_{p+1} = \mathcal{A}_p$ for impedance matching at the glottis.) When air travels through these sections of unequal areas, wave reflection occurs at each sectional boundary with reflection coefficients denoted by $k_i$, $i = 1, 2, \ldots, p$. From these area parameters it is customary to define the area ratio

$$g_i = \frac{\mathcal{A}_{i+1}}{\mathcal{A}_i} = \frac{1 - k_i}{1 + k_i}, \qquad i = 1, 2, \ldots, p \tag{4.92}$$



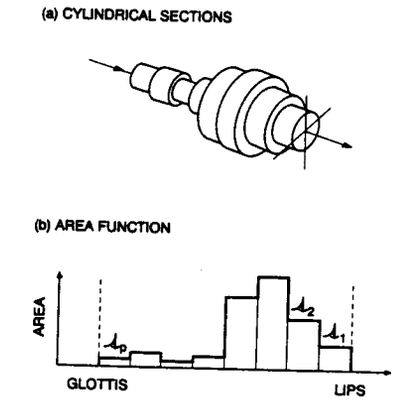



**Figure 4.32** (a) Series of cylindrical sections concatenated as an acoustic tube model of the vocal tract; (b) the area function of the cylindrical sections in (a) (after Markel and Gray [10]).

and the log area ratio

$$\log g_i = \log \frac{\mathcal{A}_{i+1}}{\mathcal{A}_i} = \log \frac{1 - k_i}{1 + k_i}, \tag{4.93}$$

and use these parameters for coding and quantization.

Yet another possible parametric representation of the all-pole spectrum is the set of line spectral frequencies (LSFs) defined as the roots of the following two polynomials based upon the inverse filter $A(z)$:

$$P(z) = A(z) + z^{-(p+1)} A(z^{-1}) \tag{4.94}$$

and

$$Q(z) = A(z) - z^{-(p+1)} A(z^{-1}). \tag{4.95}$$

These two polynomials are equivalent to artificially augmenting the $p$-section nonuniform acoustic tube of Figure 4.32, with an extra section that is either completely closed (area = 0) or completely open (area = $\infty$). LSF parameters, due to their particular structure, possess properties similar to those of the formant frequencies and bandwidths.

Given these alternative parametric representations of an all-pole spectrum, one can devise several reasonable spectral distortion measures. A Euclidean distance, defined on the predictor coefficients directly, is usually considered an inadequate measure of the spectral difference, unless the two spectra are extremely close to each other. This is because small deviations in the predictor coefficients can result in an unstable all-pole filter model, and any measurement of spectral difference involving the spectrum (spectral response) of an unstable filter usually does not have much physical significance.

For other parametric representations, such as log area ratio coefficients, a Euclidean

distance of the form

$$d_g^2 = \sum_{i=1}^{p} (\log g_i - \log g_i')^2 \tag{4.96}$$

might actually be reasonable. However, spectral distortions based on parameters such as log area ratios have not been extensively studied because it is relatively difficult to interpret the measured distortion in terms of spectral deviations.

In contrast to distortion measures involving parametric forms of predictor parameters (or their transforms), another proposed spectral distortion measure, motivated primarily by perceptual studies, is the weighted slope metric [16]. Based on a series of experiments designed to measure the subjective "phonetic" distance between pairs of synthetic vowels and fricatives, Klatt found that by controlled variations of several acoustic parameters and spectral distortions including formant frequency, formant amplitude, spectral tilt, highpass, lowpass, and notch filtering, only formant frequency deviation was phonetically relevant. (The synthetic vowels and fricatives were produced using a particular formant synthesizer structure.) The resulting distortion measure proposed by Klatt, called the weighted slope metric (WSM), attached a great deal of weight on the spectral slope difference near spectral peaks, rather than the spectral magnitude difference, and took the overall energy difference explicitly into consideration. The WSM proposed by Klatt was of the form

$$d_{\mathrm{WSM}}(S, S') = u_E\,|E_S - E_{S'}| + \sum_{i=1}^{K} u(i)[\Delta_S(i) - \Delta_S'(i)]^2, \tag{4.97}$$

where $u_E$ is the weighting constant for the absolute energy difference $|E_S - E_{S'}|$, between $S$ and $S'$, $u(i)$ is the weighting coefficient for the critical band spectral slope difference $\Delta_S(i) - \Delta_S'(i)$, between $S$ and $S'$, and $K$ is the total number of critical bands considered. In one implementation, the slope weighting coefficient is chosen to be

$$u(i) = [u_S(i) + u_S'(i)]/2, \tag{4.98}$$

where

$$u_S(i) = \left[\frac{u_{LM}}{u_{LM} + V_{LM}(i)}\right]\left[\frac{u_{GM}}{u_{GM} + V_{GM}(i)}\right] \tag{4.99}$$

and $u_S'(i)$ is defined similarly for $S'$. $V_{LM}(i)$ and $V_{GM}(i)$ are the log spectral differences (in dB) between the spectral magnitude at the $i^{\text{th}}$ critical band and its nearest local maximum (LM), and the global maximum (GM) spectral peaks, respectively. The coefficients, $u_{LM}$ and $u_{GM}$, are used to balance the contributions due to the local and the global spectral characteristics and to prevent singularity in $u_S(i)$ and $u_S'(i)$. All the weighting coefficients are determined experimentally. Figure 4.33 shows a critical band spectrum of a typical vowel sound as well as the corresponding log spectral differences $V_{LM}$ and $V_{GM}$ as functions of the critical band number. Notice that $V_{GM}$ is essentially the inverse of the critical band spectrum. The function $V_{LM}(i)$, nevertheless, has three minima at locations corresponding to the three formants. The slope weighting coefficient, $u_S(i)$, thus has large values near



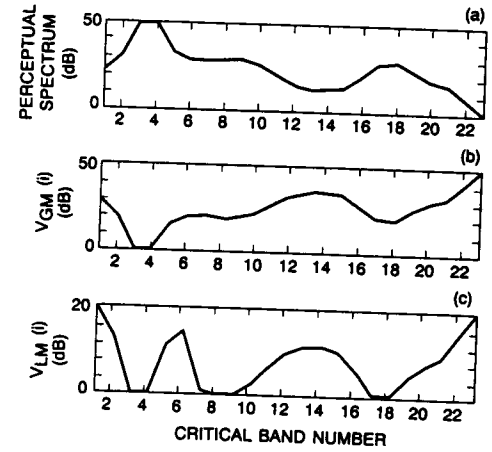



**Figure 4.33** A critical band spectrum (a) of a typical vowel sound and the corresponding log spectral differences $V_{LM}$ (b) and $V_{GM}$ (c) as functions of the critical band number (after Nocerino et al. [16]).

spectral peaks, with a maximum occurring at the global spectral peak. As a consequence, the spectral slope difference is emphasized at spectral peak locations and the significance of formant frequency shift is properly handled in the distortion measure so as to conform to the findings in the phonetic distance experiments.

Although the weighted slope metric attempts to incorporate subjectively determined characteristics, its definition has some heuristic components, such as $u_E$, $u_{LM}$ and $u_{GM}$, that are difficult to determine with high reliability and consistency. Special care is necessary in applying WSM to speech-recognition problems.

### 4.5.8 Summary of Distortion Measures—Computational Considerations

In this section, we have introduced a number of spectral distortion measures that are designed to measure the dissimilarity or distance between two (power) spectra of speech. Referring to the definition of metric and distortion in Section 4.3, we find that many of these dissimilarity measures are not metrics because they do not satisfy the symmetry property. If an objective speech-distortion measure needs to reflect the subjective reality of human perception of sound differences, or even phonetic disparity, the asymmetry seems to be actually desirable. In automatic speech-recognition applications, however, studies to date have not indicated conclusively that an asymmetric distortion measure leads to higher recognition accuracy than does a symmetric distortion measure. The question as to which particular distortion measure is the best choice often is a function of the specific

application. For example, certain distortion measures may be better for an environment where the background acoustic noise is kept at a minimal level, while others may turn out to be more robust when the background noise is a significant source of interference.

To this point we have discussed a wide range of distortion measures suitable for use in speech-recognition systems. We have discussed their derivations and shown how they can be interpreted in terms of the spectra of the signals being compared. By way of review, Table 4.4 shows, for each of nine distortion measures, the symbol used to denote the distortion measure, a mathematical expression of the measure, and the computation needed (approximately) to implement the measure. (We assume that all relevant spectral parameters are precomputed or require negligible computation.)

The computation required to implement the $L_p$ metric is extremely large because we need two FFTs to obtain $S(\omega)$ and $S'(\omega)$, logarithms of all values of $S$ and $S'$, and an integral. Each of these operations requires on the order of from $N$ to $N \log N$ operations (multiply, add), where $N$ is the FFT size (e.g., 256 or 512). The truncated and weighted cepstral distances each require just $L$ operations where $L$ is typically on the order of 12–16 for most spectral model-based systems; hence, a significant reduction in computation is achieved as compared to direct evaluation of the $L_p$ metric for the FFT spectra. The likelihood-based measures (Itakura-Saito, Itakura, Likelihood Ratio and COSH) all require on the order of $p$ operations where $p$ is the LPC order of the all-pole polynomial (typically 8–12). Hence the computation for these measures is essentially the same as for the cepstral measures. The weighted likelihood ratio distortion requires $L$ operations, similar to that of the cepstral measures, and the WSM requires $K$ operations, where $K$ is the number of frequency bands used in the computation (typically 32–64).

The bottom line on computation is that all the measures of Table 4.4 are both physically reasonable and computationally tractable for speech recognition except for the $L_p$ metrics. Hence, in practical situations, there have been proposed and studied speech-recognition systems based on all the measures shown in the table.

## 4.6 INCORPORATION OF SPECTRAL DYNAMIC FEATURES INTO THE DISTORTION MEASURE

The distortion measures discussed in the last section are designed to compare two static spectral representations $S(\omega)$ and $S'(\omega)$. In speech applications, these are usually short-time spectral estimates of the speech signal. A short-time spectrum, as discussed in Chapter 3, is normally obtained by placing a data window on the sample sequence, and the resultant spectral estimate is considered a snapshot of the speech characteristics at a particular time instant $t$. Therefore, the spectral representation is also a function of time $t$, and it is thus appropriate to denote it by $S(\omega, t)$ to emphasize the time variability of the speech. The data window slides across the signal sequence, producing a series of short-time spectral representations, $\{S(\omega, t)\}_{t=0}^{T}$. Before we explain how sequences of spectra can be compared based on the short-time spectral distortions, we first discuss an important extension of the



**TABLE 4.4.** Summary of Spectral Distortion Measures

| Distortion Measure | Notation | Expression | Computation |
|---|---|---|---|
| $L_p$ Metric | $d_p$ | $\left[\int_{-\pi}^{\pi} \left\lvert \log S(\omega) - \log S'(\omega) \right\rvert^p \frac{d\omega}{2\pi}\right]^{1/p}$ | 2 FFTs, logs, integral |
| Truncated Cepstral Distance | $d_c^2(L)$ | $\sum_{n=1}^{L} (c_n - c_n')^2$ | $L*, +$ |
| Weighted (Liftered) Cepstral Distance | $d_{cW}^2$ | $\sum_{n=1}^{L} w(n)(c_n - c_n')^2$ | $L*, +$ |
| Itakura-Saito Distortion | $d_{IS}$ | $\int_{-\pi}^{\pi} \frac{S(\omega)}{S'(\omega)} \frac{d\omega}{2\pi} - \log \frac{\sigma_\infty^2}{\sigma_\infty'^2} - 1$ <br> $\frac{\sigma_p^2}{\sigma^2} \int_{-\pi}^{\pi} \frac{\lvert A\rvert^2}{\lvert A_p\rvert^2} \frac{d\omega}{2\pi} - \log \frac{\sigma_p^2}{\sigma^2} - 1$ | $p*, +$ |
| Itakura Distortion | $d_I$ | $\log \left\{ \int_{-\pi}^{\pi} \frac{\lvert A\rvert^2}{\lvert A_p\rvert^2} \frac{d\omega}{2\pi} \right\}$ <br> $\log \frac{\mathbf{a}' R_p \mathbf{a}}{\sigma_p^2}$ | $p*, +$ |
| Likelihood Ratio Distortion | $d_{LR}$ | $\int_{-\pi}^{\pi} \frac{\lvert A\rvert^2}{\lvert A_p\rvert^2} \frac{d\omega}{2\pi} - 1$ <br> $\frac{\mathbf{a}' \mathbf{R}_p \mathbf{a}}{\sigma_p^2} - 1$ | $p*, +$ |
| COSH Distance | $d_{\text{COSH}}$ | $\int_{-\pi}^{\pi} \cosh \left[\log \frac{S(\omega)}{S'(\omega)}\right] \frac{d\omega}{2\pi} - 1$ <br> $\frac{1}{2}\left[d_{IS}(S, S') + d_{IS}(S', S)\right]$ | $2p*, +$ |
| Weighted Likelihood Ratio Distortion | $d_{\text{WLR}}$ | $\sum_{n=1}^{L} \left[\frac{\tilde{r}(n)}{\sigma^2} - \frac{\tilde{r}'(n)}{\sigma'^2}\right] (c_n - c_n')$ | $L*, +$ |
| Weighted Slope Metric | $d_{\text{WSM}}$ | $u_E \lvert E_S - E_{S'}\rvert + \sum_{i=1}^{K} u(i)[\Delta(i) - \Delta'(i)]^2$ | $K*, +$ |

short-time spectrum for distance or dissimilarity calculation that includes the dynamic as well as the static behavior (in terms of time variation) of the spectral sequence.

Spectral transitions are believed to play an important role in human speech perception. In an experimental study, Furui demonstrated, by using isolated syllables truncated at the initial or final end, that the portion of the utterance where spectral variation was locally maximum contained the most important phonetic information in the syllable [18]. It therefore seems reasonable that the use of such dynamic, variational features of the spectrum in speech-pattern comparison should contribute significantly to overall recognition performance.

Dynamic features of speech are often represented by a time differential log spectrum. A first-order differential (log) spectrum is defined by

$$\frac{\partial \log S(\omega, t)}{\partial t} = \sum_{n=-\infty}^{\infty} \frac{\partial c_n(t)}{\partial t} e^{-jn\omega}, \tag{4.100}$$

where $c_n(t)$ is the cepstrum at time $t$. Note that the time-sampled cepstral sequence, $c_n(t)$, usually cannot be expressed in a form suitable for differentiation. The time derivative, $\partial c_n(t)/\partial t$, is normally obtained by polynomial approximation. This is done by fitting the cepstral trajectory with polynomials over a finite segment of the trajectory.

Consider fitting a segment of the cepstral trajectory, $c(t)$, $t = -M, -M+1, \ldots, M$ by a second-order polynomial $h_1 + h_2 t + h_3 t^2$. (We drop the coefficient index for simplicity.) That is, we choose parameters $h_1$, $h_2$ and $h_3$ such that the fitting error

$$E = \sum_{t=-M}^{M} [c(t) - (h_1 + h_2 t + h_3 t^2)]^2 \tag{4.101}$$

is minimized. Differentiating $E$ with respect to $h_1$, $h_2$ and $h_3$ and setting the result to zero lead to three simultaneous equations:

$$\sum_{t=-M}^{M} [c(t) - h_1 - h_2 t - h_3 t^2] = 0$$

$$\sum_{t=-M}^{M} [c(t)t - h_1 t - h_2 t^2 - h_3 t^3] = 0$$

$$\sum_{t=-M}^{M} [c(t)t^2 - h_1 t^2 - h_2 t^3 - h_3 t^4] = 0.$$

The solution to these simultaneous equations can be easily shown to be

$$h_2 = \frac{\sum_{t=-M}^{M} t\, c(t)}{T_M} \tag{4.102}$$



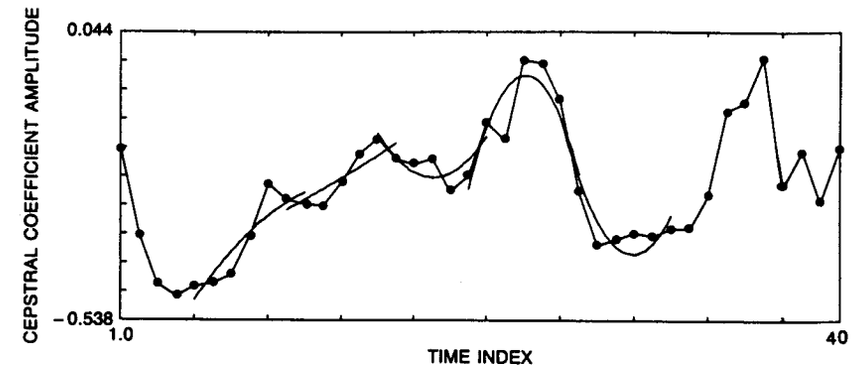



**Figure 4.34** A trajectory of the (2nd) cepstral coefficient with 2nd-order polynomial ($h_1 + h_2 t + h_3 t^2$) fitting on short portions of the trajectory; the width for polynomial fitting is 7 points.

$$h_3 = \frac{T_M \sum_{t=-M}^{M} c(t) - (2M+1) \sum_{t=-M}^{M} t^2 c(t)}{T_M^2 - (2M+1) \sum_{t=-M}^{M} t^4} \tag{4.103}$$

and

$$h_1 = \frac{1}{2M+1} \left[ \sum_{t=-M}^{M} c(t) - h_3 T_M \right], \tag{4.104}$$

where

$$T_M = \sum_{t=-M}^{M} t^2. \tag{4.105}$$

Figure 4.34 shows a typical cepstral trajectory with each nonoverlapping segment $\{c(\tau + t)\}_{t=-M}^{M}$ fitted by a second-order polynomial $h_1 + h_2 t + h_2 t^2$. (In the present formulation, it is understood that the index $t$ is with respect to $\tau$ and the polynomial fitting is done for all $\tau$s.) The curve fitting is performed individually for each of the cepstral coefficient trajectories $c_n(\tau + t)$, $n = 1, 2, \ldots, L$. Note that for a first-order fit, $h_3 = 0$ and

$$h_1 = \frac{\sum_{t=-M}^{M} c(\tau + t)}{2M+1} \tag{4.106}$$

which is simply the average, and $h_2$ remains the same as Eq. (4.102), except for a shift of origin to $\tau$.

The first and second time derivatives of $c_n$ can now be obtained by differentiating the

fitting curve, giving

$$\left.\frac{\partial c_n(\tau+t)}{\partial t}\right|_{t=0} \simeq h_2 = \sum_{t=-M}^{M} t\, c_n(\tau+t) \Big/ T_M \tag{4.107}$$

and

$$\left.\frac{\partial^2 c_n(\tau+t)}{\partial t^2}\right|_{t=0} \simeq 2h_3 \tag{4.108}$$

$$= \frac{2\left\{T_M\left[\sum_{t=-M}^{M} c(\tau+t)\right] - (2M+1)\left[\sum_{t=-M}^{M} t^2 c(\tau+t)\right]\right\}}{T_M^2 - (2M+1)\left[\sum_{t=-M}^{M} t^4\right]}. \tag{4.109}$$

Using polynomial coefficients leads to smoother estimates of the derivatives than the direct difference operation; the coefficient $h_2$ as an estimate of $\left.\frac{\partial c_n(\tau+t)}{\partial t}\right|_{t=0}$ is much smoother than $\Delta c_n(\tau) = c_n(\tau+1) - c_n(\tau)$ which is inherently noisy. We denote the estimates of $\left.\frac{\partial c_n(\tau+t)}{\partial t}\right|_{t=0}$ and $\left.\frac{\partial^2 c_n(\tau+t)}{\partial t^2}\right|_{t=0}$ by $\delta_n^{(1)}(\tau)$ and $\delta_n^{(2)}(\tau)$, respectively.

A differential spectral distance can now be defined, similar to Eq. (4.17), as

$$d^2_{2\delta^{(1)}} = \int_{-\pi}^{\pi} \left| \frac{\partial \log S(\omega,t)}{\partial t} - \frac{\partial \log S'(\omega,t)}{\partial t} \right|^2 \frac{d\omega}{2\pi} \tag{4.110}$$

$$\simeq \sum_{n=-\infty}^{\infty} \left(\delta_n^{(1)} - \delta_n'^{(1)}\right)^2, \tag{4.111}$$

where it is understood that the distance as well as $\delta_n^{(1)}$ and $\delta_n'^{(1)}$ is a function of time, $t$. Similarly, a second differential spectral distance can be defined as

$$d^2_{2\delta^{(2)}} = \int_{-\pi}^{\pi} \left| \frac{\partial^2 \log S(\omega,t)}{\partial t^2} - \frac{\partial^2 \log S'(\omega,t)}{\partial t^2} \right|^2 \frac{d\omega}{2\pi} \tag{4.112}$$

$$\simeq \sum_{n=-\infty}^{\infty} \left(\delta_n^{(2)} - \delta_n'^{(2)}\right)^2. \tag{4.113}$$

The first and second differential spectral distances can be straightforwardly combined with the cepstral distance, resulting in

$$d^2_{2\Delta} = \gamma_1 d_2^2 + \gamma_2 d^2_{2\delta^{(1)}} + \gamma_3 d^2_{2\delta^{(2)}}, \tag{4.114}$$

where $\gamma_1$, $\gamma_2$ and $\gamma_3$ are the weights used to adjust the respective significance of the associated distance components. Usually, $\gamma_1 + \gamma_2 + \gamma_3 = 1$.

For speech applications, it is interesting to examine the correlations among these distance components. If these distances, $d_2^2$, $d^2_{2\delta^{(1)}}$ and $d^2_{2\delta^{(2)}}$, are highly correlated, simply adding more terms will not lead to better discrimination for speech sound patterns. Figure 4.35 shows a scatter diagram where the $x$-axis and $y$-axis represent $d_2^2$ and $d^2_{2\delta^{(1)}}$,



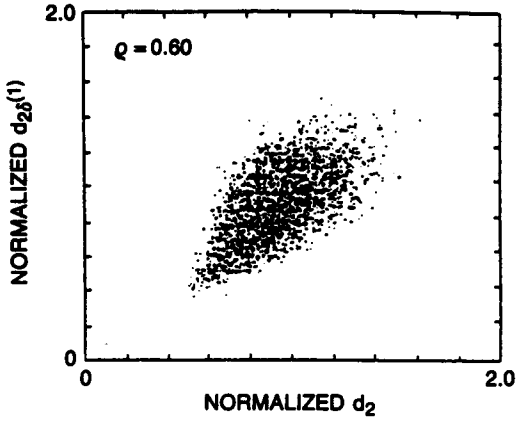



**Figure 4.35** Scatter diagram showing the correlation between the "instantaneous" cepstral distance, $d_2$, and the "differential" or "dynamic" cepstral distance, $d_{2\delta^{(1)}}$; the correlation index is 0.6.

each normalized by their corresponding standard deviations, respectively. The data were collected from a single talker, and the distances were calculated with respect to the closest typical cepstral vector in a finite set of spectral prototypes (or vector quantization code words). The correlation was found to be 0.6, which is quite small relative to correlations between spectral representations observed in speech. Therefore, one expects that the differential spectral distance should provide significant additional discriminability for speech-pattern comparison. Practical evaluations of recognition systems have shown this to be the case [19].

Cepstral weighting or liftering can be applied to the differential cepstrum in much the same manner as it is applied to the cepstrum. One form of cepstral weighting, for example, stems from differentiating $\log S(\omega,t)$ with respect to the frequency $\omega$, as shown in Eqs. (4.27) and (4.28). We rewrite Eq. (4.27) and further take the first time derivative to obtain

$$\frac{\partial^2}{\partial t\, \partial \omega}[\log S(\omega,t)] = \sum_{n=-\infty}^{\infty} -jn \frac{\partial c_n(t)}{\partial t} e^{-jn\omega}$$

$$\cong \sum_{n=-\infty}^{\infty} -jn\delta_n^{(1)}(t) e^{-jn\omega}. \tag{4.115}$$

Therefore, we can define a weighted differential cepstral distance in a manner similar to Eq. (4.110), as

$$d^2_{2W\delta} = \int_{-\pi}^{\pi} \left| \frac{\partial^2 \log S(\omega,t)}{\partial t\, \partial \omega} - \frac{\partial^2 \log S'(\omega,t)}{\partial t\, \partial \omega} \right|^2 \frac{d\omega}{2\pi} \tag{4.116}$$

$$\cong \sum_{n=-\infty}^{\infty} n^2 \left(\delta_n^{(1)} - \delta_n'^{(1)}\right)^2. \tag{4.117}$$

Note that linear operations, such as scaling and differentiation, can be interchanged in order, and thus a weighted differential cepstral distance is the same as a differential cepstral distance defined on liftered cepstra.

The time-frequency derivative expression for the log spectrum of Eq. (4.115) suggests that other operators can be added to produce a combined representation of the spectrum and the differential spectra. As an example, let us consider adding the frequency derivative and the time-frequency derivative together to form

$$\left(\frac{\partial}{\partial\omega} + \frac{\partial^2}{\partial\omega\,\partial t}\right)[\log S(\omega,t)] = \sum_{n=-\infty}^{\infty} -jn[c_n(t) + \delta_n^{(1)}(t)]e^{-jn\omega}. \tag{4.118}$$

Taking the $L_2$ distance, similar to Eq. (4.116), we have

$$\begin{aligned}
d_{2W+\delta}^2 &= \int_{-\pi}^{\pi} \left|\left(\frac{\eta_1\partial}{\partial\omega} + \frac{\eta_2\partial^2}{\partial\omega\,\partial t}\right)[\log S(\omega,t) - \log S'(\omega,t)]\right|^2 \frac{d\omega}{2\pi} \\
&= \sum_{n=-\infty}^{\infty} n^2 \left[\eta_1 c_n(t) - \eta_1 c_n'(t) + \eta_2\delta_n^{(1)}(t) - \eta_2\delta_n'^{(1)}(t)\right]^2 \\
&= \eta_1^2\left\{\sum_{n=-\infty}^{\infty} n^2[c_n(t) - c_n'(t)]^2\right\} + \eta_2^2\left\{\sum_{n=-\infty}^{\infty} n^2[\delta_n^{(1)}(t) - \delta_n'^{(1)}(t)]^2\right\} \\
&\quad + 2\eta_1\eta_2 \sum_{n=-\infty}^{\infty} n^2\left[c_n(t) - c_n'(t)\right]\left[\delta_n^{(1)}(t) - \delta_n'^{(1)}(t)\right] \\
&= \eta_1^2 d_{2W}^2 + \eta_2^2 d_{2W\delta}^2 + 2\eta_1\eta_2 \sum_{n=-\infty}^{\infty} n^2\left[c_n(t) - c_n'(t)\right]\left[\delta_n^{(1)}(t) - \delta_n'^{(1)}(t)\right].
\end{aligned} \tag{4.119}$$

This distance has an extra cross-product term, in addition to the weighted sum of $d_{2W}^2$ and $d_{2W\delta}^2$. There are many other possible ways of augmenting the conventional cepstral distances with differential (dynamic) spectral features. Most of these combinations have not yet been studied extensively.

## 4.7 TIME ALIGNMENT AND NORMALIZATION

In the previous sections, we discussed the fundamentals of comparing speech spectra, defined essentially on a short-time basis (on the order of 10-ms frames). These short-time speech patterns are represented by a set of measurements of acoustic features or simply by a short-time spectral model. An utterance, which is to be recognized, however, is



more complex than a steady sound, and thus a speech pattern almost always involves a sequence of short-time acoustic representations. The pattern-comparison technique for speech recognition, therefore, has to be able to compare sequences of acoustic features. At the end of the last section, we touched upon the notion of dynamic features that extend the short-time, snapshot-like (static) spectral representations to include its (dynamic) relationship with adjacent short-time data frames. However, the resulting spectral representation is still considered a short-time representation because of the time interval involved, which is usually in the range of several tens of milliseconds, and because of the rather simplistic, albeit adequate, treatment of short-time spectral variations.

The problem associated with spectral sequence comparison for speech comes from the fact that different acoustic renditions, or tokens, of the same speech utterance (e.g., word, phrase, sentence) are seldom realized at the same speed (speaking rate) across the entire utterance. Hence when comparing different tokens of the same utterance, speaking rate variation as well as duration variation should not contribute to the (linguistic) dissimilarity score. Thus there is a need to normalize out speaking rate fluctuation in order for the utterance comparison to be meaningful before a recognition decision can be made.

Consider two speech patterns $\mathcal{X}$ and $\mathcal{Y}$, represented by the (spectral) sequences $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{T_x})$ and $(\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{T_y})$, respectively, where $\mathbf{x}_i$ and $\mathbf{y}_i$ are parameter vectors of the short-time acoustic features. (Any set of acoustic features can be used so long as the distance measure for comparing pairs of feature vectors is known and its properties understood.) We use $i_x$ and $i_y$ to denote the time indices of $\mathcal{X}$ and $\mathcal{Y}$, respectively. The durations, $T_x$ and $T_y$, need not be identical. The dissimilarity between $\mathcal{X}$ and $\mathcal{Y}$ is defined by considering some function of the short-time spectral distortions $d(\mathbf{x}_{i_x}, \mathbf{y}_{i_y})$, which will be denoted for simplicity of notation as $d(i_x, i_y)$ where $i_x = 1, 2, \ldots, T_x$ and $i_y = 1, 2, \ldots, T_y$ without ambiguity. Since the sequential order of the sounds is (in most cases) critical in the definition of an "utterance," it is necessary that the indices of the spectral pairs being compared, $(\mathbf{x}_{i_x}, \mathbf{y}_{i_y})$, satisfy certain order constraints. The interaction between these sequential constraints and the natural speaking rate variation constitutes one of the central problems in speech recognition—namely, that of time alignment and normalization.

Perhaps the simplest solution to the problem of time alignment and normalization is a linear time normalization technique. In linear time normalization, the dissimilarity between $\mathcal{X}$ and $\mathcal{Y}$ is defined simply as

$$d(\mathcal{X}, \mathcal{Y}) = \sum_{i_x=1}^{T_x} d(i_x, i_y), \tag{4.120}$$

where $i_x$, $i_y$ satisfy

$$i_y = \frac{T_y}{T_x} i_x. \tag{4.121}$$

(Since the indices $i_x$ and $i_y$ are integer, some round-off rule is implied in Eq. (4.121).) The summation in Eq. (4.120) can be taken from $i_y = 1$ to $i_y = T_y$ as well, depending on the desired direction of time normalization. (In the above, we have used $d$ to denote the

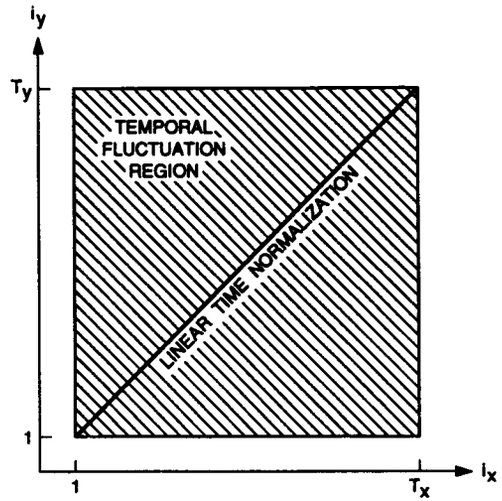



**Figure 4.36** Linear time alignment for two sequences of different durations.

dissimilarity or distortion between a pair of short-time spectra as well as a pair of spectral sequences. The interpretation of $d$ should be clear from the arguments of the dissimilarity function without ambiguity.)

Linear time normalization alignment implicitly assumes that the speaking rate variation is proportional to the duration of the utterance and is independent of the sound being spoken. Therefore, evaluation of the distortion measure takes place along the diagonal straight line of the rectangle in the $(i_x, i_y)$ plane as shown in Figure 4.36. Each point in the $(i_x, i_y)$ plane along the diagonal represents a distortion $d(i_x, i_y)$—i.e., the distance between the appropriate spectral feature vectors of $\mathcal{X}$ and $\mathcal{Y}$ at frames $i_x$ and $i_y$. Obviously, this rigid constraint of invariant speaking rate fluctuation does not adequately model the true situation for real speech utterances and thereby points to the need for a more realistic way of time aligning and normalizing an arbitrary pair of speech patterns.

A more general time alignment and normalization scheme involves the use of two warping functions, $\phi_x$ and $\phi_y$, which relate the indices of the two speech patterns, $i_x$ and $i_y$, respectively, to a common, "normal" time axis $k$, i.e.,

$$i_x = \phi_x(k), \qquad k = 1, 2, \ldots, T \tag{4.122}$$

and

$$i_y = \phi_y(k), \qquad k = 1, 2, \ldots, T. \tag{4.123}$$

A global pattern dissimilarity measure $d_\phi(\mathcal{X}, \mathcal{Y})$ can be defined based on the warping



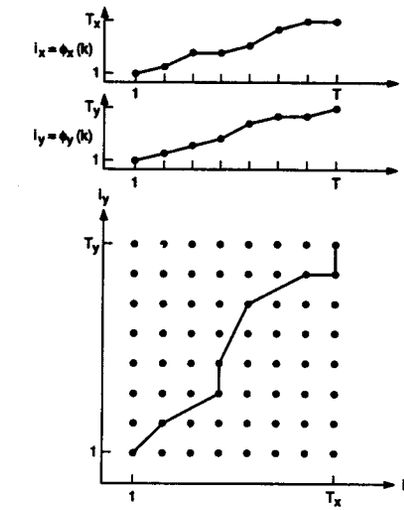



**Figure 4.37** An example of time normalization of two sequential patterns to a common time index; the time warping functions $\phi_x$ and $\phi_y$ map the individual time index $i_x$ and $i_y$, respectively, to the common time index $k$.

function pair $\phi = (\phi_x, \phi_y)$ as the accumulated distortion over the entire utterance, namely

$$d_\phi(\mathcal{X}, \mathcal{Y}) = \sum_{k=1}^{T} d(\phi_x(k), \phi_y(k)) m(k) / M_\phi \tag{4.124}$$

where $d(\phi_x(k), \phi_y(k))$ is again a short-time spectral distortion defined for $\mathbf{x}_{\phi_x(k)}$ and $\mathbf{y}_{\phi_y(k)}$, $m(k)$ is a nonnegative (path) weighting coefficient and $M_\phi$ is a (path) normalizing factor. Figure 4.37 shows an example of the above general time normalization scheme; the solid line in the lower grid represents the path along which $d_\phi(\mathcal{X}, \mathcal{Y})$ is evaluated. The grid points on the path are labeled incrementally from $k = 1$ to $k = T$ where $T$ is the "normal" duration of the two patterns on the normal time scale. The indices $i_x$ and $i_y$, as functions of the normal time scale $k$, are shown at the top of the figure. The requirement to maintain temporal order in the spectral representations of $\mathcal{X}$ and $\mathcal{Y}$ means that the warping functions $\phi_x$ and $\phi_y$ must be monotonically nondecreasing as depicted in the curves at the top of Figure 4.37.

To complete the definition of a dissimilarity measure for the $(\mathcal{X}, \mathcal{Y})$ pair of patterns, we need to specify the path $\phi = (\phi_x, \phi_y)$ as indicated in Eq. (4.124). There is obviously an extremely large number of possible warping function pairs. The key issue then is which path should be chosen such that the overall path dissimilarity can be measured with consistency.

One natural and popular choice [20] is to define the dissimilarity $d(\mathcal{X},\mathcal{Y})$ as the minimum of $d_\phi(\mathcal{X},\mathcal{Y})$, over all possible paths, such that

$$d(\mathcal{X},\mathcal{Y}) \triangleq \min_{\phi} d_\phi(\mathcal{X},\mathcal{Y}), \tag{4.125}$$

where $\phi$ must satisfy a set of requirements (to be discussed below). Intuitively, the definition of Eq. (4.125) is quite appealing when $\mathcal{X}$ and $\mathcal{Y}$ represent utterances of the same word, because choosing the best path so as to minimize the accumulated distortion along the alignment path means the dissimilarity is measured based on the best possible alignment in order to compensate for the nonlinear speaking rate differences between the two patterns of the same word. When $\mathcal{X}$ and $\mathcal{Y}$ represent utterances of different words, the implications of the best alignment in Eq. (4.125) are not immediately obvious. We shall come back to this issue later when we discuss classifier design methodology.

The fundamental point is that finding the "best" alignment between a pair of patterns is functionally equivalent to finding the "best" path through a grid mapping the acoustic features of one pattern to the acoustic features of the other pattern. Finding this best path requires solving a minimization problem to evaluate the dissimilarity between two speech patterns. The specific form of the accumulated distortion of Eq. (4.124) suggests that dynamic programming techniques are readily applicable in the solution process. Before we formally introduce and consider the set of sequential constraints that have to be imposed on the warping functions in order for them to be meaningful in speech recognition, we first discuss a general solution procedure for sequential minimization based on the dynamic programming principle.

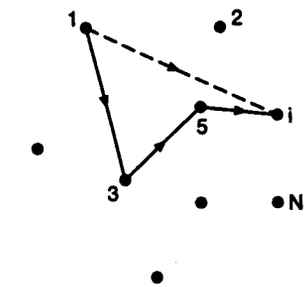
### 4.7.1 Dynamic Programming—Basic Considerations

Dynamic programming is an extensively studied and widely used tool in operations research for solving sequential decision problems. To illustrate its applicability in speech recognition, and particularly for the problem of time alignment and normalization, we discuss two typical problems in which dynamic programming has been extensively used.

The first problem is an optimal path problem that can be stated as follows. Consider a set of points labeled from 1 to $N$. Associated with every pair of points $(i,j)$ is a nonnegative cost $\zeta(i,j)$ that represents the cost of moving directly from the $i^{\text{th}}$ point to the $j^{\text{th}}$ point in one step. The problem is to find the minimum cost, as well as the corresponding sequence of moves, of moving from, say, point 1 in the set to another point, say $i$, using as many steps as needed. This problem is illustrated in Figure 4.38. Since the sequence of moves has an unspecified number of transitions (steps), from one point to another, we call this an asynchronous sequential decision problem.

Using traditional terminology, we call the decision rule for determining the next point to be visited after point $i$ a "policy." Since the policy determines the sequence of points traversed from the (fixed) originating point 1 to the destination point $i$, the cost is therefore completely defined by the policy and the destination point $i$. The question is what policy






$$\Psi(i) = \zeta(1,3) + \zeta(3,5) + \zeta(5,i) < \zeta(1,i)$$

**Figure 4.38** The optimal path problem—finding the minimum cost path from point 1 to point $i$ in as many moves as needed.

leads to the minimum cost, moving from point 1 to point $i$. We denote this minimum cost by $\varphi(1,i)$.

The principle of optimality, which is the basis of a class of computational algorithms for the above optimization problem is, according to Bellman [21],

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

To put Bellman's principle of optimality into a functional equation suitable for computational algorithms, consider first moving from the initial point 1 to an intermediate point $j$ in one or more steps. The minimum cost, as defined, is $\varphi(1,j)$. Since moving from point $j$ to point $i$ in one step incurs a cost $\zeta(j,i)$, the optimal policy, which determines which intermediate point $(j)$ to pass through, (should one exist) satisfies the following equation

$$\varphi(1,i) = \min_{j}[\varphi(1,j) + \zeta(j,i)]. \tag{4.126}$$

Generalizing Eq. (4.126) to the case in which we are interested in obtaining the optimal sequence of moves and the associated minimum cost from any point $i$ to any other point $j$, we have

$$\varphi(i,j) = \min_{\ell}[\varphi(i,\ell) + \varphi(\ell,j)], \tag{4.127}$$

where $\varphi(i,j)$ is the minimum cost from $i$ to $j$ in as many steps as necessary. Equation (4.127) implies that any partial, consecutive sequence of moves of the optimal sequence from $i$ to $j$ must also be optimal, and that any intermediate point must be the optimal point linking

the optimal partial sequences before and after that point.

To actually determine the minimum cost path between points $i$ and $j$, in any number of steps, the following simple dynamic program would be used:

$$\varphi_1(i,\ell) = \zeta(i,\ell) \qquad \ell = 1,2,\ldots,N$$

$$\varphi_2(i,\ell) = \min_k(\varphi_1(i,k) + \zeta(k,\ell)), \qquad k = 1,2,\ldots,N \quad \ell = 1,2,\ldots,N$$

$$\varphi_3(i,\ell) = \min_k(\varphi_2(i,k) + \zeta(k,\ell)), \qquad k = 1,2,\ldots,N \quad \ell = 1,2,\ldots,N$$

$$\vdots$$

$$\varphi_S(i,\ell) = \min_k(\varphi_{S-1}(i,k) + \zeta(k,\ell)), \qquad k = 1,2,\ldots,N \quad \ell = 1,2,\ldots,N$$

$$\varphi(i,j) = \min_{1\le s\le S} \varphi_s(i,j),$$

where $\varphi_s(i,\ell)$ is the $s$-step best path from point $i$ to point $\ell$, and $S$ is the maximum number of steps allowed in the path.

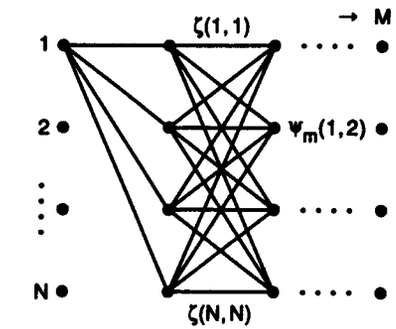
A second dynamic programming problem is the synchronous sequential decision problem, which differs from the asynchronous one in the regularity of the decision process. In terms of the optimal path problem, the objective now is to find the optimal sequence of a fixed number, say $M$, of moves, starting from point $i$ and ending at point $j$, and the associated minimum cost, $\varphi_M(i,j)$. (Note that if $M$ is large enough, the sequence could become periodic.) The regularity of the problem can best be explained with the trellis structure shown in Figure 4.39. The $N$ points are plotted vertically and the $M$ transitions progress horizontally to the right in the figure. Since there are $N$ possible moves for each point, at every moment, the total number of one-step moves is thus $N^2$. Furthermore, the total number of sequences of moves, which will be called "paths" in the rest of the chapter, connecting point $i$ at the beginning of the move and point $j$ at the end of the $M^{\text{th}}$ move, is $N^{(M-1)}$.

The principle of optimality is equally applicable in this case. After the $m^{\text{th}}$ move, $m < M$, the path can end up at any point $\ell$, $\ell = 1,2,\ldots,N$, with the associated minimum cost $\varphi_m(i,\ell)$. Suppose the $(m+1)^{\text{th}}$ move is to go to point $n$. Then, similar to Eq. (4.126), $\varphi_{m+1}(i,n)$ satisfies

$$\varphi_{m+1}(i,n) = \min_\ell[\varphi_m(i,\ell) + \zeta(\ell,n)]. \tag{4.128}$$

Equation (4.128) describes a recursion that allows the optimal path search to be conducted incrementally, in a progressive manner. Although there are $N$ possible moves that end at point $\ell$, the optimality principle indicates that only the best move is necessary to be considered according to Eq. (4.128). The algorithm can be summarized as follows.






$$\Psi_m(1,2) = \min_\ell [\Psi_{m-1}(1,\ell) + \zeta(\ell,2)]$$

**Figure 4.39** A trellis structure that illustrates the problem of finding the optimal path from point $i$ to point $j$ in $M$ steps.

1. Initialization

$$\varphi_1(i,n) = \zeta(i,n) \tag{4.129}$$

$$\xi_1(n) = i \tag{4.130}$$

for $n = 1,2,\ldots,N$.

2. Recursion

$$\varphi_{m+1}(i,n) = \min_{1\le\ell\le N}[\varphi_m(i,\ell) + \zeta(\ell,n)] \tag{4.131}$$

$$\xi_{m+1}(n) = \arg\min_{1\le\ell\le N}[\varphi_m(i,\ell) + \zeta(\ell,n)] \tag{4.132}$$

for $n = 1,2,\ldots,N$ and $m = 1,2,\ldots,M-2$

3. Termination

$$\varphi_M(i,j) = \min_{1\le\ell\le N}[\varphi_{M-1}(i,\ell) + \zeta(\ell,j)] \tag{4.133}$$

$$\xi_M(j) = \arg\min_{1\le\ell\le N}[\varphi_{M-1}(i,\ell) + \zeta(\ell,j)] \tag{4.134}$$

4. Path Backtracking

$$\text{optimal path} = (i, i_1, i_2, \ldots, i_{M-1}, j),$$

where

$$i_m = \xi_{m+1}(i_{m+1}), \qquad m = M-1, M-2, \ldots, 1 \tag{4.135}$$

with $i_M = j$.

The algorithm, as a result of the optimality principle, keeps track of only $N$ paths, ending at each of the $N$ points, at the end of every potential move. When the destination is reached, the optimal path and the associated minimum cost are the natural results of the algorithm as indicated by Eqs. (4.133) and (4.134) without having to reexamine any of the previously incurred partial costs. The low algorithmic complexity (i.e., computation on the order of $NM$ calculations) is particularly worth noting.

For time normalization and alignment, where the pattern dissimilarity requires finding the best match according to Eq. (4.125), the above dynamic programming algorithm is an almost-indispensable tool. Applications of asynchronous and synchronous sequential decision processes can be found in different aspects of the speech-recognition problem and will be discussed extensively in this and in subsequent chapters. We now return to the problem of finding the best constrained time-alignment path for comparing a pair of speech patterns.

### 4.7.2 Time-Normalization Constraints

Forthe alignment process to be meaningful in terms of time normalization for different renditions of an utterance, some constraints on the warping functions are necessary. Unconstrained minimization in Eq. (4.125) can conceivably result in a near-perfect match between two linguistically different utterances, thus making the comparison meaningless for recognition purposes. For example, consider the words "we" and "you," utterances that have almost time-reversed spectral sequences with respect to each other. If there is no constraint imposed on $\phi_x$ and $\phi_y$ so that time reversal is allowed, the resultant dissimilarity measure of Eq. (4.125) could be extremely small for utterances of these two words, making speech-pattern comparison virtually meaningless.

Typical warping constraints that are considered necessary and reasonable for time alignment between utterances include the following:

- endpoint constraints
- monotonicity conditions
- local continuity constraints
- global path constraints
- slope weighting.

In the remainder of this section, we discuss the nature of each of these constraints and their effects on the time-alignment procedure.

#### 4.7.2.1 Endpoint Constraints

When the speech patterns being compared are isolated tokens of the utterances to be recognized, they usually have well-defined endpoints that mark the beginning and the ending frames of the pattern. The endpoint information is usually derived as a result of the speech-detection operation as discussed in Section 4.2. In this sense, the endpoints of



a speech pattern are considered given a priori, and the temporal variations occur within the range defined by the endpoints. For time normalization, therefore, the endpoints are the fixed temporal limits of the utterances, leading to a set of constraints for the warping functions of the form

$$\text{beginning point} \qquad \phi_x(1) = 1, \qquad \phi_y(1) = 1 \tag{4.136a}$$

$$\text{ending point} \qquad \phi_x(T) = T_x, \qquad \phi_y(T) = T_y. \tag{4.136b}$$

(We assume the first frame of the utterance is labeled as frame 1.) In cases where endpoints cannot be reliably determined (e.g., utterances in noisy environments), the above endpoint constraints need to be modified to take the uncertainty into consideration. We will discuss this situation later in this chapter.

#### 4.7.2.2 Monotonicity Conditions

As discussed previously, the temporal order of the spectral sequence in a speech pattern is of crucial importance to linguistic meaning. To maintain the temporal order while performing time normalization, it is therefore reasonable to impose a monotonicity constraint of the form

$$\phi_x(k+1) \geq \phi_x(k) \tag{4.137a}$$

$$\phi_y(k+1) \geq \phi_y(k). \tag{4.137b}$$

The monotonicity constraint, as shown in Figure 4.37, implies that any path along which $d_\phi(\mathcal{X}, \mathcal{Y})$ is evaluated will not have a negative slope. The constraint eliminates the possibility of (time) reverse warping, along the time axis, even within a short time interval.

#### 4.7.2.3 Local Continuity Constraints

In speech utterances, the presence of a particular sound (phoneme) sometimes is the only discerning factor that facilitates correct recognition. Time normalization by way of finding the best temporal match, as defined in Eq. (4.125), therefore should not result in the omission of any important information-bearing sound segment. To ensure proper time alignment while keeping any potential loss of information to a minimum, we generally incorporate a set of local continuity constraints on the warping function. The local continuity constraints can take many forms. One example, proposed by Sakoe and Chiba [20], is

$$\phi_x(k+1) - \phi_x(k) \leq 1 \tag{4.138a}$$

$$\phi_y(k+1) - \phi_y(k) \leq 1. \tag{4.138b}$$

Such constraint specifications are often quite complicated and it is therefore convenient to express them in terms of incremental path changes.

We therefore define a path $\mathcal{P}$ as a sequence (concatenation) of moves, each specified by a pair of coordinate increments,

$$\mathcal{P} \rightarrow (p_1, q_1)(p_2, q_2) \ldots (p_T, q_T) \tag{4.139}$$

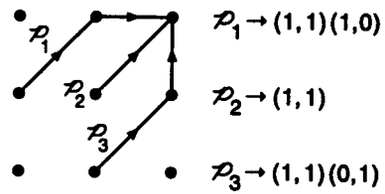



**Figure 4.40** An example of local continuity constraints expressed in terms of coordinate increments (after Myers et al. [23]).

where the subscript is in terms of the normal time $k$. Figure 4.40 illustrates three paths, $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}_3$, which can be specified by $\mathcal{P}_1 \rightarrow (1,1)(1,0)$, $\mathcal{P}_2 \rightarrow (1,1)$, and $\mathcal{P}_3 \rightarrow (1,1)(0,1)$. The path in the example of Figure 4.37, similarly, has the following specification:

$$\mathcal{P} \rightarrow (1,1)(2,1)(0,1)(1,2)(2,1)(1,0)(0,1).$$

For a path that begins at $(1,1)$, which point we designate $k = 1$ according to Eq. (4.136a), we normally set $p_1 = q_1 = 1$ (as if the path originates from $(0,0)$) and have

$$\phi_x(k) = \sum_{i=1}^{k} p_i \tag{4.140}$$

$$\phi_y(k) = \sum_{i=1}^{k} q_i. \tag{4.141}$$

Furthermore, if the path ends at $(T_x, T_y)$, then

$$\sum_{k=1}^{T} p_k = T_x \tag{4.142a}$$

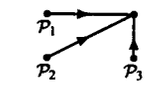
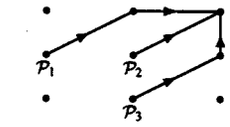
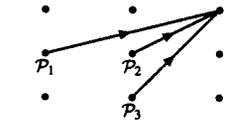
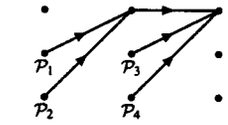
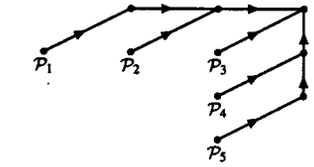
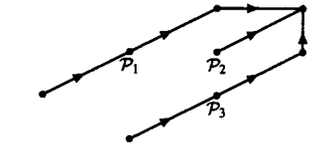
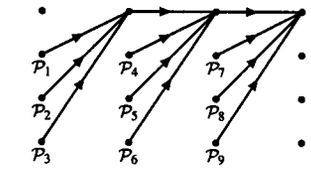
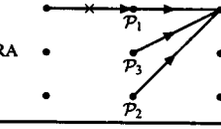
$$\sum_{k=1}^{T} q_k = T_y. \tag{4.142b}$$

Any partial sequence can be similarly characterized.

Using the above notation, any simple type of local continuity constraints can be easily specified. Table 4.5 shows a summary of the types of local constraints that have been considered. These local constraints are expressed in terms of the allowable paths to reach a given point in the $(i_x, i_y)$ plane. Type I constraints are essentially identical to what is specified by Eq. (4.138). (Obviously, $\phi_x(k+1) = \phi_x(k)$ and $\phi_y(k+1) = \phi_y(k)$ cannot occur simultaneously.) Type II and III constraints have the same originating points; however, in Type II constraints, $\mathcal{P}_1$ and $\mathcal{P}_3$ both take two moves and thus two normal time increments while $\mathcal{P}_1$ and $\mathcal{P}_3$ in Type III are single-move paths. We shall call these local paths "incremental paths" to signify the local constraints. Other types of constraints can be interpreted accordingly because the allowable paths in the table completely specify



**TABLE 4.5.** Summary of sets of local constraints and the resulting path specifications

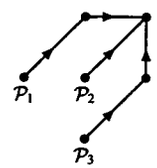
| Type | | Allowable Path Specification |
|---|---|---|
| I |  | $\mathcal{P}_1 \rightarrow (1,0)$<br>$\mathcal{P}_2 \rightarrow (1,1)$<br>$\mathcal{P}_3 \rightarrow (0,1)$ |
| II |  | $\mathcal{P}_1 \rightarrow (1,1)(1,0)$<br>$\mathcal{P}_2 \rightarrow (1,1)$<br>$\mathcal{P}_3 \rightarrow (1,1)(0,1)$ |
| III |  | $\mathcal{P}_1 \rightarrow (2,1)$<br>$\mathcal{P}_2 \rightarrow (1,1)$<br>$\mathcal{P}_3 \rightarrow (1,2)$ |
| IV |  | $\mathcal{P}_1 \rightarrow (1,1)(1,0)$<br>$\mathcal{P}_2 \rightarrow (1,2)(1,0)$<br>$\mathcal{P}_3 \rightarrow (1,1)$<br>$\mathcal{P}_4 \rightarrow (1,2)$ |
| V |  | $\mathcal{P}_1 \rightarrow (1,1)(1,0)(1,0)$<br>$\mathcal{P}_2 \rightarrow (1,1)(1,0)$<br>$\mathcal{P}_3 \rightarrow (1,1)$<br>$\mathcal{P}_4 \rightarrow (1,1)(0,1)$<br>$\mathcal{P}_5 \rightarrow (1,1)(0,1)(0,1)$ |
| VI |  | $\mathcal{P}_1 \rightarrow (1,1)(1,1)(1,0)$<br>$\mathcal{P}_2 \rightarrow (1,1)$<br>$\mathcal{P}_3 \rightarrow (1,1)(1,1)(0,1)$ |
| VII |  | $\mathcal{P}_1 \rightarrow (1,1)(1,0)(1,0)$<br>$\mathcal{P}_2 \rightarrow (1,2)(1,0)(1,0)$<br>$\mathcal{P}_3 \rightarrow (1,3)(1,0)(1,0)$<br>$\mathcal{P}_4 \rightarrow (1,1)(1,0)$<br>$\mathcal{P}_5 \rightarrow (1,2)(1,0)$<br>$\mathcal{P}_6 \rightarrow (1,3)(1,0)$<br>$\mathcal{P}_7 \rightarrow (1,1)$<br>$\mathcal{P}_8 \rightarrow (1,2)$<br>$\mathcal{P}_9 \rightarrow (1,3)$ |
| ITAKURA |  | $\mathcal{P}_1 \rightarrow (1,0)$, consecutive $(1,0)(1,0)$ disallowed<br>$\mathcal{P}_2 \rightarrow (1,1)$<br>$\mathcal{P}_3 \rightarrow (1,2)$ |

the constraints. An exception is the type of constraint proposed by Itakura (bottom of Table 4.5), which has an extra (nonregular) rule that prohibits consecutive $(1,0)(1,0)$ moves [22, 23]. There is a subtle difference between Type IV constraints and the Itakura constraints. We shall discuss this in the following section.

These local continuity constraints are clearly based on heuristics. The only commonality to all these sets of path specification is the linear path of a single move of the form $(1,1)$. Clearly such a path is required to preserve linear time alignments. The speaking rate as well as the temporal variation in speech utterances are difficult to model, and therefore the significance of these local constraints in speech-pattern comparison cannot be assessed analytically. Only experimental results—that is, recognition performance in real tasks—can be used to determine their utility in various applications.

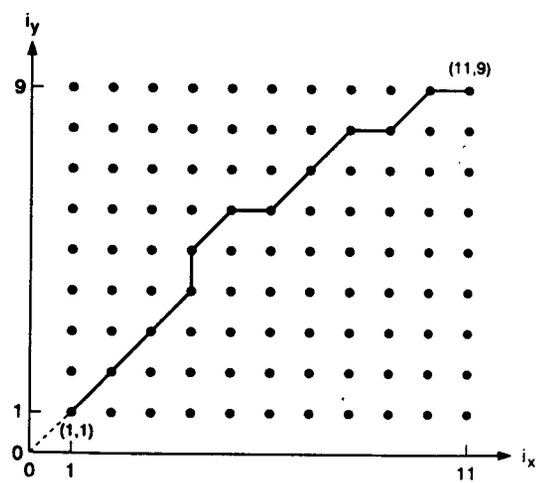
**Exercise 4.5**

Consider Type II local continuity constraints (see Figure 4.40 or Table 4.5).




$\mathcal{P}_1 \rightarrow (1,1)(1,0)$
$\mathcal{P}_2 \rightarrow (1,1)$
$\mathcal{P}_3 \rightarrow (1,1)(0,1)$

Find the sequence of path moves that match the sample path shown below:



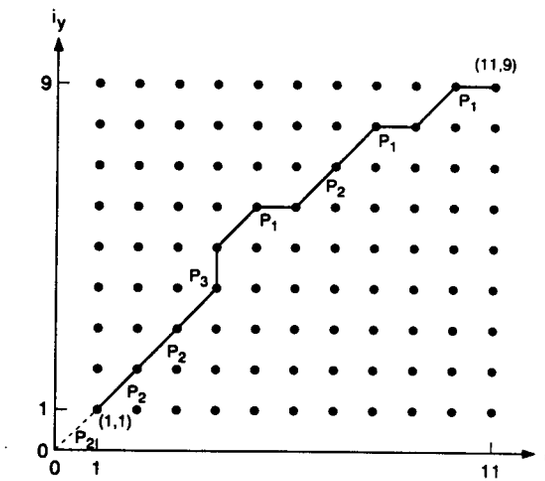



**Solution 4.5**

Beginning at the grid point $(N, M)$ and working backward to the grid point $(1,1)$, the backward path moves are $\mathcal{P}_1\ \mathcal{P}_1\ \mathcal{P}_2\ \mathcal{P}_1\ \mathcal{P}_3\ \mathcal{P}_2\ \mathcal{P}_2$ and $\mathcal{P}_2$ (to go to the point $(0,0)$); thus the complete path can be described by

$$\mathcal{P} \rightarrow \mathcal{P}_2\ \mathcal{P}_2\ \mathcal{P}_2\ \mathcal{P}_3\ \mathcal{P}_1\ \mathcal{P}_2\ \mathcal{P}_1\ \mathcal{P}_1.$$

The resulting labeled path is as shown below.




### 4.7.2.4 Global Path Constraints

Because of the local continuity constraints, certain portions of the $(i_x, i_y)$ plane are excluded from the region the optimal warping path can traverse. The allowable regions can be defined using the following two parameters for each type of local constraints

$$Q_{\max} = \max_{\ell}\left[\sum_{i=1}^{T_\ell} p_i^{(\ell)} \Big/ \sum_{i=1}^{T_\ell} q_i^{(\ell)}\right] \tag{4.143}$$

$$Q_{\min} = \min_{\ell}\left[\sum_{i=1}^{T_\ell} p_i^{(\ell)} \Big/ \sum_{i=1}^{T_\ell} q_i^{(\ell)}\right], \tag{4.144}$$

where $\ell$ signifies the index of the allowable path, $\mathcal{P}_\ell$, in the constraint set and $T_\ell$ is the total number of moves in $\mathcal{P}_\ell$. For example, in Type II constraints, $\ell = 1,2,3$, and $T_\ell = 2,1,2$, respectively, for $\mathcal{P}_1, \mathcal{P}_2$ and $\mathcal{P}_3$. These two parameters, $Q_{\max}$ and $Q_{\min}$, specify the maximum and minimum possible expansion in time warping. Normally, $Q_{\max} = 1/Q_{\min}$. Table 4.6 shows the values of $Q_{\max}$ and $Q_{\min}$ for each constraint type listed in Table 4.5.

Using the values of maximum and minimum possible path expansion, we can define

**TABLE 4.6.** Values of $Q_{max}$ and $Q_{min}$ for different types of paths

| TYPE | $Q_{max}$ | $Q_{min}$ |
|---|---|---|
| I | $\infty$ | 0 |
| II | 2 | $1/2$ |
| III | 2 | $1/2$ |
| IV | 2 | $1/2$ |
| V | 3 | $1/3$ |
| VI | $3/2$ | $2/3$ |
| VII | 3 | $1/3$ |
| ITAKURA | 2 | $1/2$ |

global path constraints as follows:

$$1 + \frac{[\phi_x(k) - 1]}{Q_{max}} \le \phi_y(k) \le 1 + Q_{max}[\phi_x(k) - 1] \tag{4.145}$$

$$T_y + Q_{max}[\phi_x(k) - T_x] \le \phi_y(k) \le T_y + \frac{[\phi_x(k) - T_x]}{Q_{max}}. \tag{4.146}$$

Eq. (4.145) specifies the range of the points in the $(i_x, i_y)$ plane that can be reached from the beginning point $(1, 1)$ via an allowable path according to the local constraints. Similarly, Eq. (4.146) specifies the range of points that have a legal path to the ending point $(T_x, T_y)$.

Figure 4.41 illustrates the effects of the global path constraints of Eqs. (4.145) and (4.146) in the $(i_x, i_y)$ plane for $Q_{max} = 1/Q_{min} = 2$. The range of valid paths form a parallelogram defined by the lines of slope 2 and $1/2$. As implied in Eq. (4.146) and illustrated in Figure 4.41, the durations, $T_x$ and $T_y$, are important factors in defining the allowable regions. Note that if $T_x - 1 = Q_{max}(T_y - 1)$, the second inequality in Eq. (4.146) becomes

$$\phi_y(k) \le 1 + \frac{[\phi_x(k) - 1]}{Q_{max}} \tag{4.147}$$

which, when combined with the first inequality of Eq. (4.145), leads to the global path constraint

$$\phi_y(k) = 1 + \frac{[\phi_x(k) - 1]}{Q_{max}}. \tag{4.148}$$

This global path constraint is the straight line connecting points $(1, 1)$ and $(T_x, T_y)$, and is the only allowable path as shown in Figure 4.42; the difference in durations of the two patterns being compared is so large that, even with a maximum expansion of 2, only a single path is possible. A similar situation occurs when $T_y - 1 = Q_{max}(T_x - 1)$. In these cases, therefore, only linear time normalization is possible. Furthermore, no time warping can be accomplished if $T_x - 1 > Q_{max}(T_y - 1)$ or $T_y - 1 > Q_{max}(T_x - 1)$. The interaction among the pattern durations and the chosen local continuity constraints is thus of crucial significance.

An additional global path constraint proposed by Sakoe and Chiba [20] is

$$|\phi_x(k) - \phi_y(k)| \le T_0, \tag{4.149}$$



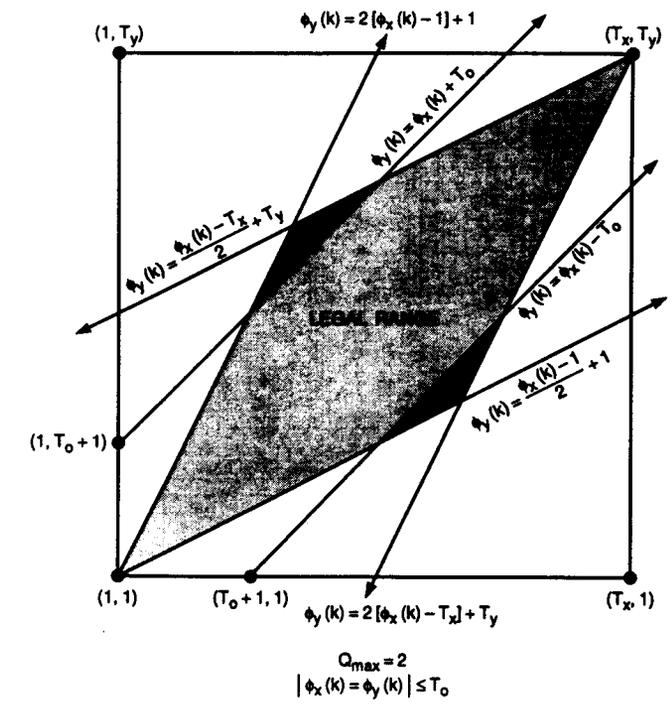
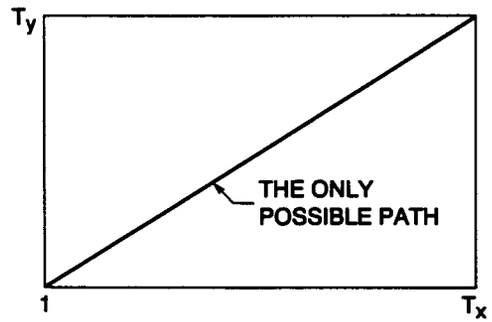



**Figure 4.41** The effects of global path constraints and range limiting on the allowable regions for time warping functions.

where $T_0$ is the maximum allowable absolute time deviation between the two patterns at any moment (frame). This additional global path constraint precludes any path that involves excessive time stretch or compression and effectively reduces the area of the parallelogram by cutting off the corners as shown in Figure 4.41. Constraints of the form of Eq. (4.149) are called range-limiting constraints because they limit absolute differences in the warped time scales.

#### 4.7.2.5 Slope Weighting

Slope weighting along the path adds yet another dimension of control in the search for the optimal warping path for, linguistically as well as acoustically meaningful, time normalization to account for the inherent temporal variability in speech utterances. As defined in Eq. (4.124), the weighting function $m(k)$ controls the contribution of each short-time distortion $d(\phi_x(k), \phi_y(k))$. On a "global" scale (i.e., for the entire utterance duration), the weighting function can be designed to implement an optimal discriminant analysis for




$$T_x - 1 = Q_{max}(T_y - 1)$$
or
$$T_y - 1 = Q_{max}(T_x - 1)$$

**Figure 4.42** Illustration of the extreme cases, $T_x - 1 = Q_{max}(T_y - 1)$ or $T_y - 1 = Q_{max}(T_x - 1)$, where only linear time warping (single straight path) is allowed.

improved recognition accuracy. This type of global weighting design will be discussed in more detail in Chapter 5 when overall speech-recognition systems and generalized weighting techniques are presented. On a "local" scale, it is also possible to associate the weighting function with the prescribed path constraints so that a preference for locally allowable paths can be applied. These locally defined weighting functions are thus called slope weighting functions, because they are usually related to the slope of the local path constraints. Although local slope weighting also appears in the same form as global weighting as in Eq. (4.124), the difference between their purposes—a *prescribed* preference for local paths versus a discriminant function design—deserves careful attention.

As with local continuity constraints, many heuristic slope weighting functions are possible. The following is a set of four types of slope weighting that were proposed by Sakoe and Chiba [20]:

$$\text{Type (a)} \quad m(k) = \min[\phi_x(k) - \phi_x(k-1), \phi_y(k) - \phi_y(k-1)] \tag{4.150}$$

$$\text{Type (b)} \quad m(k) = \max[\phi_x(k) - \phi_x(k-1), \phi_y(k) - \phi_y(k-1)] \tag{4.151}$$

$$\text{Type (c)} \quad m(k) = \phi_x(k) - \phi_x(k-1) \tag{4.152}$$

$$\text{Type (d)} \quad m(k) = \phi_x(k) - \phi_x(k-1) + \phi_y(k) - \phi_y(k-1). \tag{4.153}$$

In the above, it is assumed that $\phi_x(0) = \phi_y(0) = 0$ for initialization purposes.

To illustrate the effects of slope weighting, Figure 4.43 shows the effect of applying the four types of weighting functions to Type III local continuity constraints. The number associated with each path (i.e., along the arcs) is the weighting value. Since a higher distortion represents a less favorable or less likely match, larger weighting values are used



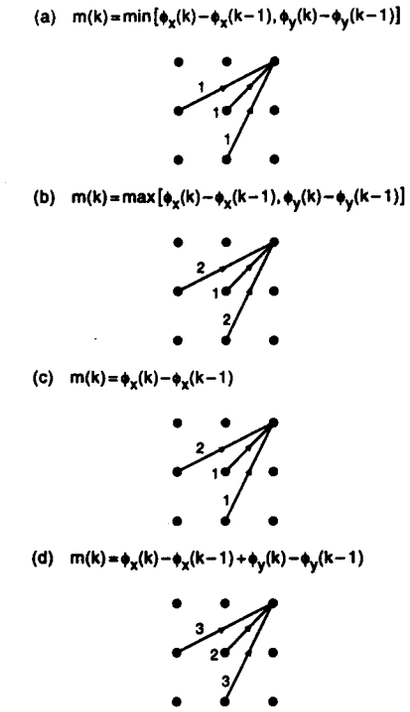



**Figure 4.43** Type III local continuity constraints with four types of slope weighting (after Myers et al. [23]).

for less preferable paths. (The actual warping path, of course, will be determined according to Eq. (4.125).) As shown in Figure 4.43, for example, the Type (b) weighting attempts to maintain a strong bias toward diagonal path movements.

The above slope weighting functions, when applied to the different types of local constraints in Table 4.5, can sometimes lead to 0 weight situations along parts of a local path. As an example, Type (a) slope weighting in combination with Type II local constraints results in 0-weighted moves along part of $\mathcal{P}_1$ and $\mathcal{P}_3$. This situation is illustrated in Figure 4.44. The horizontal move in $\mathcal{P}_1$ and the vertical move in $\mathcal{P}_3$ each have 0 weight. This means that only the distortions incurred during diagonal moves are considered in the dissimilarity measure of Eqs. (4.124) and (4.125). Using weights of this type would lead to unreasonable discontinuity in time normalization. One way to alleviate this potential difficulty is to redistribute (or smooth) the weights when the individual values change abruptly along any of the allowable paths. As shown in Figure 4.44, for Type II local

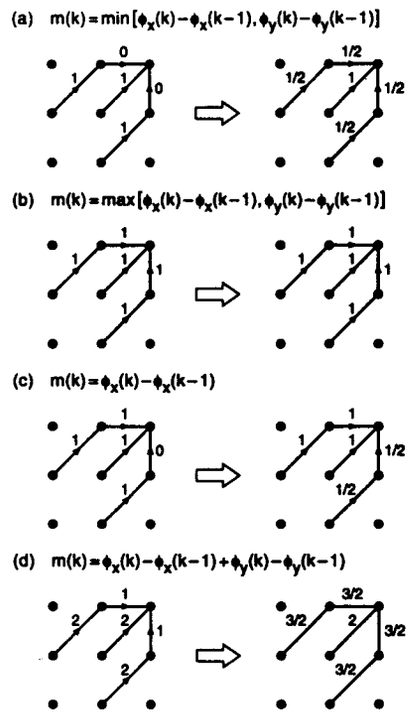



**Figure 4.44** Type II local continuity constraints with 4 types of slope weighting and their smoothed version in which the slope weights are uniformly redistributed along paths where abrupt weight changes exist (after Myers et al. [23]).

constraints, the redistribution of slope weights serves to equally divide the weight along a path where the original slope weights display abrupt changes. The smoothing scheme applies equally to other types of local constraints. Interested readers are encouraged to compute slope weights for several path types in Table 4.5 and for each of the types of slope weight of Eqs. (4.150)–(4.153).

The accumulated distortion also requires an overall normalization as indicated in Eq. (4.124). The purpose of (global) normalization is to have an average path distortion that is independent of the lengths of the two patterns being compared. The normalizing factor for a weighted sequence is customarily the sum of the components of the weighting



sequence, such that

$$M_\phi = \sum_{k=1}^{T} m(k). \tag{4.154}$$

For Types (c) and (d) slope weighting, the normalizing factors would then become

$$M_\phi^{(c)} = \sum_{k=1}^{T} [\phi_x(k) - \phi_x(k-1)] = \phi_x(T) - \phi_x(0) = T_x \tag{4.155}$$

and

$$\begin{aligned} M_\phi^{(d)} &= \sum_{k=1}^{T} [\phi_x(k) - \phi_x(k-1) + \phi_y(k) - \phi_y(k-1)] \\ &= \phi_x(T) - \phi_x(0) + \phi_y(T) - \phi_y(0) = T_x + T_y \end{aligned} \tag{4.156}$$

respectively, independent of the warping functions and the associated constraints.

For Types (a) and (b) slope weighting, however, the normalizing factor as defined by Eq. (4.154) is a strong function of the actual path. While it is possible to compute the normalizing factor for a given warping path, it makes the problem unwieldy if the minimization in Eq. (4.125) is to be solved by recursive dynamic programming algorithms. As such, an arbitrary but reasonable normalizing factor that is independent of the warping functions is chosen so that a standard dynamic programming algorithm can be used to find the optimal alignment path. Typically, for Types (a) and (b) slope weighting, we arbitrarily set

$$M_\phi^{(a)} = M_\phi^{(b)} = T_x. \tag{4.157}$$

Clearly, the (incorrect) normalizing factor of Eq. (4.157) leads to a bias in the time-alignment result. For Type (a) slope weights, there is a tendency to prefer long (indirect or fluctuating) paths (e.g., $\mathcal{P}_1$ and $\mathcal{P}_3$ in Type II local constraints), whereas for Type (b) slope weights, direct, diagonal paths tend to be preferred.

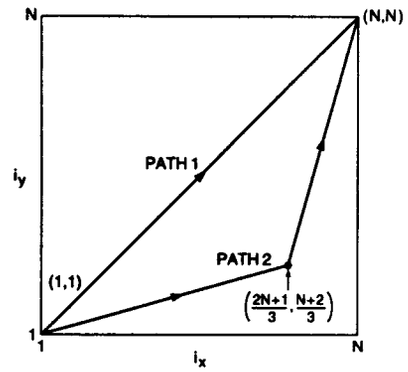
**Exercise 4.6**

To illustrate the problems associated with using a path-independent normalizing factor for the Types (a) and (b) slope weighting (Eqs. (4.150) and (4.151)), consider two extreme paths through a square grid of size $N \times N$:

**Path 1** $N$ path segments all of slope 1

**Path 2** $\frac{2N}{3}$ path segments of slope $\frac{1}{2}$, $\frac{N}{3}$ path segments of slope 2

These two paths are shown below.




1. Using Type (a) slope weighting, what is the true normalizing factor for paths 1 and 2?
2. Using Type (b) slope weighting, what is the true normalizing factor for Paths 1 and 2?
3. Assume we set the path-normalizing factor arbitrarily to path length $N$. Assume that every local distance along both Paths 1 and 2 is of value $\bar{d}$. What is the average path distance using both Types (a) and (b) slope weighting for Paths 1 and 2?
4. What type of path bias is exhibited for Types (a) and (b) slope weighting?

**Solution 4.6**

1. Using Type (a) slope weighting, the true normalizing factors are

$$M^{(a)}_{\text{Path 1}} = \sum_{k=1}^{N} 1 = N$$

$$M^{(a)}_{\text{Path 2}} = \sum_{k=1}^{\frac{2N}{3}} \frac{1}{2} + \sum_{\frac{2N}{3}+1}^{N} 1 = \frac{2N}{3} \cdot \frac{1}{2} + \frac{N}{3} \cdot 1 = \frac{2N}{3}$$

2. Using Type (b) slope weighting, the true normalizing factors are

$$M^{(b)}_{\text{Path 1}} = \sum_{k=1}^{N} 1 = N$$

$$M^{(b)}_{\text{Path 2}} = \sum_{k=1}^{\frac{2N}{3}} 1 + \sum_{\frac{2N}{3}+1}^{N} 2 = \frac{2N}{3} \cdot 1 + \frac{N}{3} \cdot 2 = \frac{4N}{3}$$



3. For Type (a) slope weights we get

$$D^{(a)}_{\text{Path 1}} = \frac{\bar{d}\sum_{k=1}^{N} 1}{N} = \bar{d}$$

$$D^{(a)}_{\text{Path 2}} = \frac{\bar{d}\left[\sum_{k=1}^{\frac{2N}{3}} \frac{1}{2} + \sum_{\frac{2N}{3}+1}^{N} 1\right]}{N} = \frac{2}{3}\bar{d}$$

For Type (b) slope weights we get

$$D^{(b)}_{\text{Path 1}} = \frac{\bar{d}\sum_{k=1}^{N} 1}{N} = \bar{d}$$

$$D^{(a)}_{\text{Path 2}} = \frac{\bar{d}\left[\sum_{k=1}^{\frac{2N}{3}} 1 + \sum_{\frac{2N}{3}+1}^{N} 2\right]}{N} = \frac{4}{3}\bar{d}$$

4. It is clear that since all local distances along the path are constant (of value $\bar{d}$), then the average path distance along *any* path should be equal. However, for Type (a) slope weights we see that the average path distance is smaller for longer paths; hence Type (a) slope weights favor long paths over short paths. Conversely, for Type (b) slope weights we see that the average path distance is smaller for shorter paths; hence Type (b) slope weights favor short paths over long paths.

### 4.7.3 Dynamic Time-Warping Solution

We are now ready to show how we can use a dynamic programming algorithm to provide an efficient way to solve the minimization problem involved in the definition of the pattern dissimilarity measure of Eq. (4.125) with embedded time-normalization and alignment. Although the local path constraints and slope weighting discussed in the last section do require adjustments to the original algorithm, the principle of optimality and the dynamic programming equations, particularly Eq. (4.128), are essentially directly and straightforwardly applicable to this problem.

Due to the endpoint constraints (Eq. (4.136)), we rewrite Eq. (4.125) in terms of $T_x$ and $T_y$ as

$$\begin{aligned} M_\phi d(\mathcal{X}, \mathcal{Y}) &\triangleq D(T_x, T_y) \\ &= \min_{\phi_x, \phi_y} \sum_{k=1}^{T} d(\phi_x(k), \phi_y(k)) m(k) \end{aligned} \tag{4.158}$$

since $\mathcal{X}$ and $\mathcal{Y}$ terminate at $T_x$ and $T_y$, respectively. Similarly, the minimum partial accumulated distortion along a path connecting $(1, 1)$ and $(i_x, i_y)$ is

$$D(i_x, i_y) \triangleq \min_{\phi_x, \phi_y, T'} \sum_{k=1}^{T'} d(\phi_x(k), \phi_y(k))m(k), \tag{4.159}$$

where

$$\phi_x(T') = i_x \quad \text{and} \quad \phi_y(T') = i_y \tag{4.160}$$

are implied. The dynamic programming recursion with constraints thus becomes

$$D(i_x, i_y) = \min_{(i'_x, i'_y)} [D(i'_x, i'_y) + \zeta((i'_x, i'_y), (i_x, i_y))], \tag{4.161}$$

where $\zeta$ is the weighted accumulated distortion (local distance) between point $(i'_x, i'_y)$ and point $(i_x, i_y)$

$$\zeta((i'_x, i'_y), (i_x, i_y)) = \sum_{\ell=0}^{L_s} d(\phi_x(T' - \ell), \phi_y(T' - \ell))m(T' - \ell) \tag{4.162}$$

with $L_s$ being the number of moves in the path from $(i'_x, i'_y)$ to $(i_x, i_y)$ according to $\phi_x$ and $\phi_y$. Again, Eq. (4.160) is implied as well as

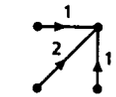
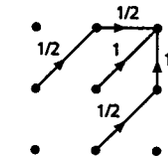
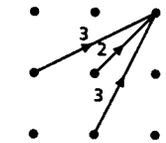
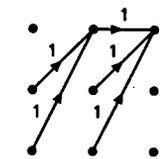
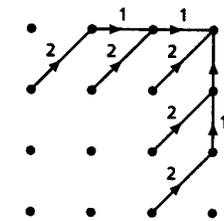
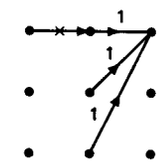
$$\phi_x(T' - L_s) = i'_x \quad \text{and} \quad \phi_y(T' - L_s) = i'_y. \tag{4.163}$$

The incremental distortion $\zeta$ is evaluated only along the allowable paths as defined by the chosen local continuity constraints for efficient implementation of the dynamic programming algorithm—that is, only along the paths defined by the various sets of local constraints. In other words, the range of $(i'_x, i'_y)$ for minimization in Eq. (4.161) is limited to those points that qualify as the legal originating points of the incremental paths as specified by the chosen set of local constraints. Table 4.7 summarizes the key recursion formula for several types of local continuity constraints with typical slope weighting. (The table is not exhaustive and the reader is encouraged to work out a couple of specific examples not covered here.)

As mentioned earlier, the distinction between Type IV local constraints and the Itakura constraints is subtle and worth noting at this point. For the Itakura constraints, a function $g(k)$ is introduced to prevent paths that traverse horizontally for more than one move. Note that when the best path to reach $(i_x - 1, i_y)$ is from $(i_x - 2, i_y)$, the algorithm essentially discards all other allowable connections to $(i_x - 1, i_y)$ except that from $(i_x - 2, i_y)$. When this occurs, the best path to reach $(i_x, i_y)$ can come only from either $(i_x - 1, i_y - 1)$ or $(i_x - 1, i_y - 2)$ because two consecutive horizontal moves are not allowed. Itakura's implementation therefore does not consider the path from $(i_x - 2, i_y - 1)$ to $(i_x, i_y)$ via $(i_x - 1, i_y)$ even if the accumulated distortion at $(i_x, i_y)$ via this particular path is smaller than other paths that go through $(i_x - 1, i_y - 1)$ or $(i_x - 1, i_y - 2)$ and ranked only second to the path $(i_x - 2, i_y) \to (i_x - 1, i_y) \to (i_x, i_y)$, which is disallowed because of the constraint.



**TABLE 4.7.** Summary of sets of local constraints, slope weights, and DP recursion formulas

| Local Constraints & Slope Weights | DP Recursion Formula |
|---|---|
|  | $\min \left\{ \begin{array}{l} D(i_x-1,i_y)+d(i_x,i_y), \\ D(i_x-1,i_y-1)+2d(i_x,i_y), \\ D(i_x,i_y-1)+d(i_x,i_y) \end{array} \right\}$ |
|  | $\min \left\{ \begin{array}{l} D(i_x-2,i_y-1)+\frac{1}{2}[d(i_x-1,i_y)+d(i_x,i_y)], \\ D(i_x-1,i_y-1)+d(i_x,i_y), \\ D(i_x-1,i_y-2)+\frac{1}{2}[d(i_x,i_y-1)+d(i_x,i_y)] \end{array} \right\}$ |
|  | $\min \left\{ \begin{array}{l} D(i_x-2,i_y-1)+3d(i_x,i_y), \\ D(i_x-1,i_y-1)+2d(i_x,i_y), \\ D(i_x-1,i_y-2)+3d(i_x,i_y), \end{array} \right\}$ |
|  | $\min \left\{ \begin{array}{l} D(i_x-2,i_y-1)+d(i_x-1,i_y)+d(i_x,i_y), \\ D(i_x-2,i_y-2)+d(i_x-1,i_y)+d(i_x,i_y), \\ D(i_x-1,i_y-1)+d(i_x,i_y), \\ D(i_x-1,i_y-2)+d(i_x,i_y), \end{array} \right\}$ |
|  | $\min \left\{ \begin{array}{l} D(i_x-3,i_y-1)+2d(i_x-2,i_y)+d(i_x-1,i_y)+d(i_x,i_y), \\ D(i_x-1,i_y-1)+2d(i_x-1,i_y)+d(i_x,i_y), \\ D(i_x-1,i_y-1)+2d(i_x,i_y), \\ D(i_x-1,i_y-2)+2d(i_x,i_y-1)+d(i_x,i_y), \\ D(i_x-1,i_y-2)+2d(i_x,i_y-1)+d(i_x,i_y), \\ D(i_x-1,i_y-3)+2d(i_x,i_y-2)+d(i_x,i_y-1)+d(i_x,i_y), \end{array} \right\}$ |
|  | $\min \left\{ \begin{array}{l} D(i_x-1,i_y)g(k)+d(i_x,i_y), \\ D(i_x-1,i_y-1)+d(i_x,i_y), \\ D(i_x-1,i_y-2)+d(i_x,i_y), \end{array} \right\}$ with $g(k) = \begin{cases} 1 & \phi(k-1) \neq \phi_y(k-2) \\ \infty & \phi(k-1) = \phi_y(k-2) \end{cases}$ |

In Type IV constraints, the algorithm maintains the partial accumulated distortion at two consecutive frame levels, instead of one, as in the case of Itakura's constraints, and thus the optimal path that satisfies the constraints of no two consecutive horizontal moves can be found.

We have removed the normalizing factor $M_\phi$ out of the dynamic programming recursion above because it is assumed to be independent of the warping path. It can be reinstated once the optimal path is found as indicated in Eq. (4.158). At this point, let us summarize the dynamic programming implementation for finding the best path through a $T_x$ by $T_y$ grid, beginning at $(1, 1)$ and ending at $(T_x, T_y)$, as follows:

1. Initialization

$$D_A(1,1) = d(1,1)m(1).$$

2. Recursion
For $1 \le i_x \le T_x$, $1 \le i_y \le T_y$ such that $i_x$ and $i_y$ stay within the allowable grid, compute

$$D_A(i_x, i_y) = \min_{(i'_x, i'_y)}[D_A(i'_x, i'_y) + \zeta((i'_x, i'_y), (i_x, i_y))],$$

where $\zeta((i'_x, i'_y), (i_x, i_y))$ is defined by Eq. (4.162).

3. Termination

$$d(\mathcal{X}, \mathcal{Y}) = \frac{D_A(T_x, T_y)}{M_\phi}.$$

The key idea of the algorithm is that the recursion step is done for all local paths that reach $(i_x, i_y)$ in exactly one step (from $(i'_x, i'_y)$) using the local path constraints chosen for the implementation. Only values of $(i_x, i_y)$ that can be reached from $(1, 1)$ and can end ultimately at $(T_x, T_y)$ are evaluated in the recursion step. Figure 4.45 shows the set of grid points on a $40 \times 40$ grid that are used in the recursion when a set of local path constraints that allow a 2-to-1 time scale expansion and a 2-to-1 time scale contraction are used. It can be seen that about $(T_x T_y)/3$ grid points fall within the discrete parallelogram of Figure 4.45. (In this case there are 560 out of 1600 possible grid points.)

The number of grid points falling within the parallelogram is also the number of local distance calculations, $\zeta((i'_x, i'_y), (i_x, i_y))$, required for implementing the DTW procedure. In general, this constitutes about 80% of the total DTW computational requirements, with the rest being that of the combinatorics for optimal path search. Proper use of the global constraint of Eq. (4.149), i.e. proper choice of $T_0$ to limit the number of grid points, would also have an effect on the computational load.

It can be shown (see Exercise 4.8) that when using a set of local path constraints that allows a $k$-to-1 time scale expansion and a $k$-to-1 time scale contraction, the ratio of grid points within the parallelogram of allowable grid points to grid points within the $T_x \times T_y$ rectangle is



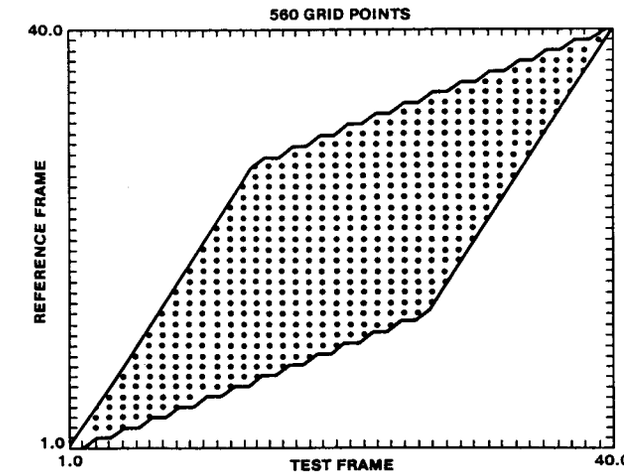



**Figure 4.45** Set of allowable grid points for dynamic programming implementation of local path expansion and contraction by 2 to 1.

$$R = \frac{\left(k - \frac{T_x}{T_y}\right)\left(k - \frac{T_y}{T_x}\right)}{(k^2 - 1)}. \tag{4.164}$$

Thusfor the case of $k = 2$, with $T_x = T_y$ we get $R = 1/3$. When $k = 3$, again with $T_x = T_y$ we get $R = 1/2$, about half the grid points are allowable with a 3:1 scale expansion and contraction.

While the definition of the dissimilarity measure of Eq. (4.125) involves a minimization process that can be effectively solved with mathematical precision by the dynamic programming algorithm described above, the heuristic nature of the dissimilarity measure should not be overlooked. The constraints for local continuity and the slope weighting are mostly based upon intuition and are not motivated by analytical results. As such, the optimal warping path that provides the best match under the given constraints and is used for time alignment and normalization is subject to various interpretations. In comparing utterances of the same word, for example, this best match *may* truly represent the best time alignment. For utterances of linguistically different words, however, the optimal path is not really meaningful, except as the solution of the dynamic programming process, because "correct" time alignment between two utterances of different words is not a well-defined linguistic concept. Furthermore, time alignment in the current context is performed on a short-time basis (using spectral slices on the order of 10 msec) with short-time constraints.

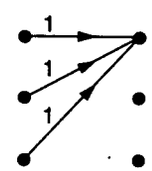
Since speech is a slowly varying signal in which the linguistic sound codes are produced at a rate of approximately 5–10 per second, the dynamic time-warping procedure and the associated constraints are thus more than likely to be unduly restrictive, particularly for those speech segments that represent steady-state sounds. In fact, a close examination of the accumulated distortion resulting from various warping paths leads to the finding that a large number of warping paths produces an accumulated distortion that is very close to the minimum attainable value, especially for utterances of the same word. The convenient definition of Eq. (4.125) for two arbitrary speech patterns has been demonstrated to be practically effective in speech-recognition applications, but as discussed, may be subject to further interpretation. One important consideration is in terms of the consistency in speech sequence representation. We shall cover this point in more detail in Chapter 5.

**Exercise 4.7**

1. Consider the problem of finding the best path through the $6 \times 6$ grid of local distances shown below:

| $i_y$ | | | | | | |
|---|---|---|---|---|---|---|
| 6 | •3 | •2 | •3 | •2 | •2 | •2 |
| 5 | •3 | •2 | •3 | •1 | •1 | •1 |
| 4 | •2 | •2 | •2 | •1 | •2 | •2 |
| 3 | •2 | •1 | •2 | •2 | •1 | •3 |
| 2 | •1 | •1 | •1 | •1 | •2 | •3 |
| 1 | •1 | •1 | •3 | •3 | •3 | •3 |
| | 1 | 2 | 3 | 4 | 5 | 6 $i_x$ |

Assume that the path must begin at $(1, 1)$ and end at $(6, 6)$. Assume local path constraints and slope weights of the form



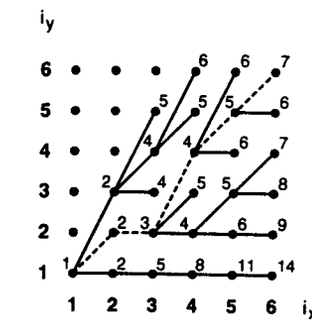

What is the best path through the grid?



2. Now assume the path can begin at any of the points $(1, 1)$, $(1, 2)$ or $(1, 3)$ and end at any of the points $(6, 4)$, $(6, 5)$ or $(6, 6)$. Now what is the best path through the grid?

**Solution 4.7**

1. Using the dynamic programming algorithm, we can solve for the accumulated distance at each (allowed path) grid point giving




The best path is shown as the dashed path with total accumulated path distance of 7, and average path distance of $7/6$.

2. Beginning the total accumulated path at either $(1, 2)$ or $(1, 3)$ only increases the total accumulated path distance, so $(1, 1)$ is the best initial path point. Ending the same path as above at $(6, 5)$ rather than at $(6, 6)$ gives a smaller total accumulated path distance (6 versus 7); hence this is now the best path through the grid.

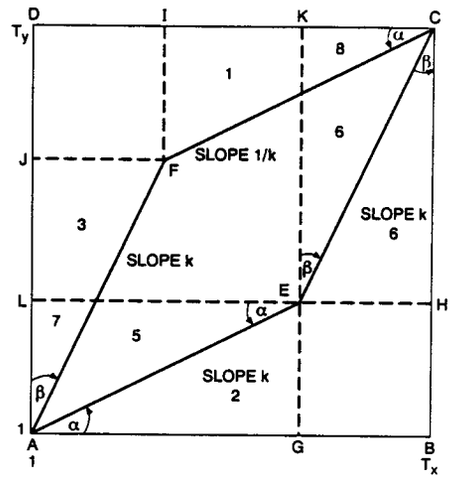
**Exercise 4.8**

In implementing the time-alignment procedure via dynamic time warping, we are generally considering a discrete grid of dimension $T_x \times T_y$. Show that for any set of local path constraints such that the global path has a maximum slope constraint of $E_{\max} = k$, and a minimum slope constraint of $E_{\min} = 1/k$, $k > 1$, the ratio of grid points within the parallelogram of allowable path grid points to grid points within the $T_x \times T_y$ rectangle is

$$R = \frac{\left[k - \frac{T_x}{T_y}\right]\left[k - \frac{T_y}{T_x}\right]}{(k^2 - 1)}.$$

**Solution 4.8**

We will solve this exercise geometrically by referring to the points defined in the figure below. The following areas, angles, and lengths can be defined:




1. Rectangle $ABCD$ (the entire grid) has area $S_{ABCD} = T_x \times T_y$
2. $\measuredangle GAE = \measuredangle ICF = \alpha$
   $\measuredangle HCE = \measuredangle JAF = \beta$
3. In parallelogram $AECF$ we have $AE = CF, AF = CE$ and

$$\Delta AGE = \Delta CIF = \Delta ELA$$
$$\Delta CKE = \Delta AJF$$

4. By recognizing that

$$\Delta ALE = S_5 + S_7 = \Delta CIF = S_1 + S_8$$
$$\Delta CKE = S_6 + S_8 = \Delta AJF = S_3 + S_7$$

we get

$$S_5 + S_6 = S_1 + S_3$$

so that

$$\begin{aligned} S_{AECF} &= S_{LEKD} - S_{JFID} + S_5 + S_6 - S_1 - S_3 \\ &= S_{LEKD} - S_{JFID} \\ &= AG \times CH - EG \times EH \end{aligned}$$

5. $\dfrac{EG}{AG} = \dfrac{1}{k}, \dfrac{EH}{CH} = \dfrac{1}{k}$
6. $S_{AECF} = AG \times CH \left(1 - \dfrac{1}{k^2}\right)$

$$AG + GB = T_x = AG + \frac{CH}{k}$$



$$BH + CH = T_y = \frac{AG}{k} + CH$$

Solve for $AG, CH$ as

$$AG = \frac{T_x - T_y/k}{1 - 1/k^2}$$
$$CH = \frac{T_y - T_x/k}{1 - 1/k^2}$$

giving

$$R = \frac{S_{AECF}}{S_{ABCD}} = \frac{(T_x - T_y/k)(T_y - T_x/k)}{\left(1 - \frac{1}{k^2}\right) T_x T_y}$$
$$= \frac{\left(k - \frac{T_x}{T_y}\right)\left(k - \frac{T_y}{T_x}\right)}{(k^2 - 1)}$$

### 4.7.4 Other Considerations in Dynamic Time Warping

We mentioned in the last section that the appearance of mathematical rigorousness of Eq. (4.125) does not alter its heuristic nature in terms of time normalization. Since the "correct" time alignment between utterances of different words does not exist linguistically, the minimization result in Eq. (4.125) cannot be meaningfully interpreted. Even for utterances of the same word, the dynamic time warping scheme to accomplish Eq. (4.125) is predicated on the given endpoint information, which is considered part of the definition of the speech patterns. This endpoint information is often treated as a set of independent parameters and appears in the form of the endpoint constraints in Eq. (4.136) for dynamic time warping. As discussed in Section 4.2, however, the determination of the beginning and ending points of a speech utterance is, at best, an imprecise calculation. This is especially a problem for words that begin or end with weak fricatives whose power level and spectral characteristics are not significantly different from the acoustic background. Also, in the presence of substantial acoustic noise, reliable detection of the endpoints is even more difficult to accomplish in a manner independent of the speech-recognition operation that follows.

Another potential problem in dynamic time-warping implementation is the restriction on the range of possible warping paths due to the interaction between continuity constraints and the utterance duration ratio $(T_x - 1)/(T_y - 1)$. When $T_x - 1 > Q_{\max}(T_y - 1)$ or $T_y - 1 > Q_{\max}(T_x - 1)$, the global constraints of Eq. (4.145) and (4.146) make time warping impossible. There is thus a desire to allow additional flexibility to exist in the path constraints such that these extreme cases are not precluded in the distortion minimization process while at the same time maintaining the local continuity constraints.

Another potential problem occurs when the durations, $T_x$ and $T_y$, are large. In this case it is obvious that a large number of grid points within the allowable warping region in the $(i_x, i_y)$ plane have to be considered in the dynamic time-warping process, thereby incurring a high computational load. Since one would expect the optimal warping path

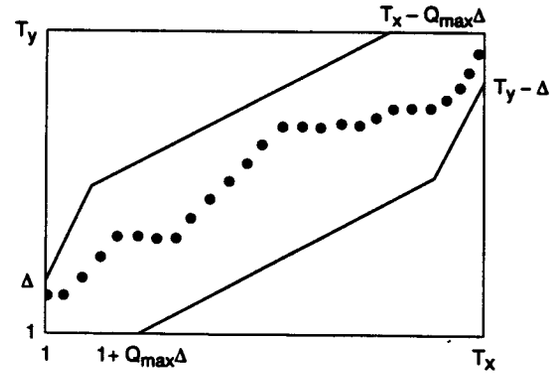



**Figure 4.46** The allowable path region for dynamic time alignment with relaxed endpoint constraints.

to be reasonably close to a linear path, much of the computation at the extremities of the allowable region may be unnecessary. Thus, possibly a suboptimal procedure could be capable of substantially reducing the computational requirements with little increase in the minimized accumulated distortion.

Consideration of the above problems has led to several proposed modifications to the standard dynamic time-warping algorithm. We now discuss two such proposals that have been extensively studied.

The first modification is to relax the endpoint constraints of (4.136). Specifically, the following new set of boundary conditions is used.

$$1 \le \phi_x(1) \le 1 + Q_{\max}\Delta \tag{4.165a}$$

$$1 \le \phi_y(1) \le 1 + \Delta \tag{4.165b}$$

$$T_x - Q_{\max}\Delta \le \phi_x(T) \le T_x \tag{4.165c}$$

$$T_y - \Delta \le \phi_y(T) \le T_y. \tag{4.165d}$$

Figure 4.46 shows the allowable path region in the $(i_x, i_y)$ plane as a result of the above relaxed endpoint constraints. The expanded range $\Delta$ represents the maximum anticipated mismatch or uncertainty in the endpoints of the patterns. The warping path, with this embedded endpoint uncertainty, can start at a grid point other than $(1, 1)$ and end at a grid point other than $(T_x, T_y)$. Typically, $Q_{\max} = 2$ and $\Delta$ is the number of frames corresponding to approximately a 75-ms region. (For a frame shift of 15 ms, $\Delta = 5$.)

To illustrate the computational effect of expanding the beginning path range by five frames and the ending path range by nine frames, Figure 4.47 shows the revised set of allowable grid points on a $40 \times 40$ grid. We see that now there are 844 grid points to be evaluated, or about 50% more grid points than the grid of Figure 4.45, where there was a unique initial and final path point.



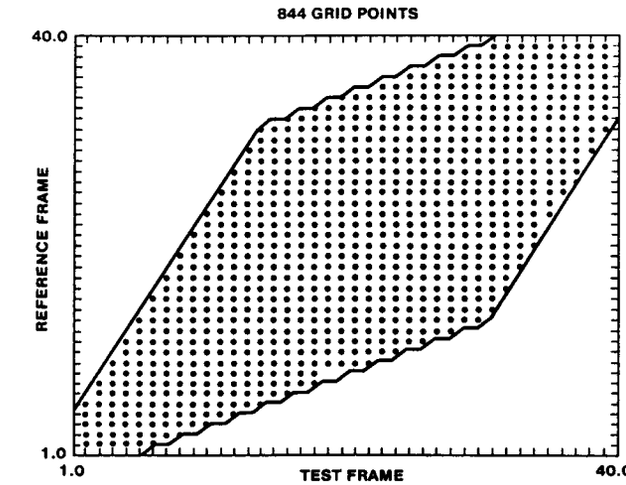



**Figure 4.47** Set of allowable grid points when opening up the initial point range to 5 frames and the final point range to 9 frames.

Since the endpoint constraints of Eq. (4.165) allow the warping function to reach the boundary prior to the last frame of either of the two patterns, the competing warping paths may not end at the same point and the accumulated distortion $d(\mathcal{X}, \mathcal{Y})$ as defined in Eq. (4.158) includes an uncertainty in the normalized duration $T$ for different paths, giving rise to the need for scaling on the accumulated distortion. The scaling factor for the accumulated distortion along a path that undergoes $T'$ moves from the beginning to the end usually takes the form $T/T'$, where $T$ is the nominal (or maximum) number of moves in terms of the normal time scale.

Another possible modification to the time-warping algorithm is to include an extra constraint that requires the dynamic path search to be limited to lie within a range of the locally optimal path. The beginning point satisfies the constraints of Eq. (4.165a) and (4.165b), but no explicit constraint is imposed a priori on the ending point. This can be conveniently introduced by considering a simple warping function $\phi_x(k) = k$, which implies that $i_x$ is treated as the normal time axis. The dynamic programming recursion of Eq. (4.161) for $i_x = k$ is now performed only for a range of $i_y$, which satisfies

$$i_y^\circ - \epsilon \le i_y \le i_y^\circ + \epsilon, \tag{4.166}$$

where

$$i_y^\circ = \arg\min_{i_y} D(k-1, i_y), \tag{4.167}$$

and $\epsilon$ is a prespecified range. The purpose of this additional constraint is to significantly reduce the computation by limiting the allowable path region to a close vicinity of an instantaneously determined, locally optimal path according to Eq. (4.167).

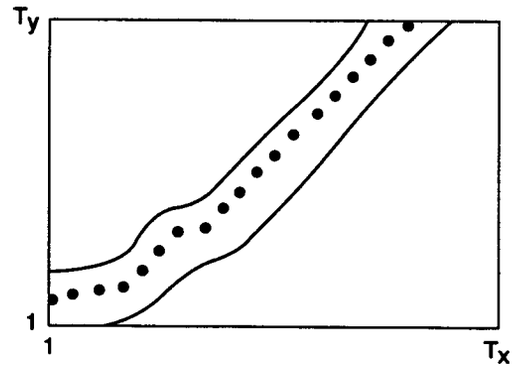



**Figure 4.48** The allowable path region for dynamic time alignment with localized range constraints.

Figure 4.48 illustrates the limited search region, resulting from the above constraints. The dynamic search range $\epsilon$ is typically four frames, corresponding to a 60-ms interval around the local minimum point.

### 4.7.5 Multiple Time-Alignment Paths

The dissimilarity definition of Eq. (4.125) relies solely on the minimum accumulated distortion. The warping path that leads to the minimum accumulated distortion is considered the "correct" time alignment to normalize the temporal variability in the utterances. As explained in Section 4.7.3, this "optimal time alignment" may be unduly restrictive, and decision processing based on a single matching path is often too sensitive and not robust enough to cope with situations that may be just minor deviations from the normal conditions. In many applications, therefore, it is desirable to consider not only the best matching path but a multiplicity of reasonable candidate paths so that reliable decision processing can be performed. In the current context, the focus is on spectral matching (decoding); however, similar situations can arise in phoneme sequence matching (decoding), word as well as phrase sequence matching (decoding), and so forth. Such cases will be discussed in later chapters when speech-recognition algorithms and systems are presented.

The fundamental tool of dynamic programming and the principle of optimality can be applied to solve the problem of finding the $K$-best paths, instead of just the one best path, requiring only minor modifications. We shall use the synchronous sequential decoding problem as the vehicle for our presentation here and for simplicity, search constraints like those discussed in Section 4.7.2 are not explicitly considered as they are not essential to the basic methodology and formulation.

Referring to Figure 4.39, we use the fact that the best $m^{\text{th}}$ move to point $n$ is



accomplished if it continues from point $j$ at the end of the $(m-1)^{\text{th}}$ move where $j$ satisfies

$$j = \arg \min_{1 \le \ell \le N} [\varphi_{m-1}(i, \ell) + \zeta(\ell, n)]. \tag{4.168}$$

(Recall that $\varphi_m(i, j)$ is the minimum accumulated cost moving from $i$ to $j$ in $m$ steps.) The best path after the $m^{\text{th}}$ move, starting out (i.e., $m = 0$) from an arbitrary initial point, $i$, is the one that ends at $n$,

$$n = \arg \min_{1 \le \ell \le N} \varphi_m(i, \ell). \tag{4.169}$$

To find the *single* best path, the recursion of (4.168) indicates that all we need to keep track of is the $N$ best paths, and the associated accumulated cost, that end at each of the $N$ points, respectively, at any moment or move. (We shall again assume that moves occur at regular unit time intervals.) This is obvious because any path that connects $i$ and $\ell$ but results in an accumulated cost greater than $\varphi_m(i, \ell)$ cannot be the best path overall.

To find the second-best path at the end of the $m^{\text{th}}$ move is slightly more complicated. To facilitate our presentation, let us denote a path that connects $i$ and $n$ in $m$ steps by ${}_m(i, n)_k$ where the additional subscript $k$ is used to indicate the rank of the path in terms of the associated accumulated cost. When $m$ or $k$ is omitted, it is assumed to be 1. Therefore, with $\zeta$ denoting the accumulated cost,

$$\zeta(i, n) = \zeta_1(i, n)_1, \tag{4.170}$$

$$\varphi_m(i, n) = \zeta_m(i, n)_1, \tag{4.171}$$

and

$$\zeta_m(i, n)_1 < \zeta_m(i, n)_2 < \zeta_m(i, n)_3 < \cdots. \tag{4.172}$$

#### 4.7.5.1 Parallel Algorithm

The second-best path after the $m^{\text{th}}$ move could reach the same point as the best path. Therefore, the accumulated costs that must be considered to find the second-best path after $m$ moves are $\{\zeta_m(i, \ell)_k,\ k = 1, 2\}_{\ell=1}^{N}$. Using $\min^{(k)}$ to signify "the $k^{\text{th}}$ smallest value of," we can thus determine $n_1$ and $n_2$

$$n_1 = \arg \min_{1 \le \ell \le N}{}^{(1)} \zeta_m(i, \ell)_1 \tag{4.173}$$

$$n_2 = \arg \min_{\substack{1 \le \ell \le N \\ k=1,2}}{}^{(2)} \zeta_m(i, \ell)_k \tag{4.174}$$

to be the ending (the $m^{\text{th}}$) points of the best and the second-best paths, originating from point $i$ at $m = 0$. Note that $n_1$ may or may not be identical to $n_2$ and the best ending point after $m$ moves also satisfies

$$n_1 = \arg \min_{1 \le \ell \le N}{}^{(1)} \zeta_m(i, \ell)_k, \qquad k \ge 1, \tag{4.175}$$

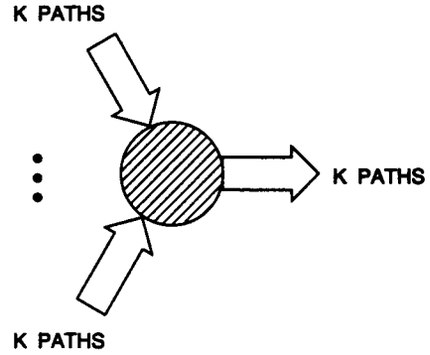



**Figure 4.49** Dynamic programming for finding $K$-best paths implemented in a parallel manner.

which is obvious because of Eq. (4.172). As the paths move forward, the recursive relationship becomes

$$\zeta_{m+1}(i,n)_1 = \min_{1\le\ell\le N}{}^{(1)}[\zeta_m(i,\ell)_1 + \zeta(\ell,n)] \tag{4.176}$$

and

$$\zeta_{m+1}(i,n)_2 = \min_{\substack{1\le\ell\le N\\ k=1,2}}{}^{(2)}[\zeta_m(i,\ell)_k + \zeta(\ell,n)]. \tag{4.177}$$

Therefore, to find the second-best path, we need to store and examine the two smallest accumulated costs for every point at every possible move.

The same principle can be extended to the case where $k > 2$. In particular, the following recursion holds

$$\zeta_{m+1}(i,n)_K = \min_{\substack{1\le\ell\le N\\ 1\le k\le K}}{}^{(K)}[\zeta_m(i,\ell)_k + \zeta(\ell,n)]. \tag{4.178}$$

The $K^{\text{th}}$ best path after $m+1$ moves ends at

$$n_K = \arg\min_{\substack{1\le\ell\le N\\ 1\le k\le N}}{}^{(K)}\zeta_{m+1}(i,\ell)_k. \tag{4.179}$$

Figure 4.49 illustrates the implication of Eq. (4.178), which indicates that for every point $n$, $NK$ accumulated costs are pooled together and $K$-smallest accumulated cost figures are augmented and retained after each move. The algorithm requires maintaining an array of $NK$ accumulated costs and an array of $NK$ point indices to store the path history at every move. When the total number of moves, $M$, is large, the complexity can be exorbitant, not just because of the storage involved but also because of the necessary sorting operation. One advantage of this approach, however, is that the $K$-best paths are found essentially simultaneously for any number of moves. This method is often referred to as the parallel algorithm.



#### 4.7.5.2 Serial Algorithm

Another way to find the $K$-best paths without requiring large data storage arrays is to search for these paths one at a time in a serial manner, starting from the very best. This procedure is thus called the serial algorithm for convenience. Consider the synchronous sequential decision problem, of Section 4.7.1, in which we are interested in finding a path of $M$ moves, starting from point $i$ and ending at point $n$. Note that if we have obtained the best path ${}_M(i,n)_1$ which passes through point $n_1$ at the $(M-1)^{\text{th}}$ move, we can be sure that the second-best path ${}_M(i,n)_2$ must have one of the following partial paths for the first $(M-1)$ moves: ${}_{M-1}(i,n_1)_2$ or ${}_{M-1}(i,\ell)_1$, $\ell = 1,2,\ldots,N$, $\ell \ne n_1$.

It is not necessary to consider ${}_{M-1}(i,\ell)_2$, $\ell \ne n_1$, because this partial path *cannot* be part of the second-best path. For paths ${}_{M-1}(i,\ell)_1$, $\ell = 1,2,\ldots,N$, $\ell \ne n_1$ the recursion of Eq. (4.168) applies and it is only necessary to maintain a single (the best) path history for each point along ${}_{M-1}(i,\ell)_1$. For each point along ${}_{M-1}(i,n_1)_1$, however, we need to maintain two path records because ${}_{M-1}(i,n_1)_2$ could be part of the second-best path. In this way, we are no longer required to keep track of $2N$ accumulated costs and $2N$ path records for every trellis point in the second-best path search case. The same idea extends to $K$-best path cases. We summarize the serial algorithm implementation in the following.

1. Initialization
   (a) Set the number of ranked paths that have been found, $k$, to zero.
   (b) Set up a path array of size $K \times M$ where $K$ is the maximum number of best paths to be found and $M$ is the designated path length (number of moves). This array is used to store the $K$-best path records (indices) to be found.
   (c) Form a "visit count" array of size $N \times M$ where $N$ is the number of points in consideration. The $jm^{\text{th}}$ element $c_{jm}$ of this array is the number of paths among the $k$ globally best that pass through point $j$ at the $m^{\text{th}}$ move. Initially, $c_{jm} = 0$ for all $j$ and $m$.
2. Recursion
   (a) Among all the paths entering point $j$ at the $m^{\text{th}}$ move, find and retain $c_{jm}+1$ paths with the minimum accumulated costs up to that point. That is, similar to Eq. (4.178), find and retain $\zeta_m(i,j)_{k'}$, $k' = 1,2,\ldots,(c_{jm}+1)$, by rank ordering all the accumulated costs to that point. The set of accumulated costs under consideration is $\{\zeta_{m-1}(i,\ell)_{k_\ell} + \zeta(\ell,j)\}$ where $\ell = 1,2,\ldots,N$ and $k_\ell = 1,2,\ldots,c_{\ell,(m-1)}$. Progress forward from $m=1$ to $m=M$. (By definition, $j=n$ at $m=M$.)
   (b) Find the $(k+1)^{\text{th}}$ best path ${}_M(i,n)_{k+1}$ after $M$ moves by

$$\zeta_M(i,n)_{k+1} = \min_{\substack{1\le\ell\le N\\ k_\ell=1,2,\ldots,c_{\ell,(M-1)}}}{}^{(k+1)}\left[\zeta_{M-1}(i,\ell)_{k_\ell} + \zeta(\ell,n)\right] \tag{4.180}$$

   Note that $\zeta_M(i,n)_1, \zeta_M(i,n)_2, \ldots, \zeta_M(i,n)_k$ are already known at this point and can be used to simplify the $\min^{(k+1)}$ operation in Eq. (4.180).
   (c) Increment the "visit count" array entries $c_{jm}$ by 1 for those points on path

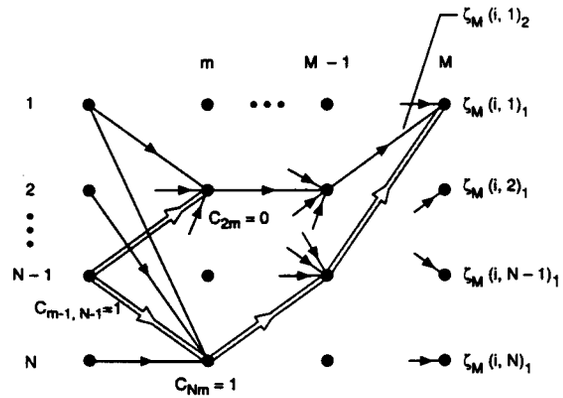



**Figure 4.50** The serial dynamic programming algorithm for finding the $K$-best paths.

${}_M(i,n)_{k+1}$.

(d) Increment $k \to k+1$.

(e) Iterate 2a through 2d until $k = K$.

Figure 4.50 illustrates the serial algorithm for $K = 2$. In the figure the shaded path is ${}_M(i,n)_1$ which passes through 1 at $m = M - 1$. The search for ${}_M(i,n)_2$ relies on the knowledge of ${}_M(i,n)_1$ which is embedded in $\{c_{jm}\}$. It can be easily verified that the algorithm does not require one to simultaneously maintain arrays of size $K \times N \times M$ as does the parallel algorithm. Particular attention should be placed, however, on the one-way, forward search nature of the algorithm; the search starts from the beginning at $m = 1$ till the end at $m = M$ for each and every $k^{\text{th}}$ optimal path. Many rank-ordering (sorting) operations are thus unnecessarily repeated, showing room for further simplification of the algorithm.

### 4.7.5.3 Tree-Trellis Search

The serial algorithm, although in a limited sense more efficient than the parallel algorithm by making use of the knowledge of ${}_M(i,n)_k$, does not take advantage of the fact that when the globally best path is found, we already have at hand all the partial minimum accumulated costs ${}_m(i,\ell)_1$, for all $m$ and $\ell$. The search in the serial algorithm is always forward and at the $m^{\text{th}}$ move, none of the information beyond the $m^{\text{th}}$ move is being used to facilitate a more efficient search. The Tree-Trellis algorithm combines the forward (trellis) search and the backward (tree) search in an intelligent way that produces the desired results much faster and easier than either the parallel or the serial algorithm.

Let us consider partial paths of ${}_M(i,n)_k$. We define

$$ {}_M^{m+1}(i,n)_k = \begin{array}{l}\{\text{a partial path consisting of the least } (M-m) \text{ points of the sequence } {}_M(i,n)_k\}\end{array} . \tag{4.181}$$



For example, if ${}_M(i,n)_k = (i, i_1, i_2, \ldots, i_m, i_{m+1}, \ldots, i_{M-1}, n)$, ${}_M^{m+1}(i,n)_k = (i_{m+1}, \ldots, i_{M-1}, n)$. With $*$ denoting concatenation of paths, we further define

$$ m(i,\ell) * {}_M^{m+1}(i,n)_k = \begin{array}{l}\{\text{the path that traverses through point } \ell \text{ at the } m^{\text{th}} \text{ move with minimum cost and merges with } {}_M(i,n)_k \text{ at the } (m+1)^{\text{th}} \text{ move.}\}\end{array} \tag{4.182}$$

The accumulated cost of ${}_m(i,\ell) * {}_M^{m+1}(i,n)_k$ is simply

$$ \zeta\left[{}_m(i,\ell) * {}_M^{m+1}(i,n)_k\right] = \zeta_m(i,\ell) + \zeta(\ell, j_{mk}) + \zeta_M^{m+1}(i,n)_k, \tag{4.183}$$

where $j_{mk}$ is the beginning point of ${}_M^{m+1}(i,n)_k$. The representation of ${}_m(i,\ell) * {}_M^{m+1}(i,n)_k$ is unique because both ${}_m(i,\ell)$ and ${}_M(i,n)_k$ are uniquely determined. Note that the essential parameters in this representation are $m, \ell$ and $k$. This representation is particularly useful in the sequential search for the $K$-best paths.

Let us assume $\tilde{k} < k$ and $m_{\tilde{k}}$ is the smallest such that

$$ {}_M^{m_{\tilde{k}}+1}(i,n)_{\tilde{k}} = {}_M^{m_{\tilde{k}}+1}(i,n)_k. \tag{4.184}$$

In other words, paths ${}_M(i,n)_{\tilde{k}}$ and ${}_M(i,n)_k$ have the longest overlap in the last part of each path, among all $\tilde{k} < k$. Then, the set

$$ \Theta_k \triangleq \left\{{}_m(i,\ell) *_M^{m+1}(i,n)_k;\ \ell = 1,2,\ldots,N, \text{ and } m < m_{\tilde{k}} \text{ with } {}_M(i,n)_k \text{ excluded}\right\} \tag{4.185}$$

where $m_{\tilde{k}}$ satisfies Eq. (4.184), and has certain properties that allow us to conduct the path search in an intelligent manner. (Note for $k = 1$, $m_{\tilde{k}} \triangleq M$.) It can be easily shown that

$$ {}_M(i,n)_{k+1} \in \{\Theta_{k'}, k' = 1,2,\ldots,k\}. \tag{4.186}$$

In fact,

$$ {}_M(i,n)_{k+1} = \arg \min_{\{\Theta_{k'}, k'=1,2,\ldots,k\}}^{(k+1)} \zeta(\theta), \tag{4.187}$$

where $\theta$ is a path in $\{\Theta_{k'}, k' = 1,2,\ldots,k\}$. Equation (4.187) implies a recursion that permits a sequential search for the $K$-best paths, starting from $k = 1$. Since $\Theta_k$ is defined after ${}_M(i,n)_k$ has been obtained, the $\min^{(k+1)}$ operation in Eq. (4.187) need not be performed exhaustively. We can accomplish this by devising a cost stack $\mathcal{L}$ of $K$ entries, each of which has a corresponding path specified by the three parameters $\ell, m$ and $k$. The cost stack is sorted and filled as $\Theta_k$ is searched one after another.

Consider when ${}_M(i,n)_k$ is being found according to Eq. (4.187). At that time, the cost stack $\mathcal{L}$ already contains a ranked list of $K$ minimum path costs among all $\zeta(\theta)$ where $\theta \in \{\Theta_{k'}, k' = 1,2,\ldots,k-1\}$. Once ${}_M(i,n)_k$ is determined, $\Theta_k$ is defined accordingly and the cost for each path in $\Theta_k$ is evaluated and compared against the lower $(K-k)$ entries in $\mathcal{L}$. The existing lower $(K-k)$ entries of $\mathcal{L}$ are adjusted, if necessary, because of the new values of $\zeta(\theta)$, $\theta \in \Theta_k$. When the search in $\Theta_k$ is complete, ${}_M(i,n)_{k+1}$ is found as the one with cost ranked $k+1$ in $\mathcal{L}$. The recursion then continues. The algorithm can be summarized as follows.

1. Start from $k = 1$. The best path can be found by the dynamic programming algorithms described previously. The results produced include ${}_m(i, \ell)$ and the associated partial accumulated cost $\zeta_m(i, \ell)$ for all $m = 1, 2, \ldots, M-1$ and $\ell = 1, 2, \ldots, N$. The results are stored for later use. Also, fill $\mathcal{L}$ with scores $\zeta_{M-1}(i, \ell) + \zeta(\ell, n)$, $\ell = 1, 2, \ldots, N$ according to their rank. If $N > K$, keep only the top $K$ scores. The best score $\zeta_M(i, n)$ is of course ranked first.
2. The $k^{\text{th}}$ best path ${}_M(i, n)_k$ is expressed in terms of the concatenated path ${}_{m_{\tilde{k}}}(i, \ell) *_M^{m_{\tilde{k}}+1} (i, n)_{\tilde{k}}$ where $\tilde{k} < k$ and $m_{\tilde{k}}$ is the smallest such that Eq. (4.184) holds. ($m_{\tilde{k}} = M$ for $k = 1$). The expression can be defined by the three essential parameters, $\ell$, $m_{\tilde{k}}$ and $\tilde{k}$. Define the set $\Theta_k$ according to Eq. (4.185). Evaluate the path cost for every path $\theta \in \Theta_k$ according to Eq. (4.183). These scores are compared and merged with the lower $K - k$ entries of $\mathcal{L}$ according to their relative values. (New path costs may enter the stack, pushing existing path costs out of the stack.)
3. Find ${}_M(i, n)_{k+1}$ which has a corresponding cost at the $(k+1)^{\text{th}}$ position in the score stack $\mathcal{L}$ according to Eq. (4.187).
4. Increment $k \to k + 1$ and repeat steps 2 and 3 until $k = K$.

A key innovation in the above search algorithm is the extensive use of the known paths, grown backward from $m = M$ like a tree. The algorithm economically defines the search range via the path set $\Theta_k$ and circumvents the suboptimality issue by making use of the exactly known partial path costs without requiring cost prediction. The latter point is distinct from other efficient one-way search algorithms that rely on cost prediction (before the search propagates) to reduce the search range.

#### 4.7.5.4 Remarks

The multiple path search procedure provides a result that enhances the robustness of the pattern-comparison process. It has many applications in speech recognition, ranging from time alignment to word sequence decoding. The generic description of the $K$-best search algorithms in this chapter serves as a foundation for other applications where some tailoring of the algorithms (e.g., incorporation of higher-level knowledge or constraints on the possible decoding results) is required for specific implementations to suit individual problems.

## 4.8 SUMMARY

In this chapter we have discussed the three basic problems in pattern comparison, namely, (1) how to detect the speech signal in a recording interval (i.e., separate speech from background), (2) how to locally compare spectra from two speech utterances (local spectral distortion measure), and (3) how to globally align and normalize the distance between two speech patterns (sequences of spectral vectors) which may or may not represent the same linguistic sequence of sounds (word, phrase, sentence, etc.). We have shown that a



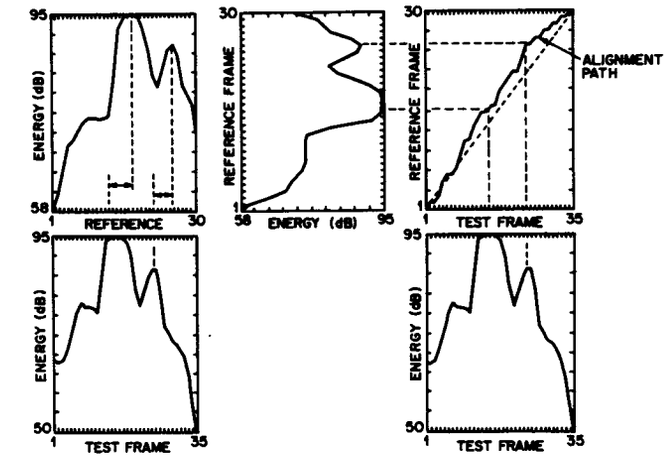



**Figure 4.51** Example illustrating the need for nonlinear time alignment of two versions of a spoken word.

variety of methods exist to either explicitly or implicitly detect locations of speech within recording intervals, a wealth of interesting and useful spectral distortion measures exist for comparing spectral representations, and a well-defined, easily implemented dynamic programming procedure (often called dynamic time warping) exists for time-aligning and normalizing highly variable pairs of speech utterance patterns.

Figure 4.51 illustrates the utility of the methods discussed in this chapter. On the left side of this figure are shown the log energy contours of two versions of a spoken word (the word "seven"). Speech detection has already taken place so that frame 1 corresponds to the first speech frame and frames 30 (for the upper utterance) and 35 (for the lower utterance) correspond to the last speech frame. It can be seen that not only are the durations of these two utterances different, but so are the temporal locations of the vowel peaks (as shown by the misalignment from the dashed lines), even after linearly compensating for the duration differences. The need for time alignment is clear. The right-hand side of the figure shows the dynamic time warping alignment path that best aligns the two versions of the word.

Figure 4.52 shows overlay plots of the log energy contours of the two utterances on a linear scale (top panel curves without time alignment) and after time alignment (bottom panel curves). The middle panel shows the optimal warping or alignment path. It should be clear that after time alignment the log energy contours line up very well in all parts of the contour. This example clearly illustrates the power of the dynamic programming time alignment procedure.

In the next chapter we will see how we can put together the spectral analysis methods of Chapter 3 along with the pattern-comparison methods of this chapter so as to implement a complete speech-recognition system.

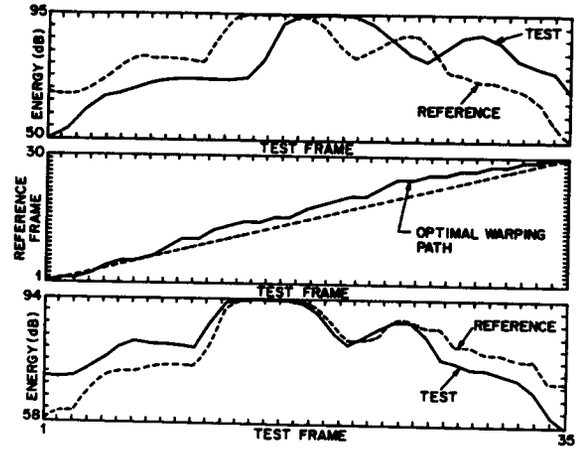



**Figure 4.52** Illustration of the effectiveness of dynamic time warping alignment of two versions of a spoken word.

## REFERENCES


[1] *Transmission Systems for Communications*, Bell Telephone Laboratories, Rev. 4th ed., New Jersey, 1970.

[2] J.G. Wilpon, L.R. Rabiner, and T.B. Martin, "An improved word-detection algorithm for telephone-quality speech incorporating both syntactic and semantic constraints," *AT&T Tech. J.*, 63 (3): 479–498, March 1984.

[3] *Handbook of Applicable Mathematics, Vol. IV: Analysis*, W. Ledermann and S. Vajda, Eds., John Wiley & Sons, ISBN0471101419, 1982.

[4] J.L. Flanagan, *Speech Analysis, Synthesis, and Perception*, 2nd ed., Springer-Verlag, Berlin, 1972.

[5] K.N. Stevens, *The perception of sounds shaped by resonant circuits*, Sc.D. Thesis, Mass. Inst. Tech., Cambridge, Mass., 1952.

[6] J.L. Flanagan and M.G. Saslow, "Pitch discrimination for synthetic vowels," *J. Acoust. Soc. Am.*, 30 (5), 435–442, 1958.

[7] A. Erell, Y. Orgad, and J.L. Goldstein, "JNDs in the LPC poles of speech and their application to quantization of the LPC filter," *IEEE Trans. Signal Proc.*, SP-39 (2): 308–317, February 1991.

[8] R.M. Gray, A. Buzo, A.H. Gray, Jr., and Y. Matsuyama, "Distortion measures for speech processing," *IEEE Trans. Acoustics, Speech, Signal Proc.*, ASSP-28 (4): 367–376, August 1980.

[9] A.H. Gray, Jr. and J.D. Markel, "Distance measures for speech processing," *IEEE Trans. Acoustics, Speech, Signal Proc.*, ASSP-24 (5): 380–391, October 1976.

[10] J.D. Markel and A.H. Gray, Jr., *Linear Prediction of Speech*, Springer-Verlag, Berlin, 1976.





[11] B.H. Juang, L.R. Rabiner, and J.G. Wilpon, "On the use of bandpass liftering in speech recognition," *IEEE Trans. Acoustics, Speech, Signal Proc.*, ASSP-35 (7): 947-954, July 1987.

[12] F. Itakura and S. Saito, "A statistical method for estimation of speech spectral density and formant frequencies," *Electronics and Communications in Japan*, 53A: 36–43, 1970.

[13] S.S. Stevens and J. Volkmann, "The relation of pitch of frequency: A revised scale," *Am. J. Psychol.*, 53: 329–353, 1940.

[14] E. Zwicker, G. Flottorp, and S.S. Stevens, "Critical bandwidth in loudness summation," *J. Acoust. Soc. Am.*, 29: 548–557, 1957.

[15] N.R. French and J.C. Steinberg, "Factors governing the intelligibility of speech sounds," *J. Acoust. Soc. Am.*, 19: 90-119, 1947.

[16] N. Nocerino, F.K. Soong, L.R. Rabiner, and D.H. Klatt, "Comparative study of several distortion measures for speech recognition," *Speech Communication*, 4: 317–331, 1985.

[17] S.B. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Trans. Acoustics, Speech, Signal Proc.*, ASSP-28 (4): 357–366, August 1980.

[18] S. Furui, "On the role of dynamic characteristics of speech spectra for syllable perception," *Fall Meeting of Acoust. Soc. Japan*, 1-1-2: October 1984.

[19] S. Furui, "Speaker Independent Isolated Word Recognition Using Dynamic Features of Speech Spectrum," *IEEE Trans. Acoustics, Speech, Signal Proc.*, ASSP-34 (1): 52–59, February 1986.

[20] H. Sakoe and S. Chiba, "Dynamic programming optimization for spoken word recognition," *IEEE Trans. Acoustics, Speech, Signal Proc.*, ASSP-26 (1): 43–49, February 1978.

[21] R.E. Bellman, *Dynamic Programming*, Princeton University Press, Princeton, New Jersey, USA, 1957.

[22] F. Itakura, "Minimum prediction residual principle applied to speech recognition," *IEEE Trans. Acoustics, Speech, Signal Proc.*, ASSP-23: 57–72, February 1975.

[23] C. Myers, L.R. Rabiner, and A.E. Rosenberg, "Performance tradeoffs in dynamic time warping algorithms for isolated word recognition," *IEEE Trans. Acoustics, Speech, Signal Proc.*, ASSP-28 (6): 623–635, December 1980.

Chapter 5

# SPEECH RECOGNITION SYSTEM DESIGN AND IMPLEMENTATION ISSUES

## 5.1 INTRODUCTION

The last two chapters dealt with several key aspects of the pattern recognition approach to speech recognition—namely, the signal-processing front-end parameter measurement system (Chapter 3) and the issues involved with detecting and comparing speech patterns (Chapter 4).

In particular, it was shown in Chapter 3 that speech parameters are most often obtained using some type of spectral analysis that concentrates on the short-time characteristics of the speech signal. Because of the nonstationary nature of speech utterances, the short-time spectral measurements are performed sequentially over time, producing a sequence of spectral feature vectors, or, as it is usually called, a speech pattern.

Using the basic pattern-recognition approach, the input (unknown class or word) speech pattern is next compared with each class (or word) reference pattern and a measure (score) of similarity between the unknown pattern and each reference pattern is calculated. In Chapter 4 we discussed the three key issues associated with such pattern comparisons—namely, speech detection, a local definition of spectral distortion (dissimilarity) between a pair of short-time spectral representations, and a global definition of pattern similarity that accounts for temporal variations in the rate of speaking through the use of an appropriate dynamic programming technique for time normalization and alignment.

Two key aspects of the pattern-recognition model must still be described to complete the processing: (1) how to create appropriate reference patterns (the "training" problem)





and (2) how to use the resulting set of pattern dissimilarity or distortion scores so as to make the best overall recognition decision for the unknown utterance. This chapter discusses both these topics, along with several related system-design and implementation issues.

The problem of how the set of reference patterns is to be obtained is generally referred to as the training problem. The objective of training is to create, for each utterance class to be recognized, a succinct, consistent pattern representation based on one or more known patterns of each individual class. The representation could be in the form of a pattern itself, which generally is called a "template," or a rich signal model that characterizes the statistical variations of the utterance class. In a strict sense, we discuss only template training in this chapter. For techniques that are considered statistical model based, we will discuss appropriate training methods in Chapter 6.

Traditionally, the notion of a speech "template" pattern has a rather strong connotation of an inflexible temporal structure that requires rigorous time-alignment procedures to obtain a meaningful pattern-comparison result. In the last chapter we cautioned that this rigid temporal structure is only artificially imposed and can be relaxed if the reference representation is constructed in a particular way. The subject of utterance class representation and comparison that does not require explicit, strict time alignment thus deserves special treatment. We begin our presentation in this chapter with a discussion of this subject.

The problem of how to make the best recognition-class decision can also be addressed in several different ways. One very popular approach is the use of the nearest-neighbor rule, which implements a nonparametric solution to the pattern-recognition problem and is applicable in many design cases. A speech recognizer using a multiplicity of reference templates to represent one utterance class often relies on the well-known k-nearest-neighbor (kNN) rule to make the recognition decision. The problem here is that the definition of "nearest neighbor," is far from unique, particularly for speech utterances that do not have uniform duration and thus cannot be straightforwardly analyzed in the traditional vector space framework. The interpretation of the kNN rule in template-based speech-recognizer designs is therefore different from the usual a posteriori probability estimate. Therefore, in this chapter, we discuss several innovative ideas which, based on our understanding of the classical pattern-recognition problem, attempt to combine the decision process and the optimization of reference-pattern parameters, leading to a new approach to speech recognizer training. The approach is categorized as "discriminative training" because its primary objective is to maximize the discriminability of the reference representations for minimum recognition error performance.

The deployment of automatic speech recognizers often requires that the machine have the capability to adapt to new application environments. A topic of particular significance is the adaptation of the reference-pattern parameters to different users (talkers), speaking styles, and acoustic backgrounds. The purpose of reference adaptation is for the recognizer to maintain or improve its performance without requiring a complete redesign of the system. Furthermore, since acoustic as well as electric noises are inevitable interferences in speech processing, the problem of robust recognition system designs that resist signal contamination and undesirable interference is not only analytically interesting but practically important. We shall discuss these two topics separately in the last two sections of the chapter.

## 5.2 APPLICATION OF SOURCE-CODING TECHNIQUES TO RECOGNITION

In communication and information theory, source coding refers to techniques that convert the output signal of an information source into a sequence of binary digits (bits) that are usually transmitted over a communications channel and then used to reproduce the original signal at a different time or location, with an acceptable level of distortion. Specifically, the goal of source coding is to achieve the minimum possible distortion for a prescribed bit rate in the binary digit sequence or, equivalently, to achieve a preset level of distortion at the lowest possible bit rate. To accomplish these goals, complete or nearly complete knowledge of the characteristics of the information source is essential.

Vector quantization is one very efficient source-coding technique. As discussed in Chapter 3, vector quantization is a procedure that encodes a vector of input (e.g., a segment of waveform or a parameter vector that represents the segment spectrum) into an integer (index) that is associated with an entry of a collection (codebook) of reproduction vectors. The reproduction vector chosen is the one that is closest to the input vector in a specified distortion sense. The coding efficiency is obviously achieved in the process of converting the (continuously valued) vector into a compact integer representation, which ranges from, for example, 1 to $N$, with $N$ being the size of (number of entries in) the codebook. The performance of the vector quantizer, however, depends on whether the set of reproduction vectors, which are often called code words, is properly chosen such that the incurred distortion is minimum on average. The block diagram of Figure 3.42 shows a binary split codebook generation algorithm that produces a good codebook based on a given training data set. Characteristics of the information source that produced the given training data are embedded in the codebook.

Source-coding techniques, like vector quantization, can also lead to good classifier designs, besides offering such advantages as reduced storage, reduced computation, and efficient representation of speech sounds. The key idea is that if a source coder is "optimally" designed for a particular source, then it will achieve a lower average distortion for signals generated by the source than any other coder not designed for the particular source. This implies that a certain degree of discriminability is present in the coder design itself. We discuss in detail how source coding techniques can be applied to speech-recognition problems.

### 5.2.1 Vector Quantization and Pattern Comparison Without Time Alignment

For convenience we review the vector quantizer design problem ([1, 2]). Consider a signal source whose output is observed at regularly spaced intervals in time, $t$. We denote the observation sequence at time $t$ by $\mathbf{x}_t$. (As in Chapter 4, $\mathbf{x}_t$ is usually a parameter vector that defines a short time spectrum.) Let $\mathbb{C} = \{\mathbf{y}_i\}_{i=1}^{N}$ be a set of reproduction vectors (code words) and $d(\mathbf{x}_t, \mathbf{y}_i)$ be a prescribed distortion measure between the input $\mathbf{x}_t$ and the code word $\mathbf{y}_i$. A vector quantizer encodes $\mathbf{x}_t$ by the index $i$ of the code word $\mathbf{y}_i \in \mathbb{C}$, which minimizes $d(\mathbf{x}_t, \mathbf{y}_i)$. The design objective of a vector quantizer is to find the best set of code words $\mathbb{C}$ to achieve the minimum expected distortion $E\{d(\mathbf{x}, \mathbf{y}_i)\}$ where $\mathbf{x}$ is considered to



be a random vector.

Practical vector quantizer designs often rely on a large training set $\{\mathbf{x}_t\}_{t=1}^{T}$, since the source distribution is usually unknown. The expected distortion of the source is then replaced by an empirical average distortion over the training set. The generalized Lloyd algorithm (as described in Section 3.4.4) is an effective codebook design procedure that guarantees a good (at least locally optimal) set of code words.

Many vector quantizers are memoryless in the sense that the encoding of the current vector, $\mathbf{x}_t$, is independent of other observations $\mathbf{x}_{t'}$, $t' \neq t$. The codebook $\mathbb{C}$ is designed to minimize

$$D = \frac{1}{T}\sum_{t=1}^{T} d(\mathbf{x}_t, \hat{\mathbf{x}}_t) \tag{5.1}$$

where

$$\hat{\mathbf{x}}_t = \arg\min_{\mathbf{y}_i \in \mathbb{C}} d(\mathbf{x}_t, \mathbf{y}_i). \tag{5.2}$$

We write $D$ as a function of $\mathbb{C}$, $D(\mathbb{C})$, to indicate the dependency of $D$ upon $\mathbb{C}$. The minimized average distortion

$$D_{\min} = \min_{\mathbb{C}} D(\mathbb{C}) \tag{5.3}$$

is an estimate of the best attainable distortion performance for the information source, conditioned on the dimension of $\mathbf{x}_t$ and the size of the codebook $\mathbb{C}, N$. If $D_{\min} = D(\mathbb{C}^\circ)$, one expects that

$$E\{d(\mathbf{x}_t, \hat{\mathbf{x}}_t^\circ)\} < E\{d(\mathbf{x}_t, \hat{\mathbf{x}}_t')\} \tag{5.4}$$

where $\hat{\mathbf{x}}_t^\circ$ and $\hat{\mathbf{x}}_t'$ are the closest code words for $\mathbf{x}_t$ chosen from the optimal codebook $\mathbb{C}^\circ$ and an arbitrary codebook $\mathbb{C}'$, $\mathbb{C}^\circ \neq \mathbb{C}'$, respectively. If the source is ergodic and the size of the training set $T \to \infty$, Eq. (5.4) can be shown to be always valid.

The concept of vector quantization can be easily applied to speech-recognizer designs. Suppose there are $M$ utterance classes (e.g., words, phrases) to be recognized. Each utterance class can be considered an information source. We thus collect $M$ sets of training data $\{\mathbf{x}_t^{(i)}\}$ where $i = 1, 2, \ldots, M$ is the class index. Each training set should contain a number of utterances of the same class. $M$ codebooks $\{\mathbb{C}^{(i)}\}_{i=1}^{M}$ are then designed using the minimum average distortion objective for the $M$ information sources, respectively. Each codebook represents a characterization of the information source (class).

During the recognition operation, the $M$ codebooks are used to implement $M$ distinct vector quantizers as shown in Figure 5.1. An unknown utterance $\{\mathbf{x}_t\}_{t=1}^{T_u}$ is vector-quantized by all $M$ quantizers, resulting in $M$ average distortion scores $D(\mathbb{C}^{(i)})$, $i = 1, 2, \ldots, M$, where

$$D(\mathbb{C}^{(i)}) = \frac{1}{T_u}\sum_{t=1}^{T_u} d(\mathbf{x}_t, \hat{\mathbf{x}}_t^{(i)}) \tag{5.5}$$

with $\hat{\mathbf{x}}_t^{(i)} \in \mathbb{C}^{(i)}$ satisfying

$$\hat{\mathbf{x}}_t^{(i)} = \arg\min_{\mathbf{y}_j^{(i)} \in \mathbb{C}^{(i)}} d(\mathbf{x}_t, \mathbf{y}_j^{(i)}). \tag{5.6}$$

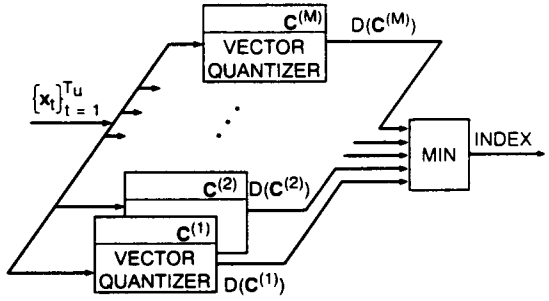



**Figure 5.1** A vector-quantizer–based speech-recognition system.

The utterance is recognized as class $k$ if

$$D(\mathbb{C}^{(k)}) = \min_i D(\mathbb{C}^{(i)}). \tag{5.7}$$

Referring to the block diagram of the pattern recognition based approach to speech recognition (Figure 2.37), we see that the $M$ codebooks are analogous to $M$ (sets of) reference patterns (or templates) and the dissimilarity measure is defined according to Eq. (5.5) and Eq. (5.6), where no explicit time alignment is required. This is distinct from the sequence-matching approach discussed in Chapter 4 that relies on dynamic programming techniques to calculate an accumulated distortion score over an *optimized alignment path*. The ability to discriminate $M$ utterance classes, with the present source coding technique, is based upon Eq. (5.4), where the expected value of the distortion is replaced by an average distortion over the utterance length. For a recognition vocabulary with words that can only be distinguished by their temporal sequential characteristics, such as "car" and "rack," or for complex speech classes that encompass long utterances with rich phonetic contents, the above simple memoryless vector quantizer method is generally not adequate and does not provide satisfactory recognition performance. However, for simple vocabularies of highly distinct words, such as the English digits, this simple method of recognition has proven quite effective.

### 5.2.2 Centroid Computation for VQ Codebook Design

The vector quantization approach to speech recognition requires that the codebook for a particular utterance class be properly designed to minimize the average distortion. The design algorithm of Lloyd for codebook generation, based on a training set, can be briefly summarized as an iterative procedure consisting of the following two steps (see Section 3.4.4):

a. Minimum distortion labeling: For each input vector $\mathbf{x}_t$, find $\hat{\mathbf{x}}_t = \mathbf{y}_j$, where $\mathbf{y}_j$ is an entry of the given codebook $\mathbb{C}$, satisfying Eq. (5.2); group training vectors according to their associated code word indices.



b. Centroid computation: For each group of vectors with the same index label, compute the new centroid which minimizes the average distortion for the members of the group (when they are reproduced by the centroid vector).

The above two steps iterate until some convergence criterion is met. The first step is straightforward as it involves only evaluation of the distortions. The second step, centroid computation, which is, in itself, an optimization problem is the focus of this section.

Consider a set of vectors $\{\mathbf{x}_i\}_{i=1}^L$ and a distortion measure, $d(\mathbf{x}, \mathbf{y})$. Without loss of generality, we assume that these vectors are assigned to the same group label (or code word) in the context of vector quantization. The centroid of $\{\mathbf{x}_i\}_{i=1}^L$ is defined as the vector $\bar{\mathbf{y}}$ that minimizes the average distortion; that is,

$$\bar{\mathbf{y}} \triangleq \arg\min_{\mathbf{y}} \frac{1}{L} \sum_{i=1}^{L} d(\mathbf{x}_i, \mathbf{y}). \tag{5.8}$$

The solution to the centroid problem, obviously, is highly dependent on the choice of the distortion measure. When $\mathbf{x}_i$ and $\mathbf{y}$ are vectors, $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{iK})$ and $\mathbf{y} = (y_1, y_2, \ldots, y_K)$, measured in a $K$-dimensional space with the $L_2$ norm (Euclidean distance), the centroid obviously is the mean of the vector set,

$$\bar{\mathbf{y}} = \frac{1}{L} \sum_{i=1}^{L} \mathbf{x}_i. \tag{5.9}$$

The centroid solution of Eq. (5.9) also applies to the case of weighted Euclidean distances (i.e., Mahalanobis distances). Another well-known result related to the $L_1$ distance (sometimes called the city block distance)

$$d(\mathbf{x}_i, \mathbf{y}) = \sum_{k=1}^{K} |x_{ik} - y_k| \tag{5.10}$$

is that the centroid $\bar{\mathbf{y}}$ is the median vector of $\{\mathbf{x}_i\}_{i=1}^L$. (In this case, each dimension is customarily treated independently, meaning that each $y_k$ is the median value of $\{x_{ik}\}_{i=1}^L$ respectively.)

As discussed in Chapter 4, $\mathbf{x}$ and $\mathbf{y}$ often represent speech spectra and $d(\mathbf{x}, \mathbf{y})$ is a spectral-distortion measure. Many spectral-distortion measures introduced in Chapter 4 are not as straightforward as the $L_p$ norm, and the solution to the centroid problem may be more involved than the previous cases. We elaborate in the following the centroid solutions for several popular spectral-distortion measures.

#### 5.2.2.1 Likelihood Distortions

The family of likelihood distortions includes the Itakura-Saito distortion (Eq. (4.39)), the Itakura distortion (Eq. (4.52)), and the likelihood ratio distortion (Eq. (4.53)).

Let $\{S_i(\omega)\}_{i=1}^L$ be the set of spectra, for which the centroid, or mean spectrum, is to be

found. The average Itakura-Saito distortion from each of the given spectra to an arbitrary $p^{\text{th}}$ order all-pole model spectrum $\sigma^2/|A(e^{j\omega})|^2$ is

$$D_{\text{IS}} = \frac{1}{L}\sum_{i=1}^{L} d_{\text{IS}}\left(S_i, \frac{\sigma^2}{|A|^2}\right) = \frac{1}{\sigma^2}\int_{-\pi}^{\pi}\left[\frac{1}{L}\sum_{i=1}^{L} S_i(\omega)\right]|A(e^{j\omega})|^2\frac{d\omega}{2\pi}$$
$$-\frac{1}{L}\sum_{i=1}^{L}\log\sigma_{\infty,i}^2 + \log\sigma^2 - 1 \tag{5.11}$$

where $\sigma_{\infty,i}^2$ is the one-step prediction error of $S_i(\omega)$. Note that

$$\frac{1}{\sigma^2}\int_{-\pi}^{\pi}\left[\frac{1}{L}\sum_{i=1}^{L} S_i(\omega)\right]|A(e^{j\omega})|^2\frac{d\omega}{2\pi} = \mathbf{a}^t\left[\frac{1}{L}\sum_{i=1}^{L}\mathbf{R}_i\right]\mathbf{a}/\sigma^2 \tag{5.12}$$

where $\mathbf{R}_i$ is a $(p+1)\times(p+1)$ autocorrelation matrix associated with $S_i(\omega)$, $\mathbf{a}$ is the coefficient vector of $A(z)$, and $t$ denotes the matrix transpose. It is easily shown that the centroid model spectrum has a filter coefficient vector $\bar{\mathbf{a}}$ obtained as

$$\bar{\mathbf{a}} = \arg\min_{\mathbf{a}}\left(\mathbf{a}^t\left[\frac{1}{L}\sum_{i=1}^{L}\mathbf{R}_i\right]\mathbf{a}\right) \tag{5.13}$$

that is, the solution for $\bar{\mathbf{a}}$ is equivalent to solving the LPC normal equations defined by the average autocorrelation $\Sigma\mathbf{R}_i/L$. (Computationally, constant scaling of the autocorrelations does not alter the optimal predictor coefficient solution, and therefore the division by $L$ is not essential for finding $\bar{\mathbf{a}}$.) Similar to Eq. (4.41), the gain term of the centroid model spectrum is

$$\bar{\sigma}^2 = \min_{\mathbf{a}}\left(\mathbf{a}^t\left[\frac{1}{L}\sum_{i=1}^{L}\mathbf{R}_i\right]\mathbf{a}\right)$$
$$= \bar{\mathbf{a}}^t\left[\frac{1}{L}\sum_{i=1}^{L}\mathbf{R}_i\right]\bar{\mathbf{a}}. \tag{5.14}$$

The average distortion is minimized by the centroid model spectrum $\bar{\sigma}^2/|\bar{A}|^2$ with

$$(D_{\text{IS}})_{\min} = \frac{1}{L}\sum_{i=1}^{L} d_{\text{IS}}\left(S_i, \bar{\sigma}^2/|\bar{A}|^2\right)$$
$$= -\frac{1}{L}\sum_{i=1}^{L}\log\sigma_{\infty,i}^2 + \log\bar{\sigma}^2. \tag{5.15}$$

The average likelihood ratio distortion from a set of $p^{\text{th}}$ order unity gain model spectra $\{1/|A_i(e^{j\omega})|^2\}_{i=1}^{L}$ to an arbitrary unity gain all-pole model spectrum $1/|A(e^{j\omega})|^2$ is defined



by

$$D_{\text{LR}} = \frac{1}{L}\sum_{i=1}^{L} d_{\text{LR}}\left(\frac{1}{|A_i|^2}, \frac{1}{|A|^2}\right)$$
$$= \frac{1}{L}\sum_{i=1}^{L}\frac{\mathbf{a}^t\mathbf{R}_i\mathbf{a}}{\sigma_i^2} - 1 \tag{5.16}$$

where $\mathbf{R}_i$ and $\mathbf{a}$ are defined as before and $\sigma_i^2$ is the minimum $p^{\text{th}}$ order residual energy associated with $\mathbf{R}_i$. The model spectrum $1/|A_i|^2$ corresponds to the normalized autocorrelation matrix $\mathbf{R}_i/\sigma_i^2$. The evaluation of the centroid model spectrum $1/|\bar{A}|^2$ becomes that of finding the LPC solution to the set of normal equations defined by the average residual-normalized autocorrelations. Specifically,

$$\bar{\mathbf{a}} = \arg\min_{\mathbf{a}}\left(\frac{1}{L}\sum_{i=1}^{L}\frac{\mathbf{a}^t\mathbf{R}_i\mathbf{a}}{\sigma_i^2}\right) \tag{5.17}$$

and

$$(D_{\text{LR}})_{\min} = \bar{\mathbf{a}}^t\left[\frac{1}{L}\sum_{i=1}^{L}\frac{\mathbf{R}_i}{\sigma_i^2}\right]\bar{\mathbf{a}} - 1. \tag{5.18}$$

The centroid solution for the Itakura distortion is difficult to find, nevertheless. (Recall that the Itakura distortion is the gain optimized version of the Itakura-Saito distortion (see Eqs. (4.51) and (4.52)).) The average distortion, in this case, is defined by

$$D_{\text{I}} = \frac{1}{L}\sum_{i=1}^{L} d_{\text{I}}(S_i, \alpha_i/|A|^2)$$
$$= \frac{1}{L}\sum_{i=1}^{L}\log\left[\frac{1}{\sigma_i^2}\int_{-\pi}^{\pi} S_i(\omega)|A(e^{j\omega})|^2\frac{d\omega}{2\pi}\right]$$
$$= \frac{1}{L}\sum_{i=1}^{L}\log\left(\frac{\alpha_i}{\sigma_i^2}\right) \tag{5.19}$$

where

$$\alpha_i = \int_{-\pi}^{\pi} S_i(\omega)|A(e^{j\omega})|^2\frac{d\omega}{2\pi} \tag{5.20}$$

and $\sigma_i^2$ is again the minimum $p^{\text{th}}$ order prediction residual energy of $S_i(\omega)$. Because of the logarithm in Eq. (5.19), we now need to minimize a geometric mean instead of an arithmetic mean, a difficult problem that cannot be as straightforwardly solved as the previous cases. Since an arithmetic mean is an upper bound of the geometric mean, however, it is possible to use the centroid model spectrum as obtained in Eq. (5.17) (from minimizing an arithmetic mean) to approximate the desired centroid without causing gross deviations in distortion evaluation. If the individual spectra in the set are close to each other, as in clustering,

such an approximation is quite reasonable because the two distortion measures are almost identical when they are small (see Eq. (4.54)).

#### 5.2.2.2 COSH Distortion

The COSH distortion is defined by Eq. (4.61) as

$$d_{\text{COSH}}(S_i, S) = \frac{1}{2}\int_{-\pi}^{\pi}\left[\frac{S_i(\omega)}{S(\omega)} + \frac{S(\omega)}{S_i(\omega)}\right]\frac{d\omega}{2\pi} - 1. \tag{5.21}$$

The average distortion is thus

$$\begin{aligned} D_{\text{COSH}} &= \frac{1}{L}\sum_{i=1}^{L} d_{\text{COSH}}(S_i, S) \\ &= \frac{1}{2}\int_{-\pi}^{\pi}\left[\frac{\frac{1}{L}\sum_{i=1}^{L} S_i(\omega)}{S(\omega)} + \frac{1}{L}\sum_{i=1}^{L}\frac{S(\omega)}{S_i(\omega)}\right]\frac{d\omega}{2\pi} - 1. \end{aligned} \tag{5.22}$$

Taking the derivative of $D_{\text{COSH}}$ with respect to $S(\omega)$ and setting it to zero, we obtain

$$\frac{dD_{\text{COSH}}}{dS(\omega)} = \frac{1}{2}\int_{-\pi}^{\pi}\left[-\frac{\frac{1}{L}\sum_{i=1}^{L} S_i(\omega)}{S^2(\omega)} + \frac{1}{L}\sum_{i=1}^{L}\frac{1}{S_i(\omega)}\right]\frac{d\omega}{2\pi} = 0, \tag{5.23}$$

which has a solution at $\overline{S}(\omega)$

$$\overline{S}(\omega) = \left\{\left[\frac{1}{L}\sum_{i=1}^{L} S_i(\omega)\right] \Big/ \left[\frac{1}{L}\sum_{i=1}^{L} S_i^{-1}(\omega)\right]\right\}^{1/2}. \tag{5.24}$$

(Since $S(\omega)$ is a power spectrum, we retain only the positive solution.) This is the centroid solution for the COSH distortion case with

$$(D_{\text{COSH}})_{\min} = \int_{-\pi}^{\pi}\left\{\left[\frac{1}{L}\sum_{i=1}^{L} S_i(\omega)\right]\left[\frac{1}{L}\sum_{i=1}^{L} S_i^{-1}(\omega)\right]\right\}^{1/2}\frac{d\omega}{2\pi} - 1. \tag{5.25}$$

#### 5.2.2.3 Cepstral Distance

The centroid problem for the truncated cepstral distance, as defined by Eq. (4.24), Eq. (4.26), or Eq. (4.37), has a straightforward solution, which is simply the mean of the cepstral vectors. This is obvious because the truncated cepstral distances are Euclidean distances. The centroid spectrum for the untruncated cepstral distance case, however, has a particular frequency domain interpretation that is worth noting. The cepstral distance is defined by

$$d_2^2(S_i, S) = \int_{-\pi}^{\pi} |\log S_i(\omega) - \log S(\omega)|^2 \frac{d\omega}{2\pi}. \tag{5.26}$$



The average cepstral distance is then

$$\begin{aligned} D_2 &= \frac{1}{L}\sum_{i=1}^{L} d_2^2(S_i, S) \\ &= \int_{-\pi}^{\pi}\frac{1}{L}\sum_{i=1}^{L}|\log S_i(\omega) - \log S(\omega)|^2 \frac{d\omega}{2\pi} \end{aligned} \tag{5.27}$$

which is minimized by $\overline{S}(\omega)$

$$\overline{S}(\omega) = \left[\prod_{i=1}^{L} S_i(\omega)\right]^{1/L} \tag{5.28}$$

the geometric mean spectrum.

#### 5.2.2.4 Other Distortions

The centroid problem for many other popular distortion measures cannot be as easily solved for as in the previous cases. This is particularly true for likelihood-based distortion measures with nonuniform frequency weighting. However, the centroid computation remains tractable by convex programming techniques for a wide range of general distortion measures. The treatment of this problem is beyond the scope of this book.

**Exercise 5.1**

1. Consider using a Euclidean distance of the form:

$$d(\mathbf{x}_i, \mathbf{y}) = (\mathbf{x}_i - \mathbf{y})^t(\mathbf{x}_i - \mathbf{y}).$$

Show that the centroid, $\bar{\mathbf{y}}$, of $\{\mathbf{x}_i\}_{i=1}^{L}$ satisfies Eq. (5.9).

2. If we now consider a weighted distance of the form:

$$d(\mathbf{x}_i, \mathbf{y}) = (\mathbf{x}_i - \mathbf{y})^t\mathbf{W}(\mathbf{x}_i - \mathbf{y})$$

where $\mathbf{W}$ is a fixed, positive definite matrix, how does the centroid $\bar{\mathbf{y}}$ of $\{\mathbf{x}_i\}_{i=1}^{L}$ change?

**Solution 5.1**

1. The centroid is defined as the vector that minimizes the average distortion

$$D = \frac{1}{L}\sum_{i=1}^{L} d(\mathbf{x}_i, \mathbf{y}).$$

Using a Euclidean distance, we have

$$D = \frac{1}{L}\sum_{i=1}^{L}(\mathbf{x}_i - \mathbf{y})^t(\mathbf{x}_i - \mathbf{y}).$$

Differentiating $D$ with respect to $\mathbf{y}$, we obtain the following set of equations for the centroid

$$\frac{\partial D}{\partial \mathbf{y}} = -\frac{2}{L}\sum_{i=1}^{L}(\mathbf{x}_i - \mathbf{y}) = 0$$

which gives

$$\bar{\mathbf{y}} = \frac{1}{L}\sum_{i=1}^{L} \mathbf{x}_i.$$

2. Using a weighted distance, we have

$$D = \frac{1}{L}\sum_{i=1}^{L} (\mathbf{x}_i - \mathbf{y})^t \mathbf{W} (\mathbf{x}_i - \mathbf{y}).$$

Again, the equation for the centroid is

$$\frac{\partial D}{\partial \mathbf{y}} = -\frac{2}{L}\sum_{i=1}^{L} \mathbf{W}(\mathbf{x}_i - \mathbf{y})$$

$$= -2\mathbf{W}\left(\left(\frac{1}{L}\sum_{i=1}^{L} \mathbf{x}_i\right) - \mathbf{y}\right) = 0.$$

Since $\mathbf{W}$ is positive definite, the solution to the above equation is thus

$$\bar{\mathbf{y}} = \frac{1}{L}\sum_{i=1}^{L} \mathbf{x}_i$$

i.e., the mean vector of $\{\mathbf{x}_i\}_{i=1}^{L}$. The weight matrix *does not* change the centroid solution.

**Exercise 5.2**

Consider the centroid problem in a one-dimensional space using an $L_1$ distance; i.e., if $x$ and $y$ are two points on the real line,

$$d(x, y) = |x - y|.$$

Given a set of points $\{x_i\}_{i=1}^{L}$, find the centroid, $\bar{y}$, that minimizes the average $L_1$ distance

$$D = \frac{1}{L}\sum_{i=1}^{L} |x_i - y|.$$

**Solution 5.2**

We reorder $x_i$, $i = 1, 2, \ldots, L$ such that

$$x_{(1)} < x_{(2)} < \cdots < x_{(L)}$$

where the parenthesized index refers to the rank of the given sample point. Suppose $y$ lies between $x_{(l)}$ and $x_{(l+1)}$, i.e.

$$x_{(l)} < y < x_{(l+1)}.$$

Then,

$$D = \frac{1}{L}\left\{\sum_{i=1}^{l} (y - x_{(i)}) + \sum_{i=l+1}^{L} (x_{(i)} - y)\right\}.$$



The derivative of $D$, with respect to $y$, is a piecewise constant function,

$$\frac{\partial D}{\partial y} = \frac{1}{L}(l - L + l) = \frac{1}{L}(2l - L), \qquad \text{for } x_{(l)} < y < x_{(l+1)}$$

which is zero at $l = L/2$. That is, the centroid, $\bar{y}$, is a value that is greater than half of the sample points and less than the other half of the sample points. Thus, the centroid solution, using an $L_1$ distance, is the median value.

More specifically, the centroid, $\bar{y}$, has the value

$$\bar{y} + \frac{1}{2}\left(x_{(\frac{L}{2})} + x_{(\frac{L}{2}+1)}\right), \quad \text{when } L \text{ is even}$$

and

$$\bar{y} = x_{\left(\frac{L+1}{2}\right)} \quad \text{when } L \text{ is odd.}$$

**Exercise 5.3**

Show that when a cepstral distortion measure is used to measure spectral dissimilarity, the centroid spectrum for a set of spectra $\{S_i(\omega)\}_{i=1}^{L}$ is the geometric mean spectrum as shown in Eq. (5.28).

**Solution 5.3**

The cepstral distortion is an $L_2$ norm measured on the log spectra; i.e.

$$d_2^2(S_i, S) = \int_{-\pi}^{\pi} |\log S_i(\omega) - \log S(\omega)|^2 \frac{d\omega}{2\pi}.$$

Therefore the average cepstral distortion is

$$D_2 = \int_{-\pi}^{\pi} \frac{1}{L}\sum_{i=1}^{L} |\log S_i(\omega) - \log S(\omega)|^2 \frac{d\omega}{2\pi}.$$

Note that $S_i(\omega)$ and $S(\omega)$ are power spectra and thus nonnegative, and that the log spectra are real valued and uniquely defined. Then, to minimize $D_2$, in order to find the centroid spectrum, we can differentiate $D_2$ with respect to $\log S(\omega)$, obtaining

$$\frac{\partial D_2}{\partial \log S(\omega)} = \int_{-\pi}^{\pi} -\frac{2}{L}\sum_{i=1}^{L} [\log S_i(\omega) - \log S(\omega)] \frac{d\omega}{2\pi}$$

which is zero at

$$\log \bar{S}(\omega) = \frac{1}{L}\sum_{i=1}^{L} \log S_i(\omega) = \log \left[\prod_{i=1}^{L} S_i(\omega)\right]^{1/L}.$$

Therefore,

$$\bar{S}(\omega) = \prod_{i=1}^{L} S_i(\omega).$$

### 5.2.3 Vector Quantizers with Memory

For utterances of limited duration (e.g., words), the phonetic content of each utterance class is usually quite different from that of any other utterance class. For such cases, memoryless vector quantizers, as described in Section 5.2.1, provide effective discriminability as well as computational simplicity. When the utterances are long enough to cause significant overlap in phonetic content among different utterance classes, or the sequential (temporal) characteristics of the utterance are the only distinguishing factor in recognition, simple memoryless vector quantizers are generally not adequate for acceptable recognition performance. One remedy to this problem is to use vector quantizers with memory so as to capture the temporal characteristics of the utterances in a particular manner.

A matrix quantizer that encodes several vectors simultaneously is a straightforward extension of the memoryless vector quantizer. Matrix quantizers, in their very simplest form, can be designed using the same Lloyd algorithm as for vector quantizers. If $n$ spectra are encoded at the same time, the codebook $\mathbb{C} = \{\mathbf{Y}_i\}_{i=1}^{N}$ is then designed to minimize

$$D = \frac{1}{T-n+1} \sum_{t=1}^{T-n+1} d'(\mathbf{X}_t, \hat{\mathbf{X}}_t) \tag{5.29}$$

where

$$\mathbf{X}_t = (\mathbf{x}_t, \mathbf{x}_{t+1}, \ldots, \mathbf{x}_{t+n-1}) \tag{5.30}$$

which is a sequence of spectral vectors (and thus a "matrix") and

$$\hat{\mathbf{X}}_t = \arg\min_{\mathbf{Y}_i \in \mathbb{C}} d'(\mathbf{X}_t, \mathbf{Y}_i). \tag{5.31}$$

Each code word $\mathbf{Y}_i \in \mathbb{C}$ accordingly has $nK$ dimensions (where $K$ is the dimensionality of each $\mathbf{x}_t$), $\mathbf{Y}_i = (\mathbf{y}_{i1}, \mathbf{y}_{i2}, \ldots, \mathbf{y}_{in})$, and the distortion $d'$ is often defined for simplicity by

$$d'(\mathbf{X}_t, \mathbf{Y}_i) = \frac{1}{n} \sum_{j=1}^{n} d(\mathbf{x}_{t+j-1}, \mathbf{y}_{ij}). \tag{5.32}$$

Simultaneous encoding of a sequence of spectral vectors, as defined by Eq. (5.31), implies that the code words $\mathbf{Y}_i$ have certain embedded block memory constraints. The iterative procedure of Lloyd is readily applicable to the matrix quantizer design since the above equations are essentially the same as those in the memoryless case. The only differences are that minimum distortion labeling applies to a sequence of spectra and that the centroid computation (a matrix of $n$ spectral vectors) now involves finding $n$ separate spectral centroids for each code word.

Another important vector quantizer with memory is the class of trellis vector quantizers in which the interdependence in the sequence of input spectra is formulated in a transition structure represented by a trellis. More precisely, a trellis vector quantizer is a finite state vector quantizer, specified by a finite state space $\mathcal{Q}$, an initial state $q_0$, and three functions: (1) an encoder $\alpha$: $\mathcal{A} \times \mathcal{Q} \to \mathcal{N}$ where $\mathcal{A}$ denotes the space of spectral observations and $\mathcal{N}$ is the index set; (2) a next state function or transition function $f : \mathcal{Q} \times \mathcal{N} \to \mathcal{Q}$; and (3) a decoder $\beta : \mathcal{Q} \times \mathcal{N} \to \hat{\mathcal{A}}$ where $\hat{\mathcal{A}}$ is the space of reproduction spectral vectors



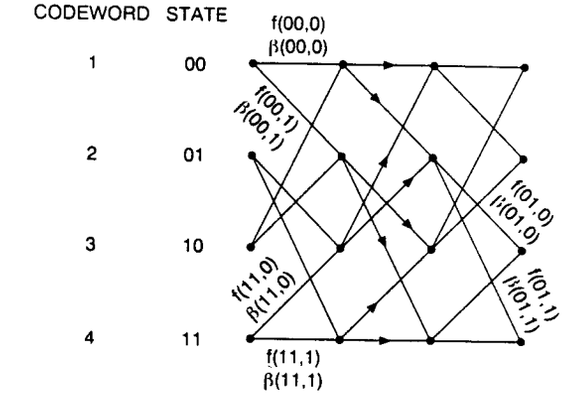



**Figure 5.2** A trellis quantizer as a finite state machine.

(i.e., code words). During encoding, the encoder $\alpha$ assigns the input $\mathbf{x}_t \in \mathcal{A}$ to a code word with index $u_t \in \mathcal{N}$ based on the current state $q_t$; i.e. $u_t = \alpha(\mathbf{x}_t, q_t)$. The state advances according to $q_{t+1} = f(q_t, u_t)$. The decoder $\beta$, upon receiving $u_t$, reconstructs $\mathbf{x}_t$ by $\hat{\mathbf{x}}_t$ based on the function $\hat{\mathbf{x}}_t = \beta(q_t, u_t)$. A trellis structure is shown in Figure 5.2 to illustrate the trellis transition mechanism. In the particular example, there is a direct correspondence between the code words and the states, and the transition follows a first-order Markov chain, meaning that encoding of the current spectrum depends only on the last state. For each given state (code word), there is a specification describing which code words are allowed to follow the particular state. For example, in Figure 5.2, only code words 1 and 2 can follow code word 1. The encoder always has a copy of the decoder so as to achieve the minimum distortion requirement:

$$u_t = \alpha(\mathbf{x}_t, q_t) = \arg\min_{u \in \mathcal{N}} d(\mathbf{x}_t, \beta(q_t, u)). \tag{5.33}$$

The codebook and the next state function are designed to minimize

$$D = \frac{1}{T} \sum_{t=1}^{T} d(\mathbf{x}_t, \beta(q_t, \alpha(\mathbf{x}_t, q_t))). \tag{5.34}$$

Note that for a given next state function $f$, Eq. (5.34) is similar to Eq. (5.1) and Eq. (5.29) and the code words can be designed using the Lloyd algorithm based on the optimal labeling and centroid computation principle. For the design of the next state function, a transition pruning procedure can be employed. The pruning method starts with a regular memoryless VQ codebook design. It then constructs the trellis, which defines the next state function in terms of allowable search range of the code words, by pruning nonessential transitions. Nonessential transitions are defined as those transitions that can be replaced by an alternate transition with a minimum degradation in distortion performance. For detailed description of the design algorithm, the reader should consult Reference [3]. The memory structure of

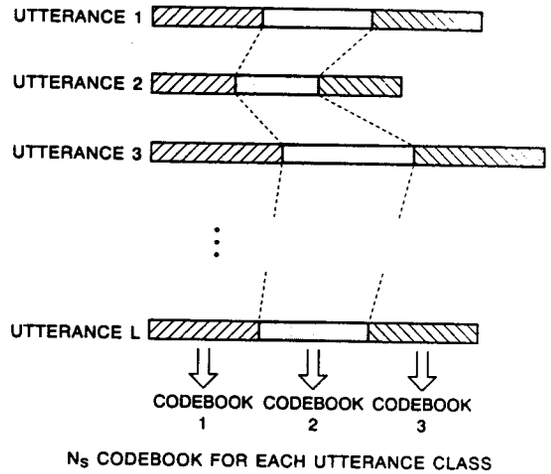



**Figure 5.3** Codebook training for segmental vector quantization.

the information source is embedded in the way the trellis constraints are constructed. This is considerably different from the block constraints in the matrix quantizer case.

There are other source coder designs that allow incorporation of memory constraints. Use of these coders in a speech recognizer can lead to improved recognition performance, because of the addition of the temporal characterization of the source.

### 5.2.4 Segmental Vector Quantization

The standard (memoryless) vector-quantization approach that uses a single vector quantizer for the entire duration of the utterance for each class is not designed to preserve the sequential characteristics of the utterance class. This lack of explicit characterization of the sequential behavior can be remedied by treating each utterance class as a concatenation of several, say $N_s$, information subsources, each of which is represented by a VQ codebook. We call this segment-specific VQ approach "segmental vector quantization." For an utterance $\{\mathbf{x}_t\}_{t=1}^{T}$, the simplest (but not necessarily the most meaningful) way to decompose it into a concatenation of $N_s$ information subsources is to equally divide the utterance into $N_s$ segments $\{\mathbf{x}_t\}_{t=1}^{T/N_s}$, $\{\mathbf{x}_t\}_{t=T/N_s+1}^{2T/N_s}, \ldots$ and so on. This simple (linear) segmentation scheme is illustrated in Figure 5.3. Other, more sophisticated, segmentation schemes obviously are possible. Given a set of training utterances of the known class, $N_s$ sets of training data are formed and used to design $N_s$ codebooks. These $N_s$ codebooks have an implicit temporal order because they correspond to different portions of the utterances. Similar to the single codebook case, each set of $N_s$ successive codebooks represents one class, and the average distortion incurred in encoding an unknown utterance with the corresponding successive vector quantizers is the discriminant score for the recognition decision.



Segmental vector quantization requires virtually the same computational complexity as the previous utterance-based VQ as long as the codebook size is the same. The only complexity increase is in the codebook storage; segmental VQ has $N_s$ codebooks while utterance-based VQ has only one for each utterance class. The preserved sequential relationship in the form of codebook concatenation, however, often proves remarkably beneficial in speech-recognition tasks. (We will return to this problem of how to exploit the ideas of sequential VQ in Chapter 6 in our discussion of the hidden Markov model approach to speech recognition.)

### 5.2.5 Use of a Vector Quantizer as a Recognition Preprocessor

Vector quantization, as discussed above, offers computational simplicity in speech-recognition applications. In some restricted cases, good recognition performance can be obtained with straightforward use of VQ as a recognizer. For example, a word-recognition accuracy of 99% was reported [1] with word-based vector quantization for speaker-trained isolated word recognition of a highly nonconfusable 20–word vocabulary. For speaker-independent word recognition, however, the same implementation achieved only 88% accuracy for the same vocabulary. Although one can further improve the performance of VQ-based recognizers by incorporating temporal information in the VQ design (e.g., segmental VQ as introduced in Section 5.2.4), the advantage of low computational complexity that the VQ-based approach is able to offer may be better applied in alternative recognition configurations. One such case is to use VQ as a preprocessor to screen out word candidates that are obviously unlikely to match the unknown utterance, thereby reducing the computational requirements of a dynamic time alignment–based processor in the latter stage of the recognition process.

A block diagram that shows how a VQ-based recognizer can be used as a preprocessor in an isolated word recognition system is given in Figure 5.4. The system consists of three major blocks. The signal processing block that analyzes the incoming speech signal and produces a sequence of spectral vectors is common in virtually every recognition system. The preprocessor block is a VQ-based speech recognizer, with a decision logic that provides a list of tentative word candidates for further processing. In the preprocessor, a set of word-based (utterance-based) VQ codebooks is used to characterize individual vocabulary words (utterance classes). The unknown spectral sequence is vector-quantized, frame by frame, by each individual codebook, and the resultant set of average distortions is then sent to the decision logic. When the decision logic chooses only one single word candidate for further processing, the decision is adopted as the final recognition result. (This happens when the minimum average distortion is significantly smaller than all other average distortion scores.) When several candidate words are considered likely by the preprocessor, they are then passed to the postprocessor block where a DTW processor performs a rigorous time-alignment procedure, aiming at critical discrimination among these potential candidates. The postprocessor can be further simplified by vector quantizing the reference patterns (see Section 5.2.6) with the same corresponding VQ codebooks stored in the preprocessor. Since the unknown spectral sequence has been vector quantized in the preprocessor and all

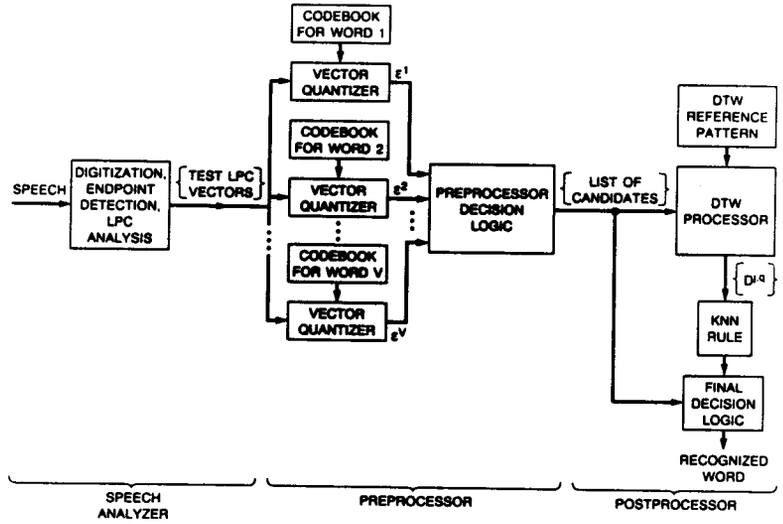



**Figure 5.4** Block diagram of isolated word recognizer incorporating a word-based VQ preprocessor and a DTW-based postprocessor (after Pan et al. [4]).

the frame distortions have been computed, the DTW process thus requires only a simple table lookup.

To understand how the preprocessor works, consider the following. We denote the average distortion, resulting from encoding the unknown utterance with the codebook for the $i^{\text{th}}$ word (utterance class), by $D(\mathbb{C}^{(i)})$. We define words $i_1$ and $i_2$ as the words whose average distortions are the two smallest among $D(\mathbb{C}^{(i)})$, $i = 1, 2, \ldots, M$. ($M$ is the vocabulary size.) That is

$$i_1 = \arg \min_{1 \le i \le M} D(\mathbb{C}^{(i)}) \tag{5.35}$$

and

$$i_2 = \arg \min_{\substack{1 \le i \le M \\ i \ne i_1}} D(\mathbb{C}^{(i)}). \tag{5.36}$$

The decision logic of the preprocessor consists of two decision rules:

**Rule 1 (Final Word Candidate Rule)**

$$\text{Choose word } i, \text{ iff } D(\mathbb{C}^{(i_1)}) \le D' \tag{5.37}$$

$$\text{and } D(\mathbb{C}^{(i_2)}) - D(\mathbb{C}^{(i_1)}) \ge D'' \tag{5.38}$$

where $D'$ and $D''$ are two distortion thresholds, empirically chosen to test the minimum average distortion value and the separation between the best and the second best candidates,



respectively. Rule 1 indicates that if the minimum average distortion for the best candidate is sufficiently small and there is no other obvious competing candidate, the decision made by the preprocessor is adopted as the final word-recognition result and no subsequent DTW processing is required. If Rule 1 is not satisfied, the decision logic passes to the postprocessor a list of valid candidates according to Rule 2:

**Rule 2 (Valid Candidate Rule)** Word $i$, $i = 1, 2, \ldots, M$, is considered a valid candidate and is passed to the postprocessor for further processing if

$$D(\mathbb{C}^{(i)}) - D(\mathbb{C}^{(i_1)}) \le D^{\circ} \tag{5.39}$$

where $D^{\circ}$ is an appropriate distortion threshold.

Obviously, the performance of such a system depends on the values of the distortion thresholds which have to be empirically determined for the best results. The performance of the system can be parametrized by the following parameters:

- $P_1$ : average fraction of correct final decisions made by the preprocessor (i.e., when the preprocessor chooses only one possible, and correct, candidate which satisfies Rule 1).
- $E_1$ : average fraction of incorrect final decisions made by the preprocessor.
- $\beta$ : average fraction of vocabulary words that are passed on to the postprocessor when the preprocessor does not produce a final decision.
- $E_2$ : average fraction of errors made by the preprocessor when the correct word is not in the candidate list passed to the DTW postprocessor.

(The fraction is with respect to the entire test set.) The sum $\gamma = P_1 + E_1$ is the average fraction of trials in which the final decision is made by the preprocessor alone. The average fraction of trials that rely on the postprocessor decision is obviously $1 - \gamma$.

These performance parameters were studied in [4] by systematically varying the distortion thresholds. (The study was based on the likelihood ratio distortion measure augmented by an energy parameter (see Section 5.4.3). Average VQ distortions for digit utterances range typically between 0.2 and 0.7 for codebook sizes from 32 to 4.) For convenience, $D^{\circ}$ was set to be equal to $D''$ in the study. Figures 5.5 and 5.6 show typical plots of the variations of these parameters as a function of $D'$, with $D^{\circ}$ and $D''$ fixed at 0.05, and for codebook sizes varying from 4 to 32. In Figure 5.5, as expected, when $D'$ is small, both $P_1$ and $E_1$ are small, indicating that very few final decisions are made by the preprocessor alone. Also plotted in the figure is the value of $(1 - \gamma)$ which is the fraction of trial decisions that are made by the postprocessor. As $D'$ increases, more decisions are made by the preprocessor and the values of $(1 - \gamma)$ drop and approach a steady-state value of approximately 0.1, corresponding to about 90% of the word decisions being made by the preprocessor. At the steady state, $P_1$ and $E_1$ are, respectively, about 0.9 and 0.0 which are consistent with the previously discussed result of 88% correct for speaker-independent isolated digit recognition [1]. Note that as the codebook size increases, the average distortion decreases and the transition points in the $P_1$ and $(1 - \gamma)$ curves are shifted to smaller values accordingly.

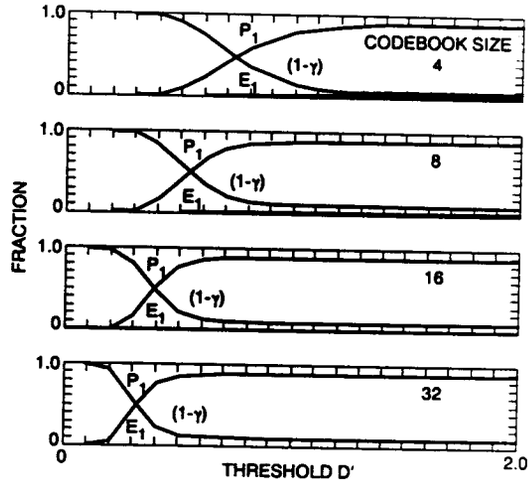



**Figure 5.5** Plots of the variation of preprocessor performance parameters $P_1$, $E_1$, and $(1-\gamma)$ as a function of the distortion threshold $D'$ for several codebook sizes for the digits vocabulary (after Pan et al. [4]).

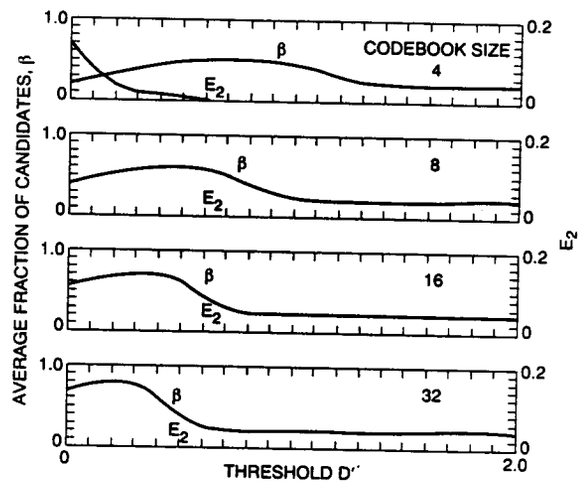



**Figure 5.6** Plots of the variation of preprocessor performance parameters $E_2$ and $\beta$ as a function of the distortion threshold $D'$ for several codebook sizes for the digits vocabulary (after Pan et al. [4]).



The curves for the $\beta$ and $E_2$ parameters as a function of $D'$ are shown in Figure 5.6. The number of trials in which the preprocessor fails to include the correct word in the candidate list for the postprocessor drops rapidly as $D'$ increases. Also, when the codebook size is 8 or larger, very rarely does the preprocessor fail to pass on a candidate list without the correct word. The behavior of the $\beta$ parameter is somewhat more complicated. This is because it is affected by both $D'$ and $D''$. The steady-state value of $\beta$ is 0.22 which, in the 10–digit vocabulary case, corresponds to 2.2 words. The DTW postprocessor thus, on average, performs dynamic time alignment on 2.2 word candidates per trial, and the correct word is essentially always one of the candidate words passed on.

These curves suggest that a good strategy to take advantage of the strength of the particular system is to choose the smallest $D'$ for which all the parameters are at their steady-state values. This is particularly true when the codebook size is very small, e.g., 4.

The effect of $D''$ is primarily on the $\gamma$ parameter, which is the average fraction of final decisions made by the preprocessor alone. Figure 5.7 shows the $\gamma$ parameter as a function of $D''$ for several codebook sizes. Recall that $D''$ corresponds to the amount of separation required between the best and the second-best candidate. As $D''$ increases, fewer trials meet the requirement of Rule 1; $\gamma$ value thus drops and the preprocessor is deferring more trial decisions to the DTW postprocessor. The number of candidate words passed on to the postprocessor, $\beta$, however, does not vary drastically with $D''$ ($D'$ fixed at 2.4) as shown in Figure 5.8. It increases only slightly from 0.22 to less than 0.3 for various codebook sizes as $D''$ increases from 0.05 to 0.15. Therefore, a reasonable value for $D''$ is approximately $0.1 \sim 0.15$.

The overall system that uses VQ as a preprocessor was found to have a recognition performance equivalent to that of the full-blown DTW system. With VQ as a preprocessor, the system requirements on computation nevertheless are greatly reduced to about 10% of what is needed in the original DTW system. The VQ preprocessor therefore warrants strong consideration for many practical systems.

**Exercise 5.4**

We wish to compare the computational complexities of a standard isolated word, DTW based recognizer, with a recognizer with a front end VQ-based preprocessor for eliminating words. Assume a vocabulary of $M$ words, with each word having a codebook with $L$ vectors. Assume each word is $N$ frames long, and that the DTW recognizer uses $Q$ reference patterns per word. Further assume that the preprocessor makes a final decision on a fraction, $\gamma = P_1 + E_1$, of the input words, and passes on the fraction, $\beta = 1 - \gamma$, of the words to the DTW processor whenever a final decision is not made by the preprocessor.

1. In terms of distance calculations, give an expression for the computation, $C_{\text{PRE}}$, of the preprocessor.
2. Assuming the cost of a combinatorics calculation is about 1/5 the cost of a distance calculation, give an expression for the computation, $C_{\text{DTW}}$, of a full DTW system (with no preprocessor).
3. Give an expression for the computation, $C_{\text{ALL}}$, of the combined preprocessor and DTW processor based on the fractions $\alpha$ and $\beta$ defined above.
4. Give an expression for the ratio, $R$, between the computation of the DTW system alone, and that of the combined system.

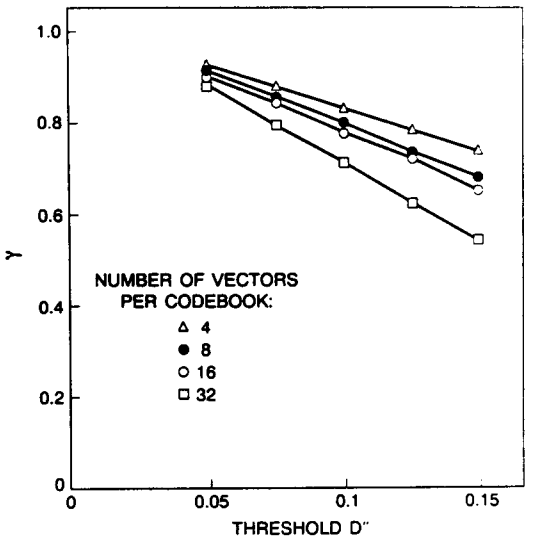



**Figure 5.7** Plots of average fraction of decisions made by the preprocessor, $\gamma$, versus preprocessor decision threshold $D''$ for several codebook sizes for the digits vocabulary (after Pan et al. [4]).

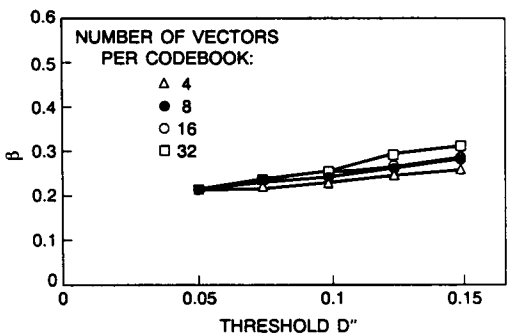



**Figure 5.8** Plots of average fraction of candidate words, $\beta$, passed on to the postprocessor, versus preprocessor decision threshold $D''$ for several codebook sizes for the digits vocabulary (after Pan et al. [4]).



5. Evaluate $R$ for the case

$$M = 10,\ Q = 12,\ N = 40,\ L = 16,\ \gamma = 0.9,\ \beta = 0.2.$$

6. If $Q$ were 1 instead of 12 (i.e., a speaker-trained case rather than a speaker-independent case), what would $R$ become? What does this imply about the effectiveness of the preprocessor for speaker-trained systems?

**Solution 5.4**

1. $C_{\text{PRE}} = M \cdot N \cdot L$ (distance calculations)
2. Assuming a grid of $N \times N$ points, there are approximately $N^2/3$ active grid points within the parallelogram defined by the usual path constraints. Thus the total computation is

$$C_{\text{DTW}} = \frac{6}{5} M \cdot Q \cdot N^2/3 \qquad \text{(distance calculations)}$$

where the $\frac{6}{5}$ is the sum of a full-distance calculation and $\frac{1}{5}$ a distance for the combinatorics.

3. The overall computation of the combined recognizer is

$$\begin{aligned} C_{\text{ALL}} &= C_{\text{PRE}} + (1-\gamma)\beta C_{\text{DTW}} \\ &= N \cdot M \left( L + (1-\gamma)\beta \frac{6}{5} QN \right) \end{aligned}$$

4. The ratio $R$ is

$$R = \frac{C_{\text{DTW}}}{C_{\text{ALL}}} = \frac{\frac{6}{5} Q \frac{N}{3}}{L + (1-\gamma)\beta \frac{6}{5} QN}.$$

5. $R \cong 10$
6. $R \approx 1 \Longrightarrow$ method is ineffective.

### 5.2.6 Vector Quantization for Efficient Pattern Matching

Another use of vector quantization in template-based speech-recognition systems is to encode the speech patterns, particularly the reference templates, for efficient pattern matching. When the reference templates are vector quantized and replaced by the corresponding codeword sequences, their representations are regularized. The enhanced regularity significantly reduces the computational as well as the storage requirements.

The VQ codebooks for this application can be trained in two ways. One is based on the entire training set of all the vocabulary words to produce a global codebook, denoted by $\mathbb{C}$. The other is to use the individual training set to generate one codebook for each word, resulting in $M$ (the vocabulary size) codebooks $\mathbb{C}^{(i)}$, $i = 1, 2, \ldots, M$. To achieve a similar level of average distortion, the single global codebook $\mathbb{C}$, so generated, usually has significantly more entries than each individual $\mathbb{C}^{(i)}$. Of course, one can also group all the $\mathbb{C}^{(i)}$'s together to form a single codebook $\mathbb{C}^{(0)} = \{\mathbb{C}^{(i)}\}_{i=1}^{M}$ which is different from $\mathbb{C}$. Use of these different codebooks leads to different performance results.

The unknown pattern $\mathcal{X}$ can be vector quantized first before being compared to the reference templates. By vector quantizing *both* the reference and test patterns with the *same* codebook in a corresponding manner, we can replace the distance computations in the DTW process for pattern matching by simple table-lookup because the distortion between each code-word pair can be computed and stored a priori. This simplification, however, comes at a price of performance degradation due to VQ distortions.

It is interesting to note that the performance degradation due to vector quantizing the unknown test pattern is unnecessary. In vector quantizing the test pattern, the distortions between each code word and each spectral frame of the unknown test pattern has to be calculated. These distortion figures, nevertheless, are the same distortions needed in the DTW process if only the reference patterns are vector quantized and represented by VQ code-word sequences. Therefore, there is no increase in computation if one chooses not to vector quantize the unknown test pattern so long as the reference patterns are properly vector quantized. (In fact, quantizing the test pattern may even result in more computation because it requires comparison of each frame to all the code words, while in the case where only the reference patterns are vector quantized, the frame distortion calculation only needs to be performed for those code words that appear within the allowable DTW region of the corresponding reference frames.) For the above reason, only the reference patterns are vector quantized in normal practice to avoid the unnecessary performance degradation.

The representation regularity is gained from the simple fact that when the reference patterns are vector quantized, each spectral frame in the reference patterns is represented by one of the code words in the finite codebook. Let $N_c$ denote the size of either the global codebook $\mathbb{C}$ that is used to encode the entire reference set, or the size of $\{\mathbb{C}^{(i)}\}_{i=1}^{M}$ if individual codebooks are used for the corresponding utterance classes. (In the following analysis, the two situations essentially make no difference in complexity considerations.) Thus, each spectrum in the unknown pattern $\mathcal{X}$ needs to be compared with these $N_c$ code words in the DTW process. This computational requirement compares favorably with the case in which the reference patterns are not vector quantized, necessitating a cumbersome frame-to-frame distance calculation of $N_w M_r$ comparisons where $N_w$ is the nominal width of the allowable DTW window and $M_r$ is the total number of reference patterns. For isolated word test utterances, such as the digits, we typically use values of $N_c = 256$ and $N_w = 15$. Computational savings become significant when $M_r$ (the vocabulary size × number of templates per word) becomes greater than 20. A similar analysis shows that a reduction in storage can also be easily obtained in this manner.

## 5.3 TEMPLATE TRAINING METHODS

One of the most fundamental modules in the pattern-matching approach to automatic speech recognition is the method for construction of the reference patterns. This is the so-called training problem. In this section, we discuss how reference patterns are created so that the pattern-matching techniques discussed in Chapter 4 can be properly applied for the best results. We use the terminology *template* and *reference pattern* interchangeably in the present context.



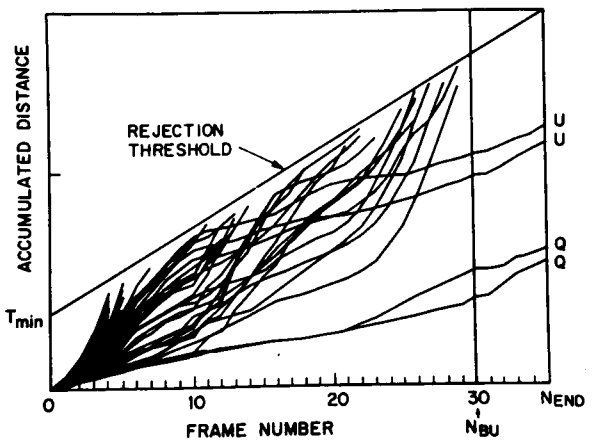



**Figure 5.9** Accumulated DTW distortion scores versus test frame based on casual training with two reference patterns per word (after Rabiner et al. [5]).

### 5.3.1 Casual Training

When the number of utterance classes in the vocabulary is not too large, and the system is designed for a specific speaker (i.e., in a speaker-trained system), a simple template training procedure that is used often is to use each token, spoken during the training session, as a reference pattern. Usually, each utterance class is represented by a multiplicity of spoken tokens, hence a multiplicity of reference patterns. This is often called casual training.

For a simple vocabulary, a speaker must be able to produce a consistent set of reference patterns for the system to be useful. However, since the method makes no attempt to estimate or predict the pattern variability, it could easily fail in systems in which the words in the vocabulary are sometimes confusable. Furthermore, because of the desire to maintain its inherent simplicity, errors committed in training, such as improper articulation, mispronunciation, mishandling of handset or microphone, acoustic noise, or other problems are often accepted as valid reference patterns without any possibility of correction. This obviously leads to poor performance under some conditions.

As an example of the use of casual training in a recognition system, Figure 5.9 shows a plot of accumulated DTW distortion scores versus test frame for a recognition system with a 39-word vocabulary (26 letters of the alphabet, 10 digits, 3 command words) based on using two casually trained templates per word [5]. The actual spoken word (the test utterance) was the word "Q." For speed of implementation, a rejection threshold curve on accumulated DTW distortion score was used to eliminate grossly bad matches before the entire DTW path was calculated. Both casual templates of the spoken word "Q" provided the two lowest accumulated scores as anticipated. However, it can also be seen that the reference templates for the word "U," which is phonetically (and acoustically) quite similar

to the word "Q," accumulated DTW scores that were the closest of all other words to the spoken "Q." For the last two-thirds of the frames of the DTW match, the slope of the accumulated distance curves for both the "Q" templates and the "U" templates are quite similar, indicating that the region of difference is at the beginning of the word. We discuss how recognizers can exploit regions of difference in phonetically similar words later in this chapter when we describe the use of discriminative training ideas in speech-recognition systems.

### 5.3.2 Robust Training

Robust training is a sequential training method in which each utterance class is spoken multiply often until a consistent pair of tokens is obtained. The resulting reference pattern is calculated as the average (along the DTW path) of the pair of consistent tokens. The averaging of tokens is normally performed in the spectral domain.

The training procedure works as follows. We consider only training for a particular utterance class. Let $\mathcal{X}_1 = (\mathbf{x}_{11}, \mathbf{x}_{12}, \mathbf{x}_{13}, \ldots, \mathbf{x}_{1T_1})$ be the first spoken token. When another token $\mathcal{X}_2 = (\mathbf{x}_{21}, \mathbf{x}_{22}, \ldots, \mathbf{x}_{2T_2})$ is recorded, the two patterns are compared via a DTW process, resulting in a DTW distortion score $d(\mathcal{X}_1, \mathcal{X}_2)$. If $d(\mathcal{X}_1, \mathcal{X}_2)$ is smaller than a prescribed threshold, say $\epsilon$, the pair of tokens are considered to be consistent. The reference pattern $\mathcal{Y} = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{T_y})$ is then computed as a warped average of $\mathcal{X}_1$ and $\mathcal{X}_2$. Following the development in Chapter 4, the dynamic time-warping problem is formulated for a pair of speech patterns, $\mathcal{X}_1 = (\mathbf{x}_{11}, \mathbf{x}_{12}, \ldots, \mathbf{x}_{1i_1}, \ldots, \mathbf{x}_{1T_1})$ and $\mathcal{X}_2 = (\mathbf{x}_{21}, \mathbf{x}_{22}, \ldots, \mathbf{x}_{2i_2}, \ldots, \mathbf{x}_{2T_2})$, as minimization of the accumulated distortion (see Eq. (4.122)–(4.125)).

$$d(\mathcal{X}_1, \mathcal{X}_2) \triangleq d_\phi(\mathcal{X}_1, \mathcal{X}_2) = \min_{\phi'} d_{\phi'}(\mathcal{X}_1, \mathcal{X}_2) \tag{5.40}$$

where

$$d_{\phi'}(\mathcal{X}_1, \mathcal{X}_2) \triangleq \sum_{k=1}^{T_y} d(\phi_1'(k), \phi_2'(k)) m(k) / M_{\phi'} \tag{5.41}$$

is based on a spectral distortion measure $d(i_1, i_2) = d(\mathbf{x}_{1i_1}, \mathbf{x}_{2i_2})$ and a set of warping functions

$$i_1 = \phi_1'(k) \quad \text{and} \quad i_2 = \phi_2'(k). \tag{5.42}$$

The pair $\phi = (\phi_1, \phi_2)$ is the optimal warping function for $\mathcal{X}_1$ and $\mathcal{X}_2$. The index $k$ is the "normal" time. Based on the optimal path $\phi$,

$$\mathbf{y}_k = \frac{1}{2}(\mathbf{x}_{1\phi_1(k)} + \mathbf{x}_{2\phi_2(k)}), \qquad k = 1, 2, \ldots, T_y. \tag{5.43}$$

($T_y$ is the maximum of the normal time $k$ for the optimal warping path $\phi$.) The vectors $\mathbf{x}$ and $\mathbf{y}$ in the above are, as usual, the short time spectra or spectral models.

If $d(\mathcal{X}_1, \mathcal{X}_2) > \epsilon$, the talker is asked to provide (speak) another training token $\mathcal{X}_3$. Similarly, the distortion scores $d(\mathcal{X}_1, \mathcal{X}_3)$ and $d(\mathcal{X}_2, \mathcal{X}_3)$ are evaluated and compared against the threshold $\epsilon$. If the smaller of the two scores falls within the $\epsilon$ threshold, a consistent token pair is declared and the reference pattern $\mathcal{Y}$ is calculated using Eq. (5.43). Otherwise, the procedure repeats with another new training token until a consistent pair of tokens is obtained.



For a speaker-trained system (using a tight consistency threshold, $\epsilon$), usually about 60% of the vocabulary words require two tokens, 35% of the words require three or four tokens, and only 5% of the words would need more than five tokens for a consistent pair of tokens to be obtained. The procedure is quite simple and efficient in terms of computation and storage. The reference pattern so generated is usually robust and relatively insensitive to botched tokens. However, if each vocabulary word is represented by a single robust pattern, it may still be inadequate for words that have more than one mode, such as words with released stops such as "eight." The inadequacy becomes even more serious when dealing with speaker-independent tasks.

### 5.3.3 Clustering

For speaker-independent speech recognition, template training by clustering is required to achieve high word recognition accuracy for practical tasks. The problem of pattern clustering is straightforward. We are given a set of $L$ utterances (speech patterns), each of which is a realization of a particular utterance (perhaps word) class to be recognized. The task is to cluster the $L$ patterns into $N$ clusters such that within each cluster, the patterns are highly similar under the specific pattern dissimilarity measure chosen for the recognizer design, and hence can be efficiently represented by a typical template. In this way, we create $N$ representative templates from a set of $L$ training patterns for each utterance class. The main advantages of pattern clustering are the statistical consistency of the generated templates and their ability to cope with a wide range of individual speech variations in a speaker-independent environment.

Early clustering algorithms [5, 6] for isolated word recognition used some fairly sophisticated pattern classification techniques such as ISODATA and chainmap. but relied on manual intervention to guide the clustering. Typically, the goal was to maximize the ratio of average intercluster distance to average intracluster distance. The total number of clusters per utterance class was a variable that was determined interactively by examining the aforementioned distance ratio. These semiautomatic procedures, although providing acceptable performance, were not amenable to widespread use, because of their lack of repeatability of the results and their need of inordinate time to complete the clustering task. To alleviate this problem, a class of automatic clustering procedures that require no human intervention was developed. In the remainder of this section we discuss these fully automatic clustering techniques.

Let $\Omega$ be a set of $L$ training patterns, $\Omega = \{\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_L\}$ where each pattern $\mathcal{X}_i$ is a realization of one specific utterance class. Following the development in Chapter 4, we define a dissimilarity measure between a pair of speech patterns as $d(\mathcal{X}_i, \mathcal{X}_j)$. (The recognition decision will be made based on the same dissimilarity measure.) An $L \times L$ dissimilarity or distance matrix $\mathbf{D}$ can be defined with $ij^{\text{th}}$ entry, $d_{ij}$, calculated as

$$d_{ij} = \frac{1}{2}[d(\mathcal{X}_i, \mathcal{X}_j) + d(\mathcal{X}_j, \mathcal{X}_i)] = \delta(\mathcal{X}_i, \mathcal{X}_j). \tag{5.44}$$

This symmetrized distance avoids potential difficulties in clustering that one may encounter in asymmetric distance measures. Because of the symmetry, we therefore need to save only $L(L-1)/2$ terms of $\delta_{ij}$. This distance matrix is the basis of several clustering algorithms that will be discussed here.

As stated previously, the objective of the algorithm is to cluster the training set $\Omega$ into $N$ disjoint clusters $\{\omega_i,\ i = 1, 2, \ldots, N\}$ such that

$$\Omega = \bigcup_{i=1}^{N} \omega_i \tag{5.45}$$

and such that speech patterns in the same cluster are "close" to each other. The total number of clusters, $N$, need not be specified a priori. Each cluster $\omega_i$ will be represented by a typical pattern $\mathcal{Y}(\omega_i) \triangleq \mathcal{Y}_i$. However, $\mathcal{Y}_i$ need not be a member of $\omega_i$. We now discuss two approaches for solving for $\{\omega_i\}_{i=1}^{N}$, the $N$ cluster solutions to Eq. (5.45).

### 5.3.3.1 Unsupervised Clustering Without Averaging (UWA)

The idea of unsupervised clustering is rather straightforward: Hone in on the largest cluster, find all training patterns close to the "center" of this cluster, exclude these patterns from the training set, and recluster the remaining patterns [7].

A flow diagram of the UWA algorithm is given in Figure 5.10. Let us denote the partial coverage set by $\Omega_j$ which includes all training patterns in the first $j$ clusters (sequentially obtained); that is,

$$\Omega_j = \bigcup_{i=1}^{j} \omega_i = \Omega_{j-1} + \omega_j. \tag{5.46}$$

The complement set $\overline{\Omega}_j$

$$\overline{\Omega}_j = \Omega - \Omega_j \tag{5.47}$$

is thus the set of all remaining training patterns after the $j^{\text{th}}$ cluster is formed.

The clustering procedure is iterative and we use $k$ to designate the iteration index. We define $\omega_j^k$ as the set of training patterns in the $j^{\text{th}}$ cluster at the $k^{\text{th}}$ iteration. For each cluster $\omega$, we have a minimax "center" $\mathcal{Y}(\omega)$,

$$\mathcal{Y}(\omega) = \mathcal{X}_{i_c} \in \omega \tag{5.48}$$

such that

$$\max_m d_{i_c,m} = \min_i \max_m d_{i,m} \tag{5.49}$$

for all $\mathcal{X}_i \in \omega$. In other words, the minimax center of a set is the pattern in the set whose maximum distance to any other pattern in the set is the smallest.

The UWA clustering algorithm, as shown in Figure 5.10, can be stated as follows.

1. Initialization: $j = 0, k = 0, \overline{\Omega}_1 = \Omega, \omega_1^{-1} = \Omega$.



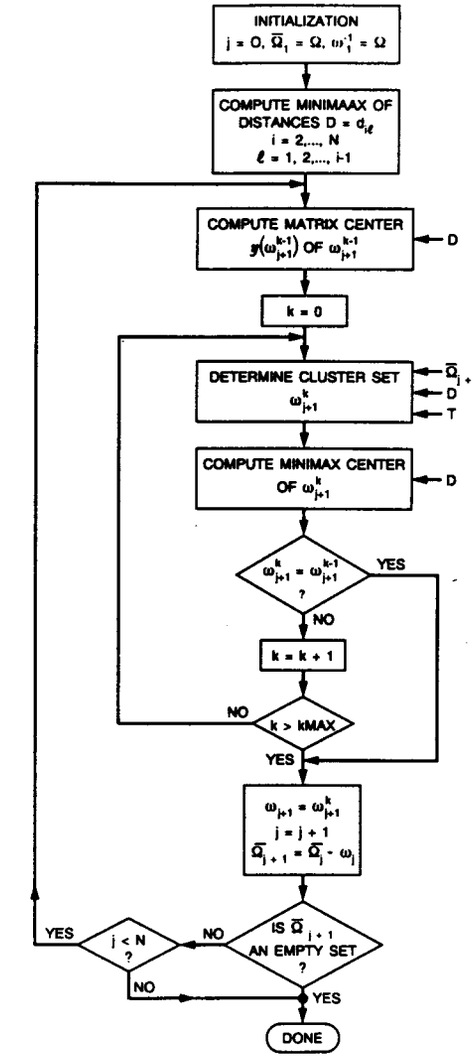



**Figure 5.10** A flow diagram of the UWA clustering procedure (after Wilpon and Rabiner [7]).

2. Determine $\mathcal{Y}(\omega_{j+1}^{k-1})$ according to Eq. (5.48) and Eq. (5.49) by making use of **D**, the distortion matrix.
3. Form $\omega_{j+1}^{k}$ by

$$\omega_{j+1}^{k} = \bigcup_{m} \mathcal{X}_m \tag{5.50}$$

where

$$\mathcal{X}_m \in \overline{\Omega}_{j+1} \tag{5.51}$$

and

$$\delta\left(\mathcal{Y}(\omega_{j+1}^{k-1}), \mathcal{X}_m\right) \leq \delta_{th}, \tag{5.52}$$

including within $\omega_{j+1}^{k}$ all patterns in $\overline{\Omega}_{j+1}$ that are within a distance threshold $\delta_{th}$ of the minimax center $\mathcal{Y}(\omega_{j+1}^{k-1})$.
4. Determine $\mathcal{Y}(\omega_{j+1}^{k})$, the new minimax center of cluster $\omega_{j+1}^{k}$ according to Eqs. (5.48–49).
5. If $\omega_{j+1}^{k} = \omega_{j+1}^{k-1}$—that is, the cluster composition is unchanged after one iteration—convergence is obtained and $\omega_{j+1} \triangleq \omega_{j+1}^{k}$, increment $j$, $j = j + 1$, and form the new partial training set $\overline{\Omega}_{j+1}$

$$\overline{\Omega}_{j+1} = \overline{\Omega}_j - \omega_j. \tag{5.53}$$

If $\overline{\Omega}_{j+1}$ is not an empty set and $j$ is smaller than the maximum number of clusters allowed, go back to step 2 and iterate; otherwise, stop.

In step 5 above, when the iteration index $k$ reaches the maximum allowable number of iterations, it is treated as if convergence were obtained.

It can be shown that prominent, distinct clusters will be readily found by this procedure because the cluster sets at successive iterations are identical. For highly overlapping data where the clusters are ambiguous, however, it often needs several iterations for the cluster composition to become stable.

Some inherent problems associated with the above unsupervised clustering algorithm are worth noting. First, as required in step 3, a distance threshold $\delta_{th}$ has to be prescribed by the user to define the compactness or closeness of a cluster. The proper way in which the distance threshold is defined is not readily defined analytically for performance optimization. Different words or utterance classes, as spoken by different talkers, generally require different thresholds. Second, the procedure does not automatically guarantee coverage of the entire training set. Depending on the distance threshold, a typical case of 12 clusters generated from 100 training patterns (word utterances) will cover only 85 to 90 of the training patterns. Those training patterns not included in the generated clusters are called "outliers," and they have the property that their distance to the closest cluster center exceeds the distance threshold. Such outlier patterns have been shown to contain useful class information that is lost in the UWA procedure.



#### 5.3.3.2 Modified $K$-means Algorithms (MKM)

The modified $K$-means algorithm [7] for speech-pattern clustering is reminiscent of the standard vector quantization codebook design algorithm of Lloyd, as described earlier. The key idea shared by the two algorithms is that of iteratively refining the clusters and cluster centers (or centroids) such that some optimality criterion is met. The main difference is that the MKM algorithm deals with a temporal sequence of spectral vectors rather than a single vector.

In Figure 5.11 we show a flow diagram of the MKM algorithm. We denote the $i^{\text{th}}$ cluster of a $j$-cluster set at the $k^{\text{th}}$ iteration as $\omega_{j,i}^{k}$ where $i = 1, 2, \ldots, j$ and $k = 1, \ldots, k_{\max}$ with $k_{\max}$ being the maximum allowable iteration count. Again, $\mathcal{Y}(\omega)$ is the representative pattern for cluster $\omega$. $\mathcal{Y}(\omega)$ can be defined as the centroid or the minimax center of $\omega$ as will be explained shortly. The algorithm finds $j$ clusters incrementally from $j = 1$ to $j = j_{\max}$, where $j_{\max}$ is the maximum number of clusters to be obtained. The MKM algorithm as shown in Figure 5.11 with a precomputed distance matrix **D** can be stated as follows.

1. Initialization: Set $j = 1$, $k = 1$, $i = 1$; set $\omega_{1,1}^{1} = \Omega$ and compute the cluster center $\mathcal{Y}(\Omega)$ of $\Omega$, the entire training set.
2. Optimal (minimum distance) classification: Label each pattern $\mathcal{X}_\ell$, $\ell = 1, 2, \ldots, L$, in $\Omega$ by index $i$ according to the minimum distance principle:

$$\mathcal{X}_\ell \in \omega_{j,i}^{k} \quad \text{iff} \quad \delta(\mathcal{X}_\ell, \mathcal{Y}(\omega_{j,i}^{k})) = \min_{i'} \delta(\mathcal{X}_\ell, \mathcal{Y}(\omega_{j,i'}^{k})). \tag{5.54}$$

Accumulate the total intracluster distance for each cluster $\omega_{j,i}^{k}$ defined by

$$\Delta_i^k = \sum \delta(\mathcal{X}_\ell, \mathcal{Y}(\omega_{j,i}^{k})) \tag{5.55}$$

where the summation is over all $\mathcal{X}_\ell \in \omega_{j,i}^{k}$.
3. Revision of clusters and cluster centers: Form $\omega_{j,i}^{k+1}$ by grouping all $\mathcal{X}_\ell$'s with label $i$ as a result of the optimal classification step above; find the new cluster centers for $\omega_{j,i}^{k+1}$, $i = 1, 2, \ldots, j$.
4. Convergence check: Go to step 5 when one of the following conditions is met:
   a. $\omega_{j,i}^{k+1} = \omega_{j,i}^{k}$ for all $i = 1, 2, \ldots, j$.
   b. $k = k_{\max}$, a preset maximum allowable number of iterations.
   c. The change in average (or the total accumulated) distance is below a preset threshold $\Delta_{th}$; i.e.

$$\left(\sum_{i=1}^{j} \Delta_i^k - \sum_{i=1}^{j} \Delta_i^{k-1}\right) \Big/ \sum_{i=1}^{j} \Delta_i^{k-1} < \Delta_{th}. \tag{5.56}$$

   Otherwise, increment $k \to k + 1$ and repeat steps 2 through 4.
5. Record the $j$-cluster solution: When the convergence condition is met, the resultant clusters and cluster centers, $\omega_{j,i}^{k+1}$ and $\mathcal{Y}(\omega_{j,i}^{k+1})$, $i = 1, 2, \ldots, j$ are the $j$-cluster solution for the training set $\Omega$. If $j = j_{\max}$, $j_{\max}$ being the maximum number of clusters specified

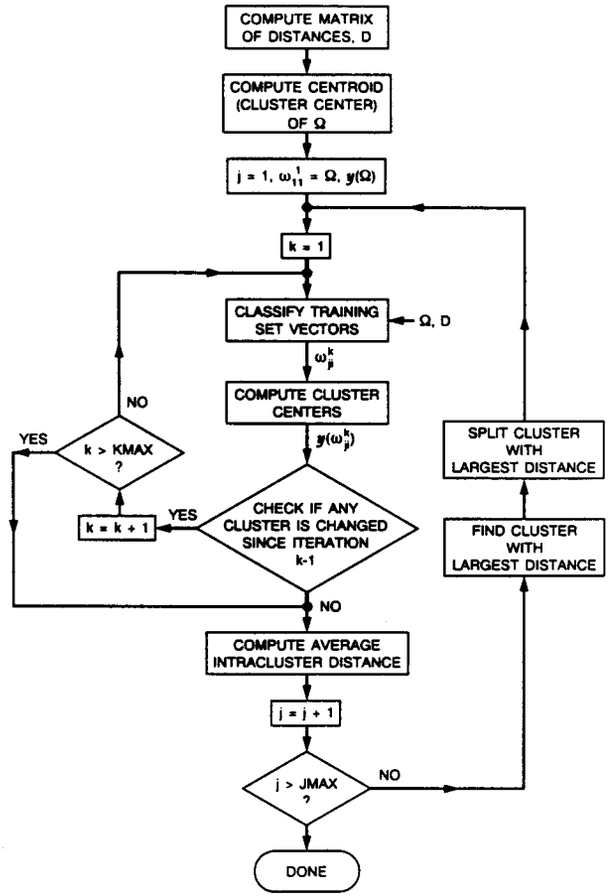



**Figure 5.11** A flow diagram of the MKM clustering procedure (after Wilpon and Rabiner [7]).

a priori, the entire clustering process is complete; otherwise, continue on to step 6.

6. Cluster splitting: Split the cluster that has the largest intracluster distance into two. There are two possibilities here, one based on the largest total intracluster distance $\max_i(\Delta_i^{k+1})$, and the other based on the largest average intracluster distance $\max_i\left(\Delta_i^{k+1}/\|\omega_{j,i}^{k+1}\|\right)$ where $\|\cdot\|$ denotes the number of patterns in the cluster. The splitting is accomplished by finding the pair of patterns, $\mathcal{X}_{\ell_1}$ and $\mathcal{X}_{\ell_2}$, in the cluster



designated to undergo splitting such that $\delta(\mathcal{X}_{\ell_1}, \mathcal{X}_{\ell_2}) \geq \delta(\mathcal{X}_{\ell_3}, \mathcal{X}_{\ell_4})$ for any other pair $\mathcal{X}_{\ell_3}, \mathcal{X}_{\ell_4}$ in the cluster. The two patterns $\mathcal{X}_{\ell_1}$ and $\mathcal{X}_{\ell_2}$ are used as the new cluster centers to replace the original single center of the designated cluster. Increment $j \to j+1$, reset $k = 1$, and repeat steps 2–5 above.

The above procedure is essentially identical to the vector quantizer design algorithm discussed previously. The procedure guarantees a 100% coverage of the training set $\Omega$, and a set of reference templates ranging from 1 to $j_{\max}$ can be easily created for an utterance class.

We complete the description of the MKM algorithm with a discussion on the computation of a cluster center. Let us assume that the cluster $\omega$ consists of the set $\{\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_{L_\omega}\}$ where the pattern index (without ambiguity) is designated only for the patterns in $\omega$ (not the entire training set $\Omega$) and $L_\omega$ is the number of patterns in $\omega$. One definition of the cluster center is the minimax center defined by Eqs. (5.48–5.49) used in the UWA algorithm. The minimax center, by this definition, is a member pattern of the cluster set. Since the distance matrix, whose elements are the distances between every pair of patterns in the cluster, exists, the minimax center can be easily computed by looking up the distance values in the matrix directly.

Another way of defining a cluster center is to find the particular pattern in the cluster that has the largest population of patterns (subset of the cluster) whose distance to the particular pattern falls within a threshold. If $\bar{\delta}$ and $\sigma_\delta$ are the mean and standard deviation of the distances $\delta(\mathcal{X}_\ell, \mathcal{X}_m)$, respectively; that is,

$$\bar{\delta} = \frac{1}{L_\omega(L_\omega - 1)} \sum_{\ell=1}^{L_\omega} \sum_{\substack{m=1 \\ m \neq \ell}}^{L_\omega} \delta(\mathcal{X}_\ell, \mathcal{X}_m) \tag{5.57}$$

$$\sigma_\delta^2 = \frac{1}{L_\omega(L_\omega - 1)} \sum_{\ell=1}^{L_\omega} \sum_{\substack{m=1 \\ \ell \neq m}}^{L_\omega} \delta^2(\mathcal{X}_\ell, \mathcal{X}_m) - \bar{\delta}^2 \tag{5.58}$$

the threshold is often empirically chosen as

$$\delta_t = \bar{\delta} + 0.5\sigma_\delta. \tag{5.59}$$

If several patterns have the same largest count of patterns with distances below the threshold, then the pattern that has the smallest average distance to all patterns in the subcluster is chosen as the cluster center. The pattern so chosen is called a *pseudoaverage center*. Note that a pseudoaverage center is again a member pattern of the cluster set. The computation again involves only table lookups.

Yet another possibility for computing the cluster center is to perform certain averaging on the patterns in the cluster. This is often done after either the minimax center or the pseudoaverage center has been computed. Since the patterns have temporal variations, the averaging process generally involves time alignment as discussed in Section 5.3.2 for robust training. In the present case, the patterns $\mathcal{X}_\ell$ in the cluster are warped to a "typical"

pattern $\mathcal{Y}$, which is either the minimax center or the pseudoaverage center of the cluster. The optimal warping path $\phi$ that satisfies $d_\phi(\mathcal{X}_\ell, \mathcal{Y}) = \min_{\phi'} d_{\phi'}(\mathcal{X}_\ell, \mathcal{Y})$ thus establishes a correspondence between $\mathbf{x}_{\ell\phi(i)}$ and $\mathbf{y}_i$. (We let $\phi_y(i) = i$, so the duration of the "typical" pattern $\mathcal{Y}$ is the normalized duration without loss of generality.) We then group the $L_\omega$ patterns (including $\mathcal{Y}$) according to their individual warping paths with respect to $\mathcal{Y}$. Those vectors that are aligned to the same index $i$ are then averaged to produce an average spectrum. The resultant pattern sequence with vectors indexed from 1 to $T_y$ (duration of $\mathcal{Y}$) is then the desired "average" cluster center.

It is important to note that the average spectrum for each time index $i$ of the typical pattern is the centroid of all the spectral vectors that are warped to $\mathbf{y}_i$. It is therefore a function of the particular spectral distortion measure chosen in the design. The centroid computation problem has been discussed in Section 5.2.2. In brief, when the distortion measure is of the Euclidean type (such as the cepstral distortion), the averaging can be performed directly on the parameters. If the distortion is likelihood based, it is necessary to average the autocorrelation (which may require residual energy normalization depending on the specific distortion measure).

It is important to note that the above three types of cluster centers are not specifically designed to minimize the average intracluster distance. Because of the temporal variations inherent in the patterns, it is not straightforward to find a pattern that minimizes the intracluster distance. The MKM algorithm is thus not guaranteed to converge in the sense of minimum average intracluster distance. This is another distinction between the MKM algorithm and the Lloyd algorithm for vector quantizer designs.

## 5.4 PERFORMANCE ANALYSIS AND RECOGNITION ENHANCEMENTS

We have discussed a set of techniques that allow us to compare speech patterns and the methods used to create reference patterns for the utterance classes. Using such methods, the modular architecture of Figure 2.37 for a speech recognizer, particularly for the recognition of isolated utterances, can be straightforwardly implemented. The performance of the system, however, depends on several factors, such as the choice of spectral-distortion measures, the exact way the reference patterns are created, the analysis conditions, and so forth. In this section, we analyze the performance of several recognition systems, so that the reader can gain insights on how these system parameters affect the recognition performance in practice.

### 5.4.1 Choice of Distortion Measures

In investigating the choice of spectral distortion measures, one must be cautious about the potential interactions between the distortion measure and other aspects of the processing, particularly the dynamic time-warping processing. One way to isolate these factors is to use a recognizer based on single frame distortion scores for a simple vocabulary such as the set of continuant vowels.

Rabiner and Soong [8] established a range of performance scores on a vocabulary



**TABLE 5.1.** Comparison of single-frame vowel-recognition error rates with various distortion measures (after Rabiner and Soong [8]).

| Distortion Measure | Training Set, VQ Codebook Size 1 | 2 | 4 | 8 | Testing Set, VQ Codebook Size 1 | 2 | 4 | 8 |
|---|---|---|---|---|---|---|---|---|
| $d_{LR}$ | 17.6 | 12.2 | 7.0 | 3.4 | 21.6 | 16.9 | 14.1 | 12.9 |
| $d_{WLR}$ | 18.8 | 13.6 | 7.2 | 3.8 | 22.4 | 18.9 | 14.2 | 12.5 |
| $d_c^2(16)$ | 18.6 | 11.8 | 7.1 | 3.5 | 21.6 | 17.4 | 13.7 | 13.4 |
| $d_{cW}^2$ | 16.5 | 11.0 | 6.7 | 3.8 | 20.0 | 16.5 | 14.2 | 13.3 |
| $d_{cWL}^2$ | 16.7 | 10.5 | 6.4 | 3.2 | 19.4 | 15.5 | 13.4 | 12.4 |

of 10 vowels, using 5 typical spectral distortion measures in a speaker-trained VQ-based recognition experiment. The 10 vowels were: /i/ embedded in the carrier word "beet," /I/ in "bit," /ɛ/ in "bet," /æ/ in "bat," /a/ in "father," /ʌ/ in "butt," /ɔ/ in "bought," /u/ in "boot," /ɝ/ in "Bert," and /U/ in "foot." Five utterances each of the 10 carrier words were recorded from each of seven talkers (four male and three female) and used to train the VQ-based vowel recognizer. An independently recorded test utterance set of five tokens each per word per talker were then employed to evaluate the performance of the VQ-based recognizers, based on various distortion measures. The spectral distortion measures compared were the likelihood ratio measure $d_{LR}$ of Eq. (4.53), the weighted likelihood ratio measure $d_{WLR}$ of Eq. (4.81) and (4.83) truncated to 16 terms, the truncated cepstral distance $d_c^2(L)$ of Eq. (4.24) with $L = 16$, the weighted cepstral distance $d_{cW}^2$ of Eq. (4.37) with the weights $w(n)$ defined as the inverse of the variance of the $n^{\text{th}}$ cepstral coefficients, and the same weighted cepstral distance of Eq. (4.37) but with the weight $w(n)$ defined according to Eq. (4.36). We denote the last case, the bandpass liftered cepstral distance, by $d_{cWL}^2$ to distinguish it from $d_{cW}^2$ with inverse variance weighting. The order of LPC analysis in the signal processing stage was 8 with a data window width of 45 ms (= 300 samples at a 6.67 kHz sampling rate). Table 5.1 shows the recognition performances based on these distortion measures. The data in the table are the vowel error rates (%), averaged over the 10 vowels and the 7 talkers, as a function of the VQ codebook size, for both the training and the testing sets ($\simeq$ 4000 trial frames per set).

Several observations can be made from the results of Table 5.1. First, there are significant degradations in performance between the training and testing data sets for a VQ-based recognizer. The degradation is particularly severe when the codebook size is large, indicating the situation where the amount of training data is insufficient for a reliable calculation of the code words. The performance differences among various distortion measures for the speaker-trained VQ-based vowel recognition task are not dramatic. However, the weighted cepstral distance with a raised sine weighting according to Eq. (4.36) consistently outperformed other distortion measures. (It is interesting to note that the design of the weighting function of Eq. (4.36) was largely motivated to deal with speaker-independent cases.)

In another study of distortion measures, Nocerino et al. [9] again compared the

**TABLE 5.2.** Comparison of recognition error rates for a 39 alphadigit-word vocabulary with various distortion measures (after Nocerino et al. [9]).

| Distortion Measure | Error Rate (%) | | | | |
|---|---|---|---|---|---|
| | Talker 1 | Talker 2 | Talker 3 | Talker 4 | Average |
| $d_I$ | 5.6 | 4.4 | 10.5 | 13.3 | 8.45 |
| $d_{WSM}$ | 5.6 | 4.4 | 11.5 | 12.3 | 8.45 |
| $d_{LR}$ | 5.1 | 5.6 | 10.5 | 13.3 | 8.63 |
| $d_c^2(32)$ | 5.1 | 5.6 | 9.7 | 15.1 | 8.88 |
| $d_{WLR}$ | 8.2 | 6.9 | 7.4 | 14.1 | 9.15 |
| $d_{IS}$ | 7.7 | 5.9 | 11.3 | 20.5 | 11.35 |

recognition performance using a 39-word alpha-digit vocabulary (26 English letters plus 10 digits and 3 command words). These are whole-word utterances recorded over dialed-up telephone lines from four talkers (two male and two female). The distortion measures compared were $d_{LR}$, $d_c^2(L)$ with $L = 32$, $d_{WLR}$ with 16 terms, $d_I$ of Eq. (4.52), $d_{IS}$ of Eq. (4.45), and $d_{WSM}$ of Eq. (4.97). The signal-processing front end of the system included a first-order preemphasis filter $(1 - 0.95\,z^{-1})$, and an eighth order LPC analysis on a 45 ms (300 samples) Hamming windowed data frame with an overlap of 30 ms between adjacent data frames. The DTW procedure had a local slope constraint of 1/2 and 2 on the warping path. The recognition was performed on a speaker-trained basis. The reference pattern for each word was chosen as the token in the training set ($5 \sim 7$ tokens per word) which had the smallest average DTW distortion score to all other training tokens.

Table 5.2 shows the error rates for the recognition trials with various distortion measures. Results for different talkers as well as the average are listed in separate columns. It is interesting to observe that the performance variation across different distortion measures is not as great as that across the four talkers. The average error rates of five of the six distortion measures were within 0.65%. The Itakura-Saito distortion, which includes matching of the spectral level as well as the spectral shape simultaneously, appeared to have the highest average error rate among the six distortion measures compared. Also, the weighted likelihood ratio measure, which attempts to emphasize the spectral peak regions, did not achieve a significantly better performance than most of the other distortion measures.

The results for $d_{WSM}$ in Table 5.2 were obtained after a series of experiments designed to determine the optimal values of $u_E$, $u_{LM}$ and $u_{GM}$ required in the definition of $d_{WSM}$ according to Eqs. (4.97–4.99). The best recognition performance (which is reported in Table 5.2) was achieved with $u_E = 0$, $u_{LM} = u_{GM} = \infty$. Note that with $u_E = 0$, the loudness difference between the two speech spectra being compared is ignored. Also with $u_{LM} = u_{GM} = \infty$, the weighting for the spectral slope becomes uniform. Therefore, it is equivalent to a unweighted slope metric.

Another interesting result in the study of Nocerino et al. is that Bark-scale warping, as introduced in $\tilde{d}_2^2$ of Eq. (4.85), led to performance degradation. Such a result is somewhat different from that of Davis and Mermelstein [10] in which the mel-frequency



cepstral measure was reported to outperform several other distortion measures in a limited experiment. It should be noted, however, that the Bark-scale warping in Eq. (4.85) does not have the same spectral averaging effect of the mel-frequency cepstral measure derived from the filterbank output as proposed by Davis and Mermelstein (see Section 4.5.6).

These comparative studies point out several important issues in defining distortion measures for speech recognition. First, the incorporation of energy (or power) in pattern comparison requires careful design as seen from the inferior performance of the Itakura-Saito distortion measure. The treatment of energy information as part of the distortion measure will be discussed in Section 5.4.3. Second, it is vitally important to design a distortion measure that takes into account the inherent speaker variation and works consistently well for different speakers, particularly for a speaker-independent task. The uneven performances of the six distortion measures shown in Table 5.2 indicate that further enhancements in terms of robustness to speaker variations are necessary. Third, it is not always easy to incorporate our empirical knowledge of sound perception in the distortion measure. The two perceptually based distortion measures, namely the weighted slope measure and the weighted likelihood ratio measure, both failed to improve performance scores. In fact, the slope measure worked best when unweighted.

While more research is taking place in pursuit of the "ideal" distortion measure, we should note that the bandpass-liftered cepstral distortion of Eq. (4.37) with a raised sine weighting function of Eq. (4.36) was shown [11] to give consistently good recognition performance in speaker-independent recognition tasks. As discussed in Section 4.5.3, such a distortion measure was designed with the intention of reducing the variability due to different speakers and to suppress the spurious artifacts of the signal processing mechanism. The experimental results show that these goals were well met in practice.

### 5.4.2 Choice of Clustering Methods and kNN Decision Rule

Template training by clustering is of essential importance in speaker-independent recognition systems because of the acoustic variability of word utterances across different talkers that require multiple representations for an utterance class. Because of the use of multiple reference templates, we can consider the use of a kNN decision rule to provide both a robust class decision and improved recognition performance [12]. If we use $N$ reference patterns to represent each word, then for every unknown input pattern, $N$ distortion scores are calculated for these $N$ reference patterns. We denote these $N$ distortion scores by $d^{ij}$, $j = 1, 2, \ldots, N$, where $i$ indicates the word index, ranging from 1 to $M$, the vocabulary size. The distortion scores for a particular word $i$ can be reordered such that

$$d^{i(1)} \le d^{i(2)} \le \ldots \le d^{i(N)} \tag{5.60}$$

where the parentheses are used to indicate ordered indices. For the kNN rule, we compute the average of the $k$ smallest distortions, resulting in

$$d^i = \frac{1}{k}\sum_{j=1}^{k} d^{i(j)}. \tag{5.61}$$

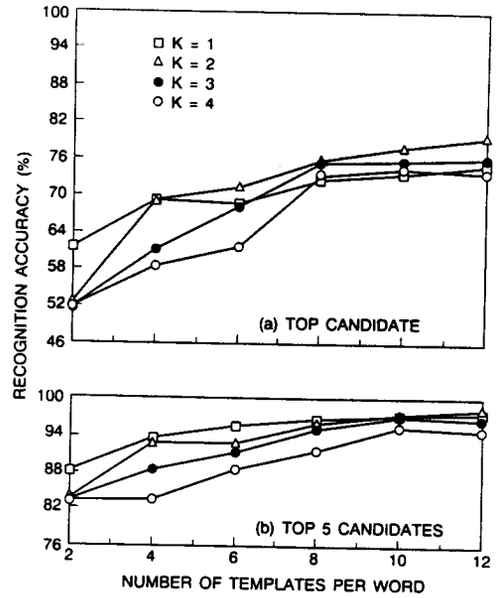



**Figure 5.12** Recognition accuracy (percent correct) as a function of the number of templates per word based on template clustering with kNN decisions ($k = 1, 2, 3, 4$); (a) the top candidate is the correct word, (b) the correct word is among the top 5 candidates (after Rabiner et al. [5]).

The index of the "recognized" word using the kNN rule above is determined as

$$i^* = \arg\min_i d^i. \tag{5.62}$$

To study the interaction of the kNN decision rule with the number of templates used to characterize each word in the vocabulary, a speaker-independent isolated word-recognition task was run using a 39-word vocabulary (26 English letters, 10 digits and 3 command words), [5]. Figure 5.12 shows plots of recognition accuracy as a function of the number of reference patterns per word used in the recognizer, the value of $k$ for the kNN rule, the method for obtaining the reference patterns, and the word candidate position in the decision outcome. Parts (a) and (b) of the figure are results based on use of a clustering algorithm and parts (c) and (d) are similar plots for templates obtained by random selection from the training tokens. Parts (a) and (c) are the results for the top recognition candidate and parts (b) and (d) are those for the top five candidates (meaning the correct word falls in the list of the top five word candidates).



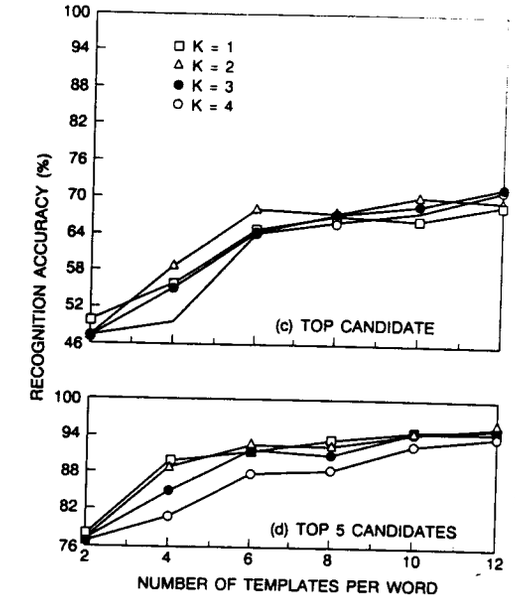



**Figure 5.12, continued** Same as Figure 5.12 a and b except that the templates are randomly chosen; (c) the top candidate is the correct word, (d) the correct word is among the top 5 candidates (after Rabiner et al. [5]).

For this particular vocabulary, it is observed that the recognition accuracy improves as the number of templates per word increases, particularly from two to about eight. For a small number of templates per word, the kNN rule performs best at $k = 1$. For a large number of templates per word, $k = 2$ produces the best results. The recognition rates for the top five candidates approaches 98% with 12 templates per word. Also, the recognition performance obtained from using the clustered templates is significantly better than that obtained from randomly chosen tokens.

A more detailed comparison on the effects of various clustering algorithms was provided by Wilpon and Rabiner [7]. The UWA algorithm (unsupervised clustering without averaging, Section 5.3.3) was found to perform worse than the MKM algorithm (modified $k$-means, also Section 5.3.3). Figure 5.13 plots the average digit error rate as a function of the number of reference templates per word for a 100-talker database of 1000 digits with clusters generated by both the UWA and the MKM procedures. (The recognition was based on isolated utterances.) It is particularly interesting to observe that for one template per word, the MKM template (which has a full "coverage" of the entire training tokens) yielded a 10% lower error rate than the UWA template. Even when more templates per word were used, the MKM templates provided about 2% reduction in error rate over that obtained

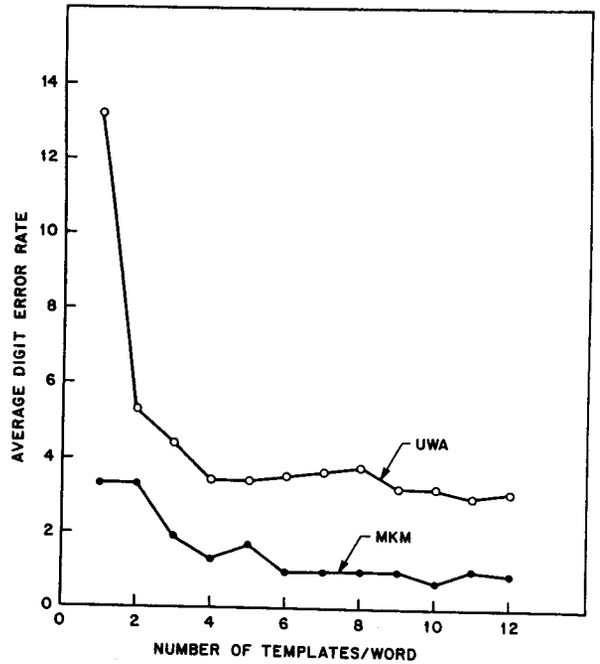



**Figure 5.13** Average digit error rate as a function of the number of templates per word for a database of 1000 digits with clusters generated by the UWA and the MKM clustering procedures respectively (after Wilpon and Rabiner [7]).

using the UWA templates. As we have discussed previously, MKM is more analytically consistent than UWA and indeed the experimental results verify the performance gain due to the increased consistency.

### 5.4.3 Incorporation of Energy Information

It is generally agreed that the energy contour of an utterance contains important information about the phonetic identity of the sounds within the utterance. For example, fricatives are much lower in energy than vowels. Proper use of such energy information (over time) can therefore be helpful in (grossly) distinguishing one word utterance from another. The results of Table 5.2, based on using the Itakura-Saito distortion measure, however, indicate that incorporation of energy information in pattern-comparison measures may in fact degrade the recognition performance if not properly handled.

The energy information appears in two forms in the speech spectral representation. One is the absolute power of the power spectrum $S(\omega)$ (of a frame of speech signal), defined



by

$$E = \int_{-\pi}^{\pi} S(\omega)\frac{d\omega}{2\pi}, \tag{5.63}$$

and the other is the gain $\sigma^2$ when $S(\omega)$ is modeled by an all-pole spectrum $\sigma^2/\left|A(e^{j\omega})\right|^2$,

$$\sigma^2 = \min_A \int_{-\pi}^{\pi} \left|A(e^{j\omega})\right|^2 S(\omega)\frac{d\omega}{2\pi}. \tag{5.64}$$

The gain term $\sigma^2$ is identical to the minimum residual energy (of a speech frame) resulting from optimal inverse filtering of $S(\omega)$. The Itakura-Saito distortion measure includes the gain term in a fixed and inflexible manner.

The most straightforward and flexible way to incorporate the energy information, either the absolute power or the gain, is to add an energy distance term to any of the spectral distortions discussed previously. As an example, in the truncated cepstral distance based on LPC cepstra, $c_0 = \log \sigma^2$, $c_0' = \log \sigma'^2$ thus the energy (or gain) distance

$$d_E(S, S') = (\log \sigma^2 - \log \sigma'^2)^2 = (c_0 - c_0')^2 \tag{5.65}$$

is just an additional term to $d_c^2(L)$ of Eq. (4.24).

A particular form of energy distance that has been extensively studied is

$$d_E(S, S') = \gamma \cdot g(|\log E_r - \log E_r'|) \tag{5.66}$$

where $\gamma$ is a constant that determines the weight of the energy distance relative to the spectral distortion, $g$ is a nonlinear function, and $E_r$ and $E_r'$ are the relative power (or gain) with respect to the maximum power $E_{\max}$ and $E_{\max}'$ of the test and reference patterns, respectively; that is,

$$E_r = E/E_{\max} \quad \text{and} \quad E_r' = E'/E_{\max}'. \tag{5.67}$$

The absolute loudness effects are thus removed from the energy distance, with the maximum set at 0 dB for both the test and the reference pattern. The nonlinear function $g(\cdot)$ is defined as

$$g(E) = \begin{cases} 0 & |E| \le E_{LO} \\ E - E_{LO} + E_{OF} & E_{LO} \le E \le E_{HI} + E_{LO} - E_{OF} \\ E_{HI} & E_{HI} + E_{LO} - E_{OF} < E. \end{cases} \tag{5.68}$$

A plot of $g(E)$ as a function of $E$ is shown in Figure 5.14. The shape of this nonlinear function reflects certain prior observations of loudness perception by human listeners. The values of $E_{LO}$, $E_{OF}$ and $E_{HI}$ are empirically set at $E_{LO} = E_{OF} = 6$ dB and $E_{HI} = 20$ dB. The weight $\gamma$ was experimentally determined to be in the range of 0.025 to 0.25, with a typical value of 0.1.

In an experiment by Nocerino et al. [9], similar to the one that produced the results of Table 5.2, use of energy information in various distortion measures was studied. Table 5.3 shows the recognition error rates of several common distortion measures with and without the energy information. The truncated cepstral distance $d_c^2(16)$ in the table is based on the LPC cepstrum, and the incorporation of gain information is achieved by adding $d_E(S, S')$ of Eq. (5.65) to the original distance. The Itakura-Saito distortion measure $d_{IS}$ of Eq. (4.45)

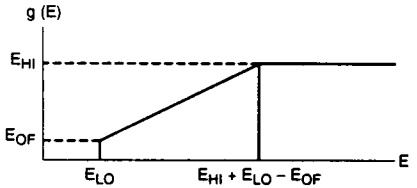



**Figure 5.14** The nonlinearity function, $g$, applied to the log energy difference between two frames for the energy distance calculation.

**TABLE 5.3.** Comparison of recognition error rate pertaining to several distortion measures with and without energy information (after Nocerino et al. [9]).

| | Distortion Measure | | Error Percentage | | | | |
|---|---|---|---|---|---|---|---|
| | Gain | Normalized Energy | Talker 1 | Talker 2 | Talker 3 | Talker 4 | Average |
| $d_c^2(16)$ | yes | no | 6.2 | 8.2 | 10.0 | 18.5 | 10.73 |
| | no | no | 6.4 | 6.4 | 10.0 | 16.2 | 9.75 |
| $d_{IS}$ | yes | no | 7.7 | 5.9 | 11.3 | 20.5 | 11.35 |
| | yes | yes | 5.6 | 5.1 | 11.0 | 15.6 | 9.33 |
| $d_{LR}$ | no | no | 5.1 | 5.6 | 10.5 | 13.3 | 8.63 |
| | no | no | 4.1 | 4.1 | 6.7 | 14.4 | 7.36 |

includes the matching of gain terms. As discussed in Section 5.4.2, such use of energy information appears to degrade the system performance as compared to the likelihood ratio distortion where $\sigma = \sigma' = 1$. One possible modification to the Itakura-Saito distortion, recognition results of which are also included in Table 5.3, is to normalize the gain terms by the maximum energy (of a frame of speech) in the utterance, similar to Eq. (5.67),

$$\bar{\sigma}^2 = \sigma^2/E_{\max} \quad \text{and} \quad \bar{\sigma}'^2 = \sigma'^2/E'_{\max}. \tag{5.69}$$

The gain terms, $\sigma^2$ and $\sigma'^2$, in the original Itakura-Saito distortion are then replaced by $\bar{\sigma}^2$ and $\bar{\sigma}'^2$, respectively. The likelihood ratio distortion $d_{LR}$ with normalized energy, as listed in the table, takes the form of $d_{LR} + d_E$ where $d_E$ is defined by Eqs. (5.66)–(5.68).

The results of Table 5.3 indicate that the energy information in the temporal pattern sequence can be useful for improving the recognition accuracy if it is properly normalized. Direct use of the absolute loudness level or gain term, however, leads to degradation in recognition accuracy.

### 5.4.4 Effects of Signal Analysis Parameters

In a typical template-based system using LPC analysis as the signal-processing front end, one has to first determine the values of several key analysis parameters—that is, the LPC analysis order, the length of the analysis window (frame) and the frame rate (often expressed



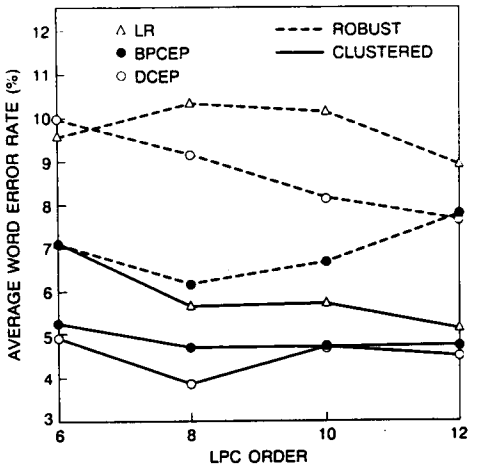



**Figure 5.15** Plots of word-recognition error rate versus the analysis frame size (in samples) with various frame shift intervals (33, 67, 100 and 200 samples) (after Rabiner and Wilpon [13]).

in terms of shift interval). These parameters are denoted by $p$ (order), $N_w$ (window length in number of samples), and $N_s$ (shift interval in number of samples), respectively. An experiment with a 4-talker, 39-word vocabulary database was conducted in a speaker-trained recognition mode to study the effects of these parameters upon the recognition performance. The database was the same one used in the study of Nocerino et al. [9] discussed in Section 5.4.1. Five tokens per word were used in training, and 10 tokens per word were used for testing. The speech material was recorded over a dialed-up telephone line and sampled at 6.67 kHz.

The values of $p$, $N_w$ and $N_s$ tested were $p = 6, 8, 10, 12$, $N_w = 33, 67, 100, 133, 200, 300$, and $N_s = 33, 67, 100, 200, 300$, respectively. Figure 5.15 shows plots of recognition error rate (%) as a function of the LPC analysis order for three LPC-related distortion measures: the likelihood ratio measure (LR), the bandpass-liftered LPC cepstral distance (with the raised sine lifter of Eq. (4.36), denoted by BPCEP), and the delta LPC cepstral distance (Eq. (4.114) with only two terms, $d_2^2$ and $d_{2\delta(1)}^2$, denoted by DCEP). For this plot, $N_w = 300$ and $N_s = 100$. The reference templates were generated by a clustering method. It is seen that both the liftered cepstral distance and the combined distance $d_2^2 + d_{2\delta(1)}^2$ performed significantly better than the likelihood ratio measure. Increasing the LPC analysis order did not, however, lead to higher recognition accuracy. The best result was found to be $p = 8$ with the delta cepstral measure. Figure 5.16 [10] shows the word error rate (%) as a function of $N_w$ for several values of $N_s$. In general, the performance at $N_w = 3N_s$ was found to be the best for a fixed $N_s$. Since larger $N_s$ means less computation, we often try to find the largest $N_s$ possible before the system seriously

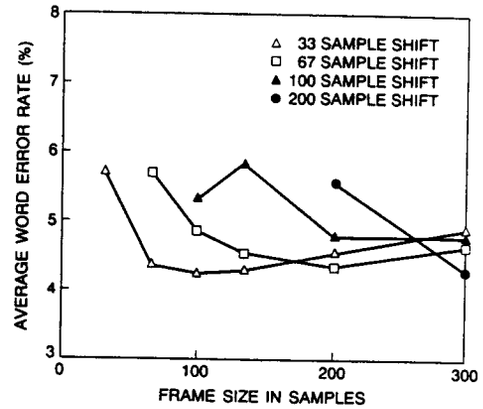



**Figure 5.16** Plots of word recognition error rate versus LPC analysis order for three LPC-related distortion measures, the likelihood ratio measure (LR), the bandpass liftered cepstral measure (BPCEP), and the cepstral measure with additional delta cepstrum (DCEP) (after Rabiner and Wilpon [13]).

degrades in performance. When $N_w$ goes beyond 300, that is an analysis window longer than 45 ms, the analysis window often encompasses nonstationary segments of the speech, reducing the precision of signal representation. A reasonable choice is $N_w = 300$ and $N_s = 100$. For a 6.67-kHz sampling rate, these are equivalent to 45-ms analysis windows and 15-ms window shifts, respectively.

### 5.4.5 Performance of Isolated Word-Recognition Systems

With proper choice of system parameters, the distortion measure, and the method to generate the reference patterns, an isolated word-recognition system usually is expected to perform well. Depending on the degree of difficulty inherent in the vocabulary, however, we observe some variation in the recognition accuracy. We summarize the laboratory performance (in terms of error rate) of a number of state-of-the-art isolated word systems evaluated on different vocabularies in different modes in Table 5.4. In the table, SI denotes speaker-independent mode, SD is speaker-trained or speaker-dependent, and MS is the "multispeaker" case in which the training data and the testing data are from the same (perhaps large) set of talkers. The 37 dialer words include the 10 digits, 7 command words, and 20 names as might be typical for a user repertory of names. The number of talkers involved in the recognition trial is listed in the comment column. The size of the vocabulary does not appear to strongly affect the recognition performance. These performance scores show that isolated word-recognition technologies are mature enough to be of practical use for many applications. (We will come back to this issue of practical applications of speech recognizers in Chapter 9.)



**TABLE 5.4.** Performance of Isolated Word-Recognition Systems

| Vocabulary | | Mode | Error Rate (%) | Comments |
|---|---|---|---|---|
| 10 | Digits | SI | 0 | 400 Talkers |
| 37 | Dialer Words | SD | 0 | 10 Talkers |
| 39 | Alphadigits | SD | 4.5 | 4 Talkers |
| | | MS | 7.0 | 100 Talkers |
| 54 | Computer Terms | SI | 3.5 | 20 Talkers |
| 129 | Airline Words | SI | 2.9 | 20 Talkers |
| 200 | Japanese Cities | SD | 2.7 | 1 Talker |
| 1109 | Basic English | SD | 4.3 | 3 Talkers |

## 5.5 TEMPLATE ADAPTATION TO NEW TALKERS

Template adaptation refers to a class of techniques that adapt the values of parameters of the reference patterns to changes in the input signal (e.g., a new speaking environment) so that the recognizer can properly deal with input signals that are somewhat different from those seen during the training phase, thereby avoiding a severe performance degradation. Among the effects that cause changes to the input signal are varying channel characteristics, ambient background noise, differences in transducers and, particularly, new talkers with accents different from those used in the training set. We focus our attention on template adaptation to new talkers in this section.

For a given speech-recognition task, a speaker-dependent (SD) system normally performs better than a speaker-independent (SI) system, so long as a sufficient amount of data is available to adequately train the speaker-dependent templates. When the amount of speaker-specific training data is limited, however, superiority of the SD system over the SI system is not guaranteed, because of the lack of reliability in the calculated reference parameters. One way to improve the performance of the recognizer under these conditions is to make use of existing knowledge, as represented in a speaker-independent template set or another well-trained speaker-specific template set. This technique is often referred to as speaker adaptation (SA) although, as we shall see, it may have a more general interpretation.

Speaker adaptation of reference patterns can be formulated in a number of ways, as illustrated in Figure 5.17. "Adaptive clustering" attempts to modify the existing speaker-independent reference set, based on a new set of speaker-independent training data, so that the new talker's characteristics (as seen in the SD data set) are properly captured in the adapted template set. In "speaker transformation" or "speaker conversion," a well-trained template set for one particular talker is converted to another template set for a new talker based on a limited amount of training data from the new talker. "Speaker adaptation" techniques attempt to modify a speaker-independent (or multispeaker) template set, using new training data from a single talker, and provide a good set of speaker-adapted templates, even when the amount of training data from the new talker is limited. Finally, in "sequential adaptation," the training data, from a given talker (or talkers), are acquired over time, and

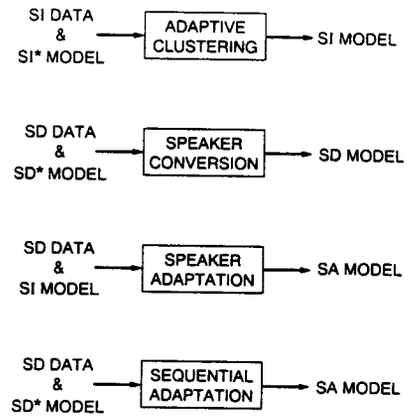



**Figure 5.17** Four different speaker adaptation scenarios.

the reference patterns are sequentially adjusted every time new training data is available. The basic adaptation techniques for these different implementations of speaker adaptation are usually very similar and we discuss in the following section the generic processing procedure rather than provide specific discussions of each of the individual adaptation techniques.

### 5.5.1 Spectral Transformation

Adaptation of the reference set of templates to a new talker's speech involves transformation of one spectrum to another that better matches the new talker's spectral characteristics. Suppose $S_A(\omega)$ is a speech spectrum produced by talker $A$. Spectral transformation is a mapping, $f$, that converts $S_A(\omega)$ into $S_B(\omega) = f(S_A(\omega))$ which is more likely to be produced by talker $B$. One classical approach to spectral transformation is to estimate the vocal cord (excitation) spectrum and the vocal tract dimensions (mainly the length) for individual talkers, and then perform the spectral transformation based on a physical vocal tract model defined by the estimated articulatory parameters. The transformation is parameterized by $\Lambda_{AB}$ and we write it in the form $S_B(\omega) = f(S_A(\omega), \Lambda_{AB})$.

Another popular approach to spectral transformation is to establish a correspondence between pairs of "typical" spectra from two talkers based on their occurrence in the same context speech (e.g., vowels embedded in carrier words). The context could simply be a word, as represented by a sequence of vector-quantization codebook entries (recall the VQ-based word recognizer introduced in Section 5.2.6). (It could also be a specified phoneme sequence.) Instead of using a single, universal transformation, parameterized by $\Lambda_{AB}$, that maps one arbitrary spectrum to another, use of VQ code words permits nonparametric mapping for a finite set of "typical" spectra. It offers a more flexible, more effective



mapping than the universal, parametric transformation $f(\cdot, \Lambda)$ because it does not postulate or rely on the existence of such a function for arbitrary spectra.

Let the set of "typical" spectra of talker $A$, as represented by VQ codebook entries, be $S_A^{(i)}(\omega)$, $i = 1, 2, \ldots, N$. The VQ-based spectral mapping is a set of transformations $\{f_i, i = 1, 2, \ldots, N\}$ that define

$$S_B^{(i)}(\omega) = f_i(S_A^{(i)}(\omega)), \quad i = 1, 2, \ldots, N, \tag{5.70}$$

or simply $S_B^{(i)}(\omega) \leftrightarrow S_A^{(i)}(\omega)$. The function $f_i$ need not be parametrized; it simply represents a correspondence between $S_A^{(i)}$ and $S_B^{(i)}$.

The procedure to determine the appropriate set of transformations is as follows. As usual, we use vector notation; $\mathbf{x}_A$ and $\mathbf{x}_B$ are spectral vectors obtained from utterances of talker $A$ and talker $B$ respectively. Based on a set of training vectors $\{\mathbf{x}_B\}$ collected from talker $B$'s utterances, we employ the generalized Lloyd algorithm to generate a codebook $\mathbb{C}_B = \{\mathbf{x}_B^{(i)}, i = 1, 2, \ldots, N\}$ of size $N$. This codebook is then used to vector quantize the spectral sequences of certain known word utterances. Without loss of generality, let us assume $\mathcal{X}_B = (\mathbf{x}_{B1}, \mathbf{x}_{B2}, \ldots, \mathbf{x}_{BT})$ is a spectral sequence for an utterance of a particular word. Vector quantizing $\mathcal{X}_B$ with $\mathbb{C}_B$ results in

$$\mathcal{X}_B \to \hat{\mathcal{X}}_B = (\hat{\mathbf{x}}_{B1}, \hat{\mathbf{x}}_{B2}, \ldots, \hat{\mathbf{x}}_{BT})$$

with the corresponding VQ index sequence $J_B = (j_1, j_2, \ldots, j_T)$ where

$$\hat{\mathbf{x}}_{Bi_B} = \mathbf{x}_B^{(j_{i_B})} \in \mathbb{C}_B, \quad i_B = 1, 2, \ldots, T. \tag{5.71}$$

The codebook $\mathbb{C}_A = \{\mathbf{x}_A^{(i)}, i = 1, 2, \ldots, N\}$ is produced in a different way. Utterances of the same word are collected from talker $A$. Again, we denote a spectral sequence of such an utterance by $\mathcal{X}_A = (\mathbf{x}_{A1}, \mathbf{x}_{A2}, \ldots, \mathbf{x}_{AT'})$. A dynamic time-warping procedure is used to align $\mathcal{X}_A$ and $\mathcal{X}_B$. The optimal alignment path $\phi = (\phi_A, \phi_B)$ defines

$$i_A = \phi_A(k) \quad \text{and} \quad i_B = \phi_B(k) \tag{5.72}$$

where $k$ is the "normal" time index. Since each $i_B$ has a corresponding VQ index $j_{i_B}$, the index label is thus transferred to $\mathcal{X}_A$ via the normal time $k$,

$$j_{i_B} \to i_B = \phi_B(k) \to i_A = \phi_A(k) \to \mathbf{x}_{Ai_A}. \tag{5.73}$$

The code word entry $\mathbf{x}_A^{(j_{i_B})}$ in $\mathbb{C}_A$ with index $j_{i_B}$ is then calculated as the centroid of all the vectors $\mathbf{x}_{Ai_A}$ that are labeled with this index. Other code words are produced in a similar manner. Since the index labels are kept identical through the conversion, the mapping is simply

$$\mathbf{x}_B^{(i)} = f_i(\mathbf{x}_A^{(i)}) \leftrightarrow \mathbf{x}_A^{(i)}. \tag{5.74}$$

Note that the codebook $\mathbb{C}_A = \{\mathbf{x}_A^{(i)}\}$ is not optimal in the sense of minimum average distortion (Eq. (5.3)). However, if $\mathcal{X}_A$ and $\mathcal{X}_B$ are reasonably close, the distortion performance of $\mathbb{C}_A$ may be nearly optimal and encoding $\mathcal{X}_A$ with $\mathbb{C}_A$ would result in an index sequence highly correlated with $J_B$ except for the time alignment.

To transform an arbitrary spectral vector $\mathbf{x}_A$ of talker $A$ into that of talker $B$, we

vector-quantize $\mathbf{x}_A$ into $\hat{\mathbf{x}}_A$ which satisfies

$$\hat{\mathbf{x}}_A = \mathbf{x}_A^{(i)} = \arg \min_{\mathbf{x}_A^{(j)} \in \mathbb{C}_A} d(\mathbf{x}_A, \mathbf{x}_A^{(j)}) \tag{5.75}$$

and then use the index $i$ to point to $\mathbf{x}_B^{(i)}$; i.e.,

$$\mathbf{x}_A \to \mathbf{x}_A^{(i)} \to \mathbf{x}_B^{(i)} = f_i(\mathbf{x}_A). \tag{5.76}$$

Therefore, the VQ-based spectral transformation maps an arbitrary spectrum of one talker to another spectrum that is a member of a finite collection of typical spectra of another talker. This type of transformation has proven effective in speech recognition.

### 5.5.2 Hierarchical Spectral Clustering

Hierarchical spectral clustering [14] is an adaptive clustering technique that performs speaker adaptation in an automatic, self-organizing manner. Again, in a VQ-based word-recognition system, each word or utterance class is represented by a set of VQ index sequences we shall refer to as a "word dictionary." The word dictionary represents all possible pronunciation variations of the word (by different talkers). Speaker adaptation is achieved by adapting the codebook entries (spectral vectors) to a particular talker while keeping the index sequence set intact.

The key idea of this adaptation method is to hierarchically cluster the spectra in the new training set in correspondence with those in the original VQ codebook. The correspondence between the centroid of a new cluster and the original code word is established by way of a deviation vector. The method works as follows.

First, a universal codebook is produced by clustering the spectral vectors collected from a set of talkers. VQ-based template clustering is then used to create the word dictionary based on the training utterances and the universal codebook. We assume the universal codebook $\mathbb{C}^u$ has $N$ code word entries, $\mathbb{C}^u = \{\mathbf{y}_i\}_{i=1}^N$. Another training set from a new talker (the target of adaptation) is given. The procedure starts by calculating the centroid of all the training spectra from the new talker and the centroid of the $N$ code word spectra in the universal codebook. We denote these two centroids by $\mathbf{u}_{11}$ and $\mathbf{v}_{11}$, respectively. A deviation vector can be calculated by

$$\mathbf{z}_{11} = \mathbf{u}_{11} - \mathbf{v}_{11} \tag{5.77}$$

such that the new centroid of the shifted code-word vectors $\{\mathbf{y}_i + \mathbf{z}_{11}\}_{i=1}^N$ coincides with $\mathbf{u}_{11}$. This can be easily verified for the case of an $L_2$-norm vector space where the norm is measured using a Euclidean distance.

We next split the new training set into two clusters and calculate the centroid of each cluster. Using the above notation, we denote these two centroids by $\mathbf{u}_{2i}$, $i = 1, 2$, respectively. Each of the shifted code-word vectors $\mathbf{y}_i + \mathbf{z}_{11}$, $i = 1, 2, \ldots, N$, is then assigned to one of the two centroids, $\mathbf{u}_{21}$ and $\mathbf{u}_{22}$, according to the nearest-neighbor principle, generating two new code-word clusters and the corresponding centroids, which we denote by $\mathbf{v}_{2i}$, $i = 1, 2$. Based on the two corresponding pairs of centroids, we define



two deviation vectors

$$\mathbf{z}_{2i} = \mathbf{u}_{2i} - \mathbf{v}_{2i}, \quad i = 1, 2 \tag{5.78}$$

and use them to further shift the already shifted code-word vectors according to the procedure described below. (The procedure aims at maintaining the continuity between adjacent clusters.) The above procedure repeats until the number of new code-word vectors reaches the preset goal or, at most $N$, the size of the original universal codebook.

The shift procedure for the code-word vectors takes into account the relative distances from the particular shifted code-word vector to all the new centroids. Let us assume that at a certain stage of hierarchical clustering, $\mathbf{y}_i$ has been shifted to $\mathbf{y}_i'$ and as a result of further clustering, $m$ pairs of centroids, $(\mathbf{u}_{mi}, \mathbf{v}_{mi})$, $i = 1, 2, \ldots, m$, are created. Note that calculation of $\mathbf{v}_{mi}$ is based on $\mathbf{y}_i'$ with the corresponding nearest-neighbor label. The $m$ deviation vectors are calculated as

$$\mathbf{z}_{mi} = \mathbf{u}_{mi} - \mathbf{v}_{mi}, \quad i = 1, 2, \ldots, m. \tag{5.79}$$

For each code word vector $\mathbf{y}_j'$, the shift vector $\mathbf{s}_j$ is calculated as a weighted sum of these deviation vectors $\mathbf{z}_{mi}$,

$$\mathbf{s}_j = \left(\sum_{i=1}^{m} w_{ji}\, \mathbf{z}_{mi}\right) \Big/ \left(\sum_{i=1}^{m} w_{ji}\right) \tag{5.80}$$

where

$$w_{ji} = 1/[d(\mathbf{y}_j', \mathbf{u}_{mi})]^{\alpha}. \tag{5.81}$$

In Eq. (5.81), $\alpha$ is a constant, typically between 0.5 and 1, and $d(\cdot, \cdot)$ is a Euclidean distance. The shift vector $\mathbf{s}_j$ is then used to adjust $\mathbf{y}_j'$,

$$\mathbf{y}_j' \to \mathbf{y}_j' + \mathbf{s}_j. \tag{5.82}$$

Figure 5.18 illustrates the hierarchical clustering procedure going from one cluster to four clusters. Note that the constant $\alpha$ plays a smoothing role in combining the effects of all the clusters.

The above hierarchical clustering procedure is progressive in that the number of clusters grows from one to the desired limit. The procedure of calculating the deviation vectors, Eq. (5.79), and the shift vectors, Eq. (5.80), is equally applicable in the situation in which all (say $m$) clusters are adapted at once, without cluster splitting and recursion. Figure 5.19 shows the VQ average spectral distortion (cepstral distance) as a result of hierarchical adaptation of the codebook to the new talker's training set, relative to the case in which no adaptation is performed (denoted NA), as a function of the codebook size. The solid curve shows the results obtained with the progressive hierarchical adaptation procedure, while the dashed curve shows the results of the direct adaptation procedure. When the number of clusters is small, the direct adaptation method appears to perform better than the progressive method. However, beyond eight clusters, the direct method does not show further reduction in the average distortion while the progressive method continues to give decreasing average spectral distortions. It was suggested [14] that the adaptation procedure start with the direct method, then follow with the progressive method when the desired number of clusters is large. Compared to the result using no adaptation,

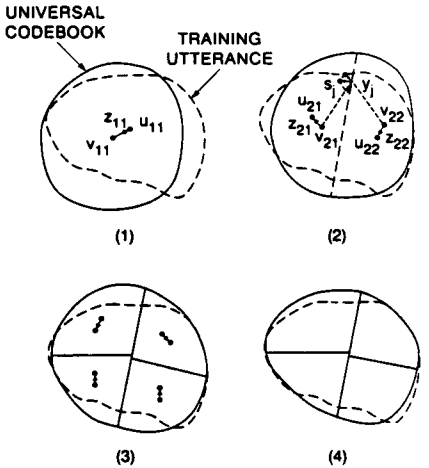



**Figure 5.18** Hierarchical codebook adaptation algorithm (after Furui [14]).

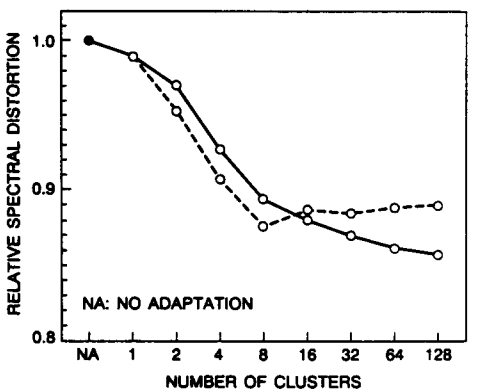



**Figure 5.19** Cepstral distortion between input speech pattern and reference templates resulted from hierarchical code-word adaptation (NA=no adaptation); — progressive adaptation, --- direct adaptation (after Furui [14]).

the reduction in spectral distortion due to adaptation is observed to be about 10–15%. The average VQ spectral distortion is highly correlated with the recognition error rate.

It is interesting to note that the hierarchical VQ clustering method relies on the



spectral similarity between the code words and the new cluster centers, and hence is essentially independent of the contextual environment in which the particular spectrum or code word appears. The spectral transformation algorithm discussed in Section 5.5.1, however, emphasizes the contextual correspondence and constraints more than the spectral similarity alone.

Template adaptation, as introduced earlier, can be viewed from a strictly statistical perspective. We will address statistical adaptation in Chapter 6 in our discussion of statistical modeling of speech signals.

**Exercise 5.5**

In hierarchical spectral adaptation using a Euclidean distance measure, a deviation vector $\mathbf{z}$ is calculated as

$$\mathbf{z} = \mathbf{u} - \mathbf{v}$$

where $\mathbf{u}$ is the centroid vector of the training data from a new talker and $\mathbf{v}$ is the centroid vector of the existing code words in the codebook $\mathbf{C} = \{\mathbf{y}_i\}_{i=1}^N$. If the deviation vector is used to shift the code words according to $\mathbf{y}_i' = \mathbf{y}_i + \mathbf{z}$, show that the centroid of $\{\mathbf{y}_i'\}_{i=1}^N$ is equal to $\mathbf{u}$.

**Solution 5.5**

By definition

$$\mathbf{v} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{y}_i.$$

The centroid of $\{\mathbf{y}_i'\}_{i=1}^N$ is $\overline{\mathbf{y}'}$,

$$\begin{aligned}\overline{\mathbf{y}'} &= \frac{1}{N}\sum_{i=1}^{N} \mathbf{y}_i' \\ &= \frac{1}{N}\sum_{i=1}^{N} \mathbf{y}_i + \mathbf{z} \\ &= \mathbf{v} + \mathbf{z} \\ &= \mathbf{u}.\end{aligned}$$

## 5.6 DISCRIMINATIVE METHODS IN SPEECH RECOGNITION

Speech-recognition tasks sometimes involve complex vocabularies in which phonetically similar words are the dominant source of recognition errors. For example, recognition of "spell mode" utterances, with vocabulary consisting of the letters of the English alphabet, would have problems with the following confusable sets:

$$\{\mathrm{A, J, K}\},\ \{\mathrm{B, C, D, E, G, P, T, V, Z}\},$$
$$\{\mathrm{Q, U}\},\ \{\mathrm{I, Y}\},\ \{\mathrm{L, M, N}\},\ \text{and}\ \{\mathrm{F, S, X}\}.$$

The problem is due to the acoustic similarity inherent in the spoken versions of the letters in each confusable set. This type of problem is essentially unrelated to the vocabulary size, because a large vocabulary could conceivably contain no phonetically confusing words, whereas a small vocabulary could contain many similar words.

Many phonetically confusable words can only be discriminated based on a small, critical portion of the utterances. For these cases, the traditional recognition approach, which relies on a simple accumulated distortion score over the entire utterance duration, generally does not place sufficient emphasis on the critical parts of the utterance. Let us consider a vocabulary in which each word is represented by a reference pattern $\mathcal{Y}_i$, $i = 1, 2, \ldots, M$. The sequence dissimilarity (distortion score) between an unknown pattern $\mathcal{X}$ and a reference pattern $\mathcal{Y}_i$ is defined in Eq. (4.158) by

$$d(\mathcal{X}, \mathcal{Y}_i) = \frac{1}{M_\phi} \min_{\phi_x, \phi_{y_i}} \sum_{k=1}^{T} d(\phi_x(k), \phi_{y_i}(k)) m(k) \tag{5.83}$$

where $\phi_x$ and $\phi_{y_i}$ are the time warping paths, $d(\phi_x(k), \phi_{y_i}(k))$ is the local frame-to-frame distortion, $m(k)$ is the *local* slope weighting and $M_\phi$ is a normalizing constant. Without loss of generality, we assume $\phi = (\phi_x, \phi_{y_i})$ is the optimal warping path pair, $m(k)$ is unity and $M_\phi = T$ such that

$$d(\mathcal{X}, \mathcal{Y}_i) = \frac{1}{T} \sum_{k=1}^{T} d(\phi_x(k), \phi_{y_i}(k)) \tag{5.84}$$

which is simply the average distortion along the optimal warping path.

We can examine the typical behavior of the distortion sequence $d(\phi_x(k), \phi_{y_i}(k))$, as a function of $k$, for the cases where $\mathcal{X}$ and $\mathcal{Y}_i$ are from the same word class and where they are from different word classes. In Figure 5.20, we plot several typical distortion sequences resulting from the dynamic time warping procedure. When $\mathcal{X}$ and $\mathcal{Y}_i$ are tokens of the same word, $d(\phi_x(k), \phi_{y_i}(k))$ varies around some expected value $\bar{d}_c = d(\mathcal{X}, \mathcal{Y}_i)$ (Figure 5.20a). (The subscript $c$ stands for correct.) When $\mathcal{X}$ and $\mathcal{Y}_i$ are from acoustically very different words, $d(\phi_x(k), \phi_{y_i}(k))$ is generally large, compared to $\bar{d}_c$, for all values of $k$ and the average distortion $\bar{d}_w$ (the subscript $w$ is for "wrong") is accordingly large. This case is illustrated in Figure 5.20b. However, when $\mathcal{X}$ and $\mathcal{Y}_i$ are from acoustically similar but linguistically different words, the distortions are, in general, quite close to those in the case of same word, except for the critical regions (i.e., when the words differ) where larger distortions are observed. Figure 5.20c clearly demonstrates an example of this type of situation. In the example, the acoustically dissimilar region occurs only at the beginning of the word (i.e., from frames 1 to $\hat{N}$).

It is obvious that if the critical region (where the words being compared differ) is short compared to the length of the entire utterance, the recognition decision based on the average distortion of Eq. (5.84) can be unreliable. The key question then is how the dissimilarity measure between two speech patterns can be modified so that the critical



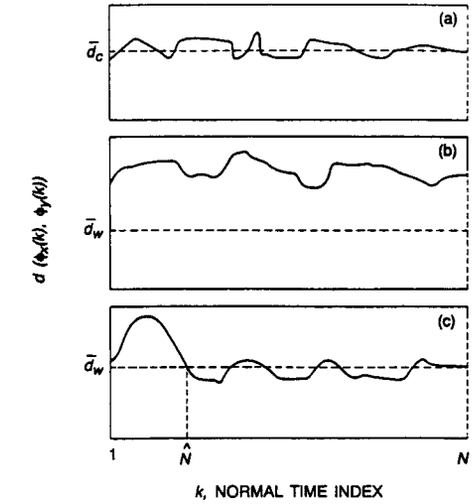



**Figure 5.20** Frame-distortion sequences between pairs of speech utterances: (a) utterances of the same word, (b) utterances of acoustically different words, and (c) utterances of acoustically confusing words (after Rabiner and Wilpon [15]).

regions of the utterance (which discriminate the words being compared) receive proper emphasis for optimal recognition results. One possible solution is to include a global, discriminative weighting in the calculation of the pattern dissimilarity measure; that is, we consider

$$d(\mathcal{X}, \mathcal{Y}_i) = \frac{\sum_{k=1}^{T} w_i(k)\, d(\phi_x(k), \phi_{y_i}(k))}{\sum_{k=1}^{T} w_i(k)} \tag{5.85}$$

where $w_i(k)$ is a weighting function to be determined. This is equivalent to the classical linear discriminant analysis and design for pattern recognition [15]. It is important to note that the local slope weighting, $m(k)$, normally used in the global dissimilarity measure is defined according to a general local constraint which is equally applied to all the vocabulary words. The goal of slope weighting is essentially to achieve reasonable time alignment and normalization. The discriminative weights $w_i(k)$ of Eq. (5.85), however, are template specific and are designed to provide a better recognition decision.

An alternative to the use of discriminant analysis is to design the reference tem-

plate so that the parameter values in the template are essentially preweighted so as to be able to discriminate differences with all other words in the vocabulary. This procedure requires a methodology that allows us to optimize the template parameter values for best discrimination among acoustically similar words. Both discriminative weighting and discriminative template optimization are essential components of a method generally called "discriminative training."

In this section, we discuss several key aspects of discriminative training. These are determination of word equivalence classes in a given vocabulary, choice of weighting functions for discrimination among confusing words, and a general discriminative training framework that offers an analytic perspective on how an "optimal" speech recognizer can be potentially designed to achieve a minimum error rate performance.

### 5.6.1 Determination of Word Equivalence Classes

Given a vocabulary of $M$ words, we would like to form a set of equivalence classes, $\phi_j$, $j = 1, 2, \ldots, J$ where $J \leq M$ such that the words belonging to the same class are similar to each other acoustically and phonetically. There are two reasonable approaches to this problem; one is phonetically based and the other acoustically based.

To form equivalence classes for words, it is necessary to define a "word-by-word" dissimilarity measure first. For the phonetically based approach, such a word-by-word dissimilarity can be derived on the basis of the phonetic description of the vocabulary words. To do this, we define a "phoneme distance" matrix, the elements of which are the distances for various phoneme pairs. We denote the phoneme distance by $d_p(\cdot, \cdot)$. There are several ways to obtain the set of phoneme distances. They could be defined by the count of the number of phonetic features that have to be changed to convert one phoneme into another. Alternatively, one could manually segment the words according to their phonetic descriptions and calculate the distortions directly from the phoneme segments. In addition, we need to define a distance cost for inserting a phoneme, $d_I$, and a distance cost for deleting a phoneme, $d_D$. A total word-by-word dissimilarity is then defined by a dynamic time-warp match between the words, with a vertical step in the warping path representing an insertion and a horizontal step representing a deletion (and vice versa, depending on the chosen warping direction).

Figure 5.21a illustrates this phonetically based procedure for the digit word "eight" and letter "J" and Figure 5.21b for the words "one" and "nine." For the words "eight" ($/e^y t/$) and "J" ($/je^y/$), the alignment path encompasses an insertion (of $j$), a match between the $/e^y/$ in "eight" and the $/e^y/$ in "J," and a deletion of $/t/$, giving a distance

$$d(/e^y t/, /je^y/) = \frac{d_I + d_p(e^y, e^y) + d_D}{3} = \frac{d_I + d_D}{3} \tag{5.86}$$



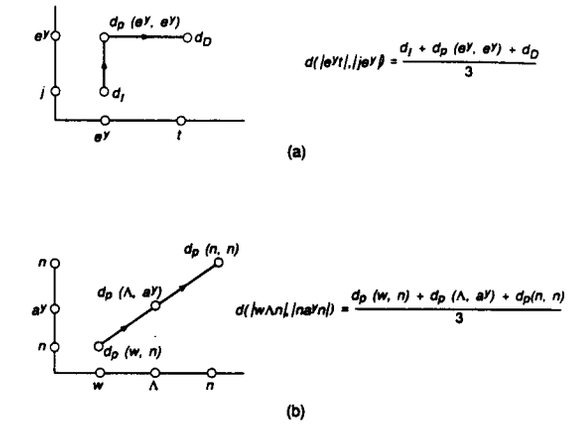



**Figure 5.21** Examples illustrating "word" alignment based on dynamic "phone" warping for word equivalence class clustering (after Rabiner and Wilpon [15]).

where we assume $d_p(e^y, e^y) = 0$. For the words "one" ($/w \Lambda n/$) and "nine"($/n a^y n/$), the alignment path is a straight line giving

$$d(/w \Lambda n/, /na^y n/) = \frac{d_p(w, n) + d_p(\Lambda, a^y) + d_p(n, n)}{3} = \frac{d_p(w, n) + d_p(\Lambda, a^y)}{3} \tag{5.87}$$

where, again, $d_p(n, n) = 0$ is assumed. This phonetically based approach, in theory, provides a simple method of generating equivalence classes directly from the phonetic descriptions of the words, without requiring actual acoustic realizations of the utterances.

The acoustically based approach to obtaining the word-by-word dissimilarity is to use real tokens (utterances) of the vocabulary words and perform actual dynamic time warping on the spectral patterns to calculate the word distances. If several tokens have been recorded, averaging of distances increases the reliability of the final results.

From the word-by-word distance matrices, equivalence classes can be obtained using the clustering procedures described in Section 5.3. The basic concept is that the vocabulary words are grouped into clusters based on pairwise (word pairs) distances. An example of the results of the clustering technique is illustrated below for the 39-word vocabulary consisting of the 26 letters of the English alphabet, the 10 digits, and the 3 command words "STOP," "ERROR," and "REPEAT":

| | Tokens |
|---|---|
| $\phi_1 = \{B, C, D, E, G, P, T, V, Z, 3, REPEAT\}$ | 11 |
| $\phi_2 = \{A, H, J, K, 8\}$ | 5 |
| $\phi_3 = \{F, S, X, 6\}$ | 4 |
| $\phi_4 = \{I, Y, 4, 5\}$ | 4 |
| $\phi_5 = \{Q, U, 2\}$ | 3 |
| $\phi_6 = \{L, M, N\}$ | 3 |
| $\phi_7 = \{O\}$ | 1 |
| $\phi_8 = \{R\}$ | 1 |
| $\phi_9 = \{W\}$ | 1 |
| $\phi_{10} = \{STOP\}$ | 1 |
| $\phi_{11} = \{ERROR\}$ | 1 |
| $\phi_{12} = \{0\}$ | 1 |
| $\phi_{13} = \{1\}$ | 1 |
| $\phi_{14} = \{7\}$ | 1 |
| $\phi_{15} = \{9\}$ | 1 |

Having obtained the equivalence classes, one can further define a pattern-to-class distance for an unknown utterance pattern in order to provide an efficient first-pass decision that determines the equivalence set of the unknown pattern before a final recognition result is computed. The pattern-to-class distance can be determined in several ways. One simple form of a pattern-to-class distance is the minimum of the pattern-to-word distances for all the words in the equivalence class—that is, for an unknown pattern $\mathcal{X}$,

$$d(\mathcal{X}, \phi_j) = \min_{\mathcal{Y}_i \in \phi_j} d(\mathcal{X}, \mathcal{Y}_i) \tag{5.88}$$

where $\mathcal{Y}_i \in \phi_j$ means the word which $\mathcal{Y}_i$ represents is in the equivalence class $\phi_j$. Alternatively, one can apply the $k$-nearest-neighbor rule to all the reference templates representing the words in the equivalence class and use a $k$-nearest-neighbor average distance (Eq. (5.60–5.61)) as the pattern-to-class distance. This simple form of pattern-to-class distance does not necessarily lead to computational savings because pattern comparison is performed over the entire set of reference templates.

Another method of obtaining the pattern-to-class distances is by explicitly defining "class-reference" templates (apart from the word-reference templates) and calculating the distances from the unknown pattern directly using the class-reference templates. Class-reference templates can also be obtained using clustering methods. In the case of multiple templates per class, again a $k$-nearest-neighbor rule can be used effectively. A ranking based on $d(\mathcal{X}, \phi_j)$ $j = 1, 2, \ldots, M$ for each unknown pattern $\mathcal{X}$ can be created for further discriminative processing.

Since the number of word classes is generally smaller than the number of words, the number of distance calculations required to establish pattern-to-class distances is smaller for class templates than for word templates. For example, for the 39-word vocabulary described above, there are 15 word equivalence classes. Hence, there is almost a 3-to-1 reduction from words to word classes in terms of template comparison. It should be noted, however, that the computational savings come with the risk that errors could occur in the fast, first-pass decision stage. Such errors are generally irreversible. This situation is similar to the VQ preprocessor case discussed in Section 5.2.5.



### 5.6.2 Discriminative Weighting Functions

Once a speech utterance is determined to be a member of an equivalence class of confusing words, discriminative weighting as defined in Eq. (5.85) can be applied to maximize the likelihood of correct recognition. The key question is how we choose the weighting function $w_i(k)$ to accomplish this goal.

We have already mentioned that methods for the design of the "optimal" weighting function, based on a set of training data, have been proposed based on classical linear discriminant analysis [12]. We discuss, in this section, an approach to the design of discriminative weights using the concept of Fisher's linear discriminant.

Assume we are given a training set, which we divide into two subsets $\{\mathcal{X}_j^{(+i)}\}$ and $\{\mathcal{X}_j^{(-i)}\}$ with $N^{(+i)}$ and $N^{(-i)}$ utterances, respectively. Speech utterances with superscript $(+i)$ are those that correspond to (are labeled as being closest to) template $\mathcal{Y}_i$, which in turn represents a certain word. The superscript $(-i)$ denotes those that are not assigned to $\mathcal{Y}_i$. We use vector notation

$$\mathbf{w}_i = (w_i(1), w_i(2), \ldots, w_i(T))^t \tag{5.89}$$

and

$$\mathbf{d}_{ij}^{(*)} = \left(d_{ij}^{(*)}(1), d_{ij}^{(*)}(2), \ldots, d_{ij}^{(*)}(T)\right)^t, \quad * = \pm i. \tag{5.90}$$

where $t$, as usual, indicates matrix (vector) transpose and

$$d_{ij}^{(*)}(k) = d\left(\phi_{x_j}^{(*)}(k), \phi_{y_i}(k)\right), \quad k = 1, 2, \ldots, T, \ * = \pm i \tag{5.91}$$

with $\phi_{x_j}^{(*)}(k)$ denoting the warping path (together with $\phi_{y_i}$) for aligning $\mathcal{X}_j^{(*)}$ and $\mathcal{Y}_i$. Note that $\mathbf{d}_{ij}^{(+i)}$ is a distance vector with elements obtained from matching two utterances of the same word represented by $\mathcal{Y}_i$ while $\mathbf{d}_{ij}^{(-i)}$ is a distance vector resulted from comparing a nonmatch utterance $\mathcal{X}_j^{(-i)}$ with the reference template $\mathcal{Y}_i$. We further assume that $\mathbf{w}_i$ has been properly normalized so that the modified discriminative measure of Eq. (5.85) can be written as

$$d(\mathcal{X}_j^{(+i)}, \mathcal{Y}_i) = \mathbf{w}_i^t \mathbf{d}_{ij}^{(+i)} \tag{5.92}$$

with a similar expression for $d(\mathcal{X}_j^{(-i)}, \mathcal{Y}_i)$. This is equivalent to projection of $T$-dimensional vectors onto a line in the direction of $\mathbf{w}_i$. Note that the recognition decision is based on the principle of *minimum* weighted distance.

Let us consider the following sample means of the distance vectors

$$\boldsymbol{\mu}^{(+i)} \triangleq \left(\mu^{(+i)}(1), \mu^{(+i)}(2), \ldots, \mu^{(+i)}(T)\right)^t \triangleq \frac{1}{N^{(+i)}} \sum_{j=1}^{N^{(+i)}} \mathbf{d}_{ij}^{(+i)} \tag{5.93a}$$

and

$$\mu^{(-i)} \triangleq \left(\mu^{(-i)}(1), \mu^{(-i)}(2), \ldots, \mu^{(-i)}(T)\right)' = \frac{1}{N^{(-i)}} \sum_{j=1}^{N^{(-i)}} \mathbf{d}_{ij}^{(-i)}. \tag{5.93b}$$

After projection, we have two mean values (scalar) for the two subsets of training data

$$\mu^{(+i)} = \mathbf{w}_i' \mu^{(+i)} = \frac{1}{N^{(+i)}} \sum_{j=1}^{N^{(+i)}} \mathbf{w}_i' \mathbf{d}_{ij}^{(+i)} \tag{5.94a}$$

and

$$\mu^{(-i)} = \mathbf{w}_i' \mu^{(-i)} = \frac{1}{N^{(-i)}} \sum_{j=1}^{N^{(-i)}} \mathbf{w}_i' \mathbf{d}_{ij}^{(-i)}. \tag{5.94b}$$

We define the "scatter" for the projected distance vectors by

$$s^{(*)} = \sum_{j=1}^{N^{(*)}} \left(\mu^{(*)} - \mathbf{w}_i' \mathbf{d}_{ij}^{(*)}\right)^2, \qquad * = \pm i. \tag{5.95a}$$

The quantity $s^{(+i)} + s^{(-i)}$ is called the total *within-class scatter* and $(s^{(+i)} + s^{(-i)}) / (N^{(+i)} + N^{(-i)})$ is an estimate of the variance of the pooled, projected data. A measure of separation can be defined as

$$\theta(\mathbf{w}_i) = \frac{(\mu^{(+i)} - \mu^{(-i)})^2}{s^{(+i)} + s^{(-i)}}. \tag{5.95}$$

In Eq. (5.95), the separation measure is expressed as a function of $\mathbf{w}_i$ and can be maximized by choosing

$$\mathbf{w}_i = \mathbf{S}^{-1}(-\mu^{(+i)} + \mu^{(-i)}) \tag{5.96}$$

where $\mathbf{S}$ is the total scatter matrix,

$$\mathbf{S} \triangleq \left[\sum_{j=1}^{N^{(+i)}} \left(\mathbf{d}_{ij}^{(+i)} - \mu^{(+i)}\right)\left(\mathbf{d}_{ij}^{(+i)} - \mu^{(+i)}\right)'\right] + \left[\sum_{j=1}^{N^{(-i)}} \left(\mathbf{d}_{ij}^{(-i)} - \mu^{(-i)}\right)\left(\mathbf{d}_{ij}^{(-i)} - \mu^{(-i)}\right)'\right]. \tag{5.97}$$

The linear discriminant based on the optimal $\mathbf{w}_i$ of Eq. (5.96) is called the *Fisher linear discriminant*. Note that the result of Eq. (5.96) also implies that $\mu^{(+i)} < \mu^{(-i)}$ for the weight vector to be consistent with the *minimum* distance decision rule.

If we assume $d_{ij}(k)$, $k = 1, 2, \ldots, T$ are independent (or uncorrelated in the case of a Gaussian distribution model), $\mathbf{S}$ has only nonzero elements on the diagonal and $w_i(k)$, $k = 1, 2, \ldots, T$ can be calculated separately according to Eq. (5.96).

By way of example, Figure 5.22 shows several plots of $\mu^{(+i)}$ and $\mu^{(-i)}$ for some typical classes; the horizontal axis represents the frame number in terms of the "normal"



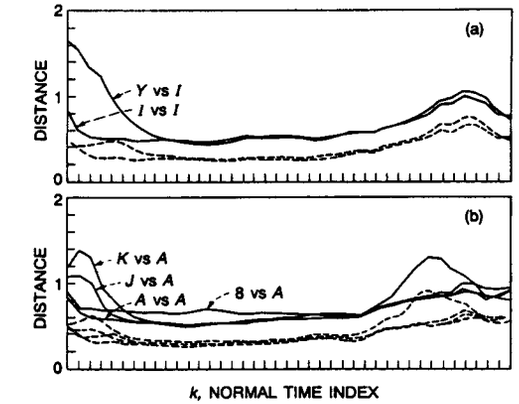



**Figure 5.22** Plots of average distortion sequences (—) and sequences of standard deviations of the frame distortion (- - - -) for various word pairs (after Rabiner and Wilpon [15]).

time index $k = 1, 2, \ldots, T$ and the vertical axis is the distance value. Also shown in the plots as dashed lines are the standard deviations

$$\sigma^{(*)}(k) = \left[\sum_{j=1}^{N^{(*)}} \left(d_{ij}^{(*)}(k) - \mu^{(*)}(k)\right)^2\right]^{1/2}, \qquad k = 1, 2, \ldots, T, \ * = \pm i. \tag{5.98}$$

In Figure 5.22a where the class of $\{Y, I\}$ is represented, we can see that $\mu^{(+i)}(k)$ is approximately constant whereas $\mu^{(-i)}(k)$ differs from $\mu^{(+i)}(k)$ only at the beginning of the word. We also see that the curves of $\sigma^{(*)}(k)$ are comparable for cases $* = +i$ and $* = -i$ with only small differences occurring in the first few frames. In Figure 5.22b, we show the cases where the letter "A" has been contrasted with the letters "J" and "K" and the digit word "8." Behavior similar to that of Figure 5.22a is seen, in that $\mu^{(-i)}(k)$ is larger than $\mu^{(+i)}(k)$ at the beginning of the word for letters "J" and "K," and at the end of the word for word "8." For the word "8," the curve of $\sigma^{(-i)}(k)$ is also fairly large at the end of the word, indicating the high degree of variability in the plosive release of the word "8."

A variation of the weighting function of Eq. (5.96) has the form

$$w_i(k) = \left|\mu^{(+i)}(k) - \mu^{(-i)}(k)\right| \Big/ \left[(\sigma^{(+i)}(k))^2 + (\sigma^{(-i)}(k))^2\right]^{1/2} \tag{5.99}$$

where $\sigma^{(*)}(k)$, $* = \pm i$, are defined in Eq. (5.98). This weighting function, studied in [15], has the same motivation as Fisher's discriminant, in that the normalized separation is the primary focus. Figure 5.23 shows a set of weighting curves according to Eq. (5.99) for the letters "I" and "Y." Several interesting properties of the curves can be noted. A large pulse is present at the beginning of the curves, followed by a residual tail. Clearly, the tail represents

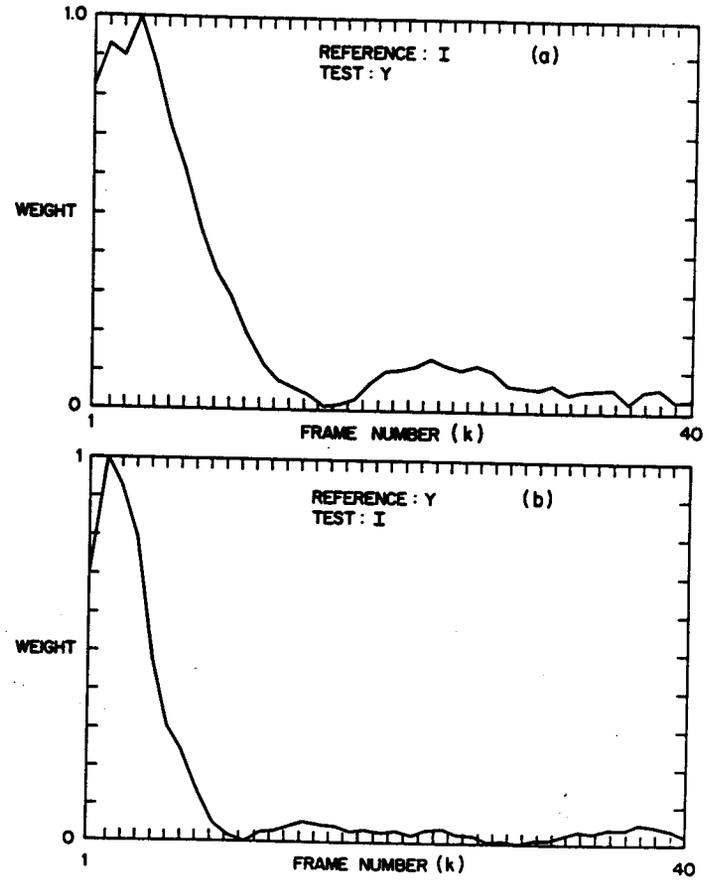



**Figure 5.23** Weighting curves for discriminatively comparing words "I" and "Y" (after Rabiner and Wilpon [15]).

the inherent statistical variation between $\mathbf{d}_{ij}^{(+i)}$ and $\mathbf{d}_{ij}^{(-i)}$ in the acoustically similar region. The pulse indicates clearly the place to be emphasized for maximal discrimination between the two words. The weighting function of Eq. (5.96) shows similar behavior except that the range of $w_i(k)$ is no longer limited to $w_i(k) \geq 0$.

**Exercise 5.6**

Show that the separation measure of Eq. (5.95) is maximized by choosing the solution of Eq. (5.96).



**Solution 5.6**

We express $\theta(\mathbf{w}_i)$ as a function of $\mathbf{w}_i$. Note that according to Eq. (5.94),

$$\mu^{(+i)} = \mathbf{w}_i^t \boldsymbol{\mu}^{(+i)}$$

and

$$\mu^{(-i)} = \mathbf{w}_i^t \boldsymbol{\mu}^{(-i)}$$

which leads to

$$\begin{aligned}(\mu^{(+i)} - \mu^{(-i)})^2 &= (\mathbf{w}_i^t \boldsymbol{\mu}^{(+i)} - \mathbf{w}_i^t \boldsymbol{\mu}^{(-i)})^2 \\ &= \mathbf{w}_i^t(\boldsymbol{\mu}^{(+i)} - \boldsymbol{\mu}^{(-i)})(\boldsymbol{\mu}^{(+i)} - \boldsymbol{\mu}^{(-i)})^t \mathbf{w}_i \\ &= \mathbf{w}_i^t \mathbf{M} \mathbf{w}_i\end{aligned}$$

where

$$\mathbf{M} = (\boldsymbol{\mu}^{(+i)} - \boldsymbol{\mu}^{(-i)})(\boldsymbol{\mu}^{(+i)} - \boldsymbol{\mu}^{(-i)})^t.$$

The denominator of $\theta$ is $s^{(+i)} + s^{(-i)}$ with

$$\begin{aligned}s^{(+i)} &= \sum_{j=1}^{N^{(+i)}} (\mathbf{w}_i^t \boldsymbol{\mu}^{(+i)} - \mathbf{w}_i^t \mathbf{d}_{ij}^{(+i)})^2 \\ &= \sum_{j=1}^{N^{(+i)}} \mathbf{w}_i^t (\boldsymbol{\mu}^{(+i)} - \mathbf{d}_{ij}^{(+i)})(\boldsymbol{\mu}^{(+i)} - \mathbf{d}_{ij}^{(+i)})^t \mathbf{w}_i\end{aligned}$$

and

$$s^{(-i)} = \sum_{j=1}^{N^{(-i)}} \mathbf{w}_i^t (\boldsymbol{\mu}^{(-i)} - \mathbf{d}_{ij}^{(-i)})(\boldsymbol{\mu}^{(-i)} - \mathbf{d}_{ij}^{(-i)})^t \mathbf{w}_i.$$

Thus,

$$s^{(+i)} + s^{(-i)} = \mathbf{w}_i^t \mathbf{S} \mathbf{w}_i$$

and

$$\theta(\mathbf{w}_i) = \frac{\mathbf{w}_i^t \mathbf{M} \mathbf{w}_i}{\mathbf{w}_i^t \mathbf{S} \mathbf{w}_i}.$$

The separation measure $\theta$ is in a form known as the generalized Rayleigh quotient and is maximized by a solution $\mathbf{w}_i$ which satisfies

$$\mathbf{M}\mathbf{w}_i = \lambda \mathbf{S} \mathbf{w}_i.$$

Since $\mathbf{M}$ projects any vector onto the direction of $\boldsymbol{\mu}^{(-i)} - \boldsymbol{\mu}^{(+i)}$ (or $\boldsymbol{\mu}^{(+i)} - \boldsymbol{\mu}^{(-i)}$), the above equation becomes

$$\lambda(\boldsymbol{\mu}^{(-i)} - \boldsymbol{\mu}^{(+i)}) = \lambda \mathbf{S} \mathbf{w}_i$$

and thus

$$\mathbf{w}_i = \mathbf{S}^{-1}(\boldsymbol{\mu}^{(-i)} - \boldsymbol{\mu}^{(+i)}).$$

### 5.6.3 Discriminative Training for Minimum Recognition Error

The classical pattern classifier or recognizer architecture is based on two fundamental steps: definition of an appropriate discriminant function $g_i(\mathbf{x}; \Lambda)$ parametrized on $\Lambda$ for each class $i$, $i = 1, 2, \ldots, M$, and use of the discriminant function in implementing the decision rule, which is often generically stated as

$$C(\mathbf{x}) = C^i \quad \text{iff } g_i(\mathbf{x}; \Lambda) = \max_j g_j(\mathbf{x}; \Lambda), \tag{5.100}$$

where $C(\mathbf{x})$ denotes the recognition decision on $\mathbf{x}$ and $C^i$ denotes class $i$. In speech-recognition applications, where the decision is based on the principle of minimum distance, we can view $g_i(\mathbf{x}; \Lambda)$ as the distance between $\mathbf{x}$ and a vector (part of the parameter set $\Lambda$) representing class $i$. In this case, the maximum in the decision rule of Eq. (5.100) is replaced by a minimum. Furthermore, since a speech pattern is represented by a sequence of vectors, the discriminant function is defined for a sequence of vectors rather than a single vector; i.e. $g_i(\mathcal{X}; \Lambda)$ instead of $g_i(\mathbf{x}; \Lambda)$ with $\Lambda$ being the template parameter set. The parameter set includes the template spectral vectors and the discriminative weighting functions. The goal of discriminative training is to obtain the values in $\Lambda$ for the "best" recognition result.

Given a set of labeled samples $\{\mathcal{X}_j^{(i)}\}_{j=1}^N$, $i = 1, 2, \ldots, M$, training is usually accomplished by optimizing a prescribed criterion defined on the given labeled set. Since speech recognizers are generally evaluated by their recognition error rate, it is therefore desirable to use recognition error as the minimization criterion. For a finite training set, i.e. $N < \infty$, unfortunately, the recognition error count (or the error rate) is a piecewise constant function of the recognizer parameter set and cannot easily be minimized with traditional optimization methods. The following three-step procedure presents a reasonable way to define a parametrically smoothed recognition error function, capable of being minimized by straightforward gradient descent algorithms.

1. Define a set of discriminant functions $g_i(\mathcal{X}, \Lambda)$, $i = 1, 2, \ldots, M$.
2. Define a misrecognition/misclassification measure

$$h_i(\mathcal{X}; \Lambda) = -g_i(\mathcal{X}, \Lambda) + \left\{ \frac{1}{M-1} \sum_{j, j \neq i} g_j(\mathcal{X}; \Lambda)^\eta \right\}^{1/\eta} \tag{5.101}$$

   where $\eta$ is the smoothing parameter. (Note that here we use the generic decision rule of Eq. (5.100). When the maximum in Eq. (5.100) is replaced by a minimum, modification of Eq. (5.101) is necessary. See Eq. (5.115).)
3. Define a smooth 0–1 loss function $\ell_i(\mathcal{X}; \Lambda)$ based on the misrecognition measure, i.e.

$$\ell_i(\mathcal{X}; \Lambda) = \ell_i(h_i(\mathcal{X}; \Lambda)). \tag{5.102}$$

   Examples of smooth 0–1 functions include

   **a.** sigmoid function

$$\ell(h) = \frac{1}{1 + e^{-\xi(h+\alpha)}}, \qquad \xi > 0 \tag{5.103}$$



   **b.** hypertangent function

$$\ell(h) = \tanh(h). \tag{5.104}$$

The misrecognition measure $h_i$ enumerates the recognition operation of Eq. (5.100). The smooth 0–1 loss function then converts the enumerated recognition operation into a smoothed error count. During training, when a labeled sample is presented, a smooth error count is thus

$$\ell(\mathcal{X}; \Lambda) = \sum_{i=1}^{M} \ell_i(\mathcal{X}; \Lambda) 1(\mathcal{X} \in C^i) \tag{5.105}$$

where $1(\cdot)$ is the indicator function

$$1(\mathcal{E}) = \begin{cases} 1, & \text{if event } \mathcal{E} \text{ is true} \\ 0, & \text{otherwise} \end{cases} \tag{5.106}$$

and $C^i$ denotes the pattern set for class $i$. (We interchange freely the notations for the pattern set and the class identity without ambiguity.)

The recognizer performance can be defined as the expected value of the above loss function

$$\mathcal{L}(\Lambda) = E\{\ell(\mathcal{X}; \Lambda)\}. \tag{5.107}$$

This performance criterion can be minimized by an adaptive, gradient descent scheme in which the parameter set $\Lambda$ is sequentially adjusted by a small amount each time a labeled training sample is evaluated. Let $\Lambda_{j+1}$ be the parameter set after $\mathcal{X}_j$ is applied. The adjustment rule for obtaining $\Lambda_{j+1}$ is

$$\Lambda_{j+1} = \Lambda_j + \Delta\Lambda_j \tag{5.108}$$

where

$$\Delta \Lambda_j = -\epsilon_j \bar{\mathbf{U}} \nabla \ell(\mathcal{X}_j; \Lambda_j). \tag{5.109}$$

In Eq. (5.109), $\epsilon_j$ is a small positive number satisfying certain stochastic convergence constraints [16], $\bar{\mathbf{U}}$ is a positive definite matrix and $\nabla$ is the gradient operator with respect to the parameter set $\Lambda$. It can be shown [16] that the adjustment rule of Eq. (5.108) leads to the following results:

1. $E[\mathcal{L}(\Lambda_{j+1}) - \mathcal{L}(\Lambda_j)] \leq 0$ (5.110)
2. $\Lambda_j$ converges with probability one to $\Lambda^*$ which results in a local minimum of $\mathcal{L}(\Lambda)$.

For discriminative training of DTW templates as well as the accompanying weighting functions, we need to make explicit the definitions involved in the above three-step procedure. Suppose each vocabulary word $C^i$ is to be represented by $N_i$ reference templates, each being a pair $(\mathcal{Y}_{in}, \mathbf{w}_{in})$, $n = 1, 2, \ldots, N_i$. The recognizer parameter set $\Lambda$ then consists of all the template sets $\{[(\mathcal{Y}_{in}, \mathbf{w}_{in}), n = 1, 2, \ldots, N_i], i = 1, 2, \ldots, M\}$. We define a word (class) distance, to be used as the discriminant function for word-based templates, between

an utterance $\mathcal{X}$ and a word $C^i$ as

$$g_i(\mathcal{X}, \Lambda) = \log \left\{ \frac{1}{N_i} \sum_{n=1}^{N_i} \exp(-f_{in}\eta_w) \right\}^{-1/\eta_w} \tag{5.111}$$

where $f_{in}$ is the template distance, as defined below, and $\eta_w$ is the smoothing factor for combining the template distances into a word distance. Note that if $\eta_w \to \infty$, $g_i(\mathcal{X}, \Lambda)$ is equivalent to the minimum template distance and the factor $1/N_i$ becomes inconsequential. The template distance is obtained from path distances as

$$f_{in} = f_{in}(\mathcal{X}, (\mathcal{Y}_{in}, \mathbf{w}_{in})) = \log \left\{ \frac{1}{\Phi} \sum_{\ell=1}^{\Phi} \exp(-d_{in\ell}\eta_t) \right\}^{-1/\eta_t} \tag{5.112}$$

where $\Phi$ is the total number of warping paths between pattern $\mathcal{X}$ and reference template $\mathcal{Y}_{in}$, $\eta_t$ is, similar to $\eta_w$, the smoothing factor for combining path distances $d_{in\ell}$. The path distance $d_{in\ell}$ is defined in the same way as Eq. (5.92)

$$d_{in\ell} = d_{\phi^\ell}(\mathcal{X}, \mathcal{Y}_{in}) = \sum_{k=1}^{T} d(\phi_x^\ell(k), \phi_y^\ell(k)) w_{in}(k)/M_{\phi^\ell} \tag{5.113}$$

where $d(\phi_x^\ell(k), \phi_y^\ell(k))$ is the frame-to-frame (local) spectral distance measure and $w_{in}(k)$ is the discriminative weighting function. Again, the value of $\eta_t$ can be varied to adjust the relative significance of path distances in calculating the template distance of Eq. (5.112). In the extreme case, $\eta_t \to \infty$ and $f_{in}$ becomes identical to the minimum distance along the optimal time warping path

$$f_{in}(\mathcal{X}, (\mathcal{Y}_{in}, \mathbf{w}_{in})) = \min_{\phi^\ell} d_{\phi^\ell}(\mathcal{X}, \mathcal{Y}_{in}). \tag{5.114}$$

Therefore, the traditional dynamic time warping distance defined between two speech patterns is a special case of the parametrically smoothed template distance of Eq. (5.112).

Having defined the discriminant functions $g_i(\mathcal{X}, \Lambda)$, we proceed to define the misrecognition measure

$$h_i(\mathcal{X}, \Lambda) = g_i(\mathcal{X}, \Lambda) - \log \left\{ \frac{1}{M-1} \sum_{j,\, j \neq i} \exp\{-g_j(\mathcal{X}, \Lambda)\eta\} \right\}^{-1/\eta} \tag{5.115}$$

which differs from Eq. (5.101) due to the fact that $g_i(\mathcal{X}, \Lambda)$ is a form of distance and the decision is based on a minimum distance rule; i.e.

$$C(\mathcal{X}) = C^i, \quad \text{iff } g_i(\mathcal{X}, \Lambda) = \min_j g_j(\mathcal{X}, \Lambda). \tag{5.116}$$

The difference is only in form rather than substance. Finally, the smooth 0–1 loss function can be any of the above mentioned functions of Eq. (5.103), (5.104).

The above formulation enables us to implement adaptive discriminative training according to the scheme of Eq. (5.108–109). We do not provide explicit derivations of



$\nabla \ell(\mathcal{X}; \Lambda)$ here. For implementational details, Ref. [16] should be consulted. We conclude this section by giving an interpretation of the adaptation scheme. Note that $\ell(\mathcal{X}_j; \Lambda)$ evaluates the "error probability" achieved by the recognizer with parameter $\Lambda$, when a labeled token pattern $\mathcal{X}_j$ is presented for training. The gradient $\nabla \ell(\mathcal{X}_j; \Lambda)$ determines the direction in which the recognizer parameters should be changed to increase or decrease this (approximate) error probability. The adaptation amount $\Delta\Lambda$ of Eq. (5.109) ensures that the (approximate) error probability will decrease in a probabilistic sense for the next token $\mathcal{X}_{j+1}$. As more training data is presented, the recognizer parameter set eventually converges to a solution which is at least a local optimum.

Adaptive discriminative training is a topic that is currently undergoing intensive research. This section was intended to present several of the key ideas behind the new training method, which in several interesting cases has proven effective and powerful.

## 5.7 SPEECH RECOGNITION IN ADVERSE ENVIRONMENTS

The performance of most speech-recognition systems, whose designs are predicated on assumptions about the ambient conditions, such as low noise background, degrades rapidly in the presence of noise and distortion. It is therefore important to consider the problem of speech recognition in adverse environments that are inevitable in real-world applications.

By way of example, Dautrich et al. [17] demonstrated that an isolated word recognizer trained in clean (virtually noise-free) conditions and capable of achieving a recognition accuracy of 95% had an order of magnitude increase in error rate when tested with noise-contaminated utterances at an SNR (signal-to-noise ratio) of 18 dB. Figure 5.24 summarizes the results of [17] and illustrates several key considerations in noisy speech recognition. First, as the curve (labeled MATCHING S/N) shows, a recognizer can provide good performance even in very noisy background conditions if the exact same testing condition (noise level) is used to provide the training material from which the reference patterns of the vocabulary are obtained. Second, the degradation in recognition performance due to noise can be significantly reduced if the stored reference patterns are trained under conditions that *approximate* the noisy test conditions (see curve labeled MISMATCHED S/N – TEST S/N = 18 dB where the abscissa represents the level of noise added to the reference patterns). Although these results indicate certain ways to get around noise problems, the inadequacy comes from the fact that training reference patterns under the exact matched noise condition of test utterances is seldom realistic and that the training of reference patterns based on noisy input leads to unacceptable performance when the test condition in fact involves no noise. Thus, the use of the term "adverse environments" implies unknown, mismatched, and often severe differences in environment and other variables between training and testing. Obviously, in considering speech recognition in adverse environments, the goal is to have an automatic recognizer with robust performance approaching that of matched conditions (as if the recognizer were trained and tested under the same conditions).

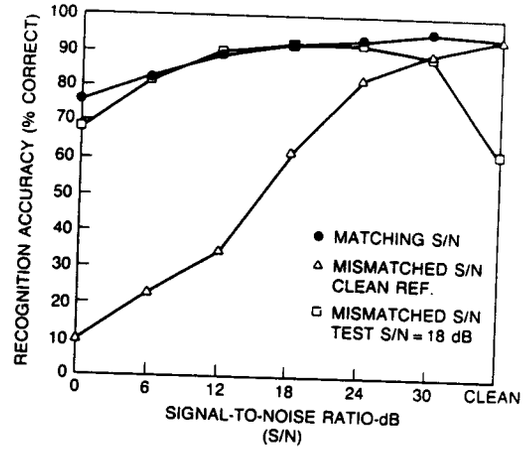



**Figure 5.24** Speech-recognition performance in noisy conditions; ●: training and testing have matched S/N as indicated by the abscissa; △: only clean training reference is used and the abscissa indicates that the test S/N; □: training and testing S/N's are mismatched with test S/N fixed at 18 dB and the abscissa indicates the training S/N (after Dautrich et al. [17]).

### 5.7.1 Adverse Conditions in Speech Recognition

A speech recognizer often encounters three main causes of adverse conditions: noise, distortion, and (human) articulation effects.

#### 5.7.1.1 Noise

Acoustic ambient noise is usually considered additive, meaning that the recorded signal is a sum of the speech signal and the ambient noise. High levels of ambient noise are one of the primary concerns for a speech recognizer. Sources of acoustic ambient noise are abundant. For example, in an office environment, sources of noise include office machinery such as a typewriter or printer, personal computers (PC), or workstations, which are usually equipped with moving components like disks and fans, telephone ringing and background conversation of other people. These noise sources often provide enough acoustic noise to cause severe performance degradation of a speech recognizer. The sound pressure level (SPL) in a normal personal office is around 45 ~ 50 dBA (noise criterion NC 40 ~ 45). Figure 5.25 shows a typical spectrum, fitted with a 16th-order all-pole smoothed model spectrum, of ambient noise recorded in a personal office with a SUN 3/110 on and operating. In a business office where secretarial duties are performed, the SPL could be 15–20 dB higher than that of the personal office. Inside an automobile, the acoustic noise level from the engine, cooling fan, wind, tire and road is usually considerably higher, particularly



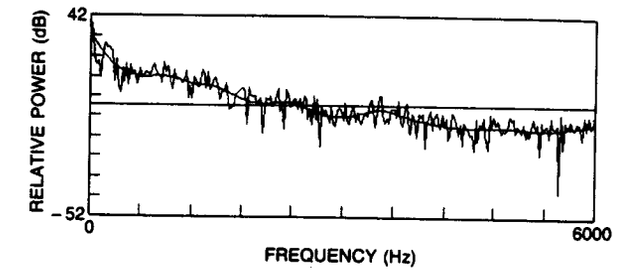



**Figure 5.25** Noise spectrum in a typical personal office with a SUN 3/110.

when the automobile is moving at high speeds. It is not unusual [18, 19] that the signal-to-noise ratio of speech signals recorded in a passenger car with a microphone mounted on the dashboard in front of the speaker/driver drops below 5 dB when the car is cruising at a highway speed of 90 km/h with the window closed and the fan turned off. In the cockpit of a modern jet fighter aircraft, SPLs of 90 dB or more across the speech frequency band have been reported. At this noise level, the speech signal is barely intelligible even to a human listener, not to mention an automatic speech-recognition machine.

The spectrum of acoustic ambient noise is usually not flat. In the case of car noise, the measurements of [18] indicate that while low-frequency noises generated by mechanical sources such as the engine, tire, and fan show a decaying trend in the power spectrum as a function of frequency, high-frequency noise due to aerodynamic phenomena essentially have a flat spectrum beyond 1 kHz.

Other types of noise such as electrical noise and quantization noise, which of course are present in any electronic speech-recognition system, are in general at a level below the threshold of concern. Nevertheless, noise due to transmission and switching equipment in a telephone network can sometimes be a factor affecting recognizer performance.

#### 5.7.1.2 Distortion

Aside from the additive contamination due to noiselike signals, the uttered signal inevitably undergoes a series of spectral distortions before being recorded and processed for speech recognition. (There are two kinds of distortions—linear and nonlinear—although most of our discussion will concentrate on linear distortions.) The room in which the speech recognizer unit is deployed almost certainly has a varying degree of reverberation that can alter the signal spectrum. The microphone transducer, depending on its type and mounting position, also can significantly distort the speech spectrum. When the transducer configuration used in testing is different from the one used during training of the reference patterns, the mismatch in spectral distortion becomes one of the major problems. For example, it was reported [20] that a large-vocabulary speech-recognition system with a baseline performance of 85% word accuracy in a matched transducer condition (Sennheiser HMD224 close-talking microphone for both training and testing) could only achieve less than 19% word accuracy when a different microphone (Crown PZM6fs desktop mounted)

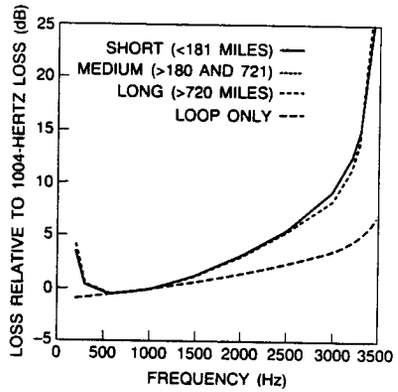



**Figure 5.26** Mean customer-premises-to-customer-premises attenuation distortion relative to 1004 Hz in a telephone channel (after Carey et al. [21]).

**TABLE 5.5.** Attenuation distortion relative to 1004 Hz: short connections.

| Freq. (Hz) | Mean | Standard Dev. | Quantile 1% | 10% | 50% | 90% | 99% |
|---|---|---|---|---|---|---|---|
| 204 | $5.1 \pm 2.9$ | 2.9 | 2.0 | 2.5 | 3.8 | 10.0 | 13.2 |
| 604 | $0.4 \pm 0.1$ | 0.3 | −0.6 | 0.0 | 0.4 | 0.9 | 1.3 |
| 1104 | $-0.1 \pm 0.1$ | 0.4 | −2.3 | −0.3 | −0.1 | 0.1 | 1.3 |
| 1604 | $-0.1 \pm 0.1$ | 0.4 | −1.2 | −0.5 | −0.1 | 0.2 | 1.0 |
| 2104 | $0.2 \pm 0.7$ | 0.7 | −1.4 | −0.4 | −0.1 | 1.0 | 2.5 |
| 2804 | $1.7 \pm 2.7$ | 2.6 | −0.6 | −0.1 | 0.3 | 5.8 | 10.0 |
| 3204 | $4.1 \pm 4.9$ | 4.6 | 0.2 | 0.5 | 1.5 | 11.5 | 18.2 |

was used during testing. (Note that in this case, noise and SNR are not characteristically matched either.)

When an automatic speech recognizer is deployed in a telephone network, the telephone channel through which the speech signal travels can cause further distortion of the signal spectrum. To illustrate typical spectral characteristics of a telephone channel, Figure 5.26 shows the mean customer-premises-to-customer-premises (end-office concatenated with loop) attenuation distortion relative to 1004 Hz, obtained in an end-office connection study and loop survey by a telephone carrier [21]. It is seen that the signal bandwidth is critically limited by the appropriate transmission system to approximately 200–3200 Hz. Within this frequency band, there is still a difference of 10 dB in spectral attenuation. Table 5.5 gives the statistics of attenuation distortion relative to 1004 Hz for short end-office connections (0–180 airline miles) alone, excluding the loop portion. At 204 Hz, the mean attenuation is 5.1 dB, but at the 10% and 90% quantile, the attenuation is 2.5 dB and 10 dB respectively. Similarly, at 3204 Hz, the mean attenuation is 4.1 dB,



with 0.5 dB and 11.5 dB attenuation at the 10% and 90% quantile, respectively. This wide range of variation in attenuation can obviously cause spectral mismatch distortion unless the telephone channel is measured/learned before every recognition trial.

#### 5.7.1.3 Articulation Effects

Many factors affect the manner of speaking of each individual talker. Even the psychological act of communicating with a speech-recognition machine could make the talker produce a noticeable difference in his or her sound formants and rhythmic pattern. Characteristic changes in articulation due to environmental influence, known as the Lombard effect, can also be dramatic. When a talker speaks in an environment with a masking noise of 90 dB SPL, Pisoni et al. reported [22] that the first formant of a vowel often increases while the second formant decreases, resulting in a potential shift in the vowel space. In a separate study [23], Junqua et al. estimated the increase in the first formant frequency due to an 85 dB SPL masking noise to be between 42 and 113 Hz. Furthermore, a significant change in spectral tilt was also observed [23]; spectral tilt in the high-band decreases, while that in the low-band increases for most of the vowels and liquids. These characteristic changes dramatically affect the performance of an automatic speech recognizer. A speaker-dependent isolated word recognizer that had an accuracy of better than 92% when the training and the testing were conducted under the same clean (virtually free from noise and the Lombard effect) condition could only achieve an accuracy of 61% when the test utterances contained the Lombard effect, even though the test tokens were free from masking noise [24]. (This result was obtained using an artificial experimental setup, which investigated the Lombard effect independent of all other effects.)

A major difficulty in dealing with articulation effects on speech is a lack of understanding as to how to quantify them. With acoustic noise or channel distortion, which usually does not vary as rapidly as the speech itself in terms of spectral characteristics, we can, to some extent, measure or model the effects so as to improve the recognizer design. For example, a white noise is usually specified by its power level, and a channel is often represented by its frequency response. Articulation effects, however, are an inherent product of the speech-production process and have been shown to be context dependent [23]. A qualitative characterization of the Lombard effect is not specific enough to provide an adequate guideline for enhancing the speech recognizer. As will be discussed later, most of the proposed solutions therefore do not have a firm theoretical foundation.

### 5.7.2 Dealing with Adverse Conditions

As pointed out previously, if the characteristics of the corrupting noise are approximately known, a speech recognizer that uses reference patterns trained from speech with the same corrupting noise components, in general, performs more robustly than one that uses clean reference patterns. This result can be extended to other cases in which the talker, because of his or her psychological reaction to environmental stress, produces speech in unusual talking styles (slow, fast, soft, loud, angry, and Lombard). The idea is simply to train the recognizer with a multi-style training procedure in which speech signals of different

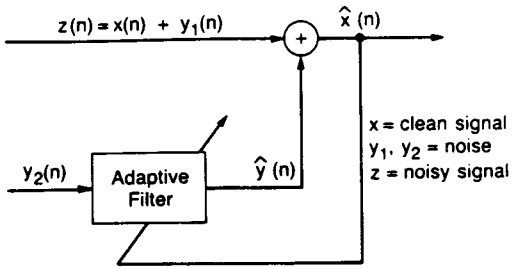



**Figure 5.27** Schematic of the two-input noise cancelling approach.

talking styles are used as the training data. In an isolated word-recognition experiment using hidden Markov models (see Chapter 6), Lippmann et al. [24] reported that the error rate of a recognizer, when tested with speech materials of various talking styles, was reduced by more than a factor of 2, from 17.5% using only normally-spoken reference patterns to 6.9% with multistyle training. Multistyle training seemed to be most effective with speech exhibiting the Lombard effect and with speech produced under highly emotional (angry) conditions.

We have argued that the use of training material reflecting actual deployment conditions usually is not practical. Therefore, straightforward solutions like noisy or multistyle training often do not solve the problem of robust speech recognition in adverse conditions. In the following, we discuss a number of methods and algorithms that have been proposed to combat the unknown environment in which a speech recognizer must operate.

### 5.7.2.1 Signal-Enhancement Preprocessing

When the adverse condition is due to additive noise alone, one can use well-established speech-enhancement methods to suppress the noise before applying the recognition algorithm. One of the most widely studied signal-enhancement methods is adaptive noise cancellation using two signal sources [25]. Figure 5.27 illustrates the principle of the two-input noise cancelling approach. The technique adaptively adjusts a set of filter coefficients so that subtracting the filtered noise input from the primary (speech plus noise) input leads to an output signal with minimum energy. This technique, however, requires that the noise component in the corrupted signal and the noise reference have high coherence. Inside a passenger car, it was found [26] that if the two microphones are located at a distance greater than 50 cm, the only coherent noise component is that of the engine. To cancel 90% of the noise energy, the two microphones cannot be more than 5 cm apart, which makes it almost impossible to prevent speech from being included in the noise reference. Therefore, one can only expect cancellation of noise related to the fundamental frequency of the engine revolution rate if the two microphones are spaced at a reasonable distance. This lack of noise coherence was also observed in the sound field of a fighter aircraft cockpit.

Other speech-enhancement techniques that do not rely on the existence of a simultaneous, separate noise reference have also been proposed. These techniques use some



estimate of the noise characteristics, such as noise power and/or SNR, to obtain improved spectral models of speech from noise-corrupted signals. (Many of these techniques are for general speech-enhancement purposes, not specifically for improving recognition performance.) The work of Porter and Boll [27] and that of Ephraim et al. [28] are worth noting as they have been directly applied to speech recognition.

The least-squares estimator of short-time independent spectral components of [27] differs from a traditional estimator in that the conditional mean of the spectral component is obtained from the sample average estimator of clean speech rather than from an assumed parametric distribution. The method uses a clean speech database and a noisy version of it by artificially adding noise to the clean set to construct a function that maps a noisy spectral component (at each single frequency) to a noise-suppressed value. For a specialized Gaussian noise model, the function can be calculated using just the clean speech database and an estimate of the SNR. (A similar technique was developed in [29] for simultaneously restoring an entire spectrum without assuming that spectral components are independent.) Under the condition that the signal and the noise levels were fixed and known (SNR=10 dB), the technique was shown to be able to reduce the recognition errors due to noise by about 75% (from ~40% to ~10%) for a speaker-dependent digit-recognition task.

The method of [28] is interesting in that the short-time noise level as well as the short-time all-pole spectral model of the clean speech are iteratively estimated to minimize the Itakura-Saito distortion between the noisy spectrum and a composite model spectrum, which is a sum of the estimated clean all-pole spectrum and the estimated noise spectrum. That is, if $|Z|^2$, $\sigma^2/|A|^2$, and $\lambda$ represent the noisy power spectral density, the all-pole model spectral density, and the noise power level respectively, the algorithm iteratively finds $\sigma/A$ and $\lambda$ such that

$$d(|Z|^2, \sigma^2/|A|^2 + \lambda) = \int_0^{2\pi} \left[ \frac{|Z(\omega)|^2}{\sigma^2/|A(e^{j\omega})|^2 + \lambda} - \ln \frac{|Z(\omega)|^2}{\sigma^2/|A(e^{j\omega})|^2 + \lambda} - 1 \right] \frac{d\omega}{2\pi} \tag{5.117}$$

is minimized. The resultant $\sigma/A$ is then used in the recognizer as the spectral measurement of speech. The method can be applied to the test utterances as well as the training utterances without explicit knowledge of the noise level. The main limitation appears to be the assumption of composite model spectrum for the noise-corrupted signal. In a speaker-dependent isolated word (alphabet plus digits) recognition experiment, the technique significantly improved the recognition accuracy, from 42% when unprocessed clean reference templates were used for 10 dB SNR test tokens to almost 70% when both the reference and the noisy test tokens were processed to produce the enhanced spectral measurement. Increased computational complexity is, however, one drawback of this method.

### 5.7.2.2 Special Transducer Arrangements

If the talker position is allowed to be held fixed, a noise-cancelling microphone can be effective in suppressing low-frequency noise in an automobile or aircraft cockpit environment. A noise-cancelling microphone is a specially designed dynamic microphone in which both sides of the diaphragm are exposed to the sound field such that sounds coming

from a relatively large distance (far field) are cancelled because the sound pressure causes virtually no net force on the diaphragm. For sound sources close to the microphone (near field), the back of the diaphragm is effectively shaded from the sound field and the sound pressure is received only by the front of the diaphragm. In a passenger car using a pressure-gradient noise-cancelling microphone (CONFIDENCER by Roanwell), Dal Degan and Prati [26] showed that the signal picked up is essentially noise free if the microphone is kept very close to the talker's mouth and parallel to the wavefront. But with a mere 10-cm shift and 30-degree rotation, the speech power drops by 15 dB, resulting in a performance degradation.

Other types of gradient microphones (first-order as well as second-order) were shown by Viswanathan and Henry [30] to be effective in a moderately noisy environment (95 dB SPL broadband acoustic noise) if the sensor location was optimized and fixed. When the noise is severe, as in a fighter aircraft cockpit at 105 dB SPL, Viswanathan and Henry [30] reported that the noise-cancelling microphone alone did not lead to satisfactory recognition results and suggested the use of a two-sensor input that combines an accelerometer output for low frequencies (up to ~1.5K Hz) and gradient microphone output for high frequencies (above ~1.5K Hz). The accelerometer is attached to the skin (often near the throat) of the talker to measure the skin vibration and is insensitive to acoustic noise. Another possibility, also suggested in [30], is to use the two sensor outputs in parallel to facilitate composite feature extraction, such as concatenated spectral coefficient vectors, for speech recognition. Although performance improvements were demonstrated with various multisensor arrangements, as compared to each single constituent transducer, the test was limited to matched training and testing conditions and thus cannot be extrapolated to the mismatched situation where the characteristics of the ambient noise may be varying due to changing flying speed and altitude, and adaptive training is not permitted.

#### 5.7.2.3 Noise Masking and Adaptive Models

In the presence of broadband noise, certain regions of the speech spectrum that are of a lower level will be more affected by the noise. This makes the calculation of spectral distortion (as a measure of spectral dissimilarity) difficult because those more corrupted regions represent less reliable spectral measurements. Recognizing this difficulty, Klatt [31] advocated use of noise masking in conjunction with a filter-bank analyzer. The key idea was to first choose, for each channel of the filter-bank output, the masking noise level as the greater of the noise level in the reference signal and that in the testing signal, and then replace that channel output by the mask value if it is below the corresponding mask level. This helps prevent spurious distortion accumulation because those channels that are determined to have been seriously corrupted by noise will have the same spectral value in both the training and the testing tokens. Many noise-compensation schemes are based on this principle.

Klatt's masking scheme, however, has practical limitations, particularly when the two patterns being compared have very different noise levels. When the test token is contaminated by a high level of noise, all the reference patterns that are of lower level than the noise would result in equally small distances, making the comparison meaningless.



Subsequent improvements to Klatt's technique were proposed by Bridle et al. [32] and Holmes and Sedgwick [33]. These revisions incorporated the following measures to overcome the above limitation: (1) maintain a running estimate of the noise spectrum, both during training and testing; (2) separately mark, rather than mask, the channel spectral values of the training and the testing tokens as speech or noise according to respective noise estimates; (3) devise individual distance-calculation rules for different marking situations (i.e., speech-speech, speech-noise, noise-speech, and noise-noise); and (4) in case of multiple reference patterns, use the maximum of the noise estimates for the particular channel during training.

A separate technique of noise compensation was employed by Roe [19] to adapt the spectral prototypes to the noise condition in the autocorrelation domain. The technique is reminiscent of the spectral transformation ideas for VQ code words discussed in Section 5.5.1 and is elaborated below. The underlying assumption, unlike the above masking model, was that the power spectra of speech and noise are additive and so are the autocorrelations. Roe obtained improvements in recognition accuracy with this simple method. Roe's method included other features that applied to stress compensation as discussed in the next section.

#### 5.7.2.4 Stress Compensation

The purpose of stress compensation is to provide offset for the spectral distortion caused by extraordinary speaking effort due to the talker's reaction to ambient conditions. Because of the difficulties in modeling such characteristic changes, no analytical result is available. However, several proposed heuristic techniques are noteworthy because of their effectiveness.

The recognizer of Roe [19] is a traditional template-based system that incorporated both vector quantization and dynamic time warping. In the system, the spectral pattern was first vector quantized and replaced by the closest spectral prototype (code word) in the VQ codebook for subsequent distortion calculation. To adapt the "stress-free" clean spectral prototypes to the stressed speech, each (known) stressed speech utterance was time-aligned, without spectral quantization, to the correct reference template. The autocorrelation vectors of the stressed speech frames were then grouped according to the corresponding VQ indices in the reference templates and averaged to yield the stress-compensated prototypes. In a speaker-trained isolated digit recognition trial, the stress-compensation scheme reduced the number of recognition errors by two-thirds, from 29.9% to 9.6%, when the extra speaking effort was caused by the noise in a car cruising at 60 mph with ventilation fan on.

Another stress-compensation technique, operating in the cepstral domain, was proposed by Chen [34]. The basic assumption of the technique was that spectral distortion induced by unusual speaking efforts could be compensated by simple linear transformation of the cepstrum. Although this assumption may be unduly optimistic, it was observed [34] that the statistics of cepstral vectors did display some systematic modification in various speaking styles. The possibility of spectral compensation in the cepstral domain thus became viable. Incorporation of cepstral compensation in a stochastic model based recognizer (see again Chapter 6) was found to be effective [34] in dealing with various articulation effects.

#### 5.7.2.5 Robust Distortion Measures

The idea of robust distortion measures is to selectively and automatically emphasize the distortion pertaining to certain regions of the spectrum that are less corrupted by noise. A noise-compensation scheme can be interpreted as implicitly defining a robust distortion measure. Clearly, noise-masking attempts to deemphasize the measured distortion in those regions contaminated by noise. For independent spectral components, such as those obtained by filter-bank methods, the practice of noise compensation and the interpretation of distortion measures are straightforward. For other spectral representations, selective emphasis of certain spectral regions to account for the noise effects can be realized in various ways.

Spectral weighting for improving speech-recognition accuracy has long been considered a viable approach. The work of Matsumoto and Imai [35] represents an early effort to investigate the sensitivity of various weighted distortion measures to noise contamination in recognition tasks. As discussed in Chapter 4, weighted spectral measures such as the weighted likelihood distortion attempt to emphasize the spectral peak regions that have more power concentration to resist noise corruption. It was reported [36] that at 18 dB SNR (white noise) using only clean reference templates, weighted measures greatly improved the recognition performance (from 60% to 90% word accuracy) in a speaker-dependent isolated word (28 Japanese city names) recognition trial. Soong and Sondhi [37] also proposed a weighted measure, similar to the above asymmetrically weighted likelihood ratio distortion, with a noise-adaptive bandwidth expansion factor in the weighting spectrum.

When proper weighting functions are used, weighted cepstral measures (Eq. (4.37)) have been found to be advantageous for speech recognition in clean as well as noisy conditions. Indeed, one can argue that additive white noise affects the cepstrum of an all-pole system mostly in the low-frequency terms and thus applying a cepstral lifter to de-emphasize these terms will be beneficial. The work of Itakura and Umezaki [38] on smoothed group delay spectra is a comprehensive study of weighted cepstral measures in noisy and distorted conditions. The cepstral lifter $w(n)$ considered in [38] has a general form of

$$w(n) = n^s \exp(-n^2/2\tau^2) \qquad s \geq 0. \tag{5.118}$$

Several test conditions were considered for a set of confusing Japanese city name pairs: high-quality microphone recording, distortion due to first-order frequency transfer characteristics deviation $(1 - \alpha z^{-1})/(1 + \alpha z^{-1})$, signal-dependent multiplicative noise and telephone speech recorded with carbon microphones. The cepstral liftering led to various degrees of performance improvements; for example, at 10 dB SNR (multiplicative noise), the recognition accuracy was enhanced from ~62% without cepstral liftering to ~82% with cepstral liftering for $s = 1.5$ or 2. Overall performance for various conditions was found to be best with parameters $s = 1 \sim 2$ and $\tau = 5$. It is interesting to note that with these parameters the smoothed group delay weighting function is very close to the raised sine lifter of [39].

The pursuit of a robust distortion measure can be more effective if we havean analytical understanding of the effects of noise upon spectral parameters. Mansour and



Juang [40] reported that analytical as well as experimental evidence indicates that additive white noise causes the cepstral vector norm (length of the cepstral vector, with the zeroth term excluded) to shrink but leaves the cepstral vector orientation more or less intact. The vector norm shrinkage is detrimental in the traditional distortion calculation where a Euclidean distance is employed. Furthermore, since the norm shrinkage was found to be a function of the noise level, the vector norm itself can be used to facilitate nonuniform weighting for each speech frame in the accumulative distance during dynamic sequence comparison. The result suggests the use of a projection operation to formulate several distortion measures to cope with the mismatched noisy condition where the reference patterns are noise-free and the test conditions are unknown. In [40], the following cepstral projection measure was found to be a good choice overall for clean as well as noisy speech:

$$d(\mathbf{x}_r, \mathbf{x}_t) = |\mathbf{x}_t| \left(1 - \frac{\mathbf{x}_r^* \mathbf{x}_t}{|\mathbf{x}_r|\,|\mathbf{x}_t|}\right) \tag{5.119}$$

where $\mathbf{x}_r$ and $\mathbf{x}_t$ are the reference and the test cepstral vector respectively and $*$ denotes matrix transpose. In a speaker-dependent isolated word recognition trial (using the 39-word vocabulary referred to several times in this chapter), the projection measure was shown to be superior to many other well known distortion measures. Figure 5.28 shows the recognition rate as a function of SNR for several distortion measures; the iterative enhancement method in the figure refers to the preprocessing enhancement of Ephraim et al. [28] discussed previously. The effectiveness of the new distortion measure is clearly seen. The robust distortion measures were also found to lead to better recognition performance for Lombard speech.

#### 5.7.2.6 Novel Representations of Speech

Apart from the above robust distortion measures, noise adaptation schemes, and speech enhancement algorithms that rely on traditional spectral measurements, there have been made many other attempts to combat the noise problem by finding characteristic representations of speech that are invariant or resistant to noise corruption. The approach, here, is to find a robust "front-end" that produces reliable measurements of speech even in the presence of noise. Two classes of such systems are noteworthy; one is from a signal-processing viewpoint and the other tries to duplicate the human auditory capability.

The short-time modified coherence (SMC) of speech proposed by Mansour and Juang [41] takes advantage of the inherent coherence in adjacent segments of the speech signal to enhance the SNR. It can be shown that an unwindowed autocorrelation operation on the impulse response of an all-pole system does not alter its pole structure and estimation of the system parameters may be more reliably accomplished from the autocorrelation function when the signal has been corrupted by noise. The autocorrelation sequence is defined by

$$\rho_i = \frac{1}{N}\sum_{j=0}^{N-1} x(j)x(j+i), \qquad i = 0, 1, \ldots, N \tag{5.120}$$

which for quasistationary signals like speech represents the coherence of adjacent signal

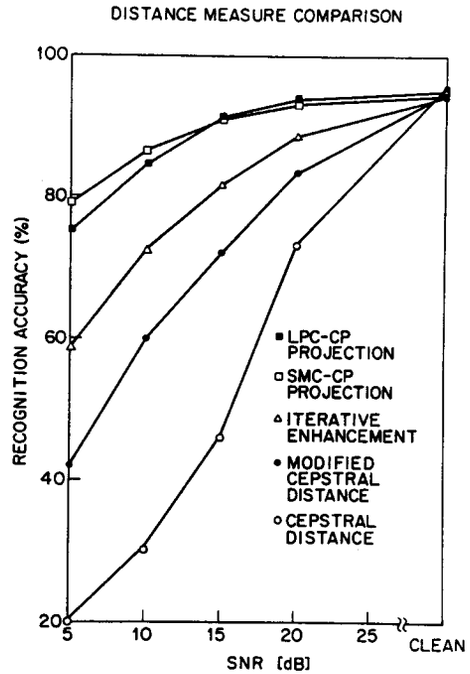



**Figure 5.28** Noisy speech-recognition performance of several distortion measures and methods (after Mansour and Juang [41]).

segments. All-pole modeling of the autocorrelation sequence (rather than the speech waveform $x(i)$) results in a more robust signal representation than that of the signal itself. The SNR improvement is data dependent (a function of the speech impulse response) and was found to be around 10~12 dB for typical SNR's between 0 and 20 dB. For noisy speech recognition at 10 dB SNR, the SMC maintains an accuracy of 98.3% for a speaker-dependent digit test, while the traditional all-pole spectrum representation suffers a severe degradation with accuracy dropping to 39.8%, from 99.2% in clean conditions.

The human auditory system seems to perceive speech better than any machine processor when noise interference is present. As previously described in Section 3.5.1, based on this premise, Ghitza [42] proposed a computational model, called the Ensemble Interval Histogram (EIH), to represent the auditory-nerve firing pattern which is assumed to be robust to noise corruption. Figure 3.53 showed a comparison of the traditional FFT spectrum and the EIH representation with their corresponding all-pole (linear prediction, LPC) spectral fit in clean as well as 0 dB SNR conditions for the vowel /e/. The robustness of the EIH to noise corruption was demonstrated by the results shown. When applied to



noisy speech recognition in conjunction with a traditional dynamic time warping scheme, the EIH was reported to yield significant accuracy improvements for male speech [42]. Other aspects of the representation, particularly how the "brain" makes best use of the "auditory-nerve firing pattern," are not yet well understood.

## 5.8 SUMMARY

In this chapter we have discussed how a speech recognition system can be built to perform a given task. We showed that a VQ-based system has the advantage of computational simplicity although its performance is generally not adequate for many standard speech recognition tasks. An alternative idea was to use the VQ system as a preprocessor which produces preliminary recognition decisions in order to reduce the computational load of a recognition system. In this implementation, vector quantizers have proven effective and are easy to implement.

In speech-recognition systems based on template matching, one critical component is the set of reference patterns that are used to represent the vocabulary words to facilitate pattern comparison and recognition. The reference template set can be "trained" in a number of ways. It was shown that classical clustering methods generally lead to better and more reliable (robust) recognition performance then alternative template training procedures. Although one has to be cautious about the fact that speech patterns are sequences of variable duration, many of the traditional clustering techniques that have been proposed for fixed dimensional vectors require only minor modifications in order to be applicable for speech template clustering.

Results of experimental studies of different task-oriented speech recognizers show that the performance of a speech-recognition system can be optimized by choosing the "right" distortion measure and using a reasonable template clustering method. The recognition rate can be often improved somewhat by incorporating energy information into the spectral distortion measure.

We also discussed several other important topics related to training and implementation of systems. Template adaptation, which attempts to adjust the reference patterns to match a new talker based on a limited amount of training data from the new talker, is often critical in many applications. Discriminative training, unlike clustering, is intended to train the reference templates so as to be able to distinguish acoustically confusing word classes. Finally, adverse noise and environmental issues that often seriously affect the performance of a speech recognizer during actual deployment, were discussed. Several promising methods to deal with various adverse conditions were presented.

## REFERENCES

[1] J.E. Shore and D.K. Burton, "Discrete utterance speech recognition without time alignment," *IEEE Trans. Information Theory*, IT-29 (4): 473–491, July 1983.

[2] L.R. Rabiner, C.K. Pan, and F.K. Soong, "On the performance of isolated word speech

recognizers using vector quantization and temporal energy contours," *AT&T Tech. J.*, 63 (7): 1245–1260, 1984.

[3] B.H. Juang, "Design and performance of trellis vector quantizers for speech signals," *IEEE Trans. Acoustics, Speech, Signal Proc.*, ASSP–36, 9: 1423–1431, September 1988.

[4] K.C. Pan, F.K. Soong, and L.R. Rabiner, "A vector-quantization-based preprocessor for speaker-independent isolated word recognition," *IEEE Trans. Acoustics, Speech, Signal Proc.*, ASSP-30 (3): 546–560, June 1985.

[5] L.R. Rabiner, S.E. Levinson, A.E. Rosenberg, and J.G. Wilpon, "Speaker independent recognition of isolated word using clustering techniques," *IEEE Trans. Acoustics, Speech, Signal Proc.*, ASSP–27, 336–349, August 1979.

[6] S.E. Levinson, L.R. Rabiner, A.E. Rosenberg, and J.G. Wilpon, "Interactive clustering techniques for selecting speaker independent reference templates for isolated word recognition," *IEEE Trans. Acoustics, Speech, Signal Proc.*, ASSP–27, 134–141, 1979.

[7] J.G. Wilpon and L.R. Rabiner, "A modified *K*-means clustering algorithm for use in isolated word recognition," *IEEE Trans. Acoustics, Speech, Signal Proc.*, ASSP-33 (3): 587–594, June 1985.

[8] L.R. Rabiner and F.K. Soong, "Single frame vowel recognition using vector quantization with several distance measures," *AT&T Tech. J.*, 64(10), 2319–2330, December 1985.

[9] N. Nocerino, F.K. Soong, L.R. Rabiner, and D.H. Klatt, "Comparative study of several distortion measures for speech recognition," *Speech Communication*, No. 4: 317–331, 1985.

[10] S. Davis and P. Mermelstein, "Comparison of parametric representation for monosyllable word recognition in continuously spoken sentences," *IEEE Trans. Acoustics, Speech, Signal Proc.*, 28 (4): 357–366, 1980.

[11] B.H. Juang, J.G. Wilpon, and L.R. Rabiner, "On the use of bandpass liftering in speech recognition," *IEEE Trans Acoustics, Speech, Signal Proc.*, 35 (7): 947–954, July 1987.

[12] R.O. Duda and P.E. Hart, *Pattern Classification and Scene Analysis*, John-Wiley & Sons, New York, 1973.

[13] L.R. Rabiner and J.G. Wilpon, "Some Performance Benchmarks for Isolated Word Speech Recognition Systems," *Computer Speech and Language*, 2: 343–357, 1987.

[14] S. Furui, "Unsupervised speaker adaptation based on hierarchical spectral clustering," *IEEE Trans. Acoustucs, Speech, Signal Proc.*, ASSP-37 (12): 1923–1930, December 1989.

[15] L.R. Rabiner and J.G. Wilpon, "A two-pass pattern-recognition approach to isolated word recognition," *Bell System Tech. J.*, 60 (5): 739–766, May–June 1987.

[16] P.C. Chang and B.H. Juang, "Discriminative training of dynamic programming based speech recognizers," *Proc. ICASSP*, ICASSP–92, San Francisco, March 1992.

[17] B.A. Dautrich, L.R. Rabiner, and T.B. Martin, "On the effects of varying filter bank parameters on isolated word recognition," *IEEE Trans. Acoustics, Speech, Signal Proc.*, ASSP–31, 793–806, August 1983.

[18] I. Lecomte et al., "Car noise processing for speech input," *Proc. ICASSP 89*, 512–515, Glasgow, UK, May 1989.

[19] D.B. Roe, "Speech recognition with a noise-adapting codebook," *Proc. ICASSP 87*, 1139–1142, Dallas, TX, April 1987.

[20] A. Acero and R.M. Stern, "Environmental robustness in automatic speech recognition," *Proc. ICASSP 90*, 849–952, Albuquerque, NM, April 1990.



[21] M.B. Carey, et al. "1982/83 end office connection study: analog voice and voiceband data transmission performance characterization of the public switched network," *AT&T Tech. J.*, 63(9), November 1984.

[22] D.B. Pisoni et al., "Some acoustic-phonetic correlates of speech produced in noise," *Proc. ICASSP 85*, 1581–1584, Tampa, FL, March 1985.

[23] J.-C. Junqua and Yolanda Angelade, "Acoustic and perceptual studies of Lombard speech: application to isolated-words automatic speech recognition," *Proc. ICASSP 90*, 841–844, Albuquerque, NM, April 1990.

[24] R.P. Lippmann, E.A. Martin, and D.B. Paul, "Multi-style training for robust isolated-word speech recognition," *Proc. ICASSP 87*, 705–708, Dallas, TX, April 1987.

[25] B. Widrow et al., "Adaptive noise cancelling: Principles and applications," *Proc. IEEE*, 63: 1692–1716, 1975.

[26] N. Dal Degan and C. Prati, "Acoustic noise analysis and speech enhancement techniques for mobile radio applications," *Signal Proc.*, 15: 43–56, 1988.

[27] J.E. Porter and S.F. Boll, "Optimal estimators for spectral restoration of noisy speech," *Proc. ICASSP 84*, 18A.2.1–18A.2.4, San Diego, CA, March 1984.

[28] Y. Ephraim, J.G. Wilpon, and L.R. Rabiner, "A linear predictive front-end processor for speech recognition in noisy environments," *Proc. ICASSP 87*, 1324–1327, Dallas, TX, April 1987.

[29] B.H. Juang and L.R. Rabiner, "Signal restoration by spectral mapping," *Proc. ICASSP 87*, 2368–2371, Dallas, TX, April 1987.

[30] V. Viswanathan and C. Henry, "Evaluation of multisensor speech input for speech recognition in high ambient noise," *Proc. ICASSP 86*, 85–88, Tokyo, JAPAN, April 1986.

[31] D.H. Klatt, "A digital filter bank for spectral matching," *Proc. ICASSP 76*, 573–576, Philadelphia, PA, 1976.

[32] J.S. Bridle et al., "A noise compensating spectrum distance measure applied to automatic speech recognition," *Proc. Institute of Acoustics (U.K.)*, Autumn Conf., Windermere, November 1984.

[33] John N. Holmes and Nigel C. Sedgwick, "Noise compensation for speech recognition using probabilistic models," *Proc. ICASSP 86*, 741–744, Tokyo, JAPAN, April 1986.

[34] Yeunung Chen, "Cepstral domain stress compensation for robust speech recognition," *Proc. ICASSP 87*, 717–720, Dallas, TX, April 1987.

[35] H. Matsumoto and H. Imai, "Comparative study of various spectrum matching measures on noise robustness," *Proc. ICASSP 86*, 769–772, Tokyo, JAPAN, April 1986.

[36] M. Sugiyama and K. Shikano, "Frequency weighted LPC spectral matching measures," *Electronics & Communications in Japan*, 65A, 12, 1–9, 1982.

[37] F.K. Soong and M.M. Sondhi, "A frequency-weighted Itakura spectral distortion measure and its application to speech recognition in noise," *Proc. ICASSP 87*, 1257–1230, Dallas, TX, April 1987.

[38] F. Itakura and T. Umezaki, "Distance measure for speech recognition based on the smoothed group delay spectrum," *Proc. ICASSP 87*, 1257–1260, Dallas, TX, April 1987.

[39] B.H. Juang, J.G. Wilpon, and L.R. Rabiner, "On the use of bandpass liftering in speech recognition," *Proc. ICASSP 86*, 765–768, Tokyo, JAPAN, April 1986.

[40] D. Mansour and B.H. Juang, "A family of distortion measures based upon projection operation for robust speech recognition," *Proc. ICASSP 88*, New York, NY, April 1988; also in *IEEE Trans. Acoustics, Speech, Signal Proc.*, ASSP-37 (11): 1659–1671, November 1989.

[41] D. Mansour and B.H. Juang, "The short-time modified coherence representation and noisy speech recognition," *IEEE Trans. Acoustics, Speech, Signal Proc.* ASSP-37 (6): 795–804, June 1989.

[42] O. Ghitza, "Auditory nerve representation as a front-end for speech recognition in a noisy environment," *Computer Speech and Language*, 1 (2): 109–130, December 1986.

Chapter 6

# THEORY AND IMPLEMENTATION OF HIDDEN MARKOV MODELS

## 6.1 INTRODUCTION

In Chapters 4 and 5 we presented one major pattern-recognition approach to speech recognition, namely the template method. One key idea in the template method is to derive typical sequences of speech frames for a pattern (e.g., a word) via some averaging procedure, and to rely on the use of local spectral distance measures to compare patterns. Another key idea is to use some form of dynamic programming to temporally align patterns to account for differences in speaking rates across talkers as well as across repetitions of the word by the same talker. The methodology of the template approach is well developed and provides good recognition performance for a variety of practical applications.

The template approach, however, is not based on the ideas of statistical signal modeling in a strict sense. Even though statistical techniques have been widely used in clustering to create reference patterns, the template approach is best classified as a simplified, nonparametric method in which a multiplicity of reference tokens (sequences) are used to characterize the variation among different utterances. As such, statistical signal characterization inherent in the template representation is only implicit and often inadequate. Consider, for example, the use of a truncated cepstral distortion measure as the local distance for template matching. The Euclidean distance form of the cepstral distance measure suggests that the reference vector can be viewed as the *mean* of some assumed distribution.

Obviously, this simple form of the sufficient statistic[1] (use of only the mean reference vector) neglects the second-order statistics—i.e., *covariances*, which, as will be seen later, are of particular significance in statistical modeling. (Note that this distribution is used to account for variations of the cepstral coefficients at the frame level since time alignment is performed so as to match appropriate frames of the patterns being compared.) There is clearly a need to use a more elaborate and analytical statistical method for speech recognition.

In this chapter we will study one well-known and widely used statistical method of characterizing the spectral properties of the frames of a pattern, namely the hidden Markov model (HMM) approach. (These models are also referred to as Markov sources or probabilistic functions of Markov chains in the communications literature.) The underlying assumption of the HMM (or any other type of statistical model) is that the speech signal can be well characterized as a parametric random process, and that the parameters of the stochastic process can be determined (estimated) in a precise, well-defined manner. We will show that the HMM method provides a natural and highly reliable way of recognizing speech for a wide range of applications and integrates well into systems incorporating both task syntax and semantics.

The basic theory of hidden Markov models was published in a series of classic papers by Baum and his colleagues ([1]–[5]) in the late 1960s and early 1970s and was implemented for speech-processing applications by Baker [6] at CMU, and by Jelinek and his colleagues at IBM ([7]–[13]) in the 1970s.

We begin this chapter with a review of the theory of Markov chains and then extend the ideas to HMMs using several simple examples. Based on the now-classical approach of Jack Ferguson of IDA (Institute for Defense Analyses), as introduced in lectures and in writing [14], we will focus our attention on the three fundamental problems for HMM design, namely: the evaluation of the probability (or likelihood) of a sequence of observations given a specific HMM; the determination of a best sequence of model states; and the adjustment of model parameters so as to best account for the observed signal. We will show that once these three fundamental problems are solved, we can readily apply HMMs to selected problems in speech recognition.

## 6.2 DISCRETE-TIME MARKOV PROCESSES

Consider a system that may be described at any time as being in one of a set of $N$ distinct states indexed by $\{1, 2, \ldots, N\}$ as illustrated in Figure 6.1 (where $N = 5$ for simplicity). At regularly spaced, discrete times, the system undergoes a change of state (possibly back to the same state) according to a set of probabilities associated with the state. We denote the time instants associated with state changes as $t = 1, 2, \ldots$, and we denote the actual

[1]Sufficient statistics are a set of measurements from a process which contain all the relevant information for estimating the parameters of that process.



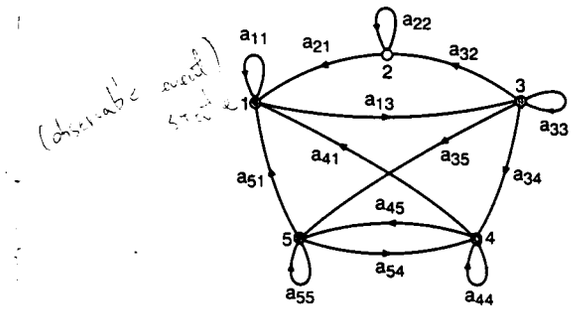



**Figure 6.1** A Markov chain with five states (labeled 1 to 5) with selected state transitions.

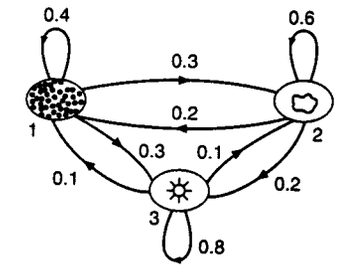



**Figure 6.2** Markov model of the weather.

state at time $t$ as $q_t$. A full probabilistic description of the above system would, in general, require specification of the current state (at time $t$), as well as all the predecessor states. For the special case of a discrete-time, first order, Markov chain, the probabilistic dependence is truncated to just the preceding state—that is,

$$P[q_t = j | q_{t-1} = i, q_{t-2} = k, \ldots] = P[q_t = j | q_{t-1} = i]. \tag{6.1}$$

Furthermore, we consider only those processes in which the right-hand side of (6.1) is independent of time, thereby leading to the set of state-transition probabilities $a_{ij}$ of the form

$$a_{ij} = P[q_t = j | q_{t-1} = i], \qquad 1 \le i, j \le N \tag{6.2}$$

with the following properties

$$a_{ij} \ge 0 \qquad \forall j, i \tag{6.3a}$$

$$\sum_{j=1}^{N} a_{ij} = 1 \qquad \forall i \tag{6.3b}$$

since they obey standard stochastic constraints.

The above stochastic process could be called an observable Markov model because the output of the process is the set of states at each instant of time, where each state corresponds to an observable event. To set ideas, consider a simple three-state Markov model of the weather as shown in Figure 6.2. We assume that once a day (e.g., at noon), the weather is observed as being one of the following:

**State 1:** precipitation (rain or snow)

**State 2:** cloudy

**State 3:** sunny.

We postulate that the weather on day $t$ is characterized by a single one of the three states above, and that the matrix $A$ of state-transition probabilities is

$$A = \{a_{ij}\} = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{bmatrix}.$$

Given the model of Figure 6.2 we can now ask (and answer) several interesting questions about weather patterns over time. For example, we can pose the following simple problem:

**Problem**

What is the probability (according to the model) that the weather for eight consecutive days is "sun-sun-sun-rain-rain-sun-cloudy-sun"?

**Solution**

We define the observation sequence, O, as

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| O | = | ( sunny, | sunny, | sunny, | rain, | rain, | sunny, | cloudy, | sunny ) |
| | = | ( 3, | 3, | 3, | 1, | 1, | 3, | 2, | 3 ) |
| day | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

corresponding to the postulated set of weather conditions over the eight-day period and we want to calculate $P(\mathbf{O}|\text{Model})$, the probability of the observation sequence O, given the model of Figure 6.2. We can directly determine $P(\mathbf{O}|\text{Model})$ as:

$$\begin{aligned} P(\mathbf{O}|\text{Model}) &= P[3, 3, 3, 1, 1, 3, 2, 3|\text{Model}] \\ &= P[3]\,P[3|3]^2 P[1|3]\,P[1|1] \\ &\quad P[3|1]\,P[2|3]\,P[3|2] \\ &= \pi_3 \cdot (a_{33})^2\, a_{31}\, a_{11}\, a_{13}\, a_{32}\, a_{23} \\ &= (1.0)(0.8)^2(0.1)(0.4)(0.3)(0.1)(0.2) \\ &= 1.536 \times 10^{-4} \end{aligned}$$

where we use the notation:

$$\pi_i = P[q_1 = i], \qquad 1 \le i \le N \tag{6.4}$$

to denote the initial state probabilities.

Another interesting question we can ask (and answer using the model) is:

**Problem**

Given that the system is in a known state, what is the probability that it stays in that state for exactly $d$ days?

**Solution**

This probability can be evaluated as the probability of the observation sequence

$$\begin{array}{lcccccc} \mathbf{O} & = ( i, & i, & i & ,\ldots, & i, & j \ne i\ ) \\ \text{day} & & 1 & 2 & 3 & d & d+1 \end{array}$$

$P(O|\text{model}) = P[i]\ P[i|i]^{d-1}\ P[j|i]$



given the model, which is

$$\begin{aligned} P(\mathbf{O}|\text{Model}, q_1 = i) &= P(\mathbf{O}, q_1 = i|\text{Model})/P(q_1 = i) \\ &= \pi_i (a_{ii})^{d-1}(1 - a_{ii})/\pi_i \\ &= (a_{ii})^{d-1}(1 - a_{ii}) \\ &= p_i(d) \end{aligned} \tag{6.5}$$

$a_{ij} = 1 - a_{ii}$

The quantity $p_i(d)$ is the probability distribution function of duration $d$ in state $i$. This exponential distribution is characteristic of the state duration in a Markov chain. Based on $p_i(d)$, we can readily calculate the expected number of observations (duration) in a state, conditioned on starting in that state as

期望值?

$$\bar{d}_i = \sum_{d=1}^{\infty} d p_i(d) \tag{6.6a}$$

$$= \sum_{d=1}^{\infty} d(a_{ii})^{d-1}(1 - a_{ii}) = \frac{1}{1 - a_{ii}}. \tag{6.6b}$$

Thus the expected number of consecutive days of sunny weather, according to the model, is $1/(0.2) = 5$; for cloudy it is 2.5; for rain it is 1.67.

**Problem**

Derive the expression for the mean of $p_i(d)$, i.e. Eq. (6.6b).

**Solution**

$$\begin{aligned} \bar{d}_i &= \sum_{d=1}^{\infty} d p_i(d) \\ &= \sum_{d=1}^{\infty} d(a_{ii})^{d-1}(1 - a_{ii}) \\ &= (1 - a_{ii}) \frac{\partial}{\partial a_{ii}} \left[ \sum_{d=1}^{\infty} a_{ii}^d \right] \\ &= (1 - a_{ii}) \frac{\partial}{\partial a_{ii}} \left( \frac{a_{ii}}{1 - a_{ii}} \right) \\ &= \frac{1}{1 - a_{ii}}. \end{aligned}$$

$\sum_{d=1}^{\infty} d\,(a_{ii})^{d-1} = \frac{\partial}{\partial a_{ii}} \left[ \sum_{d=1}^{\infty} a_{ii}^{d} \right]$

$= \frac{1 - a_{ii} + a_{ii}}{(1 - a_{ii})^2}$

## 6.3 EXTENSIONS TO HIDDEN MARKOV MODELS

So far we have considered Markov models in which each state corresponded to a deterministically observable event. Thus, the output of such sources in any given state is not random. This model is too restrictive to be applicable to many problems of interest. In this section we extend the concept of Markov models to include the case in which the observation is a probabilistic function of the state—that is, the resulting model (which is

observation 是 state 的 probabilistic function

called a hidden Markov model) is a doubly embedded stochastic process with an underlying stochastic process that is *not* directly observable (it is hidden) but can be observed only through another set of stochastic processes that produce the sequence of observations.

To illustrate the basic concepts of the hidden Markov model, we will use several simple examples including simple coin-tossing experiments. We begin with a review of some basic ideas of probability in the following exercise.

**Exercise 6.1**

Given a single fair coin, i.e., $P(\text{Heads}) = P(\text{Tails}) = 0.5$, which you toss once and observe Tails,

1. What is the probability that the next 10 tosses will provide the sequence (*HHTHTTHTTH*)?
2. What is the probability that the next 10 tosses will produce the sequence (*HHHHHHHHHH*)?
3. What is the probability that 5 of the next 10 tosses will be tails? What is the expected number of tails over the next 10 tosses?

**Solution 6.1**

1. For a fair coin, with independent coin tosses, the probability of any specific observation sequence of length 10 (10 tosses) is $(1/2)^{10}$ since there are $2^{10}$ such sequences and all are equally probable. Thus:

$$P(HHTHTTHTTH) = \left(\frac{1}{2}\right)^{10}.$$

2.

$$P(HHHHHHHHHH) = \left(\frac{1}{2}\right)^{10}.$$

Thus a specified run of length 10 is as likely as a specified run of interlaced $H$ and $T$.

3. The probability of 5 tails in the next 10 tosses is just the number of observation sequences with 5 tails and 5 heads (in any order) and this is

$$P(5H, 5T) = \binom{10}{5}\left(\frac{1}{2}\right)^{10} = \frac{252}{1024} \cong 0.25$$

since there are $\binom{10}{5}$ ways of getting $5H$ and $5T$ in 10 tosses, and each sequence has probability of $\left(\frac{1}{2}\right)^{10}$. The expected number of tails in 10 tosses is

$$E(T \text{ in 10 tosses}) = \sum_{d=0}^{10} d\binom{10}{d}\left(\frac{1}{2}\right)^{10} = 5.$$

Thus, on average, there will be $5H$ and $5T$ in 10 tosses, but the probability of exactly $5H$ and $5T$ is only 0.25.

### 6.3.1 Coin-Toss Models

Assume the following scenario. You are in a room with a barrier (e.g., a curtain) through



which you cannot see what is happening. On the other side of the barrier is another person who is performing a coin-tossing experiment (using one or more coins). The person will not tell you which coin he selects at any time; he will only tell you the result of each coin flip. Thus a sequence of *hidden* coin-tossing experiments is performed, with the observation sequence consisting of a series of heads and tails. A typical observation sequence would be

$$\begin{aligned} \mathbf{O} &= (\mathbf{o}_1\, \mathbf{o}_2\, \mathbf{o}_3 \ldots \mathbf{o}_T) \\ &= (HHTTTHTTH \ldots H) \end{aligned}$$

where $H$ stands for heads and $T$ stands for tails.

Given the above scenario, the question is, How do we build an HMM to explain (model) the observed sequence of heads and tails? The first problem we face is deciding what the states in the model correspond to, and then deciding how many states should be in the model. One possible choice would be to assume that only a single biased coin was being tossed. In this case, we could model the situation with a two-state model in which each state corresponds to the outcome of the previous toss (i.e., heads or tails). This model is depicted in Figure 6.3a. In this case, the Markov model is observable, and the only issue for complete specification of the model would be to decide on the best value for the single parameter of the model (i.e., the probability of, say, heads). Interestingly, an equivalent HMM to that of Figure 6.3a would be a degenerate one-state model in which the state corresponds to the single biased coin, and the unknown parameter is the bias of the coin.

A second HMM for explaining the observed sequence of coin toss outcomes is given in Figure 6.3b. In this case there are two states in the model, and each state corresponds to a different, biased coin being tossed. Each state is characterized by a probability distribution of heads and tails, and transitions between states are characterized by a state-transition matrix. The physical mechanism that accounts for how state transitions are selected could itself be a set of independent coin tosses or some other probabilistic event.

A third form of HMM for explaining the observed sequence of coin toss outcomes is given in Figure 6.3c. This model corresponds to using three biased coins, and choosing from among the three, based on some probabilistic event.

Given the choice among the three models shown in Figure 6.3 for explaining the observed sequence of heads and tails, a natural question would be which model best matches the actual observations. It should be clear that the simple one-coin model of Figure 6.3a has only one unknown parameter; the two-coin model of Figure 6.3b has four unknown parameters; and the three-coin model of Figure 6.3c has nine unknown parameters. Thus, with the greater degrees of freedom, the larger HMMs would seem to be inherently more capable of modeling a series of coin-tossing experiments than would equivalently smaller models. Although this is theoretically true, we will see later in this chapter that practical considerations impose some strong limitations on the size of models that we can consider. A fundamental question here is whether the observed head-tail sequence is long and rich enough to be able to specify a complex model. Also, it might just be the case that only a single coin is being tossed. Then using the three-coin model of Figure 6.3c would be inappropriate because we would be using an underspecified system.

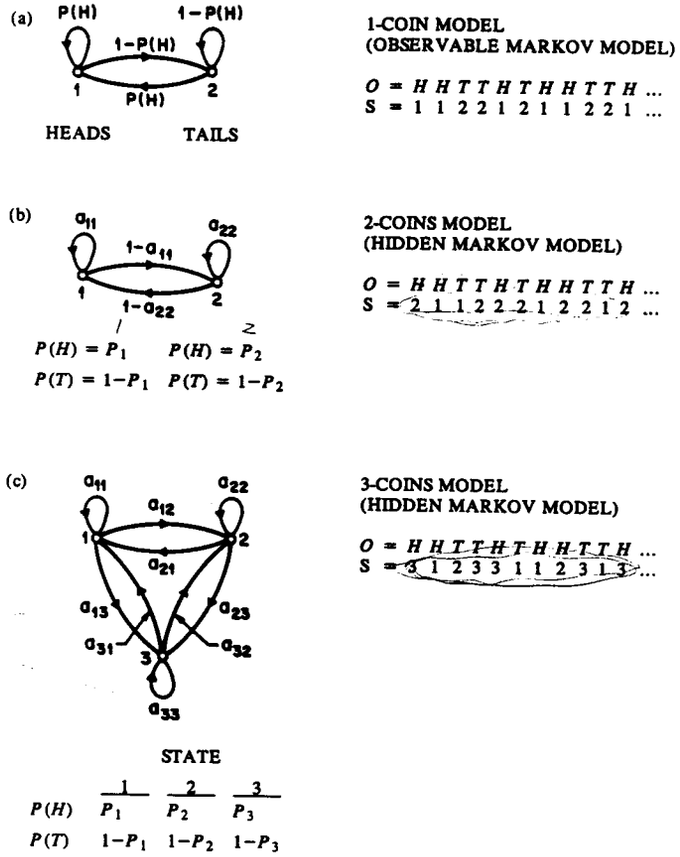



**Figure 6.3** Three possible Markov models that can account for the results of hidden coin-tossing experiments. (a) one-coin model, (b) two-coins model, (c) three-coins model.

### 6.3.2 The Urn-and-Ball Model

To extend the ideas of the HMM to a somewhat more complicated situation, consider the urn-and-ball system of Figure 6.4. We assume that there are $N$ (large) glass urns in a room. Within each urn is a large quantity of colored balls. We assume there are $M$ distinct colors of the balls. The physical process for obtaining observations is as follows. A genie is in the room, and, according to some random procedure, it chooses an initial urn. From this urn, a ball is chosen at random, and its color is recorded as the observation. The ball is



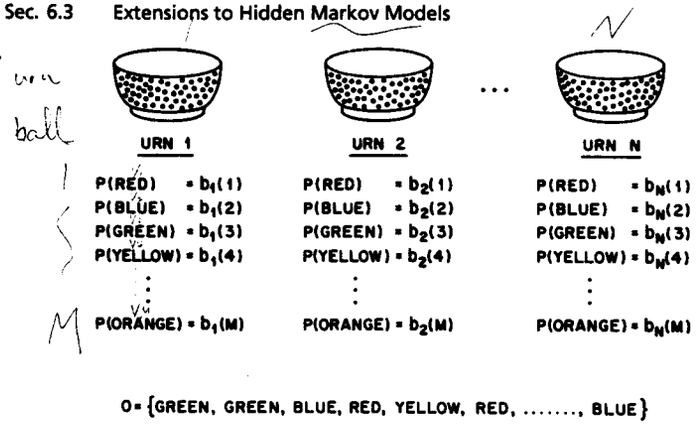



**Figure 6.4** An $N$-state urn-and-ball model illustrating the general case of a discrete symbol HMM.

then replaced in the urn from which it was selected. A new urn is then selected according to the random selection procedure associated with the current urn, and the ball selection process is repeated. This entire process generates a finite observation sequence of colors, which we would like to model as the observable output of an HMM.

It should be obvious that the simplest HMM that corresponds to the urn-and-ball process is one in which each state corresponds to a specific urn, and for which a (ball) color probability is defined for each state. The choice of urns is dictated by the state-transition matrix of the HMM.

It should be noted that the ball colors in each urn may be the same, and the distinction among various urns is in the way the collection of colored balls is composed. Therefore, an isolated observation of a particular color ball does not immediately tell which urn it is drawn from.

### 6.3.3 Elements of an HMM

The above examples give us some idea of what an HMM is and how it can be applied to some simple scenarios. We now formally define the elements of an HMM.

An HMM for discrete symbol observations such as the above urn-and-ball model is characterized by the following:

1. $N$, the number of states in the model. Although the states are hidden, for many practical applications there is often some physical significance attached to the states or to sets of states of the model. Thus, in the coin-tossing experiments, each state corresponded to a distinct biased coin. In the urn-and-ball model, the states corresponded to the urns. Generally the states are interconnected in such a way that any state can be reached from any other state (i.e., an ergodic model); however, we will see later in this chapter that other possible interconnections of states are often

of interest and may better suit speech applications. We label the individual states as $\{1, 2, \ldots, N\}$, and denote the state at time $t$ as $q_t$.

2. $M$, the number of distinct observation symbols per state—i.e., the discrete alphabet size. The observation symbols correspond to the physical output of the system being modeled. For the coin-toss experiments the observation symbols were simply heads or tails; for the ball-and-urn model they were the colors of the balls selected from the urns. We denote the individual symbols as $V = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_M\}$.
3. The state-transition probability distribution $A = \{a_{ij}\}$ where

$$a_{ij} = P[q_{t+1} = j | q_t = i], \qquad 1 \le i, j \le N. \tag{6.7}$$

For the special case in which any state can reach any other state in a single step, we have $a_{ij} > 0$ for all $i, j$. For other types of HMMs, we would have $a_{ij} = 0$ for one or more $(i, j)$ pairs.

4. The observation symbol probability distribution, $B = \{b_j(k)\}$, in which

$$b_j(k) = P[\mathbf{o}_t = \mathbf{v}_k | q_t = j], \qquad 1 \le k \le M, \tag{6.8}$$

defines the symbol distribution in state $j$, $j = 1, 2, \ldots, N$.

5. The initial state distribution $\pi = \{\pi_i\}$ in which

$$\pi_i = P[q_1 = i], \qquad 1 \le i \le N. \tag{6.9}$$

It can be seen from the above discussion that a complete specification of an HMM requires specification of two model parameters, $N$ and $M$, specification of observation symbols, and the specification of the three sets of probability measures $A, B$, and $\pi$. For convenience, we use the compact notation

$$\lambda = (A, B, \pi) \tag{6.10}$$

to indicate the complete parameter set of the model. This parameter set, of course, defines a probability measure for $\mathbf{O}$, i.e. $P(\mathbf{O}|\lambda)$, which we discuss in the next section. We use the terminology HMM to indicate the parameter set $\lambda$ and the associated probability measure interchangeably without ambiguity.

### 6.3.4 HMM Generator of Observations

Given appropriate values of $N, M, A, B$, and $\pi$, the HMM can be used as a generator to give an observation sequence

$$\mathbf{O} = (\mathbf{o}_1 \mathbf{o}_2 \ldots \mathbf{o}_T) \tag{6.11}$$

(in which each observation $\mathbf{o}_t$ is one of the symbols from $V$, and $T$ is the number of observations in the sequence) as follows:

1. Choose an initial state $q_1 = i$ according to the initial state distribution $\pi$.
2. Set $t = 1$.
3. Choose $\mathbf{o}_t = \mathbf{v}_k$ according to the symbol probability distribution in state $i$, i.e., $b_j(k)$.



4. Transit to a new state $q_{t+1} = j$ according to the state-transition probability distribution for state $i$, i.e., $a_{ij}$.
5. Set $t = t + 1$; return to step 3 if $t < T$; otherwise, terminate the procedure.

The following table shows the sequence of states and observations generated by the above procedure:

| time, $t$ | 1 | 2 | 3 | 4 | 5 | 6 | ... | $T$ |
|---|---|---|---|---|---|---|---|---|
| state | $q_1$ | $q_2$ | $q_3$ | $q_4$ | $q_5$ | $q_6$ | ... | $q_T$ |
| observation | $\mathbf{o}_1$ | $\mathbf{o}_2$ | $\mathbf{o}_3$ | $\mathbf{o}_4$ | $\mathbf{o}_5$ | $\mathbf{o}_6$ | ... | $\mathbf{o}_T$ |

The above procedure can be used as both a generator of observations and as a model to simulate how a given observation sequence was generated by an appropriate HMM.

**Exercise 6.2**

Consider an HMM representation (parametrized by $\lambda$) of a coin-tossing experiment. Assume a three-state model (corresponding to three different coins) with probabilities

| | State 1 | State 2 | State 3 |
|---|---|---|---|
| $P(H)$ | 0.5 | 0.75 | 0.25 |
| $P(T)$ | 0.5 | 0.25 | 0.75 |

and with all state-transition probabilities equal to 1/3. (Assume initial state probabilities of 1/3.)

1. You observe the sequence

$$\mathbf{O} = (HHHHTHTTTT).$$

What state sequence is most likely? What is the probability of the observation sequence and this most likely state sequence?

2. What is the probability that the observation sequence came entirely from state 1?
3. Consider the observation sequence

$$\tilde{\mathbf{O}} = (HTTHTHHTTH).$$

How would your answers to parts a and b change?

4. If the state-transition probabilities were

$$a_{11} = 0.9, \quad a_{21} = 0.45, \quad a_{31} = 0.45$$
$$a_{12} = 0.05, \quad a_{22} = 0.1, \quad a_{32} = 0.45$$
$$a_{13} = 0.05, \quad a_{23} = 0.45, \quad a_{33} = 0.1$$

that is, a new model $\lambda'$, how would your answers to parts 1–3 change? What does this suggest about the type of sequences generated by the models?

**Solution 6.2**

1. Given $\mathbf{O} = (HHHHTHTTTT)$ and that all state transitions are equiprobable, the most likely state sequence is the one for which the probability of each individual observation

is maximum. Thus for each $H$, the most likely state is 2 and for each $T$ the most likely state is 3. Thus the most likely state sequence is

$$\mathbf{q} = (2\,2\,2\,2\,3\,2\,3\,3\,3\,3).$$

The probability of $\mathbf{O}$ and $\mathbf{q}$ (given the model) is

$$P(\mathbf{O}, \mathbf{q}|\lambda) = (0.75)^{10}\left(\frac{1}{3}\right)^{10}.$$

2. The probability of $\mathbf{O}$ given that $\hat{\mathbf{q}}$ is

$$\hat{\mathbf{q}} = (1\,1\,1\,1\,1\,1\,1\,1\,1\,1)$$

is

$$P(\mathbf{O}, \hat{\mathbf{q}}|\lambda) = (0.50)^{10}\left(\frac{1}{3}\right)^{10}.$$

The ratio of $P(\mathbf{O}, \mathbf{q}|\lambda)$ to $P(\mathbf{O}, \hat{\mathbf{q}}|\lambda)$ is:

$$R = \frac{P(\mathbf{O}, \mathbf{q}|\lambda)}{P(\mathbf{O}, \hat{\mathbf{q}}|\lambda)} = \left(\frac{3}{2}\right)^{10} = 57.67$$

which shows, as expected, that $\mathbf{q}$ is more likely than $\hat{\mathbf{q}}$.

3. Given $\tilde{\mathbf{O}}$ which has the same number of $H$s and $T$s, the answers to parts 1 and 2 would remain the same, as the most likely states occur the same number of times in both cases.
4. The new probability of $\mathbf{O}$ and $\mathbf{q}$ becomes

$$P(\mathbf{O}, \mathbf{q}|\lambda') = (0.75)^{10}\left(\frac{1}{3}\right)(0.1)^6(0.45)^3.$$

The new probability of $\mathbf{O}$ and $\hat{\mathbf{q}}$ becomes

$$P(\mathbf{O}, \hat{\mathbf{q}}|\lambda') = (0.50)^{10}\left(\frac{1}{3}\right)(0.9)^9.$$

The ratio is

$$R = \left(\frac{3}{2}\right)^{10}\left(\frac{1}{9}\right)^6\left(\frac{1}{2}\right)^3 = 1.36 \times 10^{-5}.$$

In other words, because of the nonuniform transition probabilities, $\hat{\mathbf{q}}$ is more likely than $\mathbf{q}$. (The reader is encouraged to find the most likely state sequence in this case.) Now, the probability of $\tilde{\mathbf{O}}$ and $\mathbf{q}$ is not the same as the probability of $\mathbf{O}$ and $\mathbf{q}$. We have

$$P(\tilde{\mathbf{O}}, \mathbf{q}|\lambda') = \frac{1}{3}(0.1)^6(0.45)^3(0.25)^4(0.75)^6$$

$$P(\tilde{\mathbf{O}}, \hat{\mathbf{q}}|\lambda') = (0.50)^{10}\left(\frac{1}{3}\right)(0.9)^9$$

with ratio

$$R = \left(\frac{1}{9}\right)^6\left(\frac{1}{2}\right)^3\left(\frac{1}{2}\right)^4\left(\frac{3}{2}\right)^6 = 1.67 \times 10^{-7}.$$

Clearly, because $a_{11} = 0.9$, $\hat{\mathbf{q}}$ is more likely.



## 6.4 THE THREE BASIC PROBLEMS FOR HMMS

Given the form of HMM of the previous section, three basic problems of interest must be solved for the model to be useful in real-world applications. These problems are the following:

**Problem 1**

Given the observation sequence $\mathbf{O} = (\mathbf{o}_1\,\mathbf{o}\ldots\mathbf{o}_T)$, and a model $\lambda = (A, B, \pi)$, how do we efficiently compute $P(\mathbf{O}|\lambda)$, the probability of the observation sequence, given the model?

**Problem 2**

Given the observation sequence $\mathbf{O} = (\mathbf{o}_1\,\mathbf{o}\ldots\mathbf{o}_T)$, and the model $\lambda$, how do we choose a corresponding state sequence $\mathbf{q} = (q_1 q_2 \ldots q_T)$ that is optimal in some sense (i.e., best "explains" the observations)?

**Problem 3**

How do we adjust the model parameters $\lambda = (A, B, \pi)$ to maximize $P(\mathbf{O}|\lambda)$?

compute the probability of observation sequence

Problem 1 is the evaluation problem; namely, given a model and a sequence of observations, how do we compute the probability that the observed sequence was produced by the model? We can also view the problem as one of scoring how well a given model matches a given observation sequence. The latter viewpoint is extremely useful. For example, if we consider the case in which we are trying to choose among several competing models, the solution to Problem 1 allows us to choose the model that best matches the observations.

optimal find state sequence

Problem 2 is the one in which we attempt to uncover the hidden part of the model—that is, to find the "correct" state sequence. It should be clear that for all but the case of degenerate models there is no "correct" state sequence to be found. Hence for practical situations, we usually use an optimality criterion to solve this problem as best as possible. As we will see, several reasonable optimality criteria can be imposed, and hence the choice of criterion is a strong function of the intended use for the uncovered state sequence. Typical uses might be to learn about the structure of the model, to find optimal state sequences for continuous speech recognition, or to get average statistics of individual states, etc.

adjust the model parameters

Problem 3 is the one in which we attempt to optimize the model parameters to best describe how a given observation sequence comes about. The observation sequence used to adjust the model parameters is called a training sequence because it is used to "train" the HMM. The training problem is the crucial one for most applications of HMMs, because it allows us to optimally adapt model parameters to observed training data—i.e., to create best models for real phenomena.

word 于 vocabulary

To fix ideas, consider the following simple isolated-word speech recognizer. For each word of a $W$ word vocabulary, we want to design a separate $N$-state HMM. We represent the speech signal of a given word as a time sequence of coded spectral vectors. We assume that the coding is done using a spectral codebook with $M$ unique spectral vectors; hence each observation is the index of the spectral vector closest (in some spectral distortion sense) to the original speech signal. Thus, for each vocabulary word, we have a training sequence

word :

coded spectral vectors

1 2 3 N state

consisting of a number of repetitions of sequences of codebook indices of the word (by one or more talkers). The first task is to build individual word models. This task is done by using the solution to Problem 3 to optimally estimate model parameters for each word model. To develop an understanding of the physical meaning of the model states, we use the solution to Problem 2 to segment each of the word training sequences into states, and then study the properties of the spectral vectors that lead to the observations occurring in each state. The goal here is to make refinements of the model (e.g., more states, different codebook size) to improve its capability of modeling the spoken word sequences. Finally, once the set of $W$ HMMs has been designed and optimized, recognition of an unknown word is performed using the solution to Problem 1 to score each word model based upon the given test observation sequence, and select the word whose model score is highest (i.e., the highest likelihood).

In the next sections we present formal mathematical solutions to each fundamental problem for HMMs. We shall see that the three problems are tightly linked together under the probabilistic framework.

### 6.4.1 Solution to Problem 1—Probability Evaluation

We wish to calculate the probability of the observation sequence, $\mathbf{O} = (\mathbf{o}_1\,\mathbf{o}\ldots\mathbf{o}_T)$, given the model $\lambda$, i.e., $P(\mathbf{O}|\lambda)$. The most straightforward way of doing this is through enumerating every possible state sequence of length $T$ (the number of observations). There are $N^T$ such state sequences. Consider one such fixed-state sequence

$$\mathbf{q} = (q_1 q_2 \ldots q_T) \tag{6.12}$$

where $q_1$ is the initial state. The probability of the observation sequence $\mathbf{O}$ given the state sequence of Eq. (6.12) is

$$P(\mathbf{O}|\mathbf{q},\lambda) = \prod_{t=1}^{T} P(\mathbf{o}_t|q_t,\lambda) \tag{6.13a}$$

where we have assumed statistical independence of observations. Thus we get

$$P(\mathbf{O}|\mathbf{q},\lambda) = b_{q_1}(\mathbf{o}_1)\cdot b_{q_2}(\mathbf{o}_2)\ldots b_{q_T}(\mathbf{o}_T). \tag{6.13b}$$

The probability of such a state sequence $\mathbf{q}$ can be written as

$$P(\mathbf{q}|\lambda) = \pi_{q_1} a_{q_1q_2} a_{q_2q_3}\ldots a_{q_{T-1}q_T}. \tag{6.14}$$

The joint probability of $\mathbf{O}$ and $\mathbf{q}$, i.e., the probability that $\mathbf{O}$ and $\mathbf{q}$ occur simultaneously, is simply the product of the above two terms, i.e.,

$$P(\mathbf{O},\mathbf{q}|\lambda) = P(\mathbf{O}|\mathbf{q},\lambda)P(\mathbf{q}|\lambda). \tag{6.15}$$

The probability of $\mathbf{O}$ (given the model) is obtained by summing this joint probability over all possible state sequences $\mathbf{q}$, giving

$$P(\mathbf{O}|\lambda) = \sum_{\text{all } \mathbf{q}} P(\mathbf{O}|\mathbf{q},\lambda)P(\mathbf{q}|\lambda) \tag{6.16}$$



$$= \sum_{q_1,q_2,\ldots,q_T} \pi_{q_1} b_{q_1}(\mathbf{o}_1) a_{q_1q_2} b_{q_2}(\mathbf{o}_2)\ldots a_{q_{T-1}q_T} b_{q_T}(\mathbf{o}_T). \tag{6.17}$$

The interpretation of the computation in the above equation is the following. Initially (at time $t = 1$) we are in state $q_1$ with probability $\pi_{q_1}$, and generate the symbol $\mathbf{o}_1$ (in this state) with probability $b_{q_1}(\mathbf{o}_1)$. The clock changes from time $t$ to $t+1$ (time = 2) and we make a transition to state $q_2$ from state $q_1$ with probability $a_{q_1q_2}$, and generate symbol $\mathbf{o}_2$ with probability $b_{q_2}(\mathbf{o}_2)$. This process continues in this manner until we make the last transition (at time $T$) from state $q_{T-1}$ to state $q_T$ with probability $a_{q_{T-1}q_T}$ and generate symbol $\mathbf{o}_T$ with probability $b_{q_T}(\mathbf{o}_T)$.

A little thought should convince the reader that the calculation of $P(\mathbf{O}|\lambda)$, according to its direct definition (Eq. (6.17)) involves on the order of $2T\cdot N^T$ calculations, since at every $t = 1,2,\ldots,T$, there are $N$ possible states that can be reached (i.e., there are $N^T$ possible state sequences), and for each such state sequence about $2T$ calculations are required for each term in the sum of Eq. (6.17). (To be precise, we need $(2T-1)N^T$ multiplications, and $N^T - 1$ additions.) This calculation is computationally infeasible, even for small values of $N$ and $T$; e.g., for $N = 5$ (states), $T = 100$ (observations), there are on the order of $2\cdot 100\cdot 5^{100} \approx 10^{72}$ computations! Clearly a more efficient procedure is required to solve problem 1. Fortunately such a procedure (called the forward procedure) exists.

#### 6.4.1.1 The Forward Procedure

Consider the forward variable $\alpha_t(i)$ defined as

$$\alpha_t(i) = P(\mathbf{o}_1\mathbf{o}_2\ldots\mathbf{o}_t, q_t = i|\lambda) \tag{6.18}$$

that is, the probability of the partial observation sequence, $\mathbf{o}_1\,\mathbf{o}_2\ldots\mathbf{o}_t$, (until time $t$) and state $i$ at time $t$, given the model $\lambda$. We can solve for $\alpha_t(i)$ inductively, as follows:

1. Initialization

$$\alpha_1(i) = \pi_i b_i(\mathbf{o}_1), \qquad 1 \le i \le N. \tag{6.19}$$

2. Induction

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i)a_{ij}\right] b_j(\mathbf{o}_{t+1}), \qquad \begin{matrix} 1 \le t \le T-1 \\ 1 \le j \le N \end{matrix}. \tag{6.20}$$

3. Termination

$$P(\mathbf{O}|\lambda) = \sum_{i=1}^{N} \alpha_T(i). \tag{6.21}$$

Step 1 initializes the forward probabilities as the joint probability of state $i$ and initial observation $\mathbf{o}_1$. The induction step, which is the heart of the forward calculation, is illustrated in Figure 6.5(a). This figure shows how state $j$ can be reached at time $t+1$ from the $N$ possible states, $i$, $1 \le i \le N$, at time $t$. Since $\alpha_t(i)$ is the probability of the joint event that $\mathbf{o}_1\,\mathbf{o}_2\ldots\mathbf{o}_t$ are observed, and the state at time $t$ is $i$, the product $\alpha_t(i)a_{ij}$ is

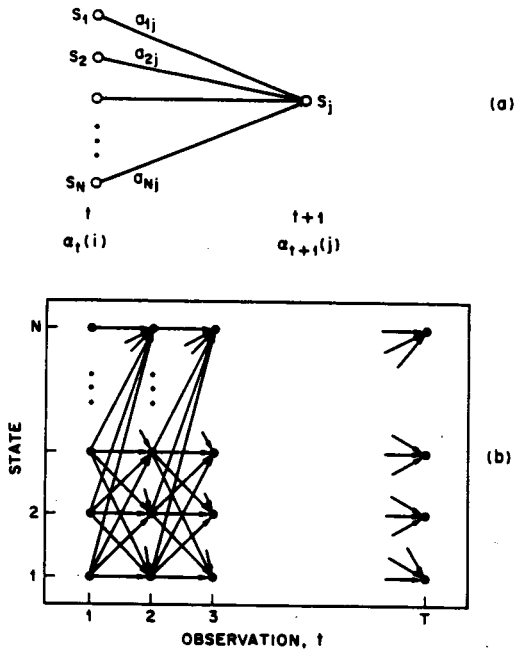



**Figure 6.5** (a) Illustration of the sequence of operations required for the computation of the forward variable $\alpha_{t+1}(j)$. (b) Implementation of the computation of $\alpha_t(i)$ in terms of a lattice of observations $t$, and states $i$.

then the probability of the joint event that $\mathbf{o}_1\,\mathbf{o}_2 \ldots \mathbf{o}_t$ are observed, and state $j$ is reached at time $t+1$ via state $i$ at time $t$. Summing this product over all the $N$ possible states, $i$, $1 \le i \le N$ at time $t$ results in the probability of $j$ at time $t+1$ with all the accompanying 部分 previous partial observations. Once this is done and $j$ is known, it is easy to see that $\alpha_{t+1}(j)$ is obtained by accounting for observation $\mathbf{o}_{t+1}$ in state $j$, i.e., by multiplying the summed quantity by the probability $b_j(\mathbf{o}_{t+1})$. The computation of Eq. (6.20) is performed for all states $j$, $1 \le j \le N$, for a given $t$; the computation is then iterated 反复 for $t = 1, 2, \ldots, T-1$. Finally, step 3 gives the desired calculation of $P(\mathbf{O}|\lambda)$ as the sum of the terminal forward variables $\alpha_T(i)$. This is the case since, by definition,

$$\alpha_T(i) = P(\mathbf{o}_1\mathbf{o}_2 \ldots \mathbf{o}_T, q_T = i|\lambda) \tag{6.22}$$

and hence $P(\mathbf{O}|\lambda)$ is just the sum of the $\alpha_T(i)$'s.

If we examine the computation involved in the calculation of $\alpha_t(j)$, $1 \le t \le T$, $1 \le j \le N$, we see that it requires on the order of $N^2T$ calculations, rather than $2TN^T$ as required by the direct calculation. (Again, to be precise, we need $N(N+1)(T-1)+N$

$N[(N+1)(T-1)+1]$



$N[(N-1)(T-1)]$

multiplications and $N(N-1)(T-1)$ additions.) For $N = 5$, $T = 100$, we need about 3000 computations for the forward method, versus $10^{72}$ computations for the direct calculation, a savings of about 69 orders of magnitude.

格子

The forward probability calculation is, in effect, based upon the lattice (or trellis) structure shown in Figure 6.5(b). The key is that, because there are only $N$ states (nodes 重合 at each time slot in the lattice), all the possible state sequences will remerge into these $N$ nodes, no matter how long the observation sequence. At time $t = 1$ (the first time slot in the lattice), we need to calculate values of $\alpha_1(i)$, $1 \le i \le N$. At times $t = 2, 3, \ldots, T$, we need only calculate values of $\alpha_t(j)$, $1 \le j \le N$, where each calculation involves only the $N$ previous values of $\alpha_{t-1}(i)$ because each of the $N$ grid points can be reached from only the $N$ grid points at the previous time slot. 格子

### 6.4.1.2 The Backward Procedure

In a similar manner, we can consider a backward variable $\beta_t(i)$ defined as

$$\beta_t(i) = P(\mathbf{o}_{t+1}\mathbf{o}_{t+2} \ldots \mathbf{o}_T | q_t = i, \lambda) \tag{6.23}$$

that is, the probability of the partial observation sequence from $t+1$ to the end, given state $i$ at time $t$ and the model $\lambda$. Again we can solve for $\beta_t(i)$ inductively, as follows:

1. Initialization

$$\beta_T(i) = 1, \qquad 1 \le i \le N. \tag{6.24}$$

2. Induction

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(\mathbf{o}_{t+1}) \beta_{t+1}(j),$$

$$t = T-1, T-2, \ldots, 1, \qquad 1 \le i \le N. \tag{6.25}$$

任意地

The initialization step 1 *arbitrarily* defines $\beta_T(i)$ to be 1 for all $i$. Step 2, which is illustrated 说明 in Figure 6.6, shows that in order to have been in state $i$ at time $t$, and to account for the observation sequence from time $t+1$ on, you have to consider all possible states $j$ at time $t+1$, accounting for the transition from $i$ to $j$ (the $a_{ij}$ term), as well as the observation $\mathbf{o}_{t+1}$ in state $j$ (the $b_j(\mathbf{o}_{t+1})$ term), and then account for the remaining partial observation sequence from state $j$ (the $\beta_{t+1}(j)$ term). We will see later how the backward as well as the forward calculations are used to help solve fundamental Problems 2 and 3 of HMMs.

Again, the computation of $\beta_t(i)$, $1 \le t \le T$, $1 \le i \le N$, requires on the order of $N^2T$ calculations, and can be computed in a lattice structure similar to that of Figure 6.5(b).

## 6.4.2 Solution to Problem 2—"Optimal" State Sequence

Unlike Problem 1, for which an exact solution can be given, there are several possible ways of solving Problem 2—namely, finding the "optimal" state sequence associated with the given observation sequence. The difficulty lies with the definition of the optimal state sequence—that is, there are several possible optimality criteria. For example, one possible

标准

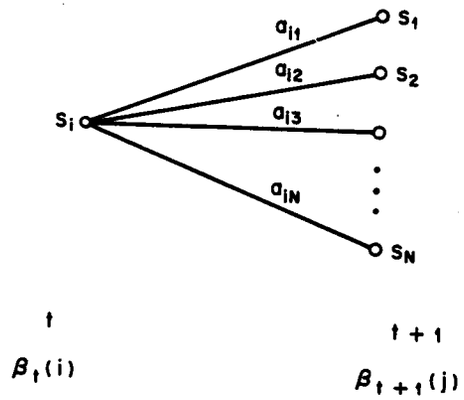



**Figure 6.6** Sequence of operations required for the computation of the backward variable $\beta_t(i)$.

在每個時間 t 最類似的 state

optimality criterion is to choose the states $q_t$ that are *individually* most likely at each time $t$. This optimality criterion maximizes the expected number of correct individual states. To implement this solution to Problem 2, we can define the a posteriori probability variable

$$\gamma_t(i) = P(q_t = i|\mathbf{O}, \lambda) \tag{6.26}$$

that is, the probability of being in state $i$ at time $t$, given the observation sequence $\mathbf{O}$, and the model $\lambda$. We can express $\gamma_t(i)$ in several forms, including

$$\begin{aligned}\gamma_t(i) &= P(q_t = i \mid \mathbf{O}, \lambda) \\ &= \frac{P(\mathbf{O}, q_t = i \mid \lambda)}{P(\mathbf{O} \mid \lambda)} \\ &= \frac{P(\mathbf{O}, q_t = i \mid \lambda)}{\sum_{i=1}^{N} P(\mathbf{O}, q_t = i \mid \lambda)}.\end{aligned} \tag{6.27}$$

Since $P(\mathbf{O}, q_t = i \mid \lambda)$ is equal to $\alpha_t(i)\beta_t(i)$, we can write $\gamma_t(i)$ as

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)} \tag{6.28}$$

where we see that $\alpha_t(i)$ accounts for the partial observation sequence $\mathbf{o}_1\,\mathbf{o}_2 \ldots \mathbf{o}_t$ and state $i$ at $t$, while $\beta_t(i)$ accounts for the remainder of the observation sequence $\mathbf{o}_{t+1}\mathbf{o}_{t+2} \ldots \mathbf{o}_T$, given state $q_t = i$ at $t$.



Using $\gamma_t(i)$, we can solve for the individually most likely state $q_t^*$ at time $t$, as

求出每一時刻最有能的 state
但整個 state sequence 未必是最有能的

$$q_t^* = \arg \max_{1 \le i \le N} [\gamma_t(i)], \qquad 1 \le t \le T. \tag{6.29}$$

Although Eq. (6.29) maximizes the expected number of correct states (by choosing the most likely state for each $t$), there could be some problems with the resulting state sequence. For example, when the HMM has state transitions which have zero probability ($a_{ij} = 0$ for some $i$ and $j$), the "optimal" state sequence may, in fact, not even be a valid state sequence. This is because the solution of Eq. (6.29) simply determines the most likely state at every instant, without regard to the probability of occurrence of sequences of states.

One possible solution to the above problem is to modify the optimality criterion. For example, one could solve for the state sequence that maximizes the expected number of correct pairs of states $(q_t, q_{t+1})$, or triples of states $(q_t, q_{t+1}, q_{t+2})$, etc. Although these criteria might be reasonable for some applications, the most widely used criterion is to find the *single* best state sequence (path)—that is, to maximize $P(\mathbf{q}|\mathbf{O}, \lambda)$, which is equivalent to maximizing $P(\mathbf{q}, \mathbf{O}|\lambda)$. A formal technique for finding this single best state sequence exists, based on dynamic programming methods, and is called the Viterbi algorithm [15, 16].

#### 6.4.2.1 The Viterbi Algorithm

演算法 最佳路徑

To find the single best state sequence, $\mathbf{q} = (q_1\, q_2 \ldots q_T)$, for the given observation sequence $\mathbf{O} = (\mathbf{o}_1\, \mathbf{o}_2 \ldots \mathbf{o}_T)$, we need to define the quantity

$$\delta_t(i) = \max_{q_1, q_2, \ldots, q_{t-1}} P[q_1 q_2 \ldots q_{t-1},\ q_t = i,\ \mathbf{o}_1 \mathbf{o}_2 \ldots \mathbf{o}_t|\lambda] \tag{6.30}$$

that is, $\delta_t(i)$ is the best score (highest probability) along a single path, at time $t$, which accounts for the first $t$ observations and ends in state $i$. By induction we have

模糊 重新得到

$$\delta_{t+1}(j) = [\max_i \delta_t(i)\, a_{ij}] \cdot b_j(\mathbf{o}_{t+1}). \tag{6.31}$$

To actually retrieve the state sequence, we need to keep track of the argument that maximized Eq. (6.31), for each $t$ and $j$. We do this via the array $\psi_t(j)$. The complete procedure for finding the best state sequence can now be stated as follows:

1. Initialization

$$\delta_1(i) = \pi_i b_i(\mathbf{o}_1), \qquad 1 \le i \le N \tag{6.32a}$$

$$\psi_1(i) = 0. \tag{6.32b}$$

2. Recursion

對每一個 state (1~N) 都有一個 $\delta_t(j)$

$$\delta_t(j) = \max_{1 \le i \le N} [\delta_{t-1}(i)\, a_{ij}]\, b_j(\mathbf{o}_t), \qquad \begin{matrix} 2 \le t \le T \\ 1 \le j \le N \end{matrix} \tag{6.33a}$$

$$\psi_t(j) = \arg \max_{1 \le i \le N} [\delta_{t-1}(i)\, a_{ij}], \qquad \begin{matrix} 2 \le t \le T \\ 1 \le j \le N. \end{matrix} \tag{6.33b}$$

3. Termination

$$P^* = \max_{1\le i\le N} [\delta_T(i)] \tag{6.34a}$$

$$q_T^* = \arg\max_{1\le i\le N} [\delta_T(i)]. \tag{6.34b}$$

4. Path (state sequence) backtracking

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \qquad t = T-1, T-2, \ldots, 1. \tag{6.35}$$

It should be noted that the Viterbi algorithm is similar (except for the backtracking step) in implementation to the forward calculation of Eqs. (6.19)–(6.21). The major difference is the maximization in Eq. (6.33a) over previous states, which is used in place of the summing procedure in Eq. (6.20). It also should be clear that a lattice (or trellis) structure efficiently implements the computation of the Viterbi procedure.

#### 6.4.2.2 Alternative Viterbi Implementation

By taking logarithms of the model parameters, the Viterbi algorithm of the preceding section can be implemented without the need for any multiplications. Thus:

**0. Preprocessing**

$$\begin{aligned} \tilde{\pi}_i &= \log(\pi_i), && 1 \le i \le N \\ \tilde{b}_i(\mathbf{o}_t) &= \log[b_i(\mathbf{o}_t)], && 1 \le i \le N,\ 1 \le t \le T \\ \tilde{a}_{ij} &= \log(a_{ij}), && 1 \le i, j \le N \end{aligned}$$

**1. Initialization**

$$\begin{aligned} \tilde{\delta}_1(i) &= \log(\delta_1(i)) = \tilde{\pi}_i + \tilde{b}_i(\mathbf{o}_1), && 1 \le i \le N \\ \psi_1(i) &= 0, && 1 \le i \le N \end{aligned}$$

**2. Recursion**

$$\tilde{\delta}_t(j) = \log(\delta_t(j)) = \max_{1\le i\le N} \left[\tilde{\delta}_{t-1}(i) + \tilde{a}_{ij}\right] + \tilde{b}_j(\mathbf{o}_t)$$

$$\psi_t(j) = \arg\max_{1\le i\le N} [\tilde{\delta}_{t-1}(i) + \tilde{a}_{ij}], \qquad 2 \le t \le T,\ 1 \le j \le N$$

**3. Termination**

$$\tilde{P}^* = \max_{1\le i\le N} [\tilde{\delta}_T(i)]$$

$$q_T^* = \arg\max_{1\le i\le N} [\tilde{\delta}_T(i)]$$

**4. Backtracking**

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \qquad t = T-1, T-2, \ldots, 1$$

The calculation required for this alternative implementation is on the order of $N^2T$ additions (plus the calculation for preprocessing). Because the preprocessing needs to be performed once and saved, its cost is negligible for most systems.



**Exercise 6.3**

Given the model of the coin-toss experiment used in Exercise 6.2 (i.e., three different coins) with probabilities

| | State 1 | State 2 | State 3 |
|---|---|---|---|
| $P(H)$ | 0.5 | 0.75 | 0.25 |
| $P(T)$ | 0.5 | 0.25 | 0.75 |

and with all state transition probabilities equal to 1/3, and with initial probabilities equal to 1/3, for the observation sequence

$$\mathbf{O} = (HHHHTHTTTT)$$

find the most likely path with the Viterbi algorithm.

**Solution 6.3**

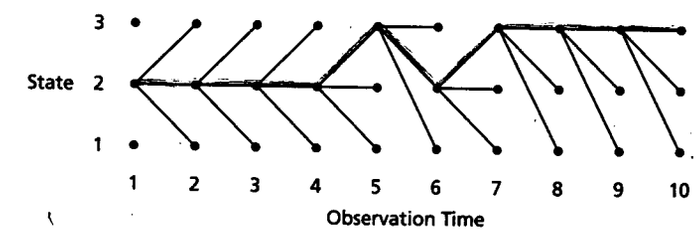
Since all $a_{ij}$ terms are equal to 1/3, we can omit these terms (as well as the initial state probability term), giving

$$\delta_1(1) = 0.5, \quad \delta_1(2) = 0.75, \quad \delta_1(3) = 0.25.$$

The recursion for $\delta_t(j)$ gives ($2 \le t \le 10$)

$$\begin{aligned}
&\delta_2(1) = (0.75)(0.5), && \delta_2(2) = (0.75)^2, && \delta_2(3) = (0.75)(0.25) \\
&\delta_3(1) = (0.75)^2(0.5), && \delta_3(2) = (0.75)^3, && \delta_3(3) = (0.75)^2(0.25) \\
&\delta_4(1) = (0.75)^3(0.5), && \delta_4(2) = (0.75)^4, && \delta_4(3) = (0.75)^3(0.25) \\
&\delta_5(1) = (0.75)^4(0.5), && \delta_5(2) = (0.75)^4(0.25), && \delta_5(3) = (0.75)^5 \\
&\delta_6(1) = (0.75)^5(0.5), && \delta_6(2) = (0.75)^6, && \delta_6(3) = (0.75)^5(0.25) \\
&\delta_7(1) = (0.75)^6(0.5), && \delta_7(2) = (0.75)^6(0.25), && \delta_7(3) = (0.75)^7 \\
&\delta_8(1) = (0.75)^7(0.5), && \delta_8(2) = (0.75)^7(0.25), && \delta_8(3) = (0.75)^8 \\
&\delta_9(1) = (0.75)^8(0.5), && \delta_9(2) = (0.75)^8(0.25), && \delta_9(3) = (0.75)^9 \\
&\delta_{10}(1) = (0.75)^9(0.5), && \delta_{10}(2) = (0.75)^9(0.25), && \delta_{10}(3) = (0.75)^{10}
\end{aligned}$$

This leads to a diagram (trellis) of the form:




Hence, the most likely state sequence is $\{2, 2, 2, 2, 3, 2, 3, 3, 3, 3\}$.

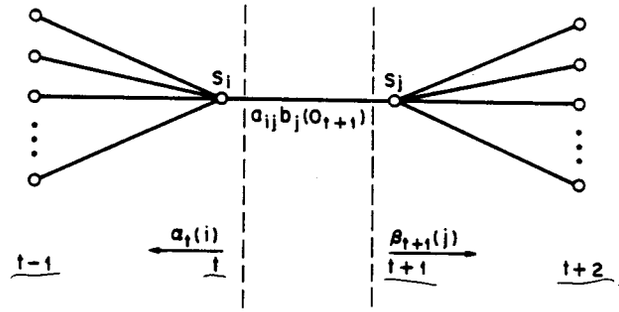



**Figure 6.7** Illustration of the sequence of operations required for the computation of the joint event that the system is in state $i$ at time $t$ and state $j$ at time $t+1$.

### 6.4.3 Solution to Problem 3—Parameter Estimation

The third, and by far the most difficult, problem of HMMs is to determine a method to adjust the model parameters $(A, B, \pi)$ to satisfy a certain optimization criterion. There is no known way to analytically solve for the model parameter set that maximizes the probability of the observation sequence in a closed form. We can, however, choose $\lambda = (A, B, \pi)$ such that its likelihood, $P(\mathbf{O}|\lambda)$, is locally maximized using an iterative procedure such as the Baum-Welch method (also known as the EM (expectation-maximization) method [17]), or using gradient techniques [18]. In this section we discuss one iterative procedure, based primarily on the classic work of Baum and his colleagues, for choosing the maximum likelihood (ML) model parameters.

To describe the procedure for reestimation (iterative update and improvement) of HMM parameters, we first define $\xi_t(i,j)$, the probability of being in state $i$ at time $t$, and state $j$ at time $t+1$, given the model and the observation sequence, i.e.

$$\xi_t(i,j) = P(q_t = i,\ q_{t+1} = j|\mathbf{O}, \lambda). \tag{6.36}$$

The paths that satisfy the conditions required by Eq. (6.36) are illustrated in Figure 6.7. From the definitions of the forward and backward variables, we can write $\xi_t(i,j)$ in the form

$$\begin{aligned}\xi_t(i,j) &= \frac{P(q_t = i,\ q_{t+1} = j,\ \mathbf{O} \mid \lambda)}{P(\mathbf{O} \mid \lambda)} \\ &= \frac{\alpha_t(i)\, a_{ij} b_j(\mathbf{o}_{t+1}) \beta_{t+1}(j)}{P(\mathbf{O}|\lambda)} \\ &= \frac{\alpha_t(i)\, a_{ij} b_j(\mathbf{o}_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_t(i)\, a_{ij} b_j(\mathbf{o}_{t+1}) \beta_{t+1}(j)}.\end{aligned} \tag{6.37}$$

We have previously defined $\gamma_t(i)$ as the probability of being in state $i$ at time $t$, given



the entire observation sequence and the model; hence, we can relate $\gamma_t(i)$ to $\xi_t(i,j)$ by summing over $j$, giving

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j). \tag{6.38}$$

If we sum $\gamma_t(i)$ over the time index $t$, we get a quantity that can be interpreted as the expected (over time) number of times that state $i$ is visited, or equivalently, the expected number of transitions made from state $i$ (if we exclude the time slot $t = T$ from the summation). Similarly, summation of $\xi_t(i,j)$ over $t$ (from $t = 1$ to $t = T-1$) can be interpreted as the expected number of transitions from state $i$ to state $j$. That is,

$$\sum_{t=1}^{T-1} \gamma_t(i) = \text{expected number of transitions from state } i \text{ in } \mathbf{O} \tag{6.39a}$$

$$\sum_{t=1}^{T-1} \xi_t(i,j) = \text{expected number of transitions from state } i \text{ to state } j \text{ in } \mathbf{O}. \tag{6.39b}$$

Using the above formulas (and the concept of counting event occurrences), we can give a method for reestimation of the parameters of an HMM. A set of reasonable reestimation formulas for $\pi, A$, and $B$ is

$$\bar{\pi}_i = \begin{array}{l}\text{expected frequency (number of times) in state } i \\ \text{at time } (t = 1) = \gamma_1(i)\end{array} \tag{6.40a}$$

$$\begin{aligned}\bar{a}_{ij} &= \frac{\text{expected number of transitions from state } i \text{ to state } j}{\text{expected number of transitions from state } i} \\ &= \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}\end{aligned} \tag{6.40b}$$

$$\begin{aligned}\bar{b}_j(k) &= \frac{\text{expected number of times in state } j \text{ and observing symbol } \mathbf{v}_k}{\text{expected number of times in state } j} \\ &= \frac{\sum_{\substack{t=1 \\ \text{s.t. } \mathbf{o}_t = \mathbf{v}_k}}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}.\end{aligned} \tag{6.40c}$$

If we define the current model as $\lambda = (A, B, \pi)$ and use that to compute the right-hand sides of Eqs. (6.40a)–(6.40c), and we define the reestimated model as $\bar{\lambda} = (\bar{A}, \bar{B}, \bar{\pi})$, as determined from the left-hand sides of Eqs. (6.40a)–(6.40c), then it has been proven by Baum and his colleagues that either (1) the initial model $\lambda$ defines a critical point of the

likelihood function, in which case $\bar{\lambda} = \lambda$; or (2) model $\bar{\lambda}$ is more likely than model $\lambda$ in the sense that $P(\mathbf{O}|\bar{\lambda}) > P(\mathbf{O}|\lambda)$; that is, we have found a new model $\bar{\lambda}$ from which the observation sequence is more likely to have been produced.

Based on the above procedure, if we iteratively use $\bar{\lambda}$ in place of $\lambda$ and repeat the reestimation calculation, we then can improve the probability of $\mathbf{O}$ being observed from the model until some limiting point is reached. The final result of this reestimation procedure is an ML estimate of the HMM. It should be pointed out that the forward-backward algorithm leads to local maxima only, and that in most problems of interest, the likelihood function is very complex and has many local maxima.

The reestimation formulas of Eqs. (6.40a)–(6.40c) can be derived directly by maximizing (using standard constrained optimization techniques) Baum's auxiliary function

$$Q(\lambda', \lambda) = \sum_{\mathbf{q}} P(\mathbf{O}, \mathbf{q}|\lambda') \log P(\mathbf{O}, \mathbf{q}|\lambda) \tag{6.41}$$

over $\lambda$. Because

$$Q(\lambda', \lambda) \geq Q(\lambda', \lambda') \Rightarrow P(\mathbf{O}|\lambda) \geq P(\mathbf{O}|\lambda') \tag{6.42}$$

we can maximize the function $Q(\lambda', \lambda)$ over $\lambda$ to improve $\lambda'$ in the sense of increasing the likelihood $P(\mathbf{O}|\lambda)$. Eventually the likelihood function converges to a critical point if we iterate the procedure.

$$\max_{\bar{\lambda}} [Q(\lambda, \bar{\lambda})] \Rightarrow P(O|\bar{\lambda}) \geq P(O|\lambda)$$

#### 6.4.3.1 Derivation of Reestimation Formulas from the $Q$ Function

The auxiliary function $Q(\lambda', \lambda)$ was defined in Eq. (6.41) as

$$Q(\lambda', \lambda) = \sum_{\mathbf{q}} P(\mathbf{O}, \mathbf{q}|\lambda') \log P(\mathbf{O}, \mathbf{q}|\lambda)$$

in which we can express $P$ and $\log P$ (in terms of the HMM parameters) as

$$P(\mathbf{O}, \mathbf{q}|\lambda) = \pi_{q_0} \prod_{t=1}^{T} a_{q_{t-1}q_t} b_{q_t}(\mathbf{o}_t)$$

$$\log P(\mathbf{O}, \mathbf{q}|\lambda) = \log \pi_{q_0} + \sum_{t=1}^{T} \log a_{q_{t-1}q_t} + \sum_{t=1}^{T} \log b_{q_t}(\mathbf{o}_t)$$

(There is a slight difference between the above equations and the expression of Eq. (6.17) in which the first observation is associated with the initial state before any state transition is made. This difference is inconsequential and should not impede our understanding of the method.) Thus we can write $Q(\lambda', \lambda)$ as

$$Q(\lambda', \lambda) = Q_\pi(\lambda', \boldsymbol{\pi}) + \sum_{i=1}^{N} Q_{a_i}(\lambda', \mathbf{a}_i) + \sum_{i=1}^{N} Q_{b_i}(\lambda', \mathbf{b}_i)$$

where

$$\boldsymbol{\pi} = [\pi_1, \pi_2, \ldots, \pi_N],$$



$$\mathbf{a}_i = [a_{i1}, a_{i2}, \ldots, a_{iN}], \quad \mathbf{b}_i \text{ is the parameter vector that defines } b_i(\cdot)$$

and

$$Q_\pi(\lambda', \boldsymbol{\pi}) = \sum_{i=1}^{N} P(\mathbf{O}, q_0 = i|\lambda') \log \pi_i$$

$$Q_{a_i}(\lambda', \mathbf{a}_i) = \sum_{j=1}^{N} \sum_{t=1}^{T} P(\mathbf{O}, q_{t-1} = i, q_t = j|\lambda') \log a_{ij}$$

$$Q_{b_i}(\lambda', \mathbf{b}_i) = \sum_{t=1}^{T} P(\mathbf{O}, q_t = i|\lambda') \log b_i(\mathbf{o}_t)$$

Because of the separability of $Q(\lambda', \lambda)$ into three independent terms, we can maximize $Q(\lambda', \lambda)$ over $\lambda$ by maximizing the individual terms separately, subject to the stochastic constraints

$$\sum_{j=1}^{N} \pi_j = 1$$

$$\sum_{j=1}^{N} a_{ij} = 1, \qquad \forall j$$

and (for discrete densities where $b_i(\mathbf{o}_t = \mathbf{v}_k) = b_i(k)$)

$$\sum_{k=1}^{K} b_i(k) = 1, \qquad \forall i.$$

Because the individual auxiliary functions all have the form

$$\sum_{j=1}^{N} w_j \log y_j$$

which, as a function of $\{y_j\}_{j=1}^{N}$, subject to the constraints $\sum_{j=1}^{N} y_j = 1$, $y_j \geq 0$, attains a global maximum at the single point

$$y_j = \frac{w_j}{\sum_{i=1}^{N} w_i}, \qquad j = 1, 2, \ldots, N$$

then the maximization leads to the model reestimate $\bar{\lambda} = [\bar{\pi}, \bar{A}, \bar{B}]$ where

$$\bar{\pi}_i = \frac{P(\mathbf{O}, q_0 = i|\lambda)}{P(\mathbf{O}|\lambda)}$$

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T} P(\mathbf{O}, q_{t-1} = i, q_t = j|\lambda)}{\sum_{t=1}^{T} P(\mathbf{O}, q_{t-1} = i|\lambda)}$$

$$\bar{b}_i(k) = \frac{\sum_{t=1}^{T} P(\mathbf{O}, q_t = i|\lambda)\ \delta(\mathbf{o}_t, \mathbf{v}_k)}{\sum_{t=1}^{T} P(\mathbf{O},\ q_t = i|\lambda)}$$

where we have defined

$$\delta(\mathbf{o}_t, \mathbf{v}_k) = 1 \quad \text{if } \mathbf{o}_t = \mathbf{v}_k$$
$$= 0 \quad \text{otherwise.}$$

Using the definitions of the forward variable, $\alpha_t(i) = P(\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_t, q_t = i|\lambda)$ and the backward variable, $\beta_t(i) = P(\mathbf{o}_{t+1}, \ldots, \mathbf{o}_T|q_t = i, \lambda)$, the reestimation transformations can be easily calculated as

$$P(\mathbf{O}, q_t = i|\lambda) = \alpha_t(i)\beta_t(i)$$
$$P(\mathbf{O}|\lambda) = \sum_{i=1}^{N} \alpha_t(i)\beta_t(i) = \sum_{i=1}^{N} \alpha_T(i)$$
$$P(\mathbf{O}, q_{t-1} = i,\ q_t = j|\lambda) = \alpha_{t-1}(i)\, a_{ij} b_j(\mathbf{o}_t)\beta_t(j)$$

giving

$$\bar{\pi}_i = \frac{\alpha_0(i)\beta_0(i)}{\sum_{j=1}^{N} \alpha_T(j)} = \gamma_0(i)$$

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T} \alpha_{t-1}(i)\, a_{ij} b_j(\mathbf{o}_t)\beta_t(j)}{\sum_{t=1}^{T} \alpha_{t-1}(i)\beta_{t-1}(i)} = \frac{\sum_{t=1}^{T} \xi_{t-1}(i,j)}{\sum_{t=1}^{T} \gamma_{t-1}(i)}$$

$$\bar{b}_i(k) = \frac{\sum_{t=1}^{T} \alpha_t(i)\beta_t(i)\delta(\mathbf{o}_t, \mathbf{v}_k)}{\sum_{t=1}^{T} \alpha_t(i)\beta_t(i)} = \frac{\sum_{\substack{t=1 \\ \text{s.t. } \mathbf{o}_t = \mathbf{v}_k}}^{T} \gamma_t(i)}{\sum_{t=1}^{T} \gamma_t(i)}$$

which are the formulas given in Eqs. (6.40a)–(6.40c).



### 6.4.4 Notes on the Reestimation Procedure

The reestimation formulas can be readily interpreted as an implementation of the EM algorithm of statistics [17] in which the E (expectation) step is the calculation of the auxiliary function $Q(\lambda', \lambda)$, (which is the expectation of log $P(\mathbf{O}, \mathbf{q}|\lambda)$), and the M (maximization) step is the maximization of $Q(\lambda', \lambda)$ over $\lambda$ to obtain $\bar{\lambda}$. Thus the Baum-Welch reestimation equations are essentially identical to the EM steps for this particular problem.

An important property of the reestimation procedure is that the stochastic constraints of the HMM parameters, namely

$$\sum_{i=1}^{N} \bar{\pi}_i = 1 \tag{6.43a}$$

$$\sum_{j=1}^{N} \bar{a}_{ij} = 1, \qquad 1 \le i \le N \tag{6.43b}$$

$$\sum_{k=1}^{M} \bar{b}_j(k) = 1, \qquad 1 \le j \le N \tag{6.43c}$$

are automatically incorporated at each iteration. By looking at the parameter estimation problem as a constrained optimization of $P(\mathbf{O}|\lambda)$ (subject to the constraints of Eq. (6.43)), we can formulate the solution procedure by use of the techniques of variational calculus to maximize $P$ (we use the notation $P = P(\mathbf{O}|\lambda)$ as shorthand in this section). Based on a standard Lagrange optimization setup using Lagrange multipliers, it can readily be shown that $P$ is maximized when the following conditions are met:

$$\pi_i = \frac{\pi_i \dfrac{\partial P}{\partial \pi_i}}{\sum_{k=1}^{N} \pi_k \dfrac{\partial P}{\partial \pi_k}} \tag{6.44a}$$

$$a_{ij} = \frac{a_{ij} \dfrac{\partial P}{\partial a_{ij}}}{\sum_{k=1}^{N} a_{ik} \dfrac{\partial P}{\partial a_{ik}}} \tag{6.44b}$$

$$b_j(k) = \frac{b_j(k) \dfrac{\partial P}{\partial b_j(k)}}{\sum_{\ell=1}^{M} b_j(\ell) \dfrac{\partial P}{\partial b_j(\ell)}}. \tag{6.44c}$$

By appropriate manipulation of Eq. (6.44), the right-hand sides of each equation can be readily shown to be *identical* to the right-hand sides of each part of Eqs. (6.40a)–(6.40c), thereby showing that the reestimation formulas are indeed exactly correct at critical points of $P$. In fact, the form of Eq. (6.44) is essentially that of a reestimation formula in which

the left-hand side is the reestimate and the right-hand side is computed using the current values of the variables.

Finally, we note that since the entire problem can be set up as an optimization problem, standard gradient techniques can be used to solve for "optimal" values of the model parameters. Such procedures have been tried and have been shown to yield solutions comparable to those of the standard reestimation procedures [18]. One critical shortcoming of standard gradient technique, as applied to the maximization of $P(\mathbf{O}|\lambda)$, is that the descent algorithms, which are critically dependent on taking a small step in the direction of the gradient, often do not produce *monotonic* improvement in the likelihood as the Baum-Welch reestimation is guaranteed by Eq. (6.42) to do.

## 6.5 TYPES OF HMMS

One way to classify types of HMMs is by the structure of the transition matrix $A$ of the Markov chain. Until now, we have only considered the special case of ergodic or fully connected HMMs in which every state of the model could be reached (in a single step) from every other state of the model. (Strictly speaking, an ergodic model has the property that every state can be reached from every other state in a finite but aperiodic number of steps.) As shown in Figure 6.8(a), for an $N = 4$ state model, this type of model has the property that every $a_{ij}$ coefficient is positive. Hence for the example of Figure 6.8(a) we have

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}.$$

For some applications, particularly those to be discussed later in this chapter, other types of HMMs have been found to account for observed properties of the signal being modeled better than the standard ergodic model. One such model is shown in Figure 6.8(b). This model is called a left-right model or a Bakis model ([11], [10]) because the underlying state sequence associated with the model has the property that, as time increases, the state index increases (or stays the same)—that is, the system states proceed from left to right. Clearly the left-right type of HMM has the desirable property that it can readily model signals whose properties change over time in a successive manner—e.g., speech. The fundamental property of all left-right HMMs is that the state-transition coefficients have the property

$$a_{ij} = 0, \qquad j < i \tag{6.45}$$

that is, no transitions are allowed to states whose indices are lower than that of the current state. Furthermore, the initial state probabilities have the property

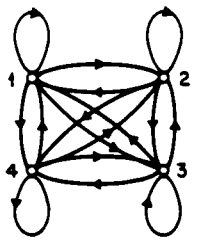
$$\pi_i = \begin{cases} 0, & i \neq 1 \\ 1, & i = 1 \end{cases} \tag{6.46}$$

because the state sequence must begin in state 1 (and end in state $N$). Often, with left-right

state 1 → (state 2 → ⋯) → state N

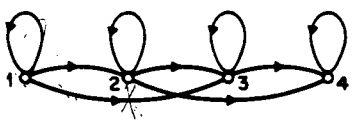


fully-connected model



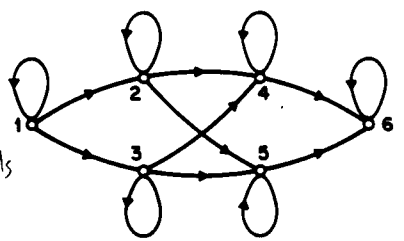

(a)

Left-right model
Bakis model




(b)

cross-coupled connection of two left-right HMMs




(c)

**Figure 6.8** Illustration of three distinct types of HMMs. (a) A 4-state ergodic model. (b) A 4-state left-right model. (c) A 6-state parallel path left-right model.

models, additional constraints are placed on the state-transition coefficients to make sure that large changes in state indices do not occur; hence a constraint of the form

相隔太遠的state

$$a_{ij} = 0, \qquad j > i + \Delta i \tag{6.47}$$

$a_{ij} = 0$

is often used. In particular, for the example of Figure 6.8(b), the value of $\Delta i$ is 2; that is, no jumps of more than two states are allowed. The form of the state-transition matrix for the example of Figure 6.8(b) is thus

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 \\ 0 & a_{22} & a_{23} & a_{24} \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{bmatrix}.$$

It should be clear that, for the last state in a left-right model, the state-transition coefficients are specified as

$$a_{NN} = 1 \tag{6.48a}$$

$$a_{Ni} = 0, \qquad i < N. \tag{6.48b}$$

Besides the above fully connected and left-right models, there are many other possible variations and combinations. By way of example, Figure 6.8(c) shows a cross-coupled connection of two parallel left-right HMMs. Strictly speaking, this model is a left-right model (it obeys all the $a_{ij}$ constraints); however, it has certain flexibility not present in a strict left-right model (i.e., one without parallel paths).

It should be clear that the imposition of the constraints of the left-right model, or those of the constrained jump model, essentially have no effect on the reestimation procedure. This is the case because any HMM parameter set to zero initially will remain at zero throughout the reestimation procedure (see Eq. (6.44)).

## 6.6 CONTINUOUS OBSERVATION DENSITIES IN HMMS

All of our discussion to this point has considered only when the observations were characterized as discrete symbols chosen from a finite alphabet, and therefore we could use a discrete probability density within each state of this model ([19]–[21]). The problem with this approach, at least for some applications, is that the observations are often continuous signals (or vectors). Although it is possible to convert such continuous signal representations into a sequence of discrete symbols via vector quantization codebooks and other methods, there might be serious degradation associated with such discretization of the continuous signal. Hence it would be advantageous to be able to use HMMs with continuous observation densities to model continuous signal representations directly.

To use a continuous observation density, some restrictions must be placed on the form of the model probability density function (pdf) to ensure that the parameters of the pdf can be reestimated in a consistent way. The most general representation of the pdf, for which a reestimation procedure has been formulated, is a finite mixture of the form

$$b_j(\mathbf{o}) = \sum_{k=1}^{M} c_{jk} \mathcal{N}(\mathbf{o}, \boldsymbol{\mu}_{jk}, \mathbf{U}_{jk}), \qquad 1 \le j \le N \tag{6.49}$$

where $\mathbf{o}$ is the observation vector being modeled, $c_{jk}$ is the mixture coefficient for the $k$th mixture in state $j$ and $\mathcal{N}$ is any log-concave or elliptically symmetric density [18] (e.g., Gaussian). Without loss of generality, we assume that $\mathcal{N}$ is Gaussian in Eq. (6.49) with mean vector $\boldsymbol{\mu}_{jk}$ and covariance matrix $\mathbf{U}_{jk}$ for the $k$th mixture component in state $j$. The mixture gains $c_{jk}$ satisfy the stochastic constraint

$$\sum_{k=1}^{M} c_{jk} = 1, \qquad 1 \le j \le N \tag{6.50a}$$

$$c_{jk} \ge 0, \; 1 \le j \le N, \; 1 \le k \le M \tag{6.50b}$$



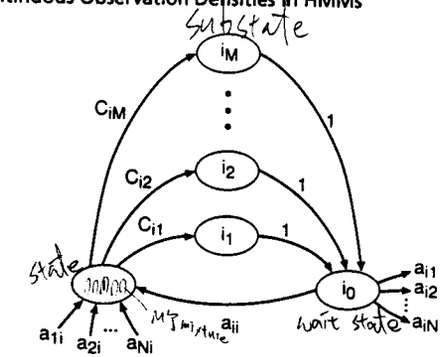



**Figure 6.9** Equivalence of a state with a mixture density to a multistate single-density distribution (after Juang et al. [21]).

so that the pdf is properly normalized, i.e.,

$$\int_{-\infty}^{\infty} b_j(\mathbf{o}) d\mathbf{o} = 1, \qquad 1 \le j \le N. \tag{6.51}$$

The pdf of Eq. (6.49) can be used to approximate, arbitrarily closely, any finite, continuous-density function. Hence it can be applied to a wide range of problems.

It has been shown that an HMM state with a mixture density is equivalent to a multistate single-mixture density model in the following way [21]. Consider a state $i$ with an $M$-mixture Gaussian density. Because the mixture gain coefficients sum up to 1, they define a set of transition coefficients to substates $i_1$ (with transition probability $c_{i1}$), $i_2$ (with transition probability $c_{i2}$) through $i_M$ (with transition probability $c_{iM}$). Within each substate $i_k$, there is a single mixture with mean $\boldsymbol{\mu}_{ik}$ and variance $\mathbf{U}_{ik}$ (see Figure 6.9 for a graphical interpretation). Each substate makes a transition to a wait state $i_0$ with probability 1. The distribution of the composite set of substates (each with a single density) is mathematically equivalent to the composite mixture density within a single state.

It can be shown that the reestimation formulas for the coefficients of the mixture density, i.e., $c_{jk}$, $\boldsymbol{\mu}_{jk}$, and $\mathbf{U}_{jk}$, are of the form

$$\bar{c}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k)}{\sum_{t=1}^{T} \sum_{k=1}^{M} \gamma_t(j,k)} \tag{6.52}$$

$$\bar{\boldsymbol{\mu}}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k) \cdot \mathbf{o}_t}{\sum_{t=1}^{T} \gamma_t(j,k)} \tag{6.53}$$

$$\bar{\mathbf{U}}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k) \cdot (\mathbf{o}_t - \boldsymbol{\mu}_{jk})(\mathbf{o}_t - \boldsymbol{\mu}_{jk})'}{\sum_{t=1}^{T} \gamma_t(j,k)} \tag{6.54}$$

transpose

where prime denotes vector transpose and where $\gamma_t(j,k)$ is the probability of being in state $j$ at time $t$ with the $k$th mixture component accounting for $\mathbf{o}_t$, i.e.,

$$\gamma_t(j,k) = \left[ \frac{\alpha_t(j)\beta_t(j)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)} \right] \left[ \frac{c_{jk}\mathcal{N}(\mathbf{o}_t, \boldsymbol{\mu}_{jk}, \mathbf{U}_{jk})}{\sum_{m=1}^{M} c_{jm}\mathcal{N}(\mathbf{o}_t, \boldsymbol{\mu}_{jm}, \mathbf{U}_{jm})} \right].$$

state j を N 了 state 中の t 時 mixture M を M 了 mixture 中の k 時

(The term $\gamma_t(j,k)$ generalizes to $\gamma_t(j)$ of Eq. (6.26) in the case of a simple mixture, or a discrete density.) The reestimation formula for $a_{ij}$ is identical to the one used for discrete observation densities (i.e., Eq. (6.40(b))). The interpretation of Eqs. (6.52)–(6.54) is fairly straightforward. The reestimation formula for $c_{jk}$ is the ratio between the expected number of times the system is in state $j$ using the $k$th mixture component, and the expected number of times the system is in state $j$. Similarly, the reestimation formula for the mean vector $\boldsymbol{\mu}_{jk}$ weights each numerator term of Eq. (6.52) by the observation, thereby giving the expected value of the portion of the observation vector accounted for by the $k$th mixture component. A similar interpretation can be given for the reestimation term for the covariance matrix $\mathbf{U}_{jk}$.

## 6.7 AUTOREGRESSIVE HMMS

Another very interesting class of HMMs that is particularly applicable to speech processing is the class of autoregressive HMMs ([22, 23]). For this class, the observation vectors are drawn from an autoregression process. (The autoregressive density is, of course, just another continuous-probability density. However, we elaborate on the subject here separately from Section 6.6 because of its importance in speech analysis as will be shown later.)

To be more specific, consider the observation vector $\mathbf{o} = (x_0, x_1, x_2, \ldots, x_{K-1})$. The elements, $x_i$, could be simply the speech waveform samples. The components of $\mathbf{o}$ are assumed to be from an autoregressive Gaussian source, satisfying the relationship

$$x_k = -\sum_{i=1}^{p} a_i x_{k-i} + e_k \tag{6.55}$$

where $e_k$, $k = 0, 1, 2, \ldots, K-1$ are Gaussian, independent, identically distributed random variables with zero mean and variance $\sigma_e^2$, and $a_i$, $i = 1, 2, \ldots, p$, are the autoregression or predictor coefficients. It can be shown that for large $K$ [22, 23], the density function for $\mathbf{o}$



is approximately

$$f(\mathbf{o}) = (2\pi\sigma_e^2)^{-K/2} \exp\left\{ -\frac{1}{2\sigma_e^2}\delta(\mathbf{o}, \mathbf{a}) \right\} \tag{6.56}$$

where

$$\delta(\mathbf{o}, \mathbf{a}) = r_a(0)r(0) + 2\sum_{i=1}^{p} r_a(i)r(i) \tag{6.57a}$$

$$\mathbf{a} = [1, a_1, a_2, \ldots, a_p]', \qquad (a_0 = 1) \tag{6.57b}$$

$$r_a(i) = \sum_{n=0}^{p-i} a_n a_{n+i}, \qquad 1 \le i \le p \tag{6.57c}$$

$$r(i) = \sum_{n=0}^{K-i-1} x_n x_{n+i} \qquad 0 \le i \le p. \tag{6.57d}$$

In the above equations $r(i)$ is the autocorrelation of the observation samples, and $r_a(i)$ is the autocorrelation of the autoregressive coefficients. Furthermore, $\delta(\mathbf{o}, \mathbf{a})$ is a form of residual energy resulting from inverse filtering the data $x_i$ with an all-zero filter defined by $\mathbf{a}$. (See Eqs. 4.40–44.)

As discussed in Chapter 4, the signal level in the observation $\mathbf{o}$ is often treated in a different fashion from the general spectral shape when it comes to speech-pattern comparison. One way to separate the signal level from the spectral shape is to use gain normalization; that is, we use $\hat{\mathbf{o}}$ instead of $\mathbf{o}$ as the observation, where

$$\hat{\mathbf{o}} = \mathbf{o}/\sigma_{eo} \tag{6.58}$$

and where $\sigma_{eo}^2$ is the minimum linear prediction residual energy per sample. (It is shown in Exercise 6.5 that $\sigma_{eo}^2 = (\sigma_e^2)_{\mathrm{ML}}$ for *the given observation* $\mathbf{o}$.) The elements of $\hat{\mathbf{o}}$, $\hat{x}_k = x_k/\sigma_{eo}$, still satisfy the autoregressive relationship

$$\hat{x}_k = -\sum_{i=1}^{p} a_i \hat{x}_{k-i} + \hat{e}_k. \tag{6.59}$$

However, the variance of $\hat{e}_k$ is now unity. Therefore, we can write the probability density function for the output of an all-pole system defined by $\mathbf{a}$, driven by a zero mean, unit variance Gaussian i.i.d. sequence, as

$$f(\hat{\mathbf{o}}) = (2\pi)^{-K/2} \exp\left\{ -\frac{1}{2}\delta(\hat{\mathbf{o}}, \mathbf{a}) \right\} \tag{6.60}$$

if the data dimension $K$ is sufficiently large. (Note that the normalization factor $\sigma_{eo}$ depends on the original data observation $\mathbf{o}$.) This type of pdf is often referred to as a "gain-independent" pdf.

The way in which we use a Gaussian autoregressive density in HMMs is straightfor-

ward. We assume a mixture density of the form

$$b_j(\mathbf{o}) = \sum_{k=1}^{M} c_{jk} b_{jk}(\mathbf{o}) \tag{6.61}$$

where each $b_{jk}(\mathbf{o})$ is the density defined by Eq. (6.60) with autoregression vector $\mathbf{a}_{jk}$ (or equivalently by autocorrelation vector $\mathbf{r}_{a_{jk}}$); that is,

$$b_{jk}(\mathbf{o}) = (2\pi)^{-K/2} \exp\left\{-\frac{1}{2}\delta(\mathbf{o}, \mathbf{a}_{jk})\right\}. \tag{6.62}$$

A reestimation formula for the sequence autocorrelation for the $j$th state, $k$th mixture component has been derived [22, 23], and is of the form

$$\bar{\mathbf{r}}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k) \cdot \mathbf{r}_t}{\sum_{t=1}^{T} \gamma_t(j,k)} \tag{6.63a}$$

where $\mathbf{r}_t = [r_t(0), r_t(1), \ldots, r_t(p)]'$ is the autocorrelation vector as defined by Eq. (6.57d) for the $t^{\text{th}}$ frame, and $\gamma_t(j,k)$ is defined as the probability of being in state $j$ at time $t$ and using mixture component $k$, i.e.,

$$\gamma_t(j,k) = \left[\frac{\alpha_t(j)\beta_t(j)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)}\right]\left[\frac{c_{jk} b_{jk}(\mathbf{o}_t)}{\sum_{k=1}^{M} c_{jk} b_{jk}(\mathbf{o}_t)}\right]. \tag{6.63b}$$

It can be seen that $\bar{\mathbf{r}}_{jk}$ is a weighted sum (by probability of occurrence) of the normalized autocorrelations of the frames in the observation sequence. From $\bar{\mathbf{r}}_{jk}$, one can solve a set of normal equations to obtain the corresponding autoregressive coefficient vector $\bar{\mathbf{a}}_{jk}$, for the $k$th mixture of state $j$. The new autocorrection vectors of the autoregression coefficients as needed in the density function can then be calculated using Eq. (6.57c), thereby closing the reestimation loop.

**Exercise 6.4**

The probability density function (pdf) [22, 23] of Eq. (6.56) is defined by parameters $\sigma_e^2$ and $\mathbf{a}$. Given a data observation vector $\mathbf{o} = (x_0, x_1, \ldots, x_{K-1})$, determine the maximum likelihood estimate of $\sigma_e^2$ and $\mathbf{a}$ that best characterizes the observed $\mathbf{o}$.

**Solution 6.4**

We write the likelihood function of $\mathbf{o}$ as a function of $\sigma_e^2$ and $\mathbf{a}$ as

$$f(\mathbf{o}|\sigma_e^2, \mathbf{a}) = (2\pi\sigma_e^2)^{-K/2} \exp\left\{-\frac{1}{2\sigma_e^2}\delta(\mathbf{o}, \mathbf{a})\right\}$$

and the log likelihood function as

$$\log f(\mathbf{o}|\sigma_e^2, \mathbf{a}) = -\frac{K}{2}\log(2\pi\sigma_e^2) - \frac{\delta(\mathbf{o}, \mathbf{a})}{2\sigma_e^2}.$$



Therefore, the maximum likelihood (ML) estimate is

$$\begin{aligned}(\mathbf{a})_{\text{ML}} &= \arg\max_{\mathbf{a}} \log f(\mathbf{o}|\sigma_e^2, \mathbf{a}) = \arg\min_{\mathbf{a}} \delta(\mathbf{o}, \mathbf{a}) \\ &= \arg\min_{\mathbf{a}} (\mathbf{a}'\mathbf{R}\mathbf{a})\end{aligned}$$

where $\mathbf{R} = [r_{ij}]$ with $r_{ij} = r(|i-j|)$ as defined by Eq. (6.57d). This establishes the relationship between maximum likelihood estimation and the classical method of autocorrelation matching (also called the method of minimization of prediction residuals) for LPC analysis (see Eq. (4.8)–(4.10)). Furthermore, it is easy to show that

$$(\sigma_e^2)_{\text{ML}} = \min_{\mathbf{a}} \delta(\mathbf{o}, \mathbf{a})/K$$

which leads to

$$\max_{\sigma_e^2, \mathbf{a}} \log f(\mathbf{o}|\sigma_e^2, \mathbf{a}) = -\frac{K}{2}\log\left[2\pi(\sigma_e^2)_{\text{ML}}\right] - \frac{K}{2}.$$

**Exercise 6.5**

The pdf of Eq. (6.56) is related to the Itakura-Saito distortion measure of Eq. (4.45). Establish this relationship.

**Solution 6.5**

Consider the following likelihood difference

$$\begin{aligned}L_d &= \left[\max_{\sigma_e^2, \mathbf{a}} \log f(\mathbf{o}|\sigma_e^2, \mathbf{a})\right] - \log f(\mathbf{o}|\sigma_e^2, \mathbf{a}) \\ &= -\frac{K}{2}\log\left[2\pi(\sigma_e^2)_{\text{ML}}\right] - \frac{K}{2} + \frac{K}{2}\log(2\pi\sigma_e^2) + \frac{\delta(\mathbf{o}, \mathbf{a})}{2\sigma_e^2} \\ &= \frac{K}{2}\left[\frac{1}{K\sigma_e^2}\delta(\mathbf{o}, \mathbf{a}) + \log\sigma_e^2 - \log(\sigma_e^2)_{\text{ML}} - 1\right].\end{aligned}$$

For distortion measures, we denote the all-pole (power) spectra by $\sigma_{e0}^2/|A_0(e^{j\omega})|^2$ and $\sigma_e^2/|A(e^{j\omega})|^2$, corresponding to parameter sets $\{(\sigma_e^2)_{\text{ML}}, (\mathbf{a})_{\text{ML}}\}$ and $\{\sigma_e^2, \mathbf{a}\}$, respectively. ($\sigma_{f0}^2 = (\sigma_e^2)_{\text{ML}}$.) Then, the bracketed term in the log likelihood difference, $L_d$, is simply $d_{\text{IS}}(\sigma_{f0}^2/|A_0|^2, \sigma_e^2/|A|^2)$ according to Eq. (4.45) because

$$\sigma_{f0}^2 \int_{-\pi}^{\pi} \frac{|A(e^{j\omega})|^2}{|A_0(e^{j\omega})|^2}\frac{d\omega}{2\pi} = \frac{1}{K}\delta(\mathbf{o}, \mathbf{a})$$

due to autocorrelation matching and Eq. (6.57a). (Note that we have defined $\sigma_e^2$ (and $\sigma_{e0}^2$) as the sample variance. The factor $K$ would disappear from the above equation if we use the total frame prediction residual variance instead of the sample variance.) Therefore,

$$\begin{aligned}f(\mathbf{o}|\sigma_e^2, \mathbf{a}) &= (2\pi\sigma_e^2)^{-K/2} \exp\left\{-\frac{K}{2}\left[d_{\text{IS}}\left(\frac{\sigma_{e0}^2}{|A_0|^2}, \frac{\sigma_e^2}{|A|^2}\right) + \log\sigma_{f0}^2 - \log\sigma_e^2 + 1\right]\right\} \\ &= G_1(\sigma_{f0}^2, \sigma_e^2) \exp\left\{-\frac{K}{2} d_{\text{IS}}\left(\frac{\sigma_{f0}^2}{|A_0|^2}, \frac{\sigma_e^2}{|A|^2}\right)\right\}\end{aligned}$$

where $G_1(\sigma_{f_0}^2, \sigma_c^2)$ encompasses only the gain terms.

**Exercise 6.6**
The pdf of Eq. (6.60) is related to the likelihood ratio distortion measure of Eq. (4.53). Establish this relationship.

**Solution 6.6**
Similar to the case of Eq. (6.56),

$$\frac{1}{K}\delta(\hat{\mathbf{o}}, \mathbf{a}) = \int_{-\pi}^{\pi} \frac{|A(e^{j\omega})|^2}{|A_0(e^{j\omega})|^2} \frac{d\omega}{2\pi}$$
$$= d_{\mathrm{LR}}\left(\frac{1}{|A_0|^2}, \frac{1}{|A|^2}\right) + 1.$$

Therefore,

$$f(\hat{\mathbf{o}}|\mathbf{a}) = (2\pi)^{-K/2} \exp\left\{-\frac{K}{2}\left[d_{\mathrm{LR}}\left(\frac{1}{|A_0|^2}, \frac{1}{|A|^2}\right) + 1\right]\right\}$$
$$= G_2 \exp\left\{-\frac{K}{2} d_{\mathrm{LR}}\left(\frac{1}{|A_0|^2}, \frac{1}{|A|^2}\right)\right\}.$$

Note that when the pdf is expressed in terms of the distortion measures ($d_{\mathrm{IS}}$ or $d_{\mathrm{LR}}$), the exponential term includes a factor $K$ that represents the data dimension. In practice, this factor $K$ is replaced by an effective frame length $\hat{K}$, which is the net shift of consecutive data frames. Thus, if consecutive data frames (vectors) have 2/3 overlap, then an effective frame length $\hat{K} = K/3$ is appropriate, so that the rate of characteristic change in terms of the spectral parameters $\mathbf{a}$ is kept at the original waveform sampling rate.

## 6.8 VARIANTS ON HMM STRUCTURES—NULL TRANSITIONS AND TIED STATES

Throughout this chapter we have considered HMMs in which the observations were associated with states of the model. It is also possible to consider models in which the observations are associated with the arcs of the model. This type of HMM has been used extensively in the IBM continuous speech recognizer [13]. It has been found useful, for this type of model, to allow transitions that produce no output; that is, jumps from one state to another that produce no observation [13]. Such transitions are called null transitions and are designated by a dashed line, with the symbol $\phi$ used to denote the null output.

Figure 6.10 illustrates three examples (from speech-processing tasks) where null arcs have been successfully utilized. The example of part (a) corresponds to an HMM (a left-right model) with a large number of states in which it is possible to omit transitions between any pair of states. Hence it is possible to generate observation sequences with as few as



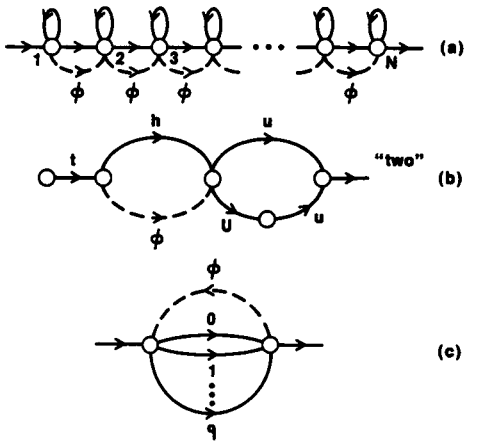



**Figure 6.10** Examples of networks incorporating null transitions. (a) Left-right model. (b) Finite state network. (c) Grammar network.

one observation and still account for a path that begins in state one and ends in state $N$.

The example of Figure 6.10(b) is a finite state network (FSN) representation of a word in terms of linguistic unit models (i.e., the sound on each arc is itself an HMM). For this model the null transition gives a compact and efficient way of describing alternative word pronunciations (i.e., symbol deletions).

Finally the FSN of Figure 6.10(c) shows how the ability to insert a null transition into a grammar network allows a relatively simple network to generate arbitrarily long word (digit) sequences. In the example shown in Figure 6.10(c), the null transition allows the network to generate arbitrary sequences of digits of arbitrary length by returning to the initial state after each individual digit is produced.

parameter tying

Another interesting variation in the HMM structure is the concept of parameter tying [13]. Basically, the idea is to set up an equivalence relation between HMM parameters in different states. In this manner the number of independent parameters in the model is reduced and the parameter estimation becomes somewhat simpler and in some cases more reliable. Parameter tieing is used when the observation density, for example, is known to be the same in two or more states. Such cases occur often in characterizing speech sounds. The technique is especially appropriate when there is insufficient training data to estimate, reliably, a large number of model parameters. For cases such as these, it is appropriate to tie model parameters so as to reduce the number of parameters (i.e., size of the model), thereby making the parameter estimation problem somewhat simpler. We will discuss this method later in this chapter.

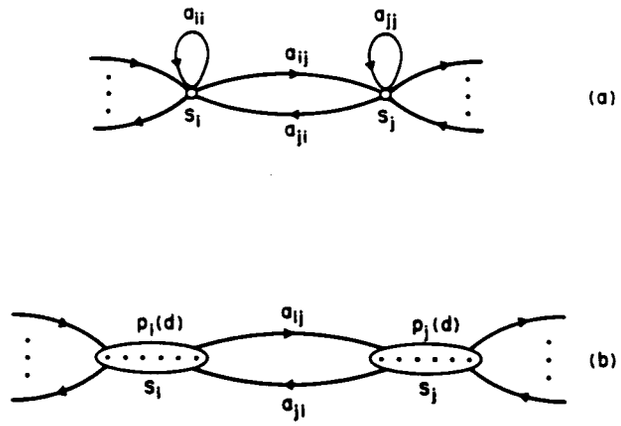



**Figure 6.11** Illustration of general interstate connections of (a) a normal HMM with exponential state duration density, and (b) a variable duration HMM with specified state densities and no self-transitions from a state back to itself.

## 6.9 INCLUSION OF EXPLICIT STATE DURATION DENSITY IN HMMS

Earlier we showed via Eq. (6.5) that the inherent duration probability density $p_i(d)$ associated with state $i$, with self-transition coefficient $a_{ii}$, was of the form

$$\begin{aligned} p_i(d) &= (a_{ii})^{d-1}(1 - a_{ii}) \\ &= \text{probability of } d \text{ consecutive observations in state } i. \end{aligned} \tag{6.64}$$

For most physical signals, this exponential state duration density is inappropriate. Instead we would prefer to explicitly model duration density in some analytic form. (An extensive treatment of state duration modeling can be found in the work of Ferguson of IDA [14], which is the basis of the material presented here. Other valuable references include [24] and [25].) Figure 6.11 illustrates, for a pair of model states $i$ and $j$, the differences between HMMs without and with explicit duration density. In part (a) the states have exponential duration densities based on self-transition coefficients $a_{ii}$ and $a_{jj}$, respectively. In part (b), the self-transition coefficients are set to zero, and an explicit duration density is specified. For this case, a transition is made only after the appropriate number of observations have occurred in the state (as specified by the duration density). Such a model is called a semi-Markov model.

Based on the simple model of Figure 6.11(b), the sequence of events of the variable duration HMM is as follows:

1. An initial state, $q_1 = i$, is chosen according to the initial state distribution $\pi_i$.



2. A duration $d_1$ is chosen according to the state duration density $p_{q_1}(d_1)$. (For expedience and ease of implementation, the duration density $p_q(d)$ is truncated at a maximum duration value $D$.)
3. Observations $\mathbf{o}_1\,\mathbf{o}_2 \ldots \mathbf{o}_{d_1}$ are chosen according to the joint observation density, $b_{q_1}(\mathbf{o}_1\,\mathbf{o}_2 \ldots \mathbf{o}_{d_1})$. Generally we assume that in each state observations are independent so that $b_{q_1}(\mathbf{o}_1\,\mathbf{o}_2 \ldots \mathbf{o}_{d_1}) = \Pi_{t=1}^{d_1} b_{q_1}(\mathbf{o}_t)$.
4. The next state, $q_2 = j$, is chosen according to the state transition probabilities, $a_{q_1 q_2}$, with the constraint that $a_{q_1 q_1} = 0$, i.e., no transition back to the same state can occur. (Clearly this is a requirement, because we assume that, in state $q_1$, exactly $d_1$ observations occur.)

A little thought should convince the reader that the variable duration HMM can be made equivalent to the standard HMM by setting $p_i(d)$ to be the exponential density of (6.64).

Using the above formulation, we must make several changes to the formulas of Section 6.4.3 to allow calculation of $P(\mathbf{O}|\lambda)$ and for reestimation of all model parameters. In particular we assume that the first state begins at $t = 1$ and the last state ends at $t = T$. We then define the forward variable $\alpha_t(i)$ as

$$\alpha_t(i) = P(\mathbf{o}_1\,\mathbf{o}_2 \ldots \mathbf{o}_t, \text{ the stay in state } i \text{ ends at } t|\lambda). \tag{6.65}$$

We assume that a total of $r$ states have been visited during the first $t$ observations, and we denote the states as $q_1, q_2, \ldots, q_r$ with durations associated with each state of $d_1, d_2, \ldots, d_r$. Thus the constraints of Eq. (6.65) are

$$q_r = i \tag{6.66a}$$

$$\sum_{s=1}^{r} d_s = t. \tag{6.66b}$$

Eq. (6.65) can then be written as

$$\begin{aligned} \alpha_t(i) = \sum_q \sum_d \pi_{q_1} \cdot p_{q_1}(d_1) \cdot P(\mathbf{o}_1\,\mathbf{o}_2 \ldots \mathbf{o}_{d_1}|q_1) \\ \cdot a_{q_1 q_2} p_{q_2}(d_2) P(\mathbf{o}_{d_1+1} \ldots \mathbf{o}_{d_1+d_2}|q_2) \ldots \\ \cdot a_{q_{r-1} q_r} p_{q_r}(d_r) P(\mathbf{o}_{d_1+d_2+\cdots+d_{r-1}+1} \ldots \mathbf{o}_t|q_r) \end{aligned} \tag{6.67}$$

where the sum is over all states $q$ and all possible state durations $d$. By induction we can write $\alpha_t(j)$ as

$$\alpha_t(j) = \sum_{i=1}^{N} \sum_{d=1}^{D} \alpha_{t-d}(i) a_{ij} p_j(d) \prod_{s=t-d+1}^{t} b_j(\mathbf{o}_s) \tag{6.68}$$

where $D$ is the maximum duration within any state. To initialize the computation of $\alpha_t(j)$ we use

$$\alpha_1(i) = \pi_i p_i(1) \cdot b_i(\mathbf{o}_1) \tag{6.69a}$$

$$\alpha_2(i) = \pi_i p_i(2) \prod_{s=1}^{2} b_i(\mathbf{o}_s) + \sum_{\substack{j=1 \\ j\neq i}}^{N} \alpha_1(j) a_{ji} p_i(1) b_i(\mathbf{o}_2) \tag{6.69b}$$

$$\alpha_3(i) = \pi_i p_i(3) \prod_{s=1}^{3} b_i(\mathbf{o}_s) + \sum_{d=1}^{2} \sum_{\substack{j=1 \\ j\neq i}}^{N} \alpha_{3-d}(j) a_{ji} p_i(d) \tag{6.69c}$$

$$\cdot \prod_{s=4-d}^{3} b_i(\mathbf{o}_s) \tag{6.69d}$$

and so on, until $\alpha_D(i)$ is computed; then Eq. (6.68) can be used for all $t > D$. It should be clear that the desired probability of $\mathbf{O}$ given the model $\lambda$ can be written in terms of the $\alpha$s as

$$P(\mathbf{O}|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{6.70}$$

as was previously used for ordinary HMMs.

To give reestimation formulas for all the variables of the variable duration HMM, we must define three more forward-backward variables, namely

$$\alpha_t^*(i) = P(\mathbf{o}_1 \mathbf{o}_2 \ldots \mathbf{o}_t, \text{ stay in state } i \text{ starts at } t+1|\lambda) \tag{6.71}$$

$$\beta_t(i) = P(\mathbf{o}_{t+1} \ldots \mathbf{o}_T | \text{ stay in state } i \text{ ends at } t, \lambda) \tag{6.72}$$

$$\beta_t^*(i) = P(\mathbf{o}_{t+1} \ldots \mathbf{o}_T | \text{ stay in state } i \text{ starts at } t+1, \lambda). \tag{6.73}$$

The relationships between $\alpha, \alpha^*, \beta$, and $\beta^*$ are as follows:

$$\alpha_t^*(j) = \sum_{i=1}^{N} \alpha_t(i) a_{ij} \tag{6.74}$$

$$\alpha_t(i) = \sum_{d=1}^{D} \alpha_{t-d}^*(i) p_i(d) \prod_{s=t-d+1}^{t} b_i(\mathbf{o}_s) \tag{6.75}$$

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} \beta_t^*(j) \tag{6.76}$$

$$\beta_t^*(i) = \sum_{d=1}^{D} \beta_{t+d}(i) p_i(d) \prod_{s=t+1}^{t+d} b_i(\mathbf{o}_s). \tag{6.77}$$

Based on the above relationships and definitions, the reestimation formulas for the variable duration HMM, with discrete observations, are

$$\bar{\pi}_i = \frac{\pi_i \beta_0^*(i)}{P(\mathbf{O}|\lambda)} \tag{6.78}$$



$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T} \alpha_t(i) a_{ij} \beta_t^*(j)}{\sum_{j=1}^{N} \sum_{t=1}^{T} \alpha_t(i) a_{ij} \beta_t^*(j)} \tag{6.79}$$

$$\bar{b}_i(k) = \frac{\sum_{\substack{t=1 \\ \text{s.t. } \mathbf{o}_t = \mathbf{v}_k}}^{T} \left[ \sum_{\tau<t} \alpha_\tau^*(i) \cdot \beta_\tau^*(i) - \sum_{\tau<t} \alpha_\tau(i) \beta_\tau(i) \right]}{\sum_{k=1}^{M} \sum_{\substack{t=1 \\ \text{s.t. } \mathbf{o}_t = \mathbf{v}_k}}^{T} \left[ \sum_{\tau<t} \alpha_\tau^*(i) \cdot \beta_\tau^*(i) - \sum_{\tau<t} \alpha_\tau(i) \beta_\tau(i) \right]} \tag{6.80}$$

$$\bar{p}_i(d) = \frac{\sum_{t=1}^{T} \alpha_t^*(i) p_i(d) \beta_{t+d}(i) \prod_{s=t+1}^{t+d} b_i(\mathbf{o}_s)}{\sum_{d=1}^{D} \sum_{t=1}^{T} \alpha_t^*(i) p_i(d) \beta_{t+d}(i) \prod_{s=t+1}^{t+d} b_i(\mathbf{o}_s)}. \tag{6.81}$$

The interpretation of the reestimation formulas is the following: The formula for $\bar{\pi}_i$ is the probability that state $i$ was the first state, given $\mathbf{O}$. The formula for $\bar{a}_{ij}$ is almost the same as for the usual HMM, except it uses the condition that the alpha terms in which a state ends at $t$, join with the beta terms in which a new state begins at $t+1$. The formula for $\bar{b}_i(k)$ is the expected number of times that observation $\mathbf{o}_t = \mathbf{v}_k$ occurred in state $i$, normalized by the expected number of times that any observation occurred in state $i$. Finally, the reestimation formula for $\bar{p}_i(d)$ is the ratio of the expected number of times state $i$ occurred with any duration.

The importance of incorporating state duration densities is reflected in the observation that, for some problems, the quality of the modeling is significantly improved when explicit state duration densities are used. However, there are drawbacks to the use of the variable duration model discussed in this section. One is the greatly increased computational load associated with using variable durations. It can be seen from the definition and initialization conditions on the forward variable $\alpha_i(i)$, from Eqs. (6.68)–(6.69), that about $D$ times the storage and $D^2/2$ times the computation is required. For $D$ on the order of 25 (as is reasonable for many speech-processing problems), computation is increased by a factor of 300. Another problem with the variable duration models is the large number of parameters ($D$), associated with each state, that must be estimated, in addition to the usual HMM parameters. Furthermore, for a fixed number of observations $T$, in the training set, there are, on average, fewer state transitions and much less data to estimate $p_i(d)$ than would be used in a standard HMM. Thus the reestimation problem is more difficult for variable duration HMMs than for the standard HMM.

One proposal to alleviate some of these problems is to use a parametric state duration density instead of the nonparametric $p_i(d)$ used above [23–24]. In particular, proposals

include the Gaussian family with

$$p_i(d) = \mathcal{N}(d, \mu_i, \sigma_i^2) \tag{6.82}$$

with parameters $\mu_i$ and $\sigma_i^2$, or the Gamma family with

$$p_i(d) = \frac{\eta_i^{\nu_i} d^{\nu_i - 1} e^{-\eta_i d}}{\Gamma(\nu_i)} \tag{6.83}$$

with parameters $\nu_i$ and $\eta_i$ and with mean $\nu_i \eta_i^{-1}$ and variance $\nu_i \eta_i^{-2}$. Reestimation formulas for $\eta_i$ and $\nu_i$ have been derived and used with good results. Another possibility, which has been used with good success, is to assume a uniform duration distribution over an appropriate range of durations and use a path-constrained Viterbi decoding procedure.

## 6.10 OPTIMIZATION CRITERION—ML, MMI, AND MDI

The standard ML design criterion is to use a training sequence of observations $\mathbf{O}$ to derive the set of model parameters $\lambda$, yielding

$$\lambda_{\text{ML}} = \arg\max_{\lambda} P(\mathbf{O}|\lambda). \tag{6.84}$$

Any of the reestimation algorithms discussed previously provides a solution to this optimization problem.

The need to consider alternative design criteria, however, comes from several concerns ([26–28]). The basic philosophy in statistical modeling methods, such as HMM, is that the signal or observation sequence can be well modeled if the parameters of the model are carefully and correctly chosen. The problem with this philosophy is that the assumed model—HMM in the present case—is sometimes inadequate to model the observed signal so that no matter how carefully the parameters are chosen, the modeling accuracy is limited. Often, this situation is described as a "model mismatch." The first alternative optimization criterion we discuss here is one that tries to overcome the problem of model mismatch in order to achieve a more accurate modeling of the observation signal.

The observed signal $\mathbf{O} = (\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_T)$ is associated with a sequence of constraints $\mathcal{R} = (\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_T)$. For example, $\mathbf{R}_t$ may be the autocorrelation matrix that characterizes the observation $\mathbf{o}_t$. Then, obviously, $\mathbf{O}$ is only one of possibly uncountably many observation sequences that satisfy the constraint sequence $\mathcal{R}$. Furthermore, in terms of the probability distributions of the observation sequences, there exists a set of such distributions that would also satisfy $\mathcal{R}$. We denote this set $\Omega(\mathcal{R})$. The minimum discrimination information (MDI) is a measure of closeness between two probability measures (one of which bears the HMM form here) under the given constraint $\mathcal{R}$ and is defined by

$$\nu(\mathcal{R}, P_\lambda) \triangleq \inf_{Q \in \Omega(\mathcal{R})} I(Q : P_\lambda) \tag{6.85}$$

where

$$I(Q : P_\lambda) = \int q(\mathbf{O}) \log \frac{q(\mathbf{O})}{p(\mathbf{O}|\lambda)} d\mathbf{O} \tag{6.86}$$



is the discrimination information between distributions $Q$ and $P_\lambda$ [27,28]. (The functions $q(\cdot)$ and $p(\cdot|\lambda)$ are the probability density functions corresponding to $Q$ and $P_\lambda$ respectively.) The discrimination information is calculated based on the given training set of observations.

The MDI criterion tries to choose a model parameter set $\lambda$ such that $\nu(\mathcal{R}, P_\lambda)$ is minimized. An interpretation of MDI is that the model parameter set $\lambda$ is chosen so that the model $p(\mathbf{O}|\lambda)$ is as close as it can be to a member of the set $\Omega(\mathcal{R})$. Since the closeness is always measured in terms of the discrimination information evaluated on the given observation, the intrinsic characteristics of the training sequences would then have substantial influence on the parameter selection. By emphasizing the measure discrimination, the model estimation is no longer solely dictated by the assumed model form. The MDI optimization problem is, however, not as straightforward as the ML optimization problem and no simple robust implementation of the procedure is known.

Another concern about the HMM optimization criterion arises when we attempt to use it to solve a class of speech-recognition problems. Consider recognition of a vocabulary of $V$ words, each of which is represented by an HMM, with parameter set $\lambda_v$, $v = 1, 2, \ldots, V$. We assume $P(v)$ to be the a priori probability for word $v$, $v = 1, 2, \ldots, V$. The set of HMMs $\Lambda = \{\lambda_v\}$ together with the a priori probabilities thus defines a probability measure for an arbitrary observation sequence $\mathbf{O}$

$$P_\Lambda(\mathbf{O}) = \sum_{v=1}^{V} P(\mathbf{O}|\lambda_v, v) P(v). \tag{6.87}$$

(The notation $P(\mathbf{O}|\lambda_v, v)$ indicates that it is a probability conditioned on the word $v$. We include the model parameter $\lambda_v$, sometimes, because of the necessity of treating $\lambda_v$ as random variables for estimation purposes. Obviously, when $\lambda_v$ is fixed, $P(\mathbf{O}|\lambda_v, v)$ is the conditional probability, parameterized by $\lambda_v$.) To train these models (i.e., to estimate the optimum parameters of the associated models), utterances of known (labeled) words are used. We denote the labeled training sequence by $\mathbf{O}^v$ where superscript $v$ reflects the fact that $\mathbf{O}^v$ is a rendition of word $v$. The standard ML criterion of Eq. (6.84) is to use $\mathbf{O}^v$ to estimate model parameters $\lambda_v$, yielding

$$(\lambda_v)_{\text{ML}} = \arg\min_{\lambda} P(\mathbf{O}^v|\lambda).$$

Each model is estimated *separately* using the correspondingly labeled training observation sequence(s). The resultant models, however, need not be the optimal solution for minimizing the probability of recognition error.

An alternative design criterion that aims at maximizing the "discrimination" of each model (i.e., the ability to distinguish between observation sequences generated by the correct word model and those generated by alternative models) is the maximum mutual information (MMI) criterion [26]. The mutual information between an observation sequence $\mathbf{O}^v$ and the word $v$, parameterized by $\Lambda = \{\lambda_v\}$, $v = 1, 2, \ldots, V$, is

$$I_\Lambda(\mathbf{O}^v, v) = \log \frac{P(\mathbf{O}^v, v|\Lambda)}{P_\Lambda(\mathbf{O}^v) P(v)}. \tag{6.88}$$

Since

$$P(\mathbf{O}^v, v|\Lambda)/P(v) = P(\mathbf{O}^v|\lambda_v),$$

$$I_\Lambda(\mathbf{O}^v, v) = \log P(\mathbf{O}^v|\lambda_v) - \log \sum_{w=1}^{V} P(\mathbf{O}^v|\lambda_w, w)P(w). \tag{6.89}$$

The MMI criterion is to find the entire model set $\Lambda$ such that the mutual information is maximized,

$$(\Lambda)_{\text{MMI}} = \max_{\Lambda}\left\{\sum_{v=1}^{V} I_\Lambda(\mathbf{O}^v, v)\right\}. \tag{6.90}$$

The MMI criterion is obviously different from the ML criterion. Both are minimum cross-entropy approaches. In the ML approach, an HMM for the distribution of the *data given the word* is matched to the empirical distribution. In the MMI approach, a model for the distribution of the *word given the data* is matched to the empirical distribution. This explains the merit of the MMI approach. The optimization procedure for the MMI approach involves the entire model parameter set $\Lambda$ even if only one labeled training sequence $\mathbf{O}^v$ is used. The ML criterion addresses the likelihood $P(\mathbf{O}^v|\lambda_v)$ alone, while the MMI criterion compares the likelihood $P(\mathbf{O}^v|\lambda_v)$ against the "probability background" $P_\Lambda(\mathbf{O}^v)$ and attempts to maximize the difference. However, $(\Lambda)_{\text{MMI}}$ is not as straightforward to obtain as $(\Lambda)_{\text{ML}}$. One often has to use general optimization procedures like the descent algorithms to solve Eq. (6.90). Such optimization procedures often lead to numerical problems in implementation.

## 6.11 COMPARISONS OF HMMS

An interesting question associated with HMMs is the following: Given two HMMs, $\lambda_1$ and $\lambda_2$, what is a reasonable measure of the similarity of the two models ([29])? Consider the case of two models

$$\lambda_1 = (A_1, B_1, \pi_1) \qquad \lambda_2 = (A_2, B_2, \pi_2)$$

with

$$A_1 = \begin{bmatrix} p & 1-p \\ 1-p & p \end{bmatrix} \quad B_1 = \begin{bmatrix} q & 1-q \\ 1-q & q \end{bmatrix} \quad \pi_1 = [1/2 \;\; 1/2]$$

and

$$A_2 = \begin{bmatrix} r & 1-r \\ 1-r & r \end{bmatrix} \quad B_2 = \begin{bmatrix} s & 1-s \\ 1-s & s \end{bmatrix} \quad \pi_2 = [1/2 \;\; 1/2].$$

For $\lambda_1$ to be equivalent to $\lambda_2$, in the sense of having the same statistical properties for the observation symbols, i.e., $E[\mathbf{o}_t = \mathbf{v}_k|\lambda_1] = E[\mathbf{o}_t = \mathbf{v}_k|\lambda_2]$, for all $\mathbf{v}_k$, we require

$$pq + (1-p)(1-q) = rs + (1-r)(1-s)$$



or, by solving for $s$, we get

$$s = \frac{p + q - 2pq}{1 - 2r}.$$

By choosing (arbitrarily) $p = 0.6, q = 0.7, r = 0.2$, we get $s = 13/30 \approx 0.433$. Thus, even when the two models, $\lambda_1$ and $\lambda_2$, look ostensibly very different (i.e., $A_1$ is very different from $A_2$ and $B_1$ is very different from $B_2$), statistical equivalence of the models can occur.

We can generalize [29] the concept of model distance (dissimilarity) by defining a distance measure $D(\lambda_1, \lambda_2)$, between two Markov models, $\lambda_1$ and $\lambda_2$, as

$$D(\lambda_1, \lambda_2) = \frac{1}{T}\left[\log P(\mathbf{O}^{(2)}|\lambda_1) - \log P(\mathbf{O}^{(2)}|\lambda_2)\right] \tag{6.91}$$

where $\mathbf{O}^{(2)} = (\mathbf{o}_1\, \mathbf{o}_2\, \mathbf{o}_3 \ldots \mathbf{o}_T)$ is a sequence of observations generated by model $\lambda_2$. Basically, Eq. (6.91) is a measure of how well model $\lambda_1$ matches observations generated by model $\lambda_2$, relative to how well model $\lambda_2$ matches observations generated by itself. Several interpretations of Eq. (6.91) exist in terms of cross-entropy, or divergence, or discrimination information [29].

One of the problems with the distance measure of Eq. (6.91) is that it is nonsymmetric. Hence a natural expression of this measure is the symmetrized version, namely

$$D_s(\lambda_1, \lambda_2) = \frac{D(\lambda_1\lambda_2) + D(\lambda_2, \lambda_1)}{2}. \tag{6.92}$$

## 6.12 IMPLEMENTATION ISSUES FOR HMMS

The discussion in the previous sections has dealt primarily with the theory of HMMs and several variations on the form of the model. In this section we deal with several practical implementation issues, including scaling, multiple observation sequences, initial parameter estimates, missing data, and choice of model size and type. For some of these implementation issues we can prescribe exact analytical solutions; for other issues we can provide only some seat-of-the-pants experience gained from working with HMMs.

### 6.12.1 Scaling

$\alpha_t(i) = P(O_1 O_2 \cdots O_t, q_t = i | \lambda)$

the probability of the partial observation (until time t) and state i at time t, given model λ

hy do
ling

To understand why scaling ([18,23]) is required for implementing the reestimation procedure of HMMs, consider the definition of $\alpha_t(i)$ of Eq. (6.18). It can be seen that $\alpha_t(i)$ consists of the sum of a large number of terms, each of the form

$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(O_{t+1})$

$$\sum\left(\prod_{s=1}^{t-1} a_{q_s q_{s+1}} \prod_{s=1}^{t} b_{q_s}(\mathbf{o}_s)\right)$$

with $q_t = i$ and $b$ is a discrete probability as defined by Eq. (6.8). Since each $a$ and $b$ term is less than 1 (generally significantly less than 1), it can be seen that as $t$ starts to get big (e.g., 10 or more), each term of $\alpha_t(i)$ starts to head exponentially to zero. For sufficiently large $t$ (e.g., 100 or more) the dynamic range of the $\alpha_t(i)$ computation will exceed the precision

range of essentially any machine (even in double precision). Hence the only reasonable way to perform the computation is to incorporate a scaling procedure.

The basic scaling procedure multiplies $\alpha_t(i)$ by a scaling coefficient that is independent of $i$ (i.e., it depends only on $t$), with the goal of keeping the scaled $\alpha_t(i)$ within the dynamic range of the computer for $1 \le t \le T$. A similar scaling is done to the $\beta_t(i)$ coefficients (since these also tend to zero exponentially fast) and then, at the end of the computation, the scaling coefficients are canceled out exactly.

To understand this scaling procedure better, consider the reestimation formula for the state-transition coefficients $a_{ij}$. If we write the reestimation formula (Eq. (6.40b)) directly in terms of the forward and backward variables, we get

$$\frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1}\sum_{j=1}^{N} \xi_t(i,j)} = \bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \alpha_t(i)a_{ij}b_j(\mathbf{o}_{t+1})\beta_{t+1}(j)}{\sum_{t=1}^{T}\sum_{j=1}^{N} \alpha_t(i)a_{ij}b_j(\mathbf{o}_{t+1})\beta_{t+1}(j)}. \tag{6.93}$$

Consider the computation of $\alpha_t(i)$. We use the notation $\alpha_t(i)$ to denote the unscaled $\alpha$s, $\hat{\alpha}_t(i)$ to denote the scaled (and iterated) $\alpha$s, and $\hat{\hat{\alpha}}_t(i)$ to denote the local version of $\alpha$ before scaling. Initially, for $t = 1$, we compute $\alpha_1(i)$ according to Eq. (6.19) and set $\hat{\hat{\alpha}}_1(i) = \alpha_1(i)$, with $c_1 = \frac{1}{\sum_{i=1}^{N} \alpha_1(i)}$ and $\hat{\alpha}_1(i) = c_1\alpha_1(i)$. For each $t$, $2 \le t \le T$, we first compute $\hat{\hat{\alpha}}_t(i)$ according to the induction formula (Eq. (6.20)), in terms of the previously scaled $\hat{\alpha}_t(i)$; that is,

$$\hat{\hat{\alpha}}_t(i) = \sum_{j=1}^{N} \hat{\alpha}_{t-1}(j)a_{ji}b_i(\mathbf{o}_t). \tag{6.94a}$$

We determine the scaling coefficient $c_t$ as

$$c_t = \frac{1}{\sum_{i=1}^{N} \hat{\hat{\alpha}}_t(i)} \tag{6.94b}$$

giving

$$\hat{\alpha}_t(i) = c_t\hat{\hat{\alpha}}_t(i) \tag{6.94c}$$

From Eq. (6.94a–c) we can write the scaled $\hat{\alpha}_t(i)$ as $c_t\hat{\hat{\alpha}}_t(i)$ or

$$\hat{\alpha}_t(i) = \frac{\sum_{j=1}^{N} \hat{\alpha}_{t-1}(j)a_{ji}b_i(\mathbf{o}_t)}{\sum_{i=1}^{N}\sum_{j=1}^{N} \hat{\alpha}_{t-1}(j)a_{ji}b_i(\mathbf{o}_t)}. \tag{6.95}$$



By induction we can write $\hat{\alpha}_{t-1}(j)$ as

$$\hat{\alpha}_{t-1}(j) = \left(\prod_{\tau=1}^{t-1} c_\tau\right)\alpha_{t-1}(j). \tag{6.96a}$$

Thus we can write $\hat{\alpha}_t(i)$ as

$$\hat{\alpha}_t(i) = \frac{\sum_{j=1}^{N} \alpha_{t-1}(j)\left(\prod_{\tau=1}^{t-1} c_\tau\right)a_{ji}b_i(\mathbf{o}_t)}{\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_{t-1}(j)\left(\prod_{\tau=1}^{t-1} c_\tau\right)a_{ji}b_i(\mathbf{o}_t)} = \frac{\alpha_t(i)}{\sum_{i=1}^{N} \alpha_t(i)} \tag{6.96b}$$

that is, each $\alpha_t(i)$ is effectively scaled by the sum over all states of $\alpha_t(i)$.

Next we compute the $\beta_t(i)$ terms from the backward recursion. The only difference here is that we use the *same* scale factors for each time $t$ for the betas as was used for the alphas. Hence the scaled $\beta$s are of the form

$$\hat{\beta}_t(i) = c_t\beta_t(i). \tag{6.97}$$

Because each scale factor effectively restores the magnitude of the $\alpha$ terms to 1, and because the magnitudes of the $\alpha$ and $\beta$ terms are comparable, using the same scaling factors on the $\beta$s as was used on the $\alpha$s is an effective way to keep the computation within reasonable bounds. Furthermore, in terms of the scaled variables, we see that the reestimation Eq. (6.93) becomes

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \hat{\alpha}_t(i)a_{ij}b_j(\mathbf{o}_{t+1})\hat{\beta}_{t+1}(j)}{\sum_{t=1}^{T-1}\sum_{j=1}^{N} \hat{\alpha}_t(i)a_{ij}b_j(\mathbf{o}_{t+1})\hat{\beta}_{t+1}(j)} \tag{6.98}$$

but each $\hat{\alpha}_t(i)$ can be written as

$$\hat{\alpha}_t(i) = \left[\prod_{s=1}^{t} c_s\right]\alpha_t(i) = C_t\alpha_t(i) \tag{6.99}$$

and each $\hat{\beta}_{t+1}(j)$ can be written as

$$\hat{\beta}_{t+1}(j) = \left[\prod_{s=t+1}^{T} c_s\right]\beta_{t+1}(j) = D_{t+1}\beta_{t+1}(j). \tag{6.100}$$

Thus Eq. (6.98) can be written as

$$\overline{a_{ij}} = \frac{\sum_{t=1}^{T-1} C_t \alpha_t(i) a_{ij} b_j(\mathbf{o}_{t+1}) D_{t+1} \beta_{t+1}(j)}{\sum_{t=1}^{T-1} \sum_{j=1}^{N} C_t \alpha_t(i) a_{ij} b_j(\mathbf{o}_{t+1}) D_{t+1}\, \beta_{t+1}(j)}. \tag{6.101}$$

Finally the term $C_t D_{t+1}$ can be seen to be of the form

$$C_t D_{t+1} = \prod_{s=1}^{t} c_s \prod_{s=t+1}^{T} c_s = \prod_{s=1}^{T} c_s = C_T \tag{6.102}$$

independent of $t$. Hence the terms $C_t D_{t+1}$ cancel out of both the numerator and denominator of Eq. (6.101) and the exact reestimation equation is therefore realized.

It should be obvious that the above scaling procedure applies equally well to reestimation of the $\pi$ or $B$ coefficients. It should also be obvious that the scaling procedure of Eq. (6.95) need not be applied at every time instant $t$, but can be performed whenever desired, or whenever necessary (e.g., to prevent underflow). If scaling is not performed at some instant $t$, the scaling coefficients $c_t$ are set to 1 at that time, and all the conditions discussed above are then met.

$(O|\lambda)$

The only real change to the HMM procedure because of scaling is the procedure for computing $P(\mathbf{O}|\lambda)$. We cannot merely sum up the $\hat{\alpha}_T(i)$ terms, because these are scaled already. However, we can use the property that

$$\prod_{t=1}^{T} c_t \sum_{i=1}^{N} \alpha_T(i) = C_T \sum_{i=1}^{N} \alpha_T(i) = 1. \tag{6.103}$$

Thus we have

$$\prod_{t=1}^{T} c_t \cdot P(\mathbf{O}|\lambda) = 1 \tag{6.104}$$

or

$$P(\mathbf{O}|\lambda) = \frac{1}{\prod_{t=1}^{T} c_t} \tag{6.105}$$

or

$$\log\,[P(\mathbf{O}|\lambda)] = \sum_{t=1}^{T} \log\, c_t. \tag{6.106}$$

Thus the log of $P$ can be computed, but not $P$, since it would be out of the dynamic range of the machine anyway.

Finally we note that when using the Viterbi algorithm to give the maximum likelihood state sequence, no scaling is required if we use logarithms as discussed in the alternate Viterbi implementation.



### 6.12.2 Multiple Observation Sequences

In Section 6.5 we discussed a form of HMM called the left-right or Bakis model, in which the state proceeds from state 1 at $t = 1$ to state $N$ at $t = T$ in a sequential manner (recall the model of Figure 6.8(b)). We have already discussed how a left-right model imposes constraints on the state-transition matrix, and the initial state probabilities Eqs. (6.45)–(6.48). However, the major problem with left-right models is that one cannot use a single observation sequence to train the model (i.e., for reestimation of model parameters). This is because the transient nature of the states within the model allows only a small number of observations for any state (until a transition is made to a successor state). Hence, to have sufficient data to make reliable estimates of all model parameters, one has to use multiple observation sequences ([18]).

The modification of the reestimation procedure is straightforward and is as follows. We denote the set of $K$ observation sequences as

$$\mathbf{O} = [\mathbf{O}^{(1)}, \mathbf{O}^{(2)}, \ldots, \mathbf{O}^{(K)}] \tag{6.107}$$

where $\mathbf{O}^{(k)} = (\mathbf{o}_1^{(k)} \mathbf{o}_2^{(k)} \ldots \mathbf{o}_{T_k}^{(k)})$ is the $k$th observation sequence. We assume each observation sequence is independent of every other observation sequence, and our goal is to adjust the parameters of the model $\lambda$ to maximize

$$P(\mathbf{O}|\lambda) = \prod_{k=1}^{K} P(\mathbf{O}^{(k)}|\lambda) \tag{6.108}$$

$P_k = P(O^{(k)}|\lambda)$

$$= \prod_{k=1}^{K} P_k. \tag{6.109}$$

Since the reestimation formulas are based on frequencies of occurrence of various events, the reestimation formulas for multiple observation sequences are modified by adding together the individual frequencies of occurrence for each sequence. Thus the modified reestimation formulas for $\overline{a}_{ij}$ and $\overline{b_j}(\ell)$ are 把每个sequence的别的事件频率相加

$$\overline{a}_{ij} = \frac{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k-1} \alpha_t^k(i) a_{ij} b_j(\mathbf{o}_{t+1}^{(k)}) \beta_{t+1}^k(j)}{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k-1} \alpha_t^k(i) \beta_t^k(i)} \tag{6.110}$$

and

$$\overline{b_j}(\ell) = \frac{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{\substack{t=1 \\ \text{s.t. } \mathbf{o}_t = v_\ell}}^{T_k-1} \alpha_t^k(i) \beta_t^k(i)}{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k-1} \alpha_t^k(i) \beta_t^k(i)} \tag{6.111}$$

and $\pi_i$ is not reestimated since $\pi_1 = 1$, $\pi_i = 0$, $i \neq 1$.

The proper scaling of Eqs. (6.110)–(6.111) is now straightforward since each observation sequence has its own scaling factor. The key idea is to remove the scaling factor from each term before summing. This can be accomplished by writing the reestimation equations in terms of the scaled variables, i.e.,

$$\bar{a}_{ij} = \frac{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k-1} \hat{\alpha}_t^k(i) a_{ij} b_j(\mathbf{o}_{t+1}^k) \hat{\beta}_{t+1}^k(j)}{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k-1} \hat{\alpha}_t^k(i) \hat{\beta}_t^k(i)}. \tag{6.112}$$

In this manner, for each sequence $\mathbf{O}^{(k)}$, the same scale factors will appear in each term of the sum over $t$ as appears in the $P_k$ term, and hence will cancel exactly. Thus using the scaled values of the $\alpha$s and $\beta$s results in an unscaled $\bar{a}_{ij}$. A similar result is obtained for the $\bar{b}_j(\ell)$ term.

### 6.12.3 Initial Estimates of HMM Parameters

In theory, the reestimation equations should give values of the HMM parameters that correspond to a local maximum of the likelihood function. A key question is, therefore, How do we choose initial estimates of the HMM parameters so that the local maximum is equal to or as close as possible to the global maximum of the likelihood function?

Basically there is no simple or straightforward answer. Instead, experience has shown that either random (subject to the stochastic and the nonzero value constraints) or uniform initial estimates of the $\pi$ and $A$ parameters are adequate for giving useful reestimates of these parameters in almost all cases. However, for the $B$ parameters, experience has shown that good initial estimates are helpful in the discrete symbol case and are essential (when dealing with multiple mixtures) in the continuous-distribution case. Such initial estimates can be obtained in a number of ways; these include (a) manual segmentation of the observation sequence(s) into states and averaging of observations within states, (b) maximum likelihood segmentation of observations and averaging, and (c) segmental $k$-means segmentation with clustering, etc. We discuss such segmentation techniques later in this chapter.

### 6.12.4 Effects of Insufficient Training Data

Another problem associated with training HMM parameters via reestimation methods is that the observation sequence used for training is, of necessity, finite ([30]). Thus there is always an inadequate number of occurrences of low-probability events (e.g., symbol occurrences within states) to give good estimates of the model parameters. By way of example, consider the case of a discrete observation HMM. Recall that the reestimation transformation of $\bar{b}_j(k)$, Eq. (6.40c), requires a count of the expected number of times in state $j$ and observing symbol $\mathbf{v}_k$ simultaneously. If the training sequence is so small that it does not have any occurrences of this event (i.e., $q_t = j$ and $\mathbf{o}_t = \mathbf{v}_k$), $b_j(k) = 0$ and



will stay 0 after reestimation. The resultant model would produce a zero probability result for any observation sequence that actually includes ($\mathbf{o}_t = \mathbf{v}_k$ and $q_t = j$). Such a singular outcome is obviously a consequence of the unreliable estimate that $b_j(k) = 0$ due to the insufficiency of the training set.

One solution to this problem is to increase the size of the training observation set. Often this is impractical. A second possible solution is to reduce the size of the model (e.g., number of states, number of symbols per state). Although this is always possible, often there are physical reasons why a given model is used, and therefore the model size cannot be changed. A third possible solution is to seek unconventional statistical estimation algorithms that can somehow enhance the reliability of the parameter estimates even based on limited training data. Deleted interpolation and parameter thresholding are two such alternatives. Since deleted interpolation is considered more an enhanced parameter estimation method, we discuss that subject in the next section.

The simplest way to handle the effects of insufficient training data is to add extra threshold constraints to the model parameters to ensure that no model parameter estimate falls below a specified level [18]. Thus, for example, we might specify the numeric floor, for a discrete symbol model, that

$$b_j(k) = \begin{cases} b_j(k), & \text{if } b_j(k) \geq \delta_b \\ \delta_b, & \text{otherwise} \end{cases} \tag{6.113a}$$

or, for a continuous distribution model, that

$$U_{jk}(r, r) = \begin{cases} U_{jk}(r,r), & \text{if } U_{jk}(r,r) \geq \delta_u \\ \delta_u, & \text{otherwise} \end{cases}. \tag{6.113b}$$

When the numeric floor is invoked in the reestimation equations, all remaining parameters need to be rescaled so that the densities obey the required stochastic constraints. Such postprocessor techniques are thus considered implementational measures to combat the insufficient data problem and have been applied with good success to several problems in speech processing. The method of parameter thresholding has a justification from a Bayes statistics point of view. It can be shown that Eq. (6.113b) is, in fact, a maximum a posteriori (MAP) estimate of the variance under the assumption that the parameter prior $P(U_{jk}(r,r))$ is an informative one with uniform distribution and $(U_{jk}(r,r))_{\min} = \delta_u$ [31]. (See Section 6.13.)

### 6.12.5 Choice of Model

The remaining issues in implementing HMMs are the choice of type of model (ergodic or left-right or some other form), choice of model size (number of states), and choice of observation symbols (discrete or continuous, single or multimixture, choice of observation parameters). Unfortunately, there is no simple, theoretically correct way of making such choices. These choices must be made depending on the signal being modeled. With these comments, we end our discussion of the theoretical aspects of hidden Markov models and proceed to a discussion of how such models have been applied to the isolated word-recognition problem.

## 6.13 IMPROVING THE EFFECTIVENESS OF MODEL ESTIMATES

We discuss three methods that have been shown to be able to enhance the effectiveness of HMM model estimates for speech recognition. These are (1) deleted interpolation, (2) Bayesian adaptation, and (3) corrective training. The first two methods are motivated by the problem of insufficient data, while the last method has the unique objective of trying to reduce recognition errors directly.

### 6.13.1 Deleted Interpolation

When training data are insufficient, reliable and robust determination of HMM parameters cannot be accomplished. The HMM obtained by the Baum-Welch reestimation method, based on the maximum likelihood criterion, may be adequate in characterizing the training data, but for new data the match may be quite poor. One parameter estimation method that aims to improve the model reliability is the method of "deleted interpolation."

The concept involves combining two (or more) separately trained models, one of which is more reliably trained than the other. A scenario in which this can happen is the case when we use tied states which forces "different" states to share an identical statistical characterization, effectively reducing the number of parameters in the model. A model with tied states is often more robust than a model without tied states when trained on the same amount of data. But a model with tied states is also less precise than a model without tied states if the training data are sufficient. Therefore, the idea of combining the two models is to allow us to fall back to the more reliable model when the supposedly more precise model is, in fact, unreliable. A similar scenario occurs when context-independent (more robust) and context-dependent (more precise) phone models are used in large vocabulary recognition (see Chapter 8).

Let the two models be defined by the parameter sets $\lambda = (A, B, \pi)$ and $\lambda' = (A', B', \pi')$, respectively. The interpolated model, $\tilde{\lambda} = (\tilde{A}, \tilde{B}, \tilde{\pi})$, is obtained as

$$\tilde{\lambda} = \epsilon\lambda + (1 - \epsilon)\lambda' \tag{6.114}$$

where $\epsilon$ represents the weighting of the parameters of the "full" model (with more detailed characterization of the observations) and $(1 - \epsilon)$ represents the weighting of the parameters of the reduced, but more reliable, model. A key issue is the determination of the optimal value of $\epsilon$, which is a strong function of the amount of training data. This is easy to see because as the amount of training data gets large, $\lambda$ becomes more reliable and we expect $\epsilon$ to tend to 1.0. Similarly, for small amounts of training data, $\lambda$ is unreliable and we expect $\epsilon$ to tend to 0.0 so as to fall back to the more reliable model $\lambda'$.

The solution to the determination of an optimal value for $\epsilon$ was provided by Jelinek and Mercer [30], who showed how the optimal $\epsilon$ could be estimated using the forward-backward algorithm by interpreting Eq. (6.114) as an expanded HMM of the type shown in Figure 6.12. Figure 6.12a shows the part of the state-transition structure related to the state $S$. Using Figure 6.12b, we can interpret the interpolated model of Eq. (6.114) as an



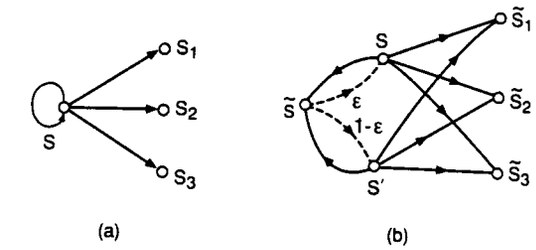



**Figure 6.12** Example of how the process of deleted interpolation can be represented using a state diagram.

expanded HMM in which each state is replaced by three states. The null transitions from the expanded state $\tilde{S}$ to $S$ and $S'$ have transition probabilities $\epsilon$ and $1 - \epsilon$, respectively. The transitions out of $S$ are characterized by those of $\lambda$ while those out of $S'$ are associated with those of $\lambda'$.

The expanded HMM interpretation suggests that the parameter $\epsilon$ can be optimally determined by the usual forward-backward algorithm. However, since the interpolation is designed to better predict unseen (future) data, rather than to account for the training data, determination of $\epsilon$ must be based on data that was not used in obtaining either of the two models, $\lambda$ and $\lambda'$. A key idea of deleted interpolation is thus to partition the training data $\mathcal{T}$ into two disjoint sets; that is, $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$. For example, one might consider a partition of the training set such that $\mathcal{T}_1$ is 90 percent of $\mathcal{T}$ and $\mathcal{T}_2$ is the remaining 10 percent of $\mathcal{T}$. Training set $\mathcal{T}_1$ is first used to train $\lambda$ and $\lambda'$. Training set $\mathcal{T}_2$ is then used to give an estimate of $\epsilon$, assuming $\lambda$ and $\lambda'$ are fixed. There are obviously a large number of ways in which such a partitioning can be accomplished, but one particularly simple one is to cycle $\mathcal{T}_2$ through the data. That is, the first partition uses the last 10 percent of the data as $\mathcal{T}_2$, the second partition uses the next-to-last 10 percent of the data as $\mathcal{T}_2$, etc. An interpretation of deleted interpolation is straightforward. If the unseen data fits the more elaborate model $\lambda$ well (thus validating the reliability of $\lambda$), the forward-backward algorithm would give a value of $\epsilon$ which is close to 1. Otherwise, the forward-backward algorithm gives a small value of $\epsilon$, indicating that the more reliable model $\lambda'$ is a better characterization of the new data than $\lambda$.

The technique of deleted interpolation has been successfully applied to a number of problems in speech recognition, including the estimation of trigram word probabilities for language models [13], and the estimation of HMM output probabilities for trigram phone models (to be discussed in Chapter 8 of this book).

### 6.13.2 Bayesian Adaptation

Another insufficient data situation occurs when we attempt to estimate a speaker-dependent model based on a limited amount of speaker-specific training data. An approach to this

problem is through speaker adaptation, in which a speaker-independent model, obtained by reliable training, is adapted to the particular talker using speaker-specific training data [31].

Speaker adaptation can be accomplished based on a Bayesian framework. Consider the HMM probability measure $P(\mathbf{O}|\lambda)$. If the HMM parameter $\lambda$ is assumed to be fixed but unknown, the maximum likelihood (ML) estimate for $\lambda$, given the training sequence $\mathbf{O}$, is obtained by solving the likelihood equation, i.e.

$$\frac{\partial}{\partial\lambda}P(\mathbf{O}|\lambda) = 0. \tag{6.115}$$

(Normally, the Baum-Welch reestimation algorithm is used to obtain certain stationary point solutions instead of directly solving for Eq. (6.115).) If $\lambda$ is assumed random with a priori distribution $P_0(\lambda)$, then the maximum a posteriori (MAP) estimate for $\lambda$ is obtained by solving

$$\frac{\partial}{\partial\lambda}P(\lambda|\mathbf{O}) = 0 \tag{6.116}$$

for the given training sequence $\mathbf{O}$. Using the Bayes theorem, we rewrite $P(\lambda|\mathbf{O})$ as

$$P(\lambda|\mathbf{O}) = \frac{P(\mathbf{O}|\lambda)P_0(\lambda)}{P(\mathbf{O})}. \tag{6.117}$$

The influence of the parameter prior $P_0(\lambda)$ in the solution process thus becomes explicit. Note that if the distribution is correctly chosen, the MAP solution attains minimum Bayes risk.

The parameter prior distribution characterizes the statistics of the parameters of interest before any measurement is made. If the prior distribution indicates no preference as to what the parameter values are likely to be, then the prior is called a noninformative prior (which is essentially constant for the entire parameter space). In this case, the MAP estimate obtained by solving Eq. (6.116) is identical to the ML estimate of Eq. (6.115). If we do have prior knowledge about the parameter values of the model, the incorporation of such prior knowledge in the form of a prior distribution would become important in the MAP estimate for minimum Bayes risk. This type of prior is often called an informative prior. Intuitively, if we know what the parameter values are likely to be before observations are made, we may be able to make good use of the data, which may be limited, to obtain a good model estimate. If this is true, the questions remaining are how to derive the informative prior and how to use it in obtaining the MAP estimate.

For mathematical tractability, conjugate priors are often used in Bayesian adaptation. A conjugate prior for a random vector is defined as the prior distribution for the parameters of the probability density function of the random vector, such that the posterior distribution $P(\lambda|\mathbf{O})$ and the prior distribution $P(\lambda)$ belong to the same distribution family for any sample observations $\mathbf{O}$. For example, it is well known that the conjugate prior for the mean of a Gaussian density is also a Gaussian density. In the following, we therefore discuss only the use of conjugate priors. We also discuss only the case of Bayesian adaptation of the Gaussian mean as it is sufficient to demonstrate the idea of Bayesian adaptation in dealing with small training set problems.



Let us focus on a Gaussian mixture component $\mathcal{N}(\mu, \sigma^2)$ in a mixture density HMM. We use one-dimensional observations for simplicity. Assume the mean $\mu$ is random with prior distribution $P_0(\mu)$ and the variance $\sigma^2$ is known and fixed. It can be shown that the conjugate prior for $\mu$ is also Gaussian; that is, if we assume $P_0(\mu)$ to be the conjugate prior of $\mu$, then $P_0(\mu)$ is Gaussian. Thus, let us denote the mean and variance of the prior for $\mu$ by $\rho$ and $\tau^2$, respectively. The MAP estimate for the mean parameter $\mu$ in Bayesian adaptation, from a set of $n$ training observations, is given by

$$\hat{\mu}_{\text{MAP}} = \frac{n\tau^2}{\sigma^2 + n\tau^2}\bar{\mathbf{o}} + \frac{\sigma^2}{\sigma^2 + n\tau^2}\rho \tag{6.118}$$

where $\bar{\mathbf{o}}$ is the sample mean of the $n$ training data. The interpretation of Eq. (6.118) is as follows. If there are no training data presented, $n = 0$ and the best estimate of $\mu$ is simply the mean $\rho$ of the prior distribution of the $\mu$ parameter. When training data are collected and used, the MAP estimate becomes a weighted average of the prior mean $\rho$ and the sample mean of the presented observations, $\bar{\mathbf{o}}$. Ultimately, $n \to \infty$ and the best estimate of $\mu$ is, as expected, the sample mean $\bar{\mathbf{o}}$. It should also be noted that if the prior variance $\tau^2$ is much larger than $\sigma^2/n$, the MAP estimate in Eq. (6.118) is essentially the ML estimate, $\bar{\mathbf{o}}$, which corresponds to the case of using noninformative priors.

A key question is, How do we determine $\rho$ and $\tau^2$? In practice, these prior parameters have to be estimated from a collection of speaker-dependent (or multispeaker) models, or from a speaker-independent model with mixture distributions in each state. For example, $\rho$ and $\tau^2$ can be estimated by

$$\rho = \sum_{m=1}^{M} c_m \rho_m \tag{6.119a}$$

and

$$\tau^2 = \sum_{m=1}^{M} c_m(\rho_m - \rho)^2 \tag{6.119b}$$

where $c_m$ is the weight assigned to the $m^{\text{th}}$ model (or the $m^{\text{th}}$ mixture component in the corresponding state of a mixture density speaker-independent HMM) and $\rho_m$ is the mean of the corresponding $m^{\text{th}}$ model (or mixture component). When using a speaker-independent Gaussian mixture HMM, the weight $c_m$ is basically the mixture gain for the $m^{\text{th}}$ mixture component, and the estimates of Eq. (6.119) are the ML estimates of the mean and variance parameters of $\mu$ before any speaker-specific training data are observed.

The concept of Bayesian adaptation based on conjugate priors can be applied to other parameters as well. The adaptation method can be shown to provide good parameter estimates even when the number of speaker-specific training tokens is extremely limited. Experiments have shown that large improvements in recognition accuracy are obtained with the Bayesian adaptation method, compared to direct training, particularly when only a small number of training tokens are available [30].

### 6.13.3 Corrective Training

In statistical pattern recognition, the minimum Bayes' risk is the theoretical recognizer performance bound, conditioned on the exact knowledge of both the class prior $P(v)$ and the conditional distributions $P(\mathbf{O}|v)$. When both distributions are not known exactly, and the classifier needs to be designed based on a finite training set, there are several ways to try to reduce the error rate. One method is based on the theoretical link between discriminant analysis and distribution estimation [32]. The idea is to design a classifier (discriminant function) such that the minimum classification error rate is attained on the training set. In particular, we wish to design a classifier that uses estimates of $P(v)$ and $P(\mathbf{O}|v)$, and that achieves a minimum error rate for the training set in the same way a discriminant function is designed. The reason for using the HMM, $P(\mathbf{O}|\lambda_v)$, for modeling $P(\mathbf{O}|v)$, as opposed to other discriminant functions, is to exploit the strengths of the HMMs—consistency, flexibility and computational ease.

Bahl et al. [33] were the first to propose an error-correction strategy, which they named corrective training, to specifically deal with the misclassification problem. Their training algorithm was motivated by analogy with an error-correction training procedure for linear classifiers. In their proposed method, the observation distribution is of a discrete type, $B = [b_i(k)]$, where $b_i(k)$ is the probability of observing a vector quantization code index (acoustic label) $k$ when the HMM source is in state $i$. Each $b_i(k)$ is obtained via the forward-backward algorithm as the weighted frequency of occurrence of the code index. The corrective training algorithm of Bahl et al. works as follows. First, use a labeled training set to estimate the parameters of the HMMs $\Lambda = \{\lambda_v\}$ with the forward-backward algorithm. For each utterance $\mathbf{O}$, labeled as $v$, for example, evaluate $P(\mathbf{O}|\lambda_v)$ for the correct class $v$ and $P(\mathbf{O}|\lambda_w)$ for each incorrect class $w$. (The evaluation of likelihood for the incorrect classes need not be exhaustive.) For every utterance where $\log P(\mathbf{O}|\lambda_w) > \log P(\mathbf{O}|\lambda_v) - \Delta$, where $\Delta$ is a prescribed threshold, modify $\lambda_v$ and $\lambda_w$ according to the following mechanism: (1) Apply the forward-backward algorithm to obtain estimates $b_i'(k)$ and $b_i''(k)$, using the labeled utterance $\mathbf{O}$ only, for the correct class $v$ and incorrect class $w$, respectively; (2) Modify the original $b_i(k)$ in $\lambda_v$ to $b_i(k) + \gamma b_i'(k)$ and the $b_i(k)$ in $\lambda_w$ to $b_i(k) - \gamma b_i''(k)$. When the state labels are tied for certain models, the above procedure is equivalent to replacing the original $b_i(k)$ by $b_i(k) + \gamma(b_i'(k) - b_i''(k))$. The prescribed adaptation parameter, $\gamma$, controls the "rate of convergence" and the threshold, $\Delta$, defines the "near-miss" cases. This corrective training algorithm therefore focuses on those parts of the model that are most important for word discrimination, a clear difference from the maximum likelihood principle.

Bahl et al. reported that the corrective training procedure worked better (in isolated word-recognition tasks) than models obtained using the maximum mutual information or the conditional maximum likelihood criterion. The method, however, is primarily experimental.

Several other forms of discriminative training were also proposed by Katagiri et al. [34] together with a framework for the analysis of related training/learning ideas for minimizing recognition errors. The discriminative training method described in Sec. 5.6.3 can be applied to HMM training without difficulty. The corrective training algorithm of Bahl et al. can be shown to be just one possible choice for the minimization of a prescribed risk function.



## 6.14 MODEL CLUSTERING AND SPLITTING

One of the basic assumptions in statistical modeling is that the variability in the observations from an information source can be modeled by the assumed statistical distributions. For speech recognition, the source could be a single word, a subword unit like a phoneme, or a word sequence. Because of variability in the production (e.g., accents, speed of talking), or the processing (e.g., transmission distortion, noise), it is often expedient to consider using more than a single HMM to characterize the source. There are two motivations behind this multiple HMM approach. First, lumping together all the variability from inhomogeneous data sources leads to unnecessarily complex models, often yielding lower modeling accuracy. Second, some of the variability, or rather the inhomogeneity in the source data, may be known *a priori*, thus warranting separate modeling of the source data sets.

Several generalized clustering algorithms exist, such as the $k$-means clustering algorithm, the generalized Lloyd algorithm as is widely used in vector quantizer designs [35], and the greedy growing algorithm found in set partition or decision tree designs [36], all of which are suitable for the purpose of separating inconsistent training data so that each divided subgroup becomes more homogeneous and therefore is better modeled by a single HMM. The nearest-neighbor rule required in these clustering algorithms is simply to assign an observation sequence $\mathbf{O}$ to cluster $i$ if

$$P(\mathbf{O}|\lambda_i) = \max_j P(\mathbf{O}|\lambda_j) \tag{6.120}$$

where $\lambda_j$s denote the models of the clusters. Successful application of the model clustering algorithms to the speech-recognition problem, using the straightforward maximum likelihood criterion, has been reported.

An alternative to model clustering is to arbitrarily subdivide a given speech source into a large number of subclasses with specialized characteristics and then consider a generalized procedure for model merging based on source likelihood considerations. By way of example, for large vocabulary speech recognition we often try to build specialized units (context sensitive) for recognition. For example, we could consider building units that are a function of the sound immediately preceding the unit (left-context) and the sound immediately following the unit (right-context). There are on the order of 10,000 such units in English. Many of the units are functionally almost identical. The problem is how to determine which pairs of units should be merged (so that the number of model units is made more manageable and the variance of the parameter estimate is reduced). To get ideas, consider two distinct models, $\lambda_a$ and $\lambda_b$, corresponding to training observation sets $\mathbf{O}_a$ and $\mathbf{O}_b$, and the merged model $\lambda_{a+b}$, corresponding to the merged observation sets $\{\mathbf{O}_a, \mathbf{O}_b\}$. We can then compute the change in entropy (i.e., loss of information) resulting from the

merged model as

$$\begin{aligned}\Delta H_{ab} &= H_a + H_b - H_{a+b} \\ &= -P(\mathbf{O}_a|\lambda_a)\log P(\mathbf{O}_a|\lambda_a) - P(\mathbf{O}_b|\lambda_b)\log P(\mathbf{O}_b|\lambda_b) \\ &\quad + P(\{\mathbf{O}_a,\mathbf{O}_b\}|\lambda_{a+b})\log P(\{\mathbf{O}_a,\mathbf{O}_b\}|\lambda_{a+b}).\end{aligned} \tag{6.121}$$

Whenever $\Delta H_{ab}$ is small enough, it means that the change in entropy resulting from merging the models will not affect system performance (at least on the training set) and the models can be merged. The question of how small is acceptable is dependent on specific applications. This model merging technique has been used successfully by Lee [37] to create a generalized set of triphone models for large vocabulary speech recognition.

## 6.15 HMM SYSTEM FOR ISOLATED WORD RECOGNITION

To illustrate the techniques discussed in this chapter, consider using HMMs to build an isolated word recognizer ([38]). Assume we have a vocabulary of $V$ words to be recognized and that each word is to be modeled by a distinct HMM. Further assume, for simplicity of notation, that for each word in the vocabulary we have a training set of $K$ utterances of the word (spoken by one or more talkers) where each utterance constitutes an observation sequence, of some appropriate representation of the (spectral and/or temporal) characteristics of the word. To do isolated word speech recognition, we must perform the following:

1. For each word $v$ in the vocabulary, we must build an HMM $\lambda_v$—that is, we must estimate the model parameters $(A, B, \pi)$ that optimize the likelihood of the training set observation vectors for the $v$th word.
2. For each unknown word to be recognized, the processing shown in Figure 6.13 must be carried out, namely, measurement of the observation sequence $\mathbf{O} = \{\mathbf{o}_1\,\mathbf{o}_2 \ldots \mathbf{o}_T\}$, via a feature analysis of the speech corresponding to the word; followed by calculation of model likelihoods for all possible models, $P(\mathbf{O}|\lambda_v)$, $1 \le v \le V$; followed by selection of the word whose model likelihood is highest—that is,

$$v^* = \arg\max_{1\le v\le V}[P(\mathbf{O}|\lambda_v)]. \tag{6.122}$$

The probability computation step is generally performed using the Viterbi algorithm (i.e., the maximum likelihood path is used) and requires on the order of $V \cdot N^2 \cdot T$ computations. For modest vocabulary sizes, e.g., $V = 100$ words, with an $N = 5$ state model, and $T = 40$ observations for the unknown word, a total of $10^5$ computations is required for recognition (where each computation is a multiply, and add, and a calculation of observation density, $b(\mathbf{o})$). Clearly this amount of computation is modest compared to the capabilities of most modern signal processor chips.



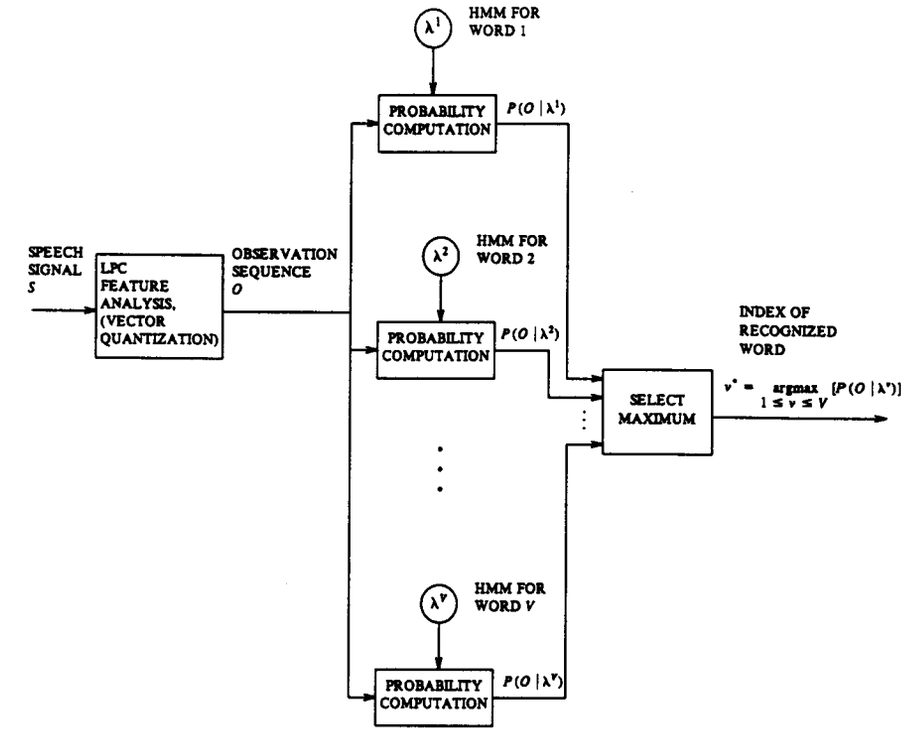



**Figure 6.13** Block diagram of an isolated word HMM recognizer (after Rabiner [38]).

### 6.15.1 Choice of Model Parameters

We now return to the issue that we have raised several times in this chapter—namely, How do we select the type of model, and how do we choose the parameters of the selected model? For isolated word recognition with a distinct HMM designed for each word in the vocabulary, it should be clear that a left-right model is more appropriate than an ergodic model, since we can then associate time with model states in a fairly straightforward manner. Furthermore, we can envision the physical meaning of the model states as distinct sounds (e.g., phonemes, syllables) of the word being modeled.

The issue of the number of states to use in each word model leads to two schools of thought. One idea is to let the number of states correspond roughly to the number of sounds (phonemes) within the word—hence, models with from 2 to 10 states would be appropriate [18]. The other idea is to let the number of states correspond roughly to the average number of observations in a spoken version of the word, the so-called Bakis model

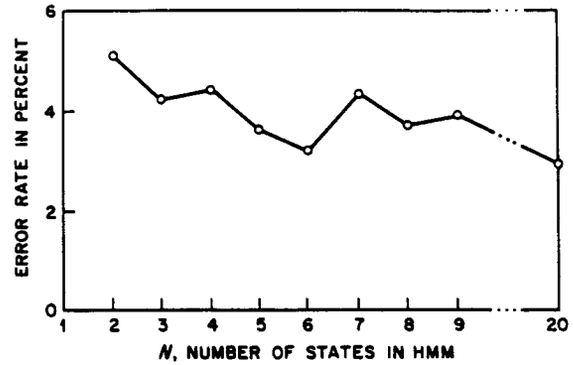



**Figure 6.14** Average word error rate (for a digits vocabulary) versus the number of states $N$ in the HMM (after Rabiner et al. [18]).

[11]. In this manner each state corresponds to an observation interval—i.e., about 10–15 ms for the standard methods of analysis. In the results to be described later in this section, we use the former approach. Furthermore, we restrict each word model to have the same number of states; this implies that the models will work best when they represent words with the same number of sounds.

To illustrate the effect of varying the number of states in a word model, Figure 6.14 shows a plot of average word error rate versus $N$, for the case of recognition of isolated digits (i.e., a 10-word vocabulary). It can be seen that the error is somewhat insensitive to $N$, achieving a local minimum at $N = 6$; however, differences in error rate for values of $N$ close to 6 are small.

The next issue is the choice of observation vector and the way it is represented. As discussed in Chapter 3, possibilities include LPC-derived weighted cepstral coefficients and weighted cepstral derivatives or (for autoregressive HMMs) the autocorrelation coefficients as the observation vectors for continuous models; for discrete symbol models we use a codebook to generate the discrete symbols. For the continuous models we use as many as $M = 64 \sim 256$ mixtures per state; for the discrete symbol models we use codebooks with as many as $M = 512 \sim 1024$ code words. Also, for the continuous models, it has been found that it is more convenient and sometimes preferable to use diagonal covariance matrices with several mixtures, rather than fewer mixtures with full covariance matrices. The reason for this is simple—namely, the difficulty in performing reliable reestimation of the off diagonal components of the covariance matrix from the necessarily limited training data. Figure 6.15 illustrates the need for using mixture densities for modeling observation vectors (eighth-order cepstral vectors derived from LPC with log energy appended as the ninth vector component). Figure 6.15 shows a comparison of marginal distributions $b_j(\mathbf{o})$ against a histogram of the actual observations within a state (as determined by a maximum likelihood segmentation of all the training observations into states). The observation vectors



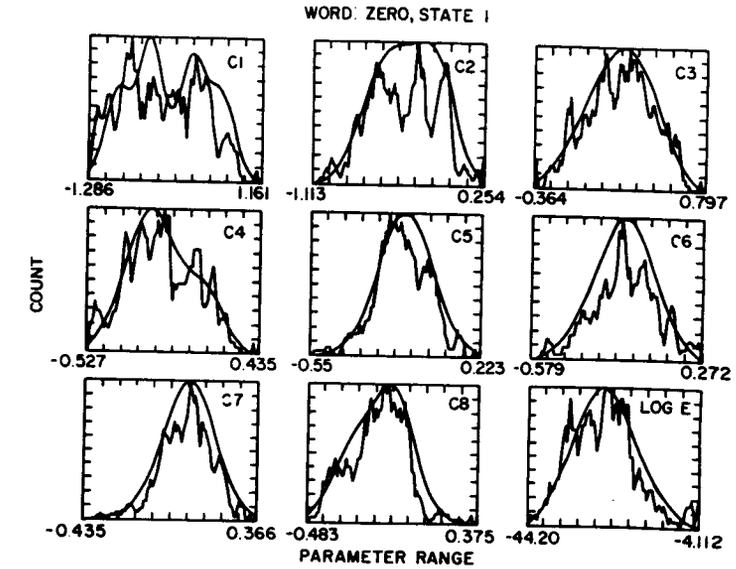



**Figure 6.15** Comparison of estimated density (jagged contour) and model density (smooth contour) for each of the nine components of the observation vector (eight cepstral components, one log energy component) for state 1 of the digit zero (after Rabiner et al. [38]).

are ninth order, and the model density uses $M = 5$ mixtures. The covariance matrices are constrained to be diagonal for each individual mixture. The results of Figure 6.15 are for the first model state of the word "zero." The need for values of $M > 1$ is clearly seen in the histogram of the first parameter (the first cepstral component) which is inherently multimodal; similarly, the second, fourth, and eighth cepstral parameters show the need for more than a single Gaussian component to provide good fits to the empirical data. Many of the other parameters appear to be well fitted by a single Gaussian; in some cases, however, even $M = 5$ mixtures do not provide a sufficiently good fit.

Another experimentally verified fact about the HMM is that it is important to limit some of the parameter estimates to prevent them from becoming too small. For example, for the discrete symbol models, the constraint that $b_j(k)$ be greater than or equal to some minimum value $\epsilon$ is necessary to ensure that even when the $k$th symbol never occurred in some state $j$ in the training observation set, there is always a finite probability of its occurrence when scoring an unknown observation set. To illustrate this point, Figure 6.16 shows a curve of average word error rate versus the parameter $\epsilon$ (on a log scale) for a standard word-recognition experiment. It can be seen that over a very broad range ($10^{-10} \leq \epsilon \leq 10^{-3}$) the average error rate remains at about a constant value; however,

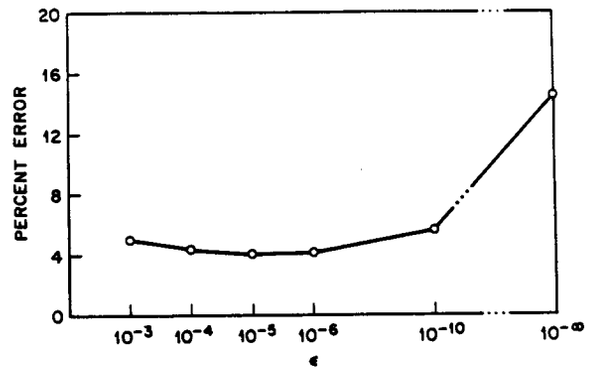



**Figure 6.16** Average word error rate as a function of the minimum discrete density value $\epsilon$ (after Rabiner et al. [18]).

when $\epsilon$ is set to 0 (i.e., $10^{-\infty}$), then the error rate increases sharply. Similarly, for continuous densities it is important to constrain the mixture gains $c_{jm}$ as well as the diagonal covariance coefficients $U_{jm}(r, r)$ to be greater than or equal to some minimum values (we use $10^{-4}$ in all cases) [18].

### 6.15.2 Segmental K-Means Segmentation into States

In this chapter, we have emphasized that good initial estimates of the parameters of the $b_j(\mathbf{o}_t)$ densities are essential for rapid and proper convergence of the reestimation formulas. Hence a procedure for providing good initial estimates of these parameters was devised and is shown in Figure 6.17. The training procedure is a variant on the well-known $K$-means iterative procedure for clustering data.

We assume that we have a training set of observations (the same as is required for parameter reestimation), and an initial estimate of all model parameters. However, unlike the one required for reestimation, the initial model estimate can be chosen randomly or on the basis of any available model appropriate to the data.

Following model initialization, the set of training observation sequences is segmented into states, based on the current model $\lambda$. This segmentation is achieved by finding the optimum state sequence, via the Viterbi algorithm, and then backtracking along the optimal path. This procedure is illustrated in Figure 6.18, which shows a log-energy plot, an accumulated log-likelihood plot, and a state segmentation for one occurrence of the word "six." The states correspond roughly to the sounds in the spoken word "six."

The result of segmenting each of the training sequences is, for each of the $N$ states, a maximum likelihood estimate of the set of the observations that occur within each state $j$ according to the current model. In the case where we are using discrete symbol densities, each of the observation vectors within a state is coded using the $M$-code-word codebook,



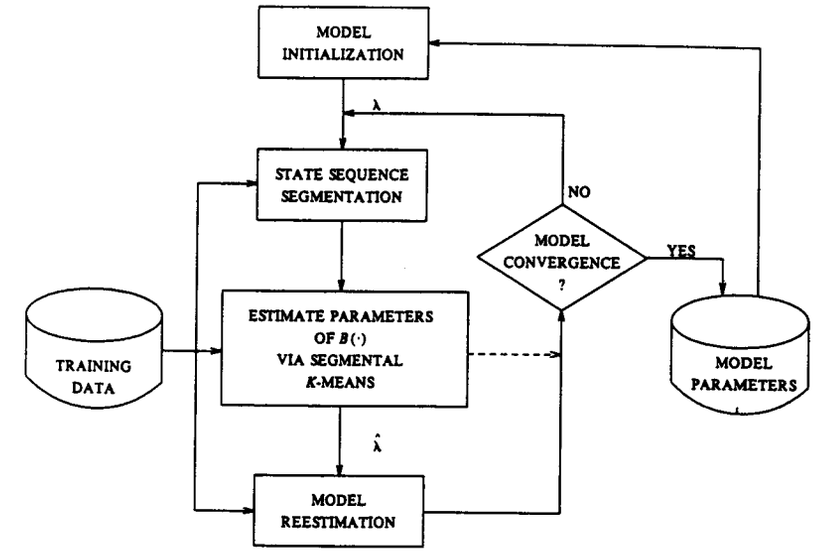



**Figure 6.17** The segmental $k$-means training procedure used to estimate parameter values for the optimal continuous mixture density fit to a finite number of observation sequences (after Rabiner et al. [38]).

and the updated estimate of the $b_j(k)$ parameters is

$$\hat{b}_j(k) = \text{number of vectors with codebook index } k \text{ in state } j \text{ divided by the number of vectors in state } j.$$

When we are using continuous observation densities, a segmental $K$-means procedure is used to cluster the observation vectors within each state $j$ into a set of $M$ clusters (using a Euclidean distortion measure), where each cluster represents one of the $M$ mixtures of the $b_j(\mathbf{o}_t)$ density. From the clustering, an updated set of model parameters is derived as follows:

$\hat{c}_{jm}$ = number of vectors classified in cluster $m$ of state $j$ divided by the number of vectors in state $j$

$\hat{\boldsymbol{\mu}}_{jm}$ = sample mean of the vectors classified in cluster $m$ of state $j$

$\hat{\mathbf{U}}_{jm}$ = sample covariance matrix of the vectors classified in cluster $m$ of state $j$.

Based on this state segmentation, updated estimates of the $a_{ij}$ coefficients can be obtained by counting the number of transitions from state $i$ to $j$ and dividing it by the number of transitions from state $i$ to any state (including itself).

An updated model $\hat{\lambda}$ is obtained from the new model parameters, and the formal reestimation procedure is used to reestimate all model parameters. The resulting model is then compared to the previous model (by computing a distance score that reflects the

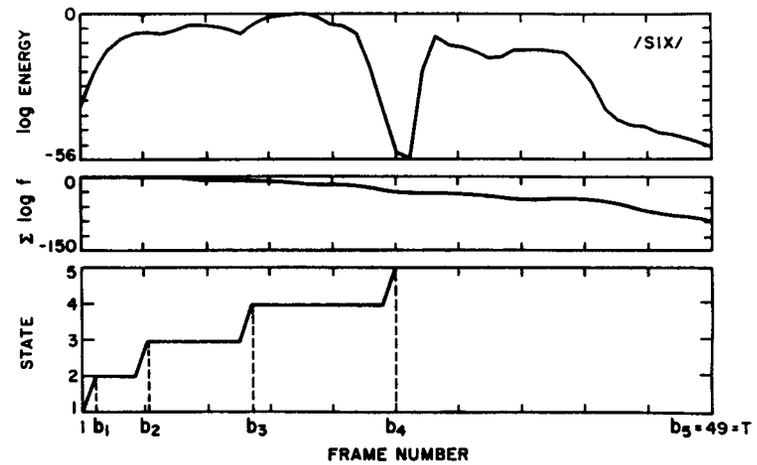



**Figure 6.18** Plots of (a) log energy; (b) accumulated log likelihood; and (c) state assignment for one occurrence of the word "six" (after Rabiner et al. [38]).

statistical similarity of the HMMs). If the model distance score exceeds a threshold, then the old model $\lambda$ is replaced by the new (reestimated) model $\overline{\lambda}$, and the overall training loop is repeated. If the model distance score falls below the threshold, then model convergence is assumed and the final model parameters are saved.

### 6.15.3 Incorporation of State Duration into the HMM

In Section 6.9 we discussed the theoretically correct method of incorporating state duration information into the mechanics of the HMM ([39]). We also showed that the cost of including duration density was rather high; namely a $D^2$-fold increase in computation and a $D$-fold increase in storage. Using a value of $D = 25$ (as is required for word recognition), the cost of the increased computation tended to make the techniques not worth using. Thus the following alternative procedure was formulated for incorporating state duration information into the HMM.

For this alternative procedure, the state duration probability $p_j(d)$ was measured directly from the segmented training sequences used in the segmental $K$-means procedure of the previous section. Hence the estimates of $p_j(d)$ are strictly heuristic ones. A typical set of histograms of $p_j(d)$ for a five-state model of the word "six" is shown in Figure 6.19. (In this figure the histograms are plotted versus normalized duration ($d/T$), rather than absolute duration $d$.) The first two states account for the initial /s/ in "six"; the third state accounts for the transition to the vowel /i/; the fourth state accounts for the vowel; and the fifth state accounts for the stop and the final /s/ sound.

The way in which the heuristic duration densities were used in the recognizer was



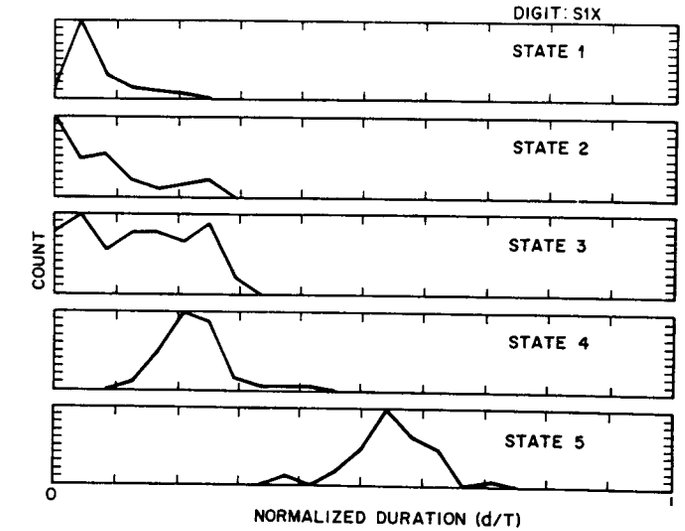



**Figure 6.19** Histograms of the normalized duration density for the five states of the digit "six" (after Rabiner et al. [38]).

as follows. First the normal Viterbi algorithm is used to give the best segmentation of the observation sequence of the unknown word into states via a backtracking procedure. The duration of each state is then measured from the state segmentation. A postprocessor then increments the log-likelihood score of the Viterbi algorithm, by the quantity

$$\log \hat{P}(\mathbf{q}, \mathbf{O}|\lambda) = \log P(\mathbf{q}, \mathbf{O}|\lambda) + \alpha_d \sum_{j=1}^{N} \log [p_j(d_j)] \tag{6.123}$$

where $\alpha_d$ is a scaling multiplier on the state duration scores, and $d_j$ is the duration of state $j$ along the optimal path as determined by the Viterbi algorithm. The incremental cost of the postprocessor for duration is essentially negligible, and experience has shown that recognition performance is essentially as good as that obtained using the theoretically correct duration model.

### 6.15.4 HMM Isolated-Digit Performance

We conclude this section on isolated word recognition using HMMs by giving a set of performance results (in terms of average word error rate) on the task of recognizing isolated digits in a speaker-independent manner. For this task, a training set consisting of 100 occurrences of each digit by 100 talkers (i.e., a single occurrence of each digit per talker) was used. Half the talkers were male and half were female. For testing the

**TABLE 6.1.** Average Digit Error Rates for Several Recognizers and Evaluation Sets

| Recognizer Type | Evaluation Set: Original Training | TS2 | TS3 | TS4 |
|---|---|---|---|---|
| LPC/DTW | 0.1 | 0.2 | 2.0 | 1.1 |
| LPC/DTW/VQ | – | 3.5 | – | – |
| HMM/VQ | – | 3.7 | – | – |
| HMM/CD | 0 | 0.2 | 1.3 | 1.8 |
| HMM/AR | 0.3 | 1.8 | 3.4 | 4.1 |

algorithm, we used the initial training set, as well as three other independent test sets with the following characteristics:

TS2 The same 100 talkers as were used in the training; 100 occurrences of each digit

TS3 A new set of 100 talkers (50 male, 50 female); 100 occurrences of each digit

TS4 Another new set of 100 talkers (50 male, 50 female); 100 occurrences of each digit

The results of the recognition tests are given in Table 6.1. The recognizers are the following:

| | |
|---|---|
| LPC/DTW | Conventional template-based recognizer using dynamic time warping (DTW) alignment |
| LPC/DTW/VQ | Conventional recognizer with vector quantization of the feature vectors ($M = 64$) |
| HMM/VQ | HMM recognizer with $M = 64$ codebook |
| HMM/CD | HMM recognizer using continuous density model with $M = 5$ mixtures per state |
| HMM/AR | HMM recognizer using autoregressive observation density |

It can be seen that, when using a VQ, the performance of the isolated word recognizer degrades in both the conventional and HMM modes. It can also be seen that the performances of the conventional template-based recognizer, and the HMM recognizer with a continuous density model are comparable. Finally Table 6.1 shows that the autoregressive density HMM gives poorer performance than the standard mixture density model.

## 6.16 SUMMARY

In this chapter we have attempted to present the theory of hidden Markov models from the simplest concepts (discrete Markov chains) to the most sophisticated models (variable duration, continuous density models). It has been our purpose to focus on physical explanations of the basic mathematics; hence we have avoided long, drawn-out proofs or derivations of the key results, and concentrated primarily on trying to interpret the meaning of the math, and how it could be implemented in practice in real-world systems. We have also attempted to illustrate one application of the theory of HMMs to a simple problem in speech recognition—namely, isolated word recognition.



## REFERENCES


[1] L.E. Baum and T. Petrie, "Statistical inference for probabilistic functions of finite state Markov chains," *Ann. Math. Stat.*, 37: 1554-1563, 1966.

[2] L.E. Baum and J.A. Egon, "An inequality with applications to statistical estimation for probabilistic functions of a Markov process and to a model for ecology," *Bull. Amer. Meteorol. Soc.*, 73: 360-363, 1967.

[3] L.E. Baum and G.R. Sell, "Growth functions for transformations on manifolds," *Pac. J. Math.*, 27 (2): 211-227, 1968.

[4] L.E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains," *Ann. Math. Stat.*, 41 (1): 164-171, 1970.

[5] L.E. Baum, "An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes," *Inequalities*, 3: 1-8, 1972.

[6] J.K. Baker, "The dragon system—An overview," *IEEE Trans. Acoustics, Speech, Signal Proc.*, ASSP-23 (1): 24-29, February 1975.

[7] F. Jelinek, "A fast sequential decoding algorithm using a stack," *IBM J. Res. Develop.*, 13: 675-685, 1969.

[8] L.R. Bahl and F. Jelinek, "Decoding for channels with insertions, deletions, and substitutions with applications to speech recognition," *IEEE Trans. Information Theory*, IT-21: 404-411, 1975.

[9] F. Jelinek, L.R. Bahl, and R.L. Mercer, "Design of a linguistic statistical decoder for the recognition of continuous speech," *IEEE Trans. Information Theory*, IT-21: 250-256, 1975.

[10] F. Jelinek, "Continuous speech recognition by statistical methods," *Proc. IEEE*, 64: 532-536, April 1976.

[11] R. Bakis, "Continuous speech word recognition via centisecond acoustic states," in *Proc. ASA Meeting* (Washington, DC), April 1976.

[12] F. Jelinek, L.R. Bahl, and R.L. Mercer, "Continuous speech recognition: Statistical methods," in *Handbook of Statistics, II*, P.R. Krishnaiad, Ed. Amsterdam, The Netherlands: North-Holland, 1982.

[13] L.R. Bahl, F. Jelinek, and R.L. Mercer, "A maximum likelihood approach to continuous speech recognition," *IEEE Trans. Pattern Anal. Machine Intell.*, PAMI-5: 179-190, 1983.

[14] J.D. Ferguson, "Hidden Markov Analysis: An Introduction," in *Hidden Markov Models for Speech*, Institute for Defense Analyses, Princeton, NJ, 1980.

[15] A.J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimal decoding algorithm," *IEEE Trans. Information Theory*, IT-13: 260-269, April 1967.

[16] G.D. Forney, "The Viterbi algorithm," *Proc. IEEE*, 61: 268-278, March 1973.

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

[18] 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 Tech. J.*, 62 (4): 1035–1074, April 1983.

[19] L.A. Liporace, "Maximum likelihood estimation for multivariate observations of Markov sources," *IEEE Trans. Information Theory*, IT-28 (5): 729-734, 1982.

[20] B.H. Juang, "Maximum likelihood estimation for mixture multivariate stochastic observations of Markov chains," *AT&T Tech. J.*, 64 (6): 1235-1249, July-Aug. 1985.

[21] B.H. Juang, S.E. Levinson, and M.M. Sondhi, "Maximum likelihood estimation for multivariate mixture observations of Markov chains," *IEEE Trans. Information Theory*, IT-32 (2): 307-309, March 1986.

[22] A.B. Poritz, "Linear predictive hidden Markov models and the speech signal," in *Proc. ICASSP 82* (Paris, France), 1291-1294, May 1982.

[23] B.H. Juang and L.R. Rabiner, "Mixture autoregressive hidden Markov models for speech signals," *IEEE Trans. Acoustics, Speech, Signal Proc.*, ASSP-33 (6): 1404-1413, December 1985.

[24] M J. Russell and R.K. Moore, "Explicit modeling of state occupancy in hidden Markov models for automatic speech recognition," in *Proc. ICASSP 85* (Tampa, FL), 5-8, March 1985.

[25] S.E. Levinson, "Continuously variable duration hidden Markov models for automatic speech recognition," *Computer Speech and Language*, 1 (1): 29-45, March 1986.

[26] L.R. Bahl, P.F. Brown, P.V. deSouza, and L.R. Mercer, "Maximum mutual information estimation of hidden Markov model parameters for speech recognition," in *Proc. ICASSP 86* (Tokyo, Japan), 49-52, April 1986.

[27] Y. Ephraim, A. Dembo, and L.R. Rabiner, "A minimum discrimination information approach for hidden Markov modeling," *IEEE Trans. Information Theory*, 35 (5): 1001-1003, September 1989.

[28] Y. Ephraim and L. Rabiner, "On the Relations Between Modeling Approaches for Speech Recognition," *IEEE Trans. Information Theory*, 36 (2): 372-380, March 1990.

[29] B.H. Juang and L.R. Rabiner, "A probabilistic distance measure for hidden Markov models," *AT&T Tech. J.*, 64 (2): 391-408, February 1985.

[30] F. Jelinek and R.L. Mercer, "Interpolated estimation of Markov source parameters from sparse data," in *Pattern Recognition in Practice*, E.S. Gelesma and L.N. Kanal, Eds. Amsterdam, The Netherlands: North-Holland, 1980, 381-397.

[31] C.-H. Lee, C.-H. Lin, and B.H. Juang, "A study on speaker adaptation of the parameters of continuous density hidden Markov models," *IEEE Trans. Signal Processing*, 39 (4): 806–841, April 1991.

[32] R.O. Duda and P.E. Hart, *Pattern Classification and Scene Analysis*, John Wiley & Sons, New York, 1973.

[33] L.R. Bahl, P.F. Brown, P.V. deSouza, and R.L. Mercer, "A new algorithm for the estimation of hidden Markov model parameters," *Proc. ICASSP 88*, 493–496, New York, April 1988.

[34] S. Katagiri, C.H. Lee, and B.H. Juang, "New Discriminative Training Algorithms Based on the Generalized Probabilistic Descent Method," *Proc. 1991 Workshop Neural Networks for Signal Processing*, 299–308, IEEE Signal Processing Society, Princeton, September 1991.



[35] Y. Linde, A. Buzo, and R.M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Communications*, COM-28: 84–95, January 1980.

[36] J. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone, *Classification and Regression Trees*, Wadsworth & Brooks, Monterey, CA, 1984.

[37] K.F. Lee, *Automatic Speech Recognition—The Development of the SPHINX System*, Kluwer Academic Publishers, Boston, 1989.

[38] L.R. Rabiner, B.H. Juang, S.E. Levinson, and M.M. Sondhi, "Recognition of Isolated Digits Using Hidden Markov Models With Continuous Mixture Densities," *AT&T Tech. J.*, 64 (6): 1211–1234, July-Aug. 1985.

[39] L.R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proc. IEEE*, 77 (2): 257–286, February 1989.

Chapter 7

# SPEECH RECOGNITION BASED ON CONNECTED WORD MODELS

## 7.1 INTRODUCTION

Up to this point we have been discussing discrete utterance (or, as it is often called, isolated word or phrase) recognition. The assumption was that the speech to be recognized comprised a single word or phrase and was to be recognized as a complete entity with no explicit knowledge or regard for the phonetic content of the word or phrase. Hence, for a vocabulary of $V$ words (or phrases), the recognition algorithm consisted of matching (via time alignment) the measured sequence of spectral vectors of the unknown spoken input against each of the set of spectral patterns for the $V$ words and selecting the pattern whose accumulated time aligned spectral distance was smallest as the recognized word. Another implicit assumption was that each spoken utterance had a clearly defined beginning and ending point that could be found using some type of speech endpoint detector. As a result, pattern matching could be reliably performed without having to be concerned about uncertainties in the endpoints of the patterns being compared. For many applications, notably those referred to as "command-and-control" applications, in which the user is required to speak the command words one at a time (i.e., with distinct pauses between command words), this paradigm works well and is entirely appropriate. (We will discuss command-and-control applications in Chapter 9.) However, for applications in which the speech to be recognized consists of a sequence of words from the recognition vocabulary (e.g., strings of digits) such a paradigm is often unacceptable for practical reasons. Figure 7.1 illustrates this point for a three-digit sequence in which the upper panel shows the log





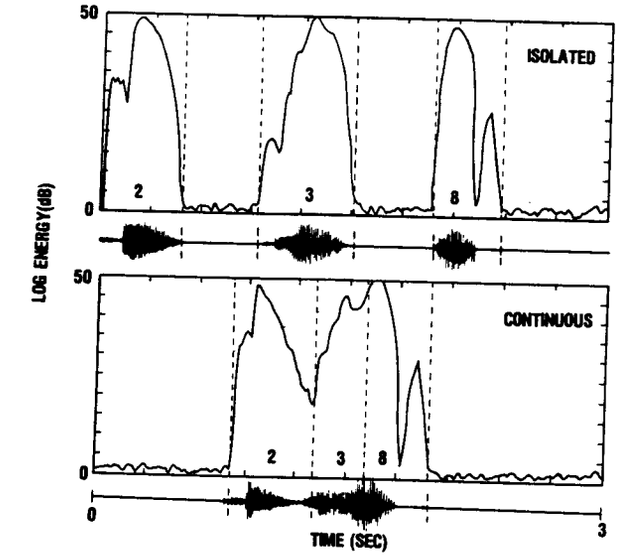



**Figure 7.1** Illustration of an isolated string of digits (upper panel) and a fluently spoken version of the same digit string (lower panel).

energy contour of a digit string spoken as a sequence of isolated digits where each digit is followed by a discernible pause (i.e., period of no speaking), and the lower panel shows the same digit string spoken in a fluent (continuous) manner. It should be self-evident that speaking a sequence of isolated digits is unnatural (from the point of view of human speech production) and grossly inefficient (it takes about 2.5 seconds to speak the isolated digit sequence versus less than 1 second to speak the same string of digits in a fluent manner). Hence, it is important to be able to extend the techniques described in earlier chapters of this book so that accurate recognition of fluent speech becomes possible.

From the point of view of speech-recognition algorithms, there are two interesting classes of fluent speech strings. One class is the set of strings derived from small-to-moderate-size vocabularies, including digit strings, spelled letter sequences, combinations of alphanumerics, and strings for accessing limited databases based on small-to-moderate-size vocabularies and highly constrained word syntax. This set has the property that the basic speech-recognition unit can be the word (or phrase), much as in the case of isolated word- (or phrase-) recognition systems. The other class is the set of continuous speech drawn from moderate-to-large-size vocabularies where the basic speech-recognition unit cannot be the word because of complexity constraints. In such a case, subword speech units are necessary for the implementation of the speech-recognition system. We defer a discussion of the second class of speech strings to Chapter 8, where we provide a complete description of techniques used in the recognition of large vocabulary continuous speech. In

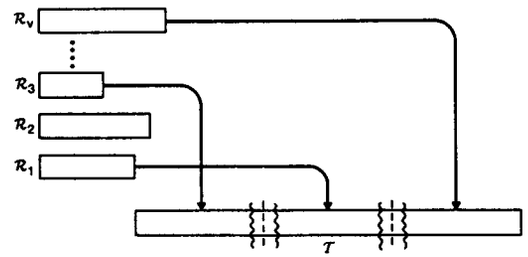



**Figure 7.2** Illustration of the connected word-recognition problem.

this chapter we discuss in depth the class of speech-recognition systems commonly referred to as connected word speech recognizers, because the basic unit of recognition is a whole word.

We can now formulate the basic problem in connected word recognition. To do this we refer to Figure 7.2, which gives a pictorial description of the connected word-recognition problem. We assume that we are given the spectral vectors from a fluently spoken string of words, $\mathcal{T} = \{\mathbf{t}(1), \mathbf{t}(2), \ldots, \mathbf{t}(M)\}$, and we are also given the spectral patterns for each of $V$ reference patterns, ($\mathcal{R}_1$ to $\mathcal{R}_V$) corresponding to $V$ unique words in the vocabulary. The connected word-recognition problem can be compactly stated as follows:

> Given a fluently spoken sequence of words, how can we determine the *optimum* match in terms of a concatenation of word reference patterns?

To solve the connected word-recognition problem we must resolve the following problems:

1. We don't usually know the number of words, $L$, in the string (although we usually have a good bound on the range, e.g., one to seven words).
2. We don't know word utterance boundaries within the spoken string; i.e. except for the beginning of the first word in the string, and the end of the last word in the string, we don't know precisely where any word begins or ends.
3. The word boundaries are often fuzzy or nonunique. That is, it is often difficult, if not impossible, to specify (i.e., find accurately and automatically) the word boundaries because of sound coarticulation. Thus, for example, the boundary between the digit 3 and the digit 8 in Figure 7.1 is fuzzy because the ending sound /i/ in 3 coarticulates strongly with the initial sound /e$^y$/ in 8. We indicate such fuzziness of the boundaries by the squiggly lines in Figure 7.2.
4. For a set of $V$ word-reference patterns, and for a given value of $L$ (the number of words in the string), there are $V^L$ possible combinations of composite matching patterns (possible $L$ word sequences); for anything but extremely small values of $V$ and $L$ the exponential number of composite matching patterns implies that the connected word-recognition problem *cannot* be solved by exhaustive means.

The bottom line is that a nonexhaustive matching algorithm is required to solve the connected word-recognition problem. In this chapter we discuss several algorithms that have this property and are capable of solving the connected word-recognition problem with various degrees of efficiency.



Although we will spend a great deal of time in this chapter discussing efficient algorithms for optimal matching of sequences of word patterns, it is also important to address the problem of word reference pattern training using connected word sequences. Unlike the isolated word case, for which training procedures were discussed in Chapters 5 and 6, for connected word sequence training we encounter a fundamental difficulty, namely the uncertainty in word boundaries within the spoken word sequence. This makes the connected word sequence training problem somewhat more difficult; therefore in this chapter we also describe an embedded word training method based on the segmental $k$-means algorithm.

## 7.2 GENERAL NOTATION FOR THE CONNECTED WORD-RECOGNITION PROBLEM

We denote the spectral sequence of vectors of the test pattern as

$$\mathcal{T} = \{\mathbf{t}(1), \mathbf{t}(2), \ldots, \mathbf{t}(M)\} = \{\mathbf{t}(m)\}_{m=1}^{M} \tag{7.1}$$

in which each $\mathbf{t}(m)$ is an appropriate spectral vector (e.g., filter bank, LPC, ear model, etc.). Similarly, we denote the set of word reference patterns (either templates or statistical models) as $\mathcal{R}_i$, $1 \le i \le V$ for a $V$ word vocabulary, where each pattern is of the form

$$\mathcal{R}_i = \{\mathbf{r}_i(1), \mathbf{r}_i(2), \ldots, \mathbf{r}_i(N_i)\} \tag{7.2}$$

where $N_i$ is the duration (in frames or states) of the $i^{\text{th}}$ word reference pattern. (For convenience we will develop the algorithms on the basis of using word templates; we will see later in this chapter that the required modifications for statistical models are essentially trivial.)

The connected word-recognition problem can now be stated as a problem in finding the optimum sequence of word reference patterns, $\mathcal{R}^*$, that best matches $\mathcal{T}$. With no loss of generality we assume that there are $L$ word patterns in $\mathcal{R}^*$ (where we vary $L$ from the minimum to the maximum possible values and take the optimum overall values of $L$). Hence the best sequence of reference patterns, $\mathcal{R}^*$, is a concatenation of $L$ reference patterns, i.e.,

$$\mathcal{R}^* = \{\mathcal{R}_{q^*(1)} \oplus \mathcal{R}_{q^*(2)} \oplus \mathcal{R}_{q^*(3)} \oplus \cdots \oplus \mathcal{R}_{q^*(L)}\} \tag{7.3}$$

in which each index, $q^*(\ell)$ is in the range $[1, V]$.

To determine $\mathcal{R}^*$—that is, to determine the sequence of word indices $q^*(\ell)$, $1 \le \ell \le L$, that give the best matching sequence—consider constructing an arbitrary "super-reference" pattern $\mathcal{R}^s$ of the form

$$\mathcal{R}^s = \mathcal{R}_{q(1)} \oplus \mathcal{R}_{q(2)} \oplus \mathcal{R}_{q(3)} \oplus \cdots \oplus \mathcal{R}_{q(L)} = \{\mathbf{r}^s(n)\}_{n=1}^{N^s} \tag{7.4}$$

in which $N^s$ is the total duration of the concatenated reference pattern $\mathcal{R}^s$. The time aligned

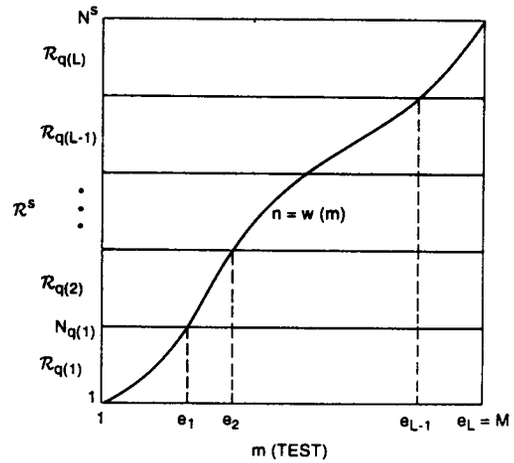



**Figure 7.3** Determination of the optimum alignment of super-reference pattern $\mathcal{R}^s$ to $\mathcal{T}$, along with corresponding word boundary frames.

distance between $\mathcal{R}^s$ and $\mathcal{T}$ is readily obtained via dynamic time warping, as illustrated in Figure 7.3. The distance obtained is

$$D(\mathcal{R}^s, \mathcal{T}) = \min_{w(m)} \sum_{m=1}^{M} d(\mathbf{t}(m), \mathbf{r}^s(w(m))) \tag{7.5}$$

in which $d(\cdot,\cdot)$ is a local spectral distance measure, $w(\cdot)$ is a warping function for the time index and the techniques of Chapter 4 can be used to evaluate $D$. By appropriate path backtracking, as illustrated in Figure 7.3, the word boundary frames in the input string can be found on the basis of the word boundary frames in the super-reference pattern. Hence the last frame in the first reference word, $\mathbf{r}_{q(1)}(N_{q(1)})$ maps to frame $e_1$ in the test pattern; similarly, the last frame in the second reference word $\mathbf{r}_{q(2)}(N_{q(2)})$ maps to frame $e_2$ in the test pattern, etc.

To determine the global best match, $\mathcal{R}^*$, we optimize Eq. (7.5) over every possible value of local reference patterns, $q(1), q(2), \ldots, q(L)$, and over every possible value of $L$, i.e., $L_{\min} \le L \le L_{\max}$, giving

$$\begin{aligned} D^* &= \min_{\mathcal{R}^s} D(\mathcal{R}^s, \mathcal{T}) \\ &= \min_{L_{\min} \le L \le L_{\max}} \; \min_{\substack{q(1),q(2),\ldots,q(L) \\ 1 \le q(i) \le V}} \; \min_{w(m)} \sum_{m=1}^{M} d(\mathbf{t}(m), \mathbf{r}^s(w(m))) \end{aligned} \tag{7.6}$$

with



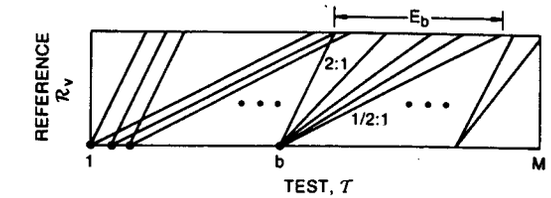



**Figure 7.4** Computation ranges for matching $\mathcal{R}_v$ against portions of $\mathcal{T}$.

$$\mathcal{R}^* = \arg\min_{\mathcal{R}^s} D(\mathcal{R}^s, \mathcal{T}). \tag{7.7}$$

For a straightforward evaluation of Eq. (7.6), the computation required is

$$C_L \approx \left[\frac{M \cdot \sum_{\ell=1}^{L} N_{q(\ell)}}{3}\right] V^L = \frac{M \cdot L \cdot \overline{N}}{3} V^L \text{ (grid points)} \tag{7.8}$$

where the bracketed term is the dynamic time warping computation for matching a given $\mathcal{R}^s$ to $\mathcal{T}$, the second term ($V^L$) is the number of possible combinations in $\mathcal{R}^s$, and $\overline{N}$ is the average number of frames in a word reference pattern. To get a feeling for how impractical the computation of Eq. (7.8) actually is, consider typical values of $M = 300$ frames, $L = 7$ words, $\overline{N} = 40$ frames, and $V = 10$ words. With these values we get $C_L \cong 2.8 \times 10^{11}$ (grid points). This computation (for a single test string) is already excessive for most modern machines. Hence, as discussed previously, alternatives to exhaustive evaluation of $D^*$ are required.

Fortunately, several algorithms have been proposed that solve Eq. (7.6) without the exponential growth in computation. These include

- the two-level dynamic programming approach [1,2]
- the level building approach [3,4]
- the one-stage approach and subsequent generalizations [5–9].

In the next several sections we discuss these individual algorithms.

## 7.3 THE TWO-LEVEL DYNAMIC PROGRAMMING (TWO-LEVEL DP) ALGORITHM

The basic idea of the two-level DP algorithm is to break up the computation of Eq. (7.6) into two stages (called levels). At the first level the algorithm matches each individual word reference pattern, $\mathcal{R}_v$, against an arbitrary portion of the test string, $\mathcal{T}$. To understand the computation involved, consider Figure 7.4.

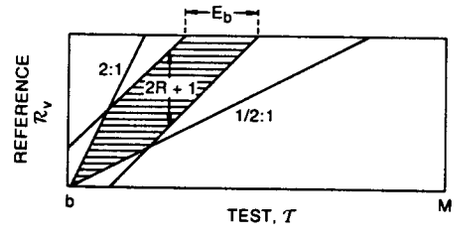



**Figure 7.5** Use of range limiting to reduce the size of individual time warps.

For the range of beginning test frames of the match, $b$, $1 \le b \le M$, and for the range of ending test frames, $e$, $1 \le e \le M$ $(e > b)$, and for word reference pattern, $\mathcal{R}_v$, $1 \le v \le V$, we need to compute

$$\hat{D}(v, b, e) = \min_{w(m)} \sum_{m=b}^{e} d(\mathbf{t}(m), \mathbf{r}_v(w(m))) \tag{7.9}$$

as illustrated in Figure 7.4. (It should be clear from our discussions in Chapter 4 that for each beginning frame $b$, a single time warp provides the time aligned distance, $\hat{D}$, for a range, $E_b$, of ending frames $e$, $b + \Delta \le e \le b + \delta$ where $\Delta$ and $\delta$ are related to $N_v$, the number of frames in the reference pattern $\mathcal{R}_v$, and the expansion/contraction factor of the DTW algorithm. Hence, using a 2-to-1 expansion and a $^1/_2$ to 1 compression, we would have $b+N_v/2 \le e \le b+2N_v$ as the range on ending frames for beginning frame $b$. Clearly if $b + 2N_v > M$ then $e$ is truncated at $M$, the end of the test pattern.)

Equation (7.9) gives the minimum distance for every possible vocabulary pattern, $\mathcal{R}_v$, between each possible pair of beginning and ending frames $(b, e)$. Clearly, we can eliminate $v$ by finding the best (smallest distance) match between $b$ and $e$ for any $v$, giving

$$\tilde{D}(b, e) = \min_{1 \le v \le V} [\hat{D}(v, b, e)] = \text{best score} \tag{7.10}$$

$$\tilde{N}(b, e) = \arg \min_{1 \le v \le V} [\hat{D}(v, b, e)] = \text{best reference index} \tag{7.11}$$

thereby significantly reducing the data storage (by a factor of $v$) with no loss of optimality. We can further reduce computation of Eq. (7.9) by using a range-limited DTW algorithm as illustrated in Figure 7.5. With appropriate choice of the range variable, $R$, the ending region, $E_b$, can be made significantly smaller than for a normal $2:1$ expansion, $^1/_2:1$ compression DTW algorithm.

Given the array of best scores, $\tilde{D}(b, e)$, of Eq. (7.10), the second level of the computation pieces together the individual reference pattern scores to minimize the overall accumulated distance over the entire test string. This can be accomplished using dynamic programming as follows. Consider ending frame $e$, as illustrated in Figure 7.6. We define



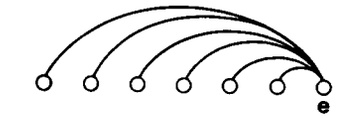



**Figure 7.6** Series of paths ending at frame $e$.

the distance of the best path ending at frame $e$ using a concatenated sequence of $\ell$ reference patterns as $\bar{D}_\ell(e)$, which can be derived as

$$\bar{D}_\ell(e) = \min_{1 \le b < e} [\tilde{D}(b, e) + \bar{D}_{\ell-1}(b-1)] \tag{7.12}$$

i.e., the best path ending at frame $e$ using exactly $\ell$ reference patterns is the one with minimum distance over all possible beginning frames, $b$, of the concatenation of the best path ending at frame $b-1$ using exactly $\ell - 1$ reference patterns plus the distance of Eq. (7.10) of the best path from frame $b$ to frame $e$ (of $\mathcal{T}$). The recursion of Eq. (7.12) is identical to the dynamic programming recursion principle discussed in Chapter 4. In this case the "local" distance used in the recursion is the word distance $\tilde{D}(b, e)$ (with assumed beginning and ending frames) which is calculated using Eq. (7.10) based on the first level dynamic programming results. (For single-word pattern matching the local distance is simply the beginning to ending frame distance.)

Based on the recursion of Eq. (7.12) we can formulate the following DP algorithm for determining the overall best path:

**Step 1, Initialization**

$$\bar{D}_0(0) = 0, \qquad \bar{D}_\ell(0) = \infty, \qquad 1 \le \ell \le L_{\max}$$

**Step 2, Loop on $e$ for $\ell = 1$**

$$\bar{D}_1(e) = \tilde{D}(1, e), \qquad 2 \le e \le M$$

**Step 3, Recursion, Loop on $e$ for $\ell = 2, 3, \ldots, L_{\max}$**

$$\bar{D}_2(e) = \min_{1 \le b < e} [\tilde{D}(b, e) + \bar{D}_1(b-1)], \quad 3 \le e \le M$$
$$\bar{D}_3(e) = \min_{1 \le b < e} [\tilde{D}(b, e) + \bar{D}_2(b-1)], \quad 4 \le e \le M$$
$$\bar{D}_\ell(e) = \min_{1 \le b < e} [\tilde{D}(b, e) + \bar{D}_{\ell-1}(b-1)], \quad \ell + 1 \le e \le M$$

**Step 4, Final Solution**

$$D^* = \min_{1 \le \ell \le L_{\max}} [\bar{D}_\ell(m)]$$

and use $\tilde{N}(b, e)$ to backtrack to obtain actual sequence of reference patterns in $\mathcal{R}^s$.

To gain a better understanding of how the above recursion is carried out in practice,

the following exercise gives an explicit example of decoding a given matrix of distances, $\tilde{D}(b, e)$, into best strings.

**Exercise 7.1**

Assume we are given the $15 \times 15$ matrix of distances, $\tilde{D}(b, e)$, representing best accumulated distances between frames $b$ and $e$, as shown below

ending frame, $e$

| beginning frame, $b$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | – | – | 9 | 10 | 13 | 17 | 22 | 25 | 29 | 33 | 37 | 41 | 45 | 50 | 60 |
| 2 | – | – | – | 7 | 10 | 14 | 17 | 21 | 25 | 29 | 32 | 36 | 40 | 45 | 53 |
| 3 | – | – | – | – | 9 | 13 | 16 | 19 | 23 | 26 | 28 | 32 | 35 | 40 | 47 |
| 4 | – | – | – | – | – | 6 | 8 | 9 | 11 | 12 | 14 | 19 | 23 | 28 | 33 |
| 5 | – | – | – | – | – | – | 4 | 6 | 8 | 10 | 12 | 15 | 19 | 23 | 30 |
| 6 | – | – | – | – | – | – | – | 9 | 12 | 16 | 19 | 23 | 27 | 30 | 33 |
| 7 | – | – | – | – | – | – | – | – | 15 | 18 | 22 | 27 | 32 | 37 | 45 |
| 8 | – | – | – | – | – | – | – | – | – | 12 | 16 | 21 | 25 | 29 | 33 |
| 9 | – | – | – | – | – | – | – | – | – | – | 6 | 8 | 10 | 13 | 16 |
| 10 | – | – | – | – | – | – | – | – | – | – | – | 5 | 7 | 9 | 12 |
| 11 | – | – | – | – | – | – | – | – | – | – | – | – | 4 | 6 | 8 |
| 12 | – | – | – | – | – | – | – | – | – | – | – | – | – | 2 | 4 |
| 13 | – | – | – | – | – | – | – | – | – | – | – | – | – | – | 2 |
| 14 | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| 15 | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |

1. Find the best path for a 1, 2, and 3 word match.
2. What are the total distance scores for the 3 string lengths? Which string length is most likely?

**Solution 7.1**

Although we can trivially determine the paths for the best one- and two-word matches, we must use the dynamic programming solution to determine the best three-word match. Hence it is worthwhile using the dynamic programming solution for all three matches. From the recursion above we get

$$\bar{D}_1(e) = \tilde{D}(1, e), \qquad 2 \le e \le 15$$

$$\bar{D}_2(e) = \min_{1 \le b < e} [\tilde{D}(b, e) + \bar{D}_1(b-1)], \qquad 3 \le e \le 15$$

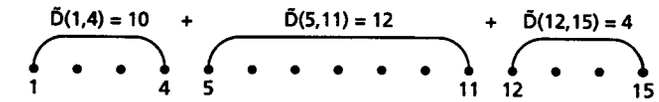
$$\bar{D}_3(e) = \min_{1 \le b < e} [\tilde{D}(b, e) + \bar{D}_2(b-1)], \qquad 4 \le e \le 15$$

From the matrix of distances we get



| $e$ | $\bar{D}_1(e)$ | $\bar{D}_2(e)$ | $\bar{D}_3(e)$ |
|---|---|---|---|
| 3 | 9 | - | - |
| 4 | 10 | - | - |
| 5 | 13 | - | - |
| 6 | 17 | $15 = (6 + 9)$ | - |
| 7 | 22 | $14 = (4 + 10)$ | - |
| 8 | 25 | $16 = (6 + 10)$ | - |
| 9 | 29 | $18 = (8 + 10)$ | $30 = (\bar{D}_2(6) + 15)$ |
| 10 | 33 | $20 = (10 + 10)$ | $26 = (\bar{D}_2(7) + 12)$ |
| 11 | 37 | $22 = (12 + 10)$ | $22 = (\bar{D}_2(8) + 6)$ |
| 12 | 41 | $25 = (15 + 10)$ | $23 = (\bar{D}_2(9) + 5)$ |
| 13 | 45 | $29 = (19 + 10)$ | $24 = (\bar{D}_2(10) + 4)$ |
| 14 | 50 | $33 = (23 + 10)$ | $24 = (\bar{D}_2(11) + 2)$ |
| 15 | 60 | $40 = (30 + 10)$ | $26 = (\bar{D}_2(12) + 4)$ |

1. For a one-word match we have $D_1^* = \bar{D}_1(15) = 60$ with path (1, 15). For a two-word match we have $D_2^* = \bar{D}_2(15) = 40$ with path (1, 4)(5, 15); i.e., word 1 spans frames 1 to 4 and word 2 spans frames 5 to 15. For a three-word match we have $D_3^* = \bar{D}_3(15) = 26$ with path (1, 4) (5, 11) (12, 15); i.e. word 1 spans frames 1 to 4, word 2 spans frames 5 to 11, and word 3 spans frames 12 to 15.
2. The distance scores for one-, two-, and three-word matches are 60, 40, and 26. Clearly, string length 3 has the smallest total distance and is most likely. The overall path of the 3 word string is




### 7.3.1 Computation of the Two-Level DP Algorithm

The total computation of the two-level DP algorithm is essentially the computation required to determine $\tilde{D}(v, b, e)$. This computation is essentially the computation of $V \cdot M$ time warps corresponding to a single time warp for each reference pattern ($V$) and for each starting frame ($M$). The size of each time warp (which provides distance score for the range of ending frames associated with each starting frame) is approximately $\bar{N}(2R + 1)$ for an $R$ frame range limited DTW search with a reference pattern of average length $\bar{N}$ frames. Hence the total computation of the two-level DP algorithm is

$$C_{2L} = V \cdot M \cdot \bar{N}(2R + 1) \text{ (grid points)} \tag{7.13}$$

in which the computation for each grid point constitutes a distance computation and the associated combinatorics. The storage of the two-level DP algorithm is essentially the

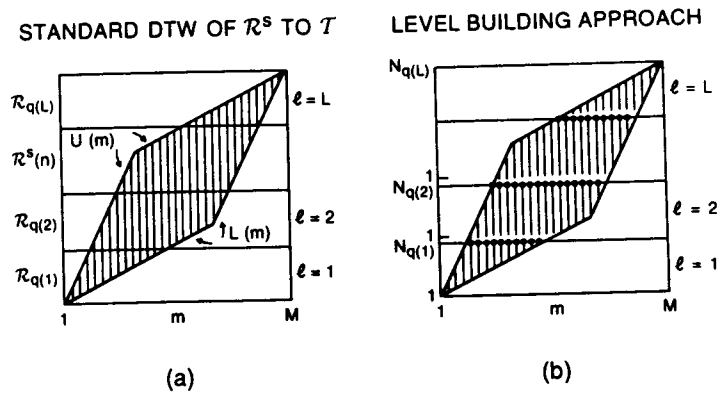



**Figure 7.7** Illustration of standard DTW alignment of super-reference and test patterns (a), and level building alignment (b) (after Myers and Rabiner [4]).

storage of the reduced distance matrix $[\tilde{D}(b,e)]$, since only temporary storage of the full array $\hat{D}(v,b,e)$ is required. The required storage of the range reduced algorithm is

$$S_{2L} = 2M(2R+1) \tag{7.14}$$

since for each of the $M$ starting frames, distance scores and best word identifications are saved for an ending range of $(2R+1)$ frames.

For a typical set of values—i.e., $M = 300$ frames, $\overline{N} = 40$ frames, $V = 10$ words, $R = 5$ frames—we require computation for 1,320,000 grid points and storage of 6600 locations for $\tilde{D}(b,e)$.

## 7.4 THE LEVEL BUILDING (LB) ALGORITHM

At the beginning of this chapter we discussed an exhaustive approach to connected word recognition in which we construct a set of super-reference patterns, $\mathcal{R}^s$, by concatenating every possible combination of word patterns (in sequence), and performing a global time alignment of each $\mathcal{R}^s$ to the test pattern $\mathcal{T}$. Figure 7.7a illustrates the computation associated with time aligning $\mathcal{R}^s$ to $\mathcal{T}$, namely iterative computation of accumulated distance at each grid point within a constrained region (the parallelogram of Figure 7.7a) of the alignment plane. Usually the computation is performed in a frame-synchronous manner; that is, the computation for a given test frame, $m$, is performed along a vertical stripe representing the allowable range of super-reference frames. Following this the computation for test frame $m+1$ occurs, then for frame $m+2$, etc., until the entire test frame sequence is used.

An alternative way of aligning $\mathcal{R}^s$ and $\mathcal{T}$ is shown in part b of Figure 7.7. Instead of performing the computation strictly along vertical stripes, the computation, for a given test frame $m$, is truncated at a fixed horizontal level (which corresponds to the super reference



frame at the end of the first word in $\mathcal{R}^s$). (The use of the word "level" here should not be confused with its usage in the two-level DP algorithm of the previous section. In the two-level DP algorithm the first level is the calculation of local warping distances for tentative word candidates, and the second level is the search for the optimal concatenated word path.) This procedure is iterated for all test frames for which we can define a range of reference frames within the allowed alignment region. For each test frame whose vertical stripe intersects the horizontal level corresponding to the end of the first word, we store the accumulated distance so that the DTW computation can be picked up at the next level (word in the string) by beginning the search at an appropriate test frame and iterating the search along vertical stripes corresponding to regions of intersection with the second word in the super-reference pattern. This procedure is iterated through all levels (words in $\mathcal{R}^s$) until the complete alignment grid has been covered.

The differences between the sequence of computation of the standard DTW alignment along complete vertical stripes (as in Figure 7.7a) and the level building alignment along partial vertical stripes (as in Figure 7.7b) are essentially minor for a single super-reference pattern. The efficiency of the approach comes with the realization that the computation at each level can be performed on all $V$ reference patterns *before* proceeding to the next level. The advantage of doing this is that after doing the $V$ warps at a given level, we can compare accumulated distance scores *across vocabulary words*, and retain only the minimum (best) score at each path ending frame, and still pick up the computation at the next level with no loss of optimality. The key difference between the level building algorithm and the two-level DP algorithm is that (with no loss of information) partial word decisions are made during the DP search, and these decisions are used to reduce the search range in later stages of the LB algorithm, whereas the two-level DP algorithm accumulates all the word scores until the end of the entire utterance and then decides the word string in a separate and independent second level DP calculation. The advantage we accrue is that, at each level, we do exactly $V$ time warps, so that the total computation is on the order of $V \cdot L$ time warps for an $L$-word string. Since $L$ is always significantly smaller than the number of test frames, $M$, we see that the level building approach inherently involves significantly less computation than the two-level DP algorithm (which required $V \cdot M$ time warps).

With this simplified overview of the level building approach, we now formally describe the implementation of the level building algorithm.

### 7.4.1 Mathematics of the Level Building Algorithm

We define $\bar{D}_\ell^v(m)$ as the minimum accumulated distance, at level $\ell$, using reference pattern $\mathcal{R}_v$, to frame $m$ of the test pattern, $\mathcal{T}$. Clearly $\bar{D}_\ell^v(m)$ is defined for $1 \le \ell \le L_{\max}$, $1 \le v \le V$ and $1 \le m \le M$. Consider the implementation of level 1 as shown in Figure 7.8.

The first reference pattern, $\mathcal{R}_1$, is aligned with $\mathcal{T}$ beginning at frame 1 of $\mathcal{T}$ using a standard DTW alignment procedure. The warping paths intersect the last frame in $\mathcal{R}_1$ (frame $N_1$) at a range of test frames, namely $m_{11}(1) \le m \le m_{12}(1)$. (For example if we used a DTW procedure with a minimum and maximum expansion of $1/2$ and 2 we would have $N_1/2 \le m \le 2N_1$.) For each of these warp path ending frames we store

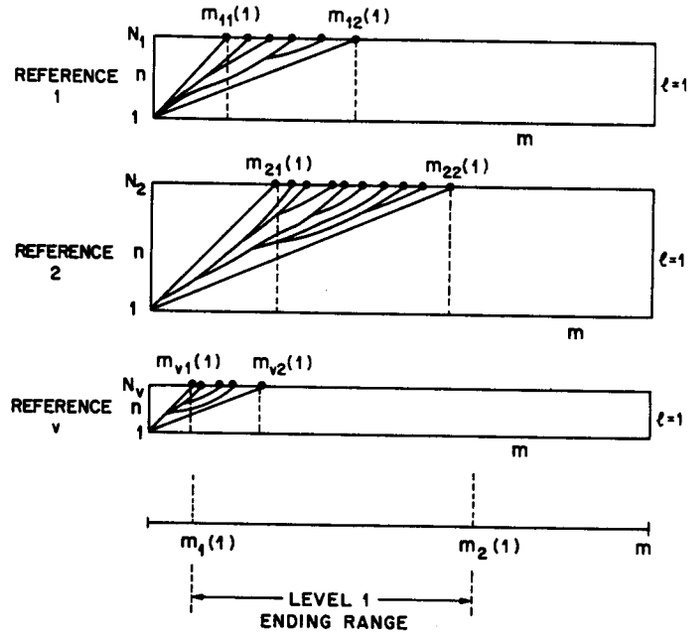



**Figure 7.8** Implementation of level 1 of the level building algorithm (after Myers and Rabiner [4]).

the accumulated distances, $\bar{D}_1^1(m)$. Similarly, for the second reference pattern, $\mathcal{R}_2$, with duration $N_2$ frames, we begin at frame 1 of $\mathcal{T}$ and obtain the best warping paths to the range $m_{21}(1) \le m \le m_{22}(1)$. (Clearly $m_{21}(1) \ne m_{11}(1)$ and $m_{22}(1) \ne m_{12}(1)$, since $N_2$ and $N_1$ are generally different. If $N_2 = N_1$, then $m_{21}(1) = m_{11}(1)$ and $m_{22}(1) = m_{12}(1)$.) This procedure is iterated for all $V$ reference patterns at level 1. Thus the output of level 1 is the array of accumulated distances and the corresponding ranges, i.e.,

$$\begin{aligned} &\bar{D}_1^1(m), \quad m_{11}(1) \le m \le m_{12}(1) \\ &\bar{D}_1^2(m), \quad m_{21}(1) \le m \le m_{22}(1) \\ &\vdots \\ &\bar{D}_1^V(m), \quad m_{V1}(1) \le m \le m_{V2}(1). \end{aligned} \tag{7.15}$$

We can now define the ending range of level 1, $m_1(1) \le m \le m_2(1)$, as the composite range of $m$ over which $\bar{D}_1^v$ is defined (and is not infinity), namely,

$$m_1(1) = \min_{1 \le v \le V} [m_{v1}(1)] \tag{7.16}$$



$$m_2(1) = \max_{1 \le v \le V} [m_{v2}(1)] \tag{7.17}$$

and for each frame, $m$, in the composite ending range, $m_1(\ell) \le m \le m_2(\ell)$, we need to store

$$\bar{D}_\ell^B(m) = \min_{1 \le v \le V} [\bar{D}_\ell^v(m)] \quad \text{– best distance at level } \ell \text{ to frame } m \tag{7.18}$$

$$\bar{N}_\ell^B(m) = \arg\min_{1 \le v \le V} [\bar{D}_\ell^v(m)] \quad \text{– reference pattern index which gave distance at level } \ell \text{ to frame } m \tag{7.19}$$

$$\bar{F}_\ell^B(m) = \bar{F}_\ell^{\bar{N}_\ell^B(m)}(m) \quad \text{– backpointer to best ending frame at previous level that achieves } \bar{D}_\ell^B(m) \tag{7.20}$$

(By definition $\bar{F}_1^B(m) = 0$ for all $m$ since the ending frame of the $0^{\text{th}}$ level is 0.) By storing only $\bar{D}_\ell^B(m)$, $\bar{N}_\ell^B(m)$ and $\bar{F}_\ell^B(m)$, we significantly reduce the storage at each level and yet we retain all the information required to pick up the optimal path through the entire grid as shown below.

The second-level computation does not begin until the first-level computation is finished. To see how the computation is picked up at level 2, consider Figure 7.9, which shows a series of time warps that span a set of beginning frames and provide a new set of ending frames.

For each reference pattern, $\mathcal{R}_v$, the range of starting frames is $m_1(1) \le m \le m_2(1)$ because every ending point of the first level is a possible starting point in the second level. Hence for each frame $m$ in the beginning range, and for each frame at the beginning of the reference pattern, we must consider paths coming from both the current reference pattern and the previous level. Other than the broadened beginning region, each DTW at level 2 is essentially identical to those at level 1. Thus for reference $\mathcal{R}_1$ the range of ending frames at level 2 is $m_{11}(2) \le m \le m_{12}(2)$; for reference $\mathcal{R}_2$ the range of ending frames at level 2 is $m_{21}(2) \le m \le m_{22}(2)$, etc. We again derive the ending range at the end of level 2 as

$$m_1(2) = \min_{1 \le v \le V} [m_{v1}(2)] \tag{7.21}$$

$$m_2(2) = \max_{1 \le v \le V} [m_{v2}(2)] \tag{7.22}$$

and for each frame in $m_1(2) \le m \le m_2(2)$ we determine the best distance $\bar{D}_2^B(m)$, the reference with the best distance, $\bar{N}_2^B(m)$, and the backpointer $\bar{F}_2^B(m)$.

We can continue the level building procedure through all levels until level $L_{\max}$ in the above manner and we obtain, as the final solution, $D^*$ as

$$D^* = \min_{1 \le \ell \le L_{\max}} [\bar{D}_\ell^B(m)]. \tag{7.23}$$

It should be clear that by performing the computation in levels (i.e., a word at a time) and by doing appropriate minimization within levels, we avoid much of the computation of the two-level DP algorithm described in the previous section. However, the negative feature of the level building algorithm is that the computation is level synchronous, not time synchronous; that is, we can go back to a given test frame at many levels. Hence, real-time hardware implementations of level building are difficult, if not impossible, to

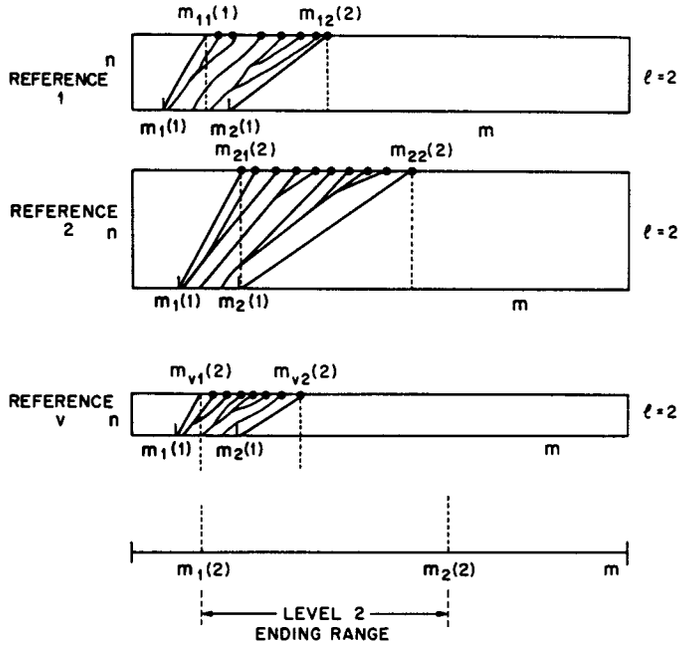



**Figure 7.9** Implementation of level 2 of the level building algorithm (after Myers and Rabiner [4]).

implement. (We will return to these issues later in this section.)

To gain a better understanding of the basic concepts of the level building algorithm, consider the simple example shown in Figure 7.10. Here we assume that the vocabulary consists of two words (which we call $A$ and $B$ for simplicity) and that the two reference patterns, $\mathcal{R}_A$ and $\mathcal{R}_B$, are of equal length. We also assume we are only interested in an $\ell = 4$ level solution. (This greatly simplifies the range of the dynamic programming search parallelogram which, as seen in Figure 7.10, is essentially identical to the case for isolated word template matching. In the next section, where multiple level considerations are discussed, the dynamic programming search range will be shown to be significantly more complicated to specify.) Since both patterns are of equal length, the ending regions for both words, at each level, are identical. Hence at each level ending region we choose the reference pattern ($A$ or $B$) that gave the smallest accumulated distance to that frame. In this simple example, there are 6 ending frames at level 1, with the best path corresponding to $\mathcal{R}_A$ for the first two frames, and $\mathcal{R}_B$ for the next 4 frames. At level 2 there are 10 ending frames; at level 3 there are 6 ending frames, and finally at level 4 there is only one ending frame corresponding to frame $M$, the end of the test utterance. By tracing back the (best)



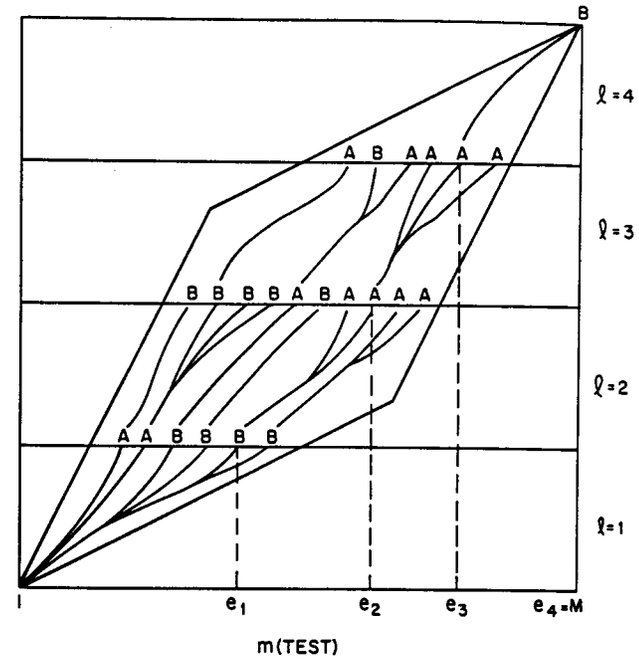



**Figure 7.10** Simple example illustrating level building on two reference patterns of equal length (after Myers and Rabiner [4]).

path ending at $m = M$ we see that the sequence of reference patterns giving the best score is

$$\mathcal{R}^* = \mathcal{R}_B \oplus \mathcal{R}_A \oplus \mathcal{R}_A \oplus \mathcal{R}_B$$

with test frames $e_1, e_2, e_3$, and $e_4 = M$ corresponding to the last frames of the 4 words in the sequence.

### 7.4.2 Multiple Level Considerations

In the more realistic case in which we want the level building solutions for all feasible levels (i.e., where there is a possible solution), there are some very simple techniques that can be used to eliminate unnecessary computation. To understand this issue consider the "standard" warping range of the level building algorithm as shown in Figure 7.11. (We assume here that we use a DTW algorithm with a maximum expansion of 2 and a minimum

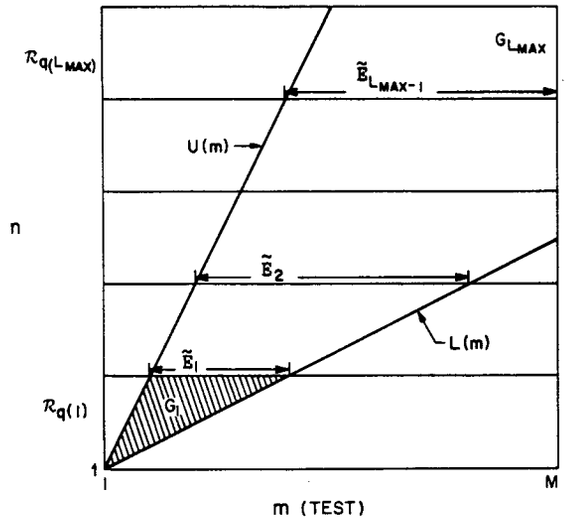



**Figure 7.11** Computation region of level building algorithm for a fixed-length reference pattern (after Myers and Rabiner [4]).

expansion of $^1/_2$.) If we define lower and upper constraints lines as

$$L(m) = (m+1)/2 \tag{7.24}$$

$$U(m) = 2(m-1)+1 \tag{7.25}$$

then for a fixed-length reference pattern we get the computation and ending regions shown in Figure 7.11. (We denote the computation region at level $\ell$ as $G_\ell$ with ending region $\tilde{E}_\ell$.) It should be clear that some of the computation of Figure 7.11 is unnecessary, since there do not exist paths from some of the computed points to ends of reference patterns at any level.

We can use the constraint that in order to do the computation at any grid point, the path from that grid point must be capable of reaching the end of the reference pattern before the end of the test pattern as shown in Figure 7.12. Here we have drawn lines, at each level, of slope 2, from the last frame of the test pattern and the last frame of the reference pattern, and used them as constraints on the grid. We have also drawn a line, at level $L_{\max}$, of slope $^1/_2$ from the last test frame and the last reference frame to further constrain the grid at the last level. The lower and upper constraints of the simplified grid can be described by the equations

$$L(m) = \max\left[\frac{m+1}{2}, 2(m-M)+\theta(\ell)\right] \tag{7.26}$$



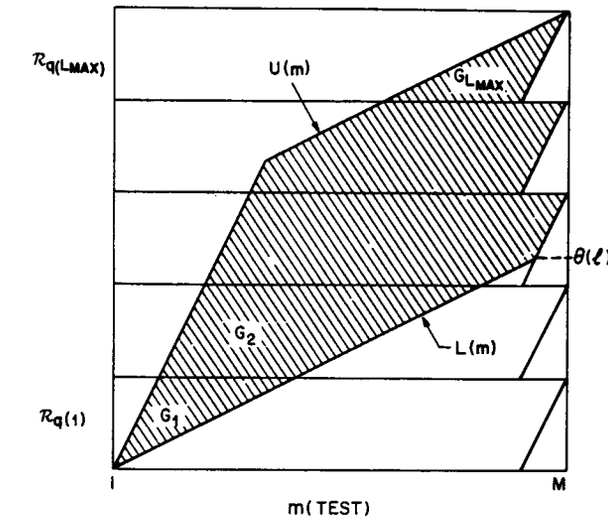



**Figure 7.12** Reduced computation region using upper- and lower-level constraints (after Myers and Rabiner [4]).

$$U(m) = \min\left[2(m-1), \frac{1}{2}(m-M)+\theta(L_{\max})\right] \tag{7.27}$$

in which the reference pattern length function, $\theta(\ell)$, is the accumulated number of frames of the reference patterns used up to level $\ell$ where $(m+1)/2 = 2(m-M)+\theta(\ell)$ at some frame $m < M$. The resulting region of computation is somewhat reduced from that shown in Figure 7.11.

Since the length of each reference pattern is different (in general) the actual computation regions for a level building search based on variable length reference patterns is shown in Figure 7.13. Here we show the computation regions, at each level, for the shortest and for the longest reference patterns. Fundamentally, the pattern of computation is similar to that of Figure 7.11.

### 7.4.3 Computation of the Level Building Algorithm

From the above discussion it should be clear that the basic computation of the level building algorithm is a series of $V$ time warps at each level, where $V$ is the size of the vocabulary. If we assume the maximum number of levels in $L_{\max}$, then we need $V L_{\max}$ time warps. An overbound on size of each time warp is $\overline{N}M/3$ grid points where $\overline{N}$ is the average length of each reference pattern and $M$ is the number of frames in the test pattern. Hence the total

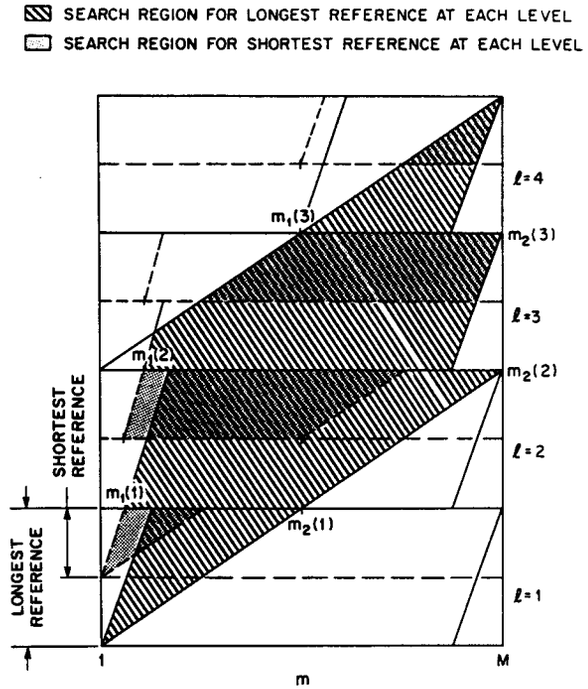



**Figure 7.13** Overall computation pattern of level building algorithm for variable length reference patterns (after Myers and Rabiner [4]).

computation of the level building algorithm is

$$C_{LB} = V \cdot L_{\max} \cdot \overline{N} \cdot M/3 \qquad \text{grid points} \tag{7.28}$$

with storage

$$S_{LB} = 3M \cdot L_{\max} \tag{7.29}$$

since we need storage for $\bar{D}^B$, $\bar{N}^B$, and $\bar{F}^B$ at each value of $m$ and for each level $\ell$.

Using the same values for $M$, $\overline{N}$, and $V$ as used in determining the computation and storage of the two-level DP method, namely $M = 300$, $\overline{N} = 40$, $V = 10$, and using $L_{\max} = 7$, we get $C_{LB} = 280,000$ and $S_{LB} = 6300$. The basic computation of the level building algorithm is a factor of 4.7 less than that of the two-level DP method; the storage of both methods is about the same.

The above calculations are based on full DTWs at each level for each reference pattern. Because of the overlaps of regions in the $(m, n)$ plane, at different levels, some of the assumed computation is redundant. The following exercise illustrates this point.



**Exercise 7.2**

In implementing different levels of the level building algorithm, we have assumed all levels are independent. Thus, at each level, we have performed a full DTW, for each reference pattern, where the size of the DTW was $\overline{N} \cdot M/3$ (grid points). An alternative, and, we hope, more efficient, approach is to realize that for each reference pattern, a significant portion of the computation at each level (namely that of distance computation) may have previously been performed at an earlier level, and therefore only the combinatorics part of the DTW is truly independent at each level.

1. Show that for the assumed parameters of the system (i.e., $M$ = number of test frames = 300, $\overline{N}$ = average number of frames per reference pattern = 40, $V$ = size of vocabulary = 10, $L_{\max}$ = maximum number of levels = 7), the alternative implementation of storing all previously computed distances is more efficient than the standard implementation. (Assume that the cost of the combinatorics in a DTW procedure is negligible.)
2. What is the ratio of computation of the standard implementation to the stored distance implementation?
3. If we assume the cost of the combinatorics at each grid point is $^1/_5$ the cost of a distance computation, how do the results to parts 1 and 2 change?

**Solution 7.2**

1. The simplest way of exploiting previously computed distances is to precompute the entire grid of distances from each reference pattern frame to each test frame. (Clearly this is *not* the most efficient implementation, since many of these full grid distances will never be required in practice.) If we do this we need

$$C_{\text{DIST}} = V \cdot M \cdot \overline{N} \text{ (distances)} = 120{,}000.$$

The number of combinatorics is just the number of grid points used in the level building method; hence

$$C_{\text{COMB}} = V \cdot L_{\max} \cdot \overline{N} \cdot \frac{M}{3} \text{ (grid points)} = 280{,}000.$$

The total computation of this approach is then

$$C_{\text{TOTAL}} = C_{\text{DIST}} + \alpha \cdot C_{\text{COMB}}$$

in which $\alpha$ is the weight for combinatorics. If we assume combinatorics are negligible, $\alpha = 0$, we get

$$C_{\text{TOTAL}} = C_{\text{DIST}} = 120{,}000$$

which is significantly less than the 280,000 distances for the standard implementation (i.e., one per grid point).

2. The ratio of computation of the two approaches is

$$\frac{C_{\text{LB}}}{C_{\text{TOTAL}}} = \frac{V \cdot L_{\max} \cdot \overline{N} \cdot M/3}{V \cdot M \cdot \overline{N}} = \frac{L_{\max}}{3} = 2.33$$

and is independent of $V$, $M$, and $\overline{N}$ but only depends on $L_{\max}$.

3. If we assume the cost of combinatorics is $^1/_5$ the cost of a distance computation we get

$$C_{\text{TOTAL}} = V \cdot M \cdot \overline{N} + V \cdot L_{\max} \cdot \overline{N} \cdot \frac{M}{3}\,\frac{1}{5} = 176{,}000$$

with the ratio in computation being

$$\frac{C_{\text{LB}}}{C_{\text{TOTAL}}} = \frac{V \cdot L_{\max} \cdot \overline{N} \cdot M/3}{V \cdot M \cdot \overline{N} + V \cdot L_{\max} \cdot \overline{N} \cdot \frac{M}{3}\frac{1}{5}} = \frac{L_{\max}}{3(1 + \frac{L_{\max}}{15})} = 1.59$$

### 7.4.4 Implementation Aspects of Level Building

Even though we have already shown that the computation for the level building approach to connected word recognition is significantly less than that of the two-level DP approach of Section 7.2, there are several ways of even further reducing the computational load of the algorithm. These include the following:

1. Beginning range reduction in which we reduce the size of the initial region (range of $m$) for which valid paths to a given level can begin.
2. Global range reduction in which we reduce the size of the region (width of the template) that is tracked, within a level, to determine a best path to each possible level ending frame, $m$.
3. Test pattern ending range in which we *increase* the range over which the global path can match the connected word string; this procedure provides some robustness to ending frame errors.
4. Reference pattern uncertainty ranges in which we increase the range of search at the beginning and end of reference patterns to allow for some degree of word coarticulation at reference word boundaries.

The way in which we implement these features is as follows.

1. **Beginning range reduction—$M_T$**
For the complete level building algorithm, at level $\ell - 1$, we retain the best score, $\bar{D}^B_{\ell-1}(m)$, for each frame $m$ in the ending region $m_1(\ell-1) \le m \le m_2(\ell-1)$. It should be obvious that, in general, the best global path is not at either boundary but somewhere in the middle range of $m$. Hence if we could eliminate some of the ending range at level $\ell - 1$, the search at level $\ell$ would involve less computation. (To see this, consider the limiting case where we make the ending region, at each level, a single frame; then the computation at each new level is a simple DTW with a single initial frame, much as occurs at level 1.)
To determine how to reduce the ending range at level $\ell - 1$ we need to normalize the best accumulated distance scores by the number of frames. Thus we first find the locally best (minimum) normalized accumulated distance as:

$$\phi_{\ell-1} = \min_{m_1(\ell-1) \le m \le m_2(\ell-1)} \left[\frac{\bar{D}^B_{\ell-1}(m)}{m}\right]. \tag{7.30}$$



We now define a reduced-range level threshold as $M_T \cdot \phi_{\ell-1}$ where $M_T$ is a defined parameter, and search the range $m_1(\ell-1) \le m \le m_2(\ell-1)$ to find the indices $S^1_\ell$ and $S^2_\ell$ such that

$$S^1_\ell = \arg \max_{m_1(\ell-1) \le m \le m_2(\ell-1)} \left[\frac{\bar{D}^B_{\ell-1}(m)}{m} > M_T \cdot \phi_{\ell-1} \quad \forall\, m \le S^1_\ell\right] \tag{7.31}$$

$$S^2_\ell = \arg \max_{m_1(\ell-1) \le m \le m_2(\ell-1)} \left[\frac{\bar{D}^B_{\ell-1}(m)}{m} > M_T \cdot \phi_{\ell-1} \quad \forall\, m \ge S^2_\ell\right]. \tag{7.32}$$

To see what is achieved by the beginning range reduction, consider Figure 7.14, which shows a sequence of levels and the resulting ending ranges with and without range reduction (top graph), and the way in which the reduced range is determined (bottom graph). For the level shown, the best normalized distance score, $\phi_{\ell-1}$, is determined as the smallest value of the curve, and the threshold $M_T \cdot \phi_{\ell-1}$ is shown as a level above $\phi_{\ell-1}$ (clearly $M_T \ge 1.0$). The initial reduced range point, $S^1_\ell$, is the last index (beginning from $m = m_1(\ell-1)$) where the curve is always above the threshold; similarly the index $S^2_\ell$ is the last index (beginning from $m = m_2(\ell-1)$) where the curve is always above the threshold. The reduced range means that the computation at the next level is smaller as seen at the top of Figure 7.14. Depending on the value of $M_T$, the reduced range can get smaller (as $M_T \to 1$) or larger (as $M_T \to \infty$). It should be clear that too small a value of $M_T$ will allow the best path to be prematurely eliminated; hence proper choice of $M_T$ is essential.

2. **Global range reduction—$\epsilon$**
The idea behind global range reduction is to reduce the search range along the reference axis, for each test frame, by tracking the global minimum, and allowing only a range around the global minimum. Thus for each test frame, $m$, at each level $\ell$, and for each reference pattern, $\mathcal{R}_v$, we determine the local minimum, $c(m)$, as

$$c(m) = \arg \min_{c(m-1)-\epsilon \le n \le c(m-1)+\epsilon} [D^v_\ell(m-1, n)] \tag{7.33}$$

in which $D^v_\ell(m-1, n)$ is defined to be the best distance at level $\ell$ using reference $\mathcal{R}_v$ at test frame $m-1$ to reference frame $n$ (as determined by the local alignment path) and with $c(1)$ defined to be 1. Figure 7.15 illustrates the global range reduction for a typical level building search. For global range reduction to be effective we must have the reduced range, $2\epsilon + 1$, be smaller than the typical reference pattern width.

3. **Test pattern ending range—$\delta_{END}$**
The idea here is to allow a range of test pattern ending frames, rather than restricting the ending frame to $m = M$. This feature provides a measure of robustness to test pattern endpoint errors. If we extend the end of the test pattern by $\delta_{\text{END}}$ frames, the global level building solution is modified to be

$$D^* = \min_{1 \le \ell \le L_{\max}} \min_{M - \delta_{\text{END}} \le m \le M} [\bar{D}^B_\ell(m)]. \tag{7.34}$$

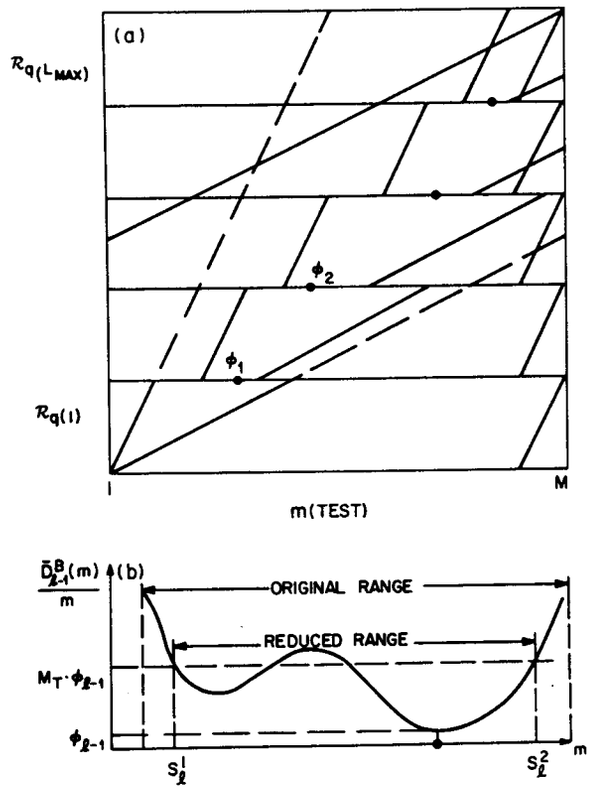



**Figure 7.14** Illustration of beginning range reduction (after Myers and Rabiner [4]).

4. **Reference pattern uncertainty regions**—$\delta_{R_1}, \delta_{R_2}$
   To account for coarticulation of words across word boundaries, the level building algorithm allows a range of beginning and ending frames of the reference pattern. In this manner, at any level, the path can begin over the range $1 \leq n \leq 1+\delta_{R_1}$ (where $n$ is the reference pattern index), and end at any frame in the range $N_v - \delta_{R_2} \leq n \leq N_v$. For appropriate values of $\delta_{R_1}$ and $\delta_{R_2}$, it is possible to have a path which skips $(\delta_{R_1} + \delta_{R_2})$ frames at highly coarticulated word boundaries (e.g., the boundary between six and seven in the string "six-seven"). Figure 7.16 illustrates the use of reference pattern uncertainty regions in level building.

A summary of the four implementation aspects of level building, as discussed in this section, is shown in Figure 7.17 in which all four features are combined in a single



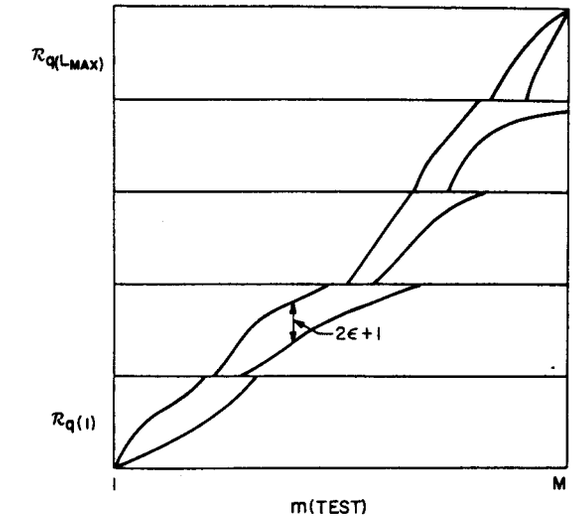



**Figure 7.15** Illustration of global range reduction (after Myers and Rabiner [4]).

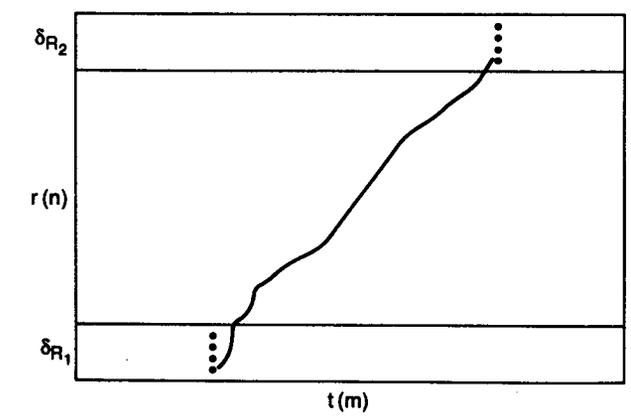



**Figure 7.16** Illustration of the use of reference pattern uncertainty regions (after Myers and Rabiner [4]).

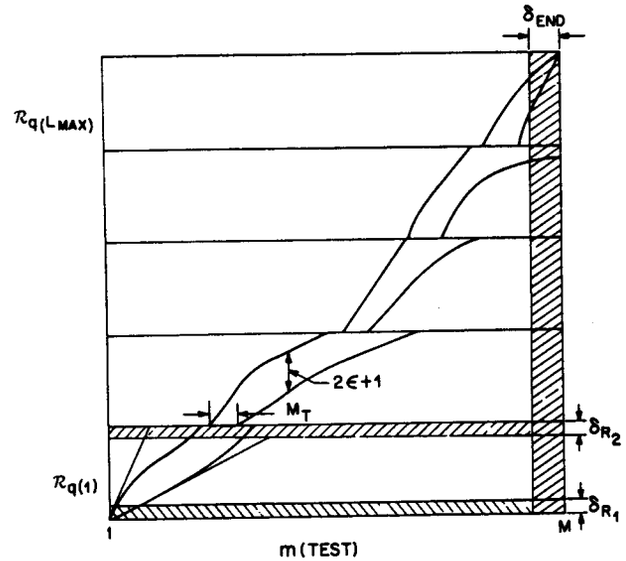



**Figure 7.17** Summary of level building implementation of computation reduction methods (after Myers and Rabiner [4]).

level building search. It can be shown that with judicious choice of the implementation parameters, $M_T$, $\epsilon$, $\delta_{END}$, $\delta_{R_1}$ and $\delta_{R_2}$, the overall computation can be substantially reduced from that of the standard level building approach.

### 7.4.5 Integration of a Grammar Network

We have been implicitly assuming that, for connected word recognition, each word in the string can be followed by any other word in the string. This implicit form of grammar is most appropriate for things like digit strings in which any digit can follow any other digit. However, for some connected word recognition tasks there is an explicit set of rules (grammar) governing which words can logically follow other words to form valid sentences in the language [10]. Although the form of such a grammar can be of several different types (we will discuss this in more detail in Chapter 8), we will restrict ourselves here to those tasks in which we can represent the grammar by a finite state network (FSN) or a finite state automata (FSA) of the form

$$G = A(Q, V, \delta, q_0, Z) \tag{7.35}$$

where

$Q$ = set of states



$V$ = set of vocabulary words

$\delta$ = set of transitions

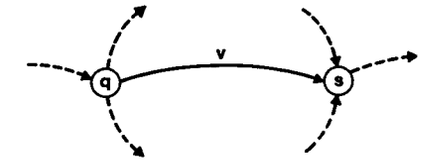
$q_0 \in Q$ = initial state

$Z \subseteq Q$ = set of terminal states

and the set of transitions obeys the rule

$$\delta(q, v) = s \tag{7.36}$$

meaning that word $v$ drives the state from $q$ to $s$



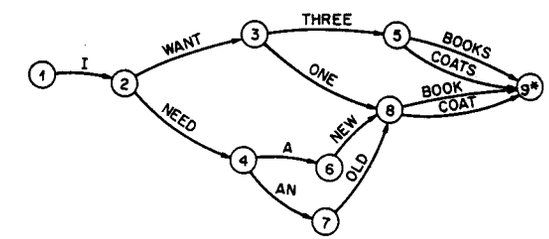

To integrate the FSN grammar network into the following level building algorithm we must do the following:

1. Identify levels with states rather than word position so that word candidates at the $\ell^{th}$ level need not be temporally contiguous to those at the $(\ell + 1)^{st}$ level
2. Partition the vocabulary so that only the reference patterns for words leaving the $\ell^{th}$ state are matched at the $\ell^{th}$ level
3. Retain state backtracking pointers for recovering the best matching string.

(It should be noted that for the most efficient computation, the states in $Q$ should be topologically sorted.)

To illustrate how a simple grammar FSN can be integrated into the LB algorithm, consider the following example:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1. | I | 5. | ONE | 9. | BOOKS | 13. | OLD |
| 2. | WANT | 6. | A | 10. | COAT | | |
| 3. | NEED | 7. | AN | 11. | COATS | | |
| 4. | THREE | 8. | BOOK | 12. | NEW | | |



| Current State | Words Used | Predecessor State | Currect Level | Predecessor Levels |
|---|---|---|---|---|
| 2 | I | 1 | 1 | 0 |
| 3 | WANT | 2 | 2 | 1 |
| 4 | NEED | 2 | 3 | 1 |
| 5 | THREE | 3 | 4 | 2 |
| 6 | A | 4 | 5 | 3 |
| 7 | AN | 4 | 6 | 3 |
| 8 | ONE | 3 | 7 | 2 |
| 8 | NEW | 6 | 8 | 5 |
| 8 | OLD | 7 | 9 | 6 |
| 9* | BOOK, COAT | 8 | 10 | 7,8,9 |
| 9* | BOOKS, COATS | 5 | 11 | 4 |

If we study this simple example, we see that levels 2 and 3 both pick up from level 1; similarly both levels 4 and 7 pick up from level 2. By building up the computation by levels (keeping track of the correct predecessor level), and by backtracking the final result by states (according to the grammar FSN) we can essentially use all the techniques described to efficiently find the best grammatically correct string.

### 7.4.6 Examples of LB Computation of Digit Strings

Figures 7.18 and 7.19 show two examples of connected digit strings matched using the LB algorithm. Figure 7.18 is for the string "51560" and shows the computation building up level by level. For this example, the locally best path at each level (shown by the digit to the right of the last test frame) is actually the globally best digit. At the end of level four the algorithm provided the four best choices with the string "5157" having the lowest score of 0.553. At the end of level 5 there were six string choices with the correct string "51560" having the lowest average accumulated distance of 0.333.

The example in Figure 7.19 is for the string "99211," which did not provide a match nearly as good as the previous example. We see here that the locally best string at each level, namely "90111" is not globally best, and actually is incorrect in two positions. Although the algorithm gets the correct string as the best score, the second best string is the string "901," which has two digit deletions, and one digit-substitution error.

## 7.5 THE ONE-PASS (ONE-STATE) ALGORITHM

The third general approach to the connected word recognition problem is a procedure which was originally proposed by Vintsyuk in 1971 [5] and which has been "rediscovered" several times in the last two decades [6–8] and generalized in scope [9]. The algorithm has been called the one-pass procedure or the one-state algorithm, or most recently, the frame-synchronous level building (FSLB) method. The basic idea behind the algorithm is illustrated in Figure 7.20, which shows a grid with the test pattern, $\mathcal{T}$, mapped to



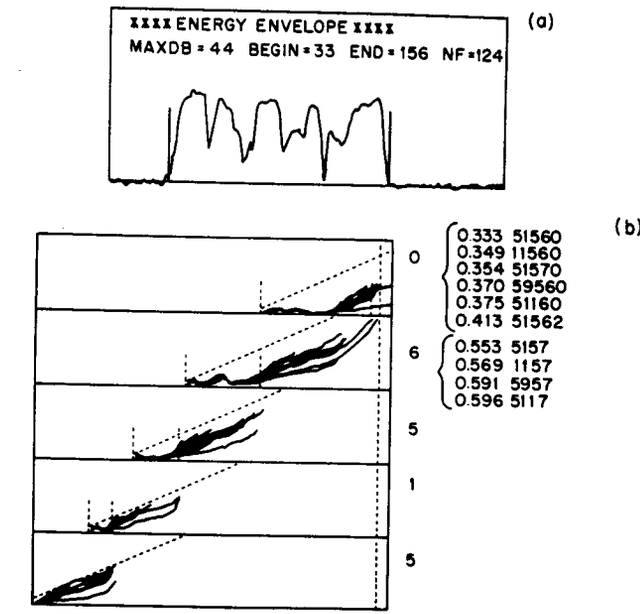



**Figure 7.18** Level building of the string "51560" (after Myers and Rabiner [4]).

the horizontal axis, and the set of reference patterns, $\{\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_V\}$ mapped to the vertical axis.

Using the standard notation of $m$ to represent the test frame index, $1 \le m \le M$, $v$ to represent the reference pattern ($\mathcal{R}_v$) index, $1 \le v \le V$, and $n$ to represent the reference frame index of pattern $\mathcal{R}_v$, $1 \le n \le N_v$, then for each test frame we calculate the accumulated distance, $d_A(m, n, v)$ as:

$$d_A(m, n, v) = d(m, n, v) + \min_{n-2 \le j \le n} (d_A(m-1, j, v)). \tag{7.37}$$

For $2 \le n \le N_v$, $1 \le v \le V$, where $d(m, n, v)$ is the local distance between test frame $\mathbf{t}(m)$ and reference from $\mathbf{r}_v(n)$, and we assume a maximum path expansion of 2 to 1 (hence we search back by 2 reference pattern frames in the combinatoric stage). The recursion of Eq. (7.37) is carried out for all internal frames of each reference pattern (i.e., $n \ge 2$). At the reference pattern boundary, i.e., $n = 1$, we have the simple recursion

$$d_A(m, 1, v) = d(m, 1, v) + \min \left[ \min_{1 \le r \le V} [d_A(m-1, N_r, r)], d_A(m-1, 1, v) \right]. \tag{7.38}$$

Thus the combinatorics for internal and boundary frames are as shown in Figure 7.21. Figure 7.21a shows that for internal frames the combinatorics choose the best internal path

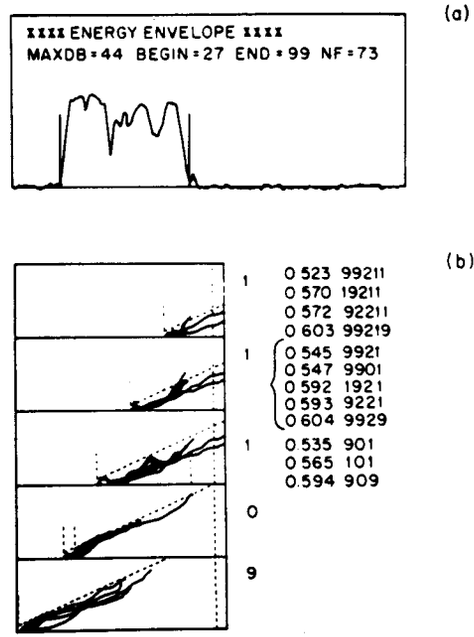



**Figure 7.19** Level building of the string "99211" (after Myers and Rabiner [4]).

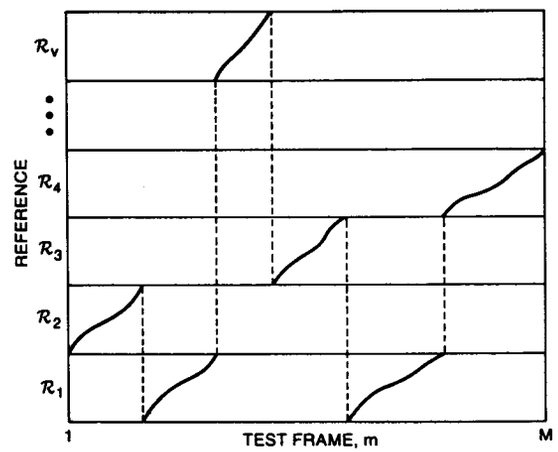



**Figure 7.20** The one-pass connected word recognition algorithm (after Bridle et al. [6]).



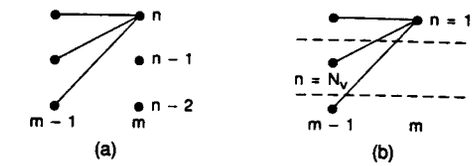



**Figure 7.21** Combinatorics for the one-pass algorithm.

within the reference pattern, whereas at boundary frames the combinatorics choose either a straight (horizontal) path from within the reference pattern (subject to the constraint that the path cannot remain constant for more than one frame) or the best ending frame of any reference pattern. (Within the dynamic programming framework, of course, incorporation of a set of *local constraints* that differ from those of Eq. (7.38) is possible, subject to pragmatic considerations.)

The final solution for the best path (corresponding to the best word string) is

$$D^* = \min_{1 \le v \le V} [d_A(M, N_v, v)]. \tag{7.39}$$

Thus the one-pass algorithm computes best paths to every reference pattern frame at every test frame and eventually is able to backtrack the best score (from Eq. (7.39)) to give the best word sequence, as shown in Figure 7.20.

The major problem with the one-pass algorithm is that no mechanism is provided for controlling the resulting string length—that is, for giving a best path for a string of arbitrary length. The algorithm inherently finds a single best path whose string length is whatever it turns out to be. Thus there is no simple way of exploiting given constraints on string length within the fundamental procedure.

There is, however, a simple and straightforward way of incorporating level (i.e., string-length) constraint in the computation. We do this by extending the accumulated distance to include level information—that is, we extend the recursion to compute the accumulated distance at level $\ell$ as

$$d_A^\ell(m, n, v) = d(m, n, v) + \min_{n-2 \le j \le n} [d_A^\ell(m-1, j, v)] \tag{7.40}$$

where the computation is for $1 \le \ell \le L_{\max}$, $2 \le n \le N_v$, $1 \le v \le V$, $1 \le m \le M$. At each boundary frame the computation now becomes

$$d_A^\ell(m, 1, v) = d(m, 1, v) + \min \left[ \min_{1 \le r \le V} d_A^{\ell-1}(m-1, N_v, r), d_A^\ell(m-1, 1, v) \right] \tag{7.41}$$

with

$$D^* = \min_{1 \le \ell \le L_{\max}} \min_{1 \le v \le V} [d_A^\ell(M, N_v, v)]. \tag{7.42}$$

The key difference is in Eq. (7.41), which only allows a path to an ending frame at level $(\ell - 1)$ to become a path at a beginning frame at level $\ell$.

The main advantage of the one-pass algorithm is that the computation for a given test frame, $m$, can be done frame synchronously; hence the one-pass algorithm is well

suited to real-time implementation on processors that are capable of doing all the necessary computations of Eqs. (7.40) and (7.41) in a single frame interval. Although at first glance it seems as though the computation of the one-pass algorithm is significantly greater than that of the LB approach, it is easily seen that the computation of $d(m, n, v)$ of Eq. (7.40) is independent of level, $\ell$; hence it can be computed once (e.g., at level 1) and stored, and used in subsequent levels with no extra computation. Because of its suitability for real-time implementation the level-based version of the one-pass algorithm is generally the one used for connected word recognition tasks.

## 7.6 MULTIPLE CANDIDATE STRINGS

In the previous sections we have been discussing ways of determining the globally best path through the extended grid, corresponding to the best match to the spoken word string. There are many ways of obtaining multiple candidate strings from which we can determine, at least in theory, the second-best string match, the third-best string match, etc. Multiple string candidates are particularly useful when using grammar networks in order to provide robust recognition decisions. The way in which we obtain multiple candidate strings is simple; namely, we keep track of both the *best* distance, and the *second-best* distance to each ending frame at each level (to get the second-best string match). Since we have computed all reference pattern distances already (as needed to determine the best distance), all that is required is additional storage (to keep track of the array of second-best distances) and the bookkeeping to determine the second best string. (In Chapter 4 we discussed the general problem of using dynamic programming methods to determine the $N$-best paths through a grid. What is described here is essentially what was called the parallel algorithm in Chapter 4, Section 4.7.5. The main difference is that here the path ranking is in terms of word candidate strings instead of frame-based time warping paths. This difference gives rise to some extra optimality considerations as discussed in this section.) We illustrate the procedure in Figure 7.22, which shows a simple two-level LB search with tracking of the two best strings at each grid point. At the end of level 2 there are two distinct paths to the last frame of the test pattern—namely, the best distance (labeled 1 and shown as a solid path) and the second best distance (labeled 2 and shown as a dashed line). The best path from level 2 branches off to two other paths in level 1, with one being a best path and the other being a second-best path. Similarly, the second-best path from level 2 branches off to two other paths in level 1. Thus a total of four paths are followed—namely, the 11 path (best from level 2, best from level 1), the 12 path (best from level 2, second best from level 1), the 21 path (second best from level 2, best from level 1 to the beginning point of the level 2 path) and the 22 path (second best from level 2, second best from level 1). The overall best path is, by definition, the 11 path through the grid. The second-best path, according to this method, is (with some pathological exceptions, to be explained next) either the 12 path, or the 21 path; the 22 path cannot ever be as good as the 21 path.

The procedure described above can be extended trivially to the best three candidates at each level, in which case a total of $3^L$ scores are obtained after $L$ levels. This is illustrated at the bottom of Figure 7.22 for a two-level match in which there are nine string candidates.



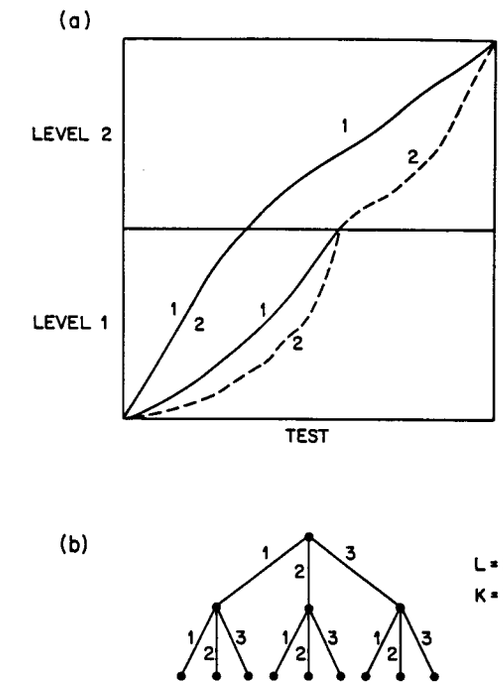



**Figure 7.22** Description of procedure for determining multiple candidate scores (after Myers and Rabiner [4]).

Figure 7.23 shows a case of using $L = 4$ levels with two best candidates. There are now four possible choices for the second-best string, including the 1112 path, the 1121 path, the 1211 path and the 2111 path. A sorting algorithm can be used to order the $K^L$ paths obtained when keeping track of $K$ candidates at $L$ levels.

It should be noted that the procedure described above, implemented using the level building algorithm, *cannot* guarantee that the computed "second-best" string is actually the true second-best string. This is because in keeping track of multiple strings to any ending frame the procedure inherently requires that the candidate strings come from different reference patterns. (Recall that in the LB algorithm, partial word decisions are made at each level before reaching the end of the test pattern.) The real requirement should be that they come from different overall strings (i.e., any word in the built-up strings can be different, not just the immediate last word). Hence, in theory, the two best paths to a given ending frame could indeed be from the same reference pattern. Figure 7.24 [10] illustrates the level building flaw via an example in which the true second-best string is the sequence

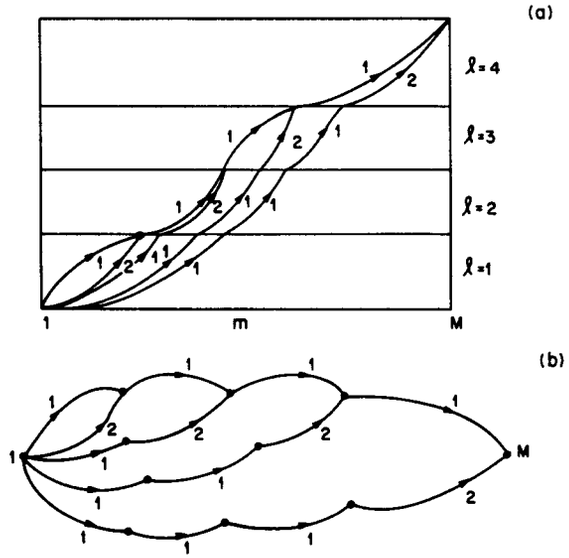



**Figure 7.23** Candidate strings for a four-level search (after Myers and Rabiner [4]).

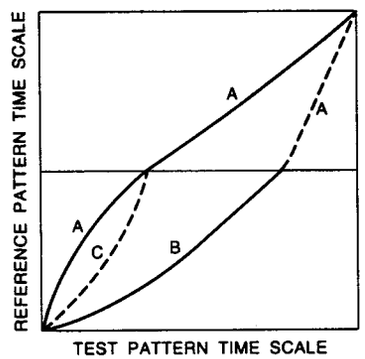



**Figure 7.24** Illustration of level building flaw for determining the second-best candidate string (after Myers and Rabiner [10]).



*BA* (i.e., reference pattern *B* followed by *A*); however, since the best string match is *AA*, the reference pattern *A* is not allowed as the second candidate to the last frame at level 2. This situation occurs extremely infrequently; however, it can and does occur from time to time.

## 7.7 SUMMARY OF CONNECTED WORD RECOGNITION ALGORITHMS

We have presented three approaches to solving the "connected word recognition" problem—namely, the two-level DP algorithm, the LB method, and the (frame synchronous) one-pass method. The algorithms all are fundamentally identical in that they provide the identical best matching string with the identical matching score. Basically the algorithms differ in computational efficiency, storage requirements, and ease of realization in real-time hardware.

Although we have concentrated primarily on word template patterns, it is easy to see that the basic procedures are virtually identical for statistical models like HMMs. To see this, consider the case of level building using $N$-state HMMs instead of $N_v$-frame reference patterns. If we denote the test frame index as $t$, $1 \le t \le T$, (rather than $n$ as is conventionally done for HMMs), and we denote the test frame vector as $\mathbf{o}_t$ (rather than $\mathbf{t}_n$ as we have done throughout this chapter), then the local log likelihood for state $j$ of reference model $\lambda^v$ is (for an $M$-mixture density)

$$b_j^v(\mathbf{o}_t) = \log\left[\sum_{m=1}^{M} c_m \prod_{d=1}^{D} e^{-(\mathbf{o}_t(d)-\mu_{jm}(d))^2/2\,\mathrm{U}_{jm}(d)}\right]. \tag{7.43}$$

The level building computation is therefore a calculation of $P_\ell^v(t)$, $1 \le t \le T$, $1 \le v \le V$, $1 \le \ell \le L_{\max}$, the accumulated log likelihood to frame $t$, at level $\ell$, for reference model $\lambda^v$ along the best path along with $F_\ell^v(t)$, the backpointer indicating where the best path started at the beginning of the level. At the end of each level we compute the "level best" scores as

$$P_\ell^B(t) = \max_{1\le v\le V} P_\ell^v(t), \qquad 1 \le t \le T \tag{7.44}$$

$$N_\ell^B(t) = \arg\max_{1\le v\le V} P_\ell^v(t), \qquad 1 \le t \le T \tag{7.45}$$

$$F_\ell^B(t) = F_\ell^{N_\ell^B(t)}(t), \qquad 1 \le t \le T \tag{7.46}$$

with solution

$$P^* = \max_{1\le \ell\le L_{\max}} [P_\ell^B(T)]. \tag{7.47}$$

Figure 7.25 illustrates the grid for the level building computation for a set of $N$-state HMMs. The regularity of the grid (the trellis shape) is due to the lack of constraint about remaining in a state for more than one or two frames for statistical models.

Finally it should be clear that all three algorithms are equally amenable to inclusion (integration) of an FSN grammar network to constrain allowable word sequences. Consider

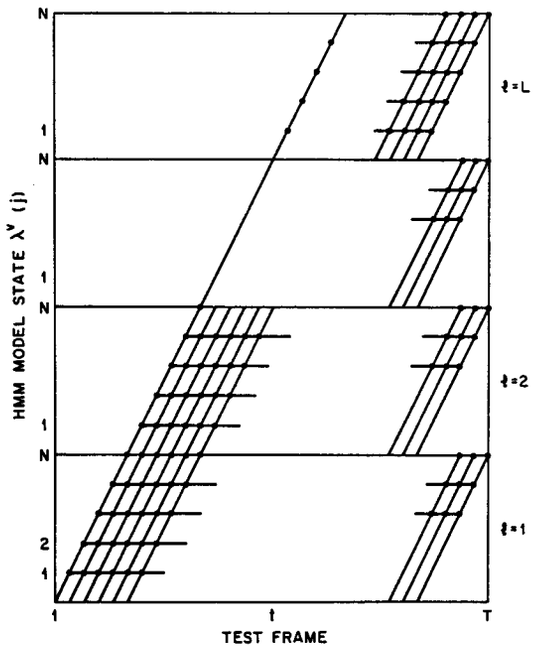



**Figure 7.25** Use of HMMs in the level building procedure.

an arbitrary FSN grammar network with a typical node, $g$, as shown in Figure 7.26. The input to the grammar node is a series of word arcs (corresponding to words in the vocabulary) from predecessor grammar nodes $i-1, i$ and $i+1$. Since all these inputs merge at grammar node $g$, the basic computation at this grammar node is to determine the maximum likelihood path over the set, $P(g)$, (which constitutes the paths of all words coming into $g$), and to propagate this path to all successor nodes, namely $j-1, j, j+1$ through the appropriate word arcs. The grammar node computation is iterated over all nodes in the grammar in an organized fashion (so that all computation necessary for node $g$ is done before considering node $g$).

The way in which computation for the grammar FSN is integrated into the connected word algorithm is shown in Figure 7.27. For each test frame, $t$, the spectral vector is computed, and then the local likelihood (or distance) scores are computed for every reference pattern state (or frame). In parallel the local combinatorics (within reference patterns) are performed (the local distance scores are added at the end of the computation). The grammar network scores (corresponding to word transitions) are then computed and the procedure is iterated until the last test frame, at which point a backtracker is used to determine the best matching string.



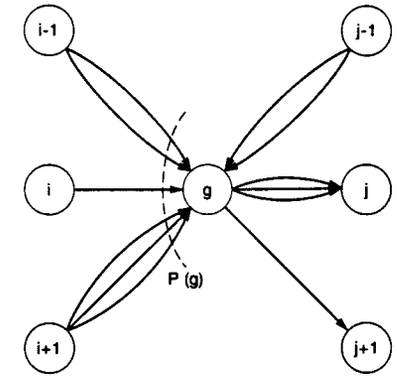



**Figure 7.26** A typical grammar node of an FSN grammar network (after Lee and Rabiner [9]).

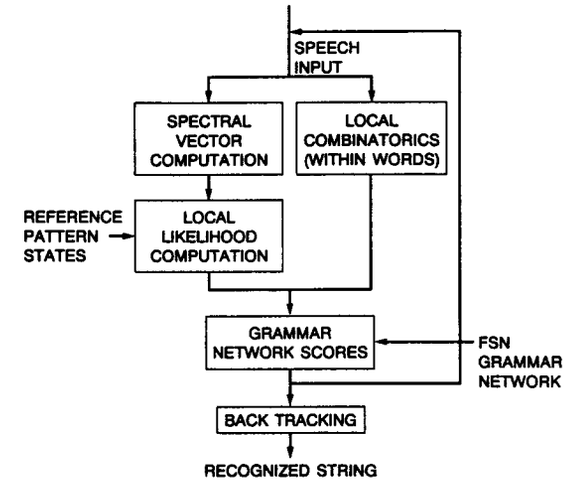



**Figure 7.27** Block diagram of connected word recognition computation.

## 7.8 GRAMMAR NETWORKS FOR CONNECTED DIGIT RECOGNITION

One of the most important applications of connected word recognition is connected digit recognition because of its potential application to credit card entry, all-digit dialing of telephone numbers, personal identification number (PIN) entry, catalog ordering, and so on.

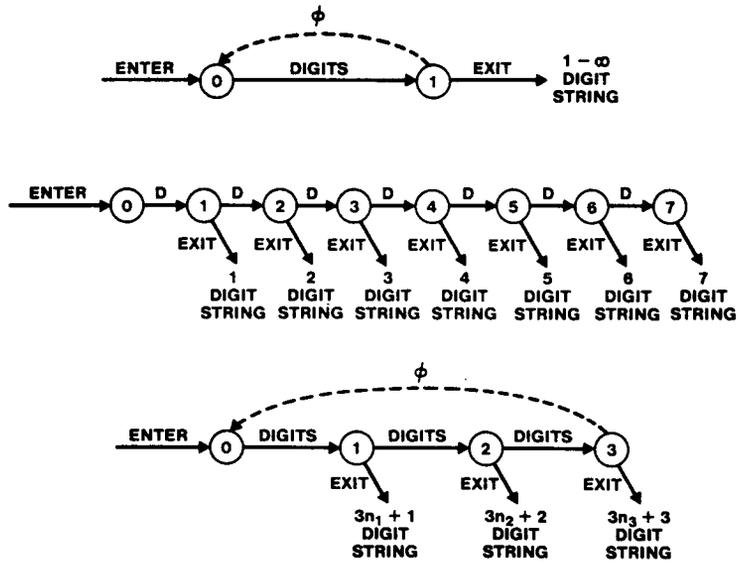



**Figure 7.28** Three possible grammar networks for connected digit recognition (after Lee and Rabiner [9]).

Thus in this and the next section we will concentrate on aspects of this important application, including forms of the grammar network that are often used, and some implementation and performance aspects of current systems.

Figure 7.28 shows three general grammar networks for connected digit strings. The network of part (a) is the simple search of the one-pass algorithm *without* level information. Thus the network will always find the best string, but cannot control the string length, and therefore is unusable for known string length tasks (e.g., choosing the best seven-digit string to represent a spoken telephone number).

The network of Figure 7.28, part b explicitly breaks out digit strings of length one to seven digits, and therefore corresponds to the level building approach with $L_{max} = 7$. As we have seen, the local combinatorics search requires about 7 times the computation of the simple grammar network of part a; however, local distance computation remains the same.

The network of Figure 7.28, part c represents a reasonable compromise between the networks of parts a and b in that it breaks out three distinct levels corresponding to string lengths of $3n_1 + 1$, $3n_2 + 2$, and $3n_3 + 3$ digits where $n_1$, $n_2$ and $n_3$ are arbitrary integers. The idea here is that if the input is an $n$-digit string, the most likely errors are single-digit insertions or deletions, at which point one of the three outputs will most likely not have the single insertion or deletion, and will be the correct string. The computation is only three times the combinatorics search of part a, with again the same computation



for local distances.

## 7.9 SEGMENTAL K-MEANS TRAINING PROCEDURE

The discussion in the previous sections focused primarily on efficient and optimal solutions to the so-called decoding problem, in which the likelihood (or distance) of a given speech pattern (the unknown test pattern) corresponding to a *string of words* is evaluated. Of equal importance is the connected word training problem in which the object is to derive appropriate word reference patterns or models from a labeled training set of many connected word sequences. The major problem here is the lack of an exact (precise) correspondence between speech segments and the spoken words they correspond to. One could consider manually segmenting and labeling each spoken training utterance into the individually spoken words of each string. However, this process is a tedious one that is well known to be error prone because of inconsistencies in determination of the exact boundaries between adjacent words in the string. Hence what is required for connected word training is a fully automatic procedure for both segmentation of a connected word string into individual words, and model (pattern) training from the segmented strings. Such an algorithm is described in this section.

The way in which the individual digit models are derived from connected word training strings is a procedure called the segmental $k$-mean training procedure, which is a straightforward variation on the well-known $k$-means iteration (e.g., as used for vector quantizer design) [11–13]. The basic idea is to have a training set of labeled connected digit strings, and an initial set of digit models (e.g., isolated digit models). (The procedure to be described works even without an initial set of digit models, but we will not describe how this is accomplished here.)

The segmental $k$-means training procedure (as shown in Figure 7.29) works as follows (when used for training HMMs):

1. Given the initial model (the set of word pattern files) and the (labeled) training files, any of the connected word recognition procedures is used to segment each training string into individual digit tokens which are stored in appropriate files according to the identity of the digits. (This is the segmentation phase into words.)
2. Each file of word tokens (e.g., the file for all the 1s in the training set) is then segmented into states and within each state the parameters of the mixture density (the mixture weights, means, and covariances) are determined using a standard VQ clustering procedure. The result of this procedure (which we call the word pattern building algorithm) is an updated set of word models.
3. A test for convergence is made, either based on a set of testing files or based on the likelihood scores of the training set files. If the convergence test shows continuing improvement in performance, the procedure is iterated (i.e., steps 1 and 2 are repeated using the updated set of word models); otherwise the procedure is terminated and the updated set of models is the final set of models.

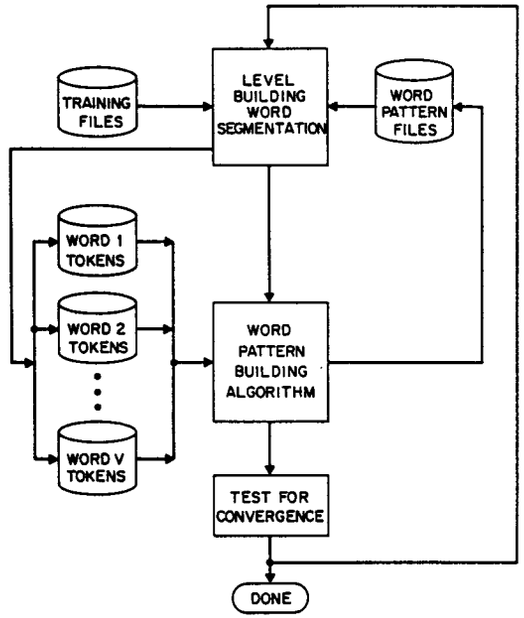



**Figure 7.29** The segmental $k$-means training algorithm for connected word strings (after Rabiner et al. [13]).

## 7.10 CONNECTED DIGIT RECOGNITION IMPLEMENTATION

A block diagram of a canonic system for connected digit recognition is shown in Figure 7.30. There are three basic steps in the recognition process, namely:

1. **Spectral analysis**, in which the speech signal, $s(n)$, is converted to an appropriate spectral representation, e.g., filter-bank vector, LPC-based vector, ear model vector.
2. **Connected word pattern matching**, in which the sequence of spectral vectors of the unknown (test) connected digit string is matched against whole word (single digit) patterns using any of the algorithms discussed in this chapter. The output of this process is a set of candidate digit strings, generally of different lengths, ordered by distance (likelihood, probability) score.
3. **Postprocessing**, in which the candidate digit strings are subjected to further processing (e.g., based on digit durations, word stress, etc.) so as to eliminate unreasonable (unlikely) candidates. The postprocessor chooses the most likely digit string from the ordered list of candidates which passed the postprocessor tests.



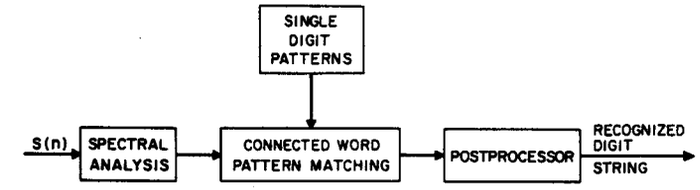



**Figure 7.30** Block diagram of connected digit recognition method (after Rabiner et al. [13]).

In the remainder of this section we present an overview of the techniques that are currently used to provide the best performance (highest string accuracy) on this task.

### 7.10.1 HMM-Based System for Connected Digit Recognition

The system that provides the highest reported string accuracy on a standard testing set is one based on LPC cepstral analysis and HMMs. In particular, the spectral analysis uses an LPC front end with the following characteristics:

- sampling rate—6.67 kHz
- analysis window size—300 samples (45 msec)
- analysis window shift—100 samples (15 msec)
- LPC order—8
- cepstrum order—12
- delta cepstrum order—12
- delta-delta (second difference) cepstrum order—12
- cepstral window—raised, sinelike window

The observation vector used was a 38-component vector consisting of 12 cepstral coefficients, 12 delta cepstrum coefficients, 12 delta-delta cepstral coefficients, delta log energy, and delta-delta log energy. (The log energy was used directly in the evaluation of model likelihoods on the basis of measured histograms, rather than as a component of the observation vector.)

The hidden Markov models used for each digit model were left-to-right models of the type shown in Figure 7.31. Each model had $N$ states ($N$ varied from 5 to 10 for different digits), and within states a continuous mixture density was used to characterize the observation vector, where the number of mixture components per state was as few as 3 (for speaker-trained models) and as many as 64 (for speaker independent models). In addition to the spectral density, an empirically derived log energy probability density (e.g., a histogram) was used within each state (with appropriate weighting), as well as an empirically derived state duration density. For the postprocessor a single Gaussian digit duration density was used based on the measured mean duration and variance from the training set.

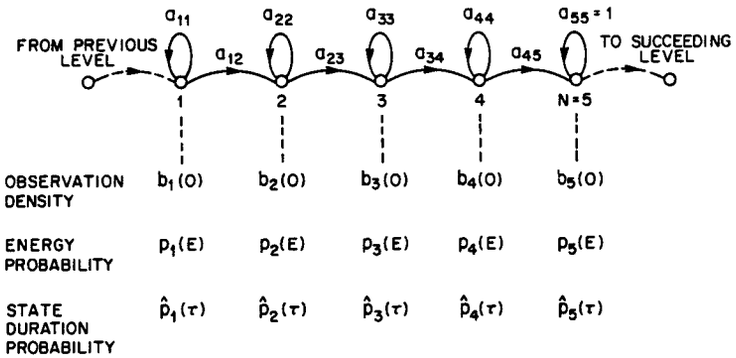



**Figure 7.31** Connected digit HMM (after Rabiner et al. [13]).

### 7.10.2 Performance Evaluation on Connected Digit Strings

To evaluate the performance of connected digit recognition algorithms ([13–15]), two databases have been used:

1. **DB50**, consisting of 50 adult talkers (25 male, 25 female), from the local, nontechnical population of Murray Hill, New Jersey. Each talker provided from 600 to 1150 digit strings, for a total of 47,336 strings. The digit strings were recorded off of local, dialed-up, telephone lines (100-3200 Hz bandwidth), and were variable in length from 1 to 7 digits. Within each string the selection of digits was random from the set of zero to nine (oh was excluded). All digit strings were spoken fluently with no pauses.
2. **TI Set**, consisting of a training set of 112 talkers (55 male, 57 female) with 22 regional accents, and a testing set of 113 additional talkers (56 male, 57 female) from the same 22 regions with no overlap between training and testing set talkers. Each talker spoke 77 connected digit strings of length 1–5 or 7 digits. The input provided was wideband (100–7000 Hz) but was lowpass filtered to telephone bandwidth at Bell Labs for compatibility with the DB50 set. There was a random selection of digits within each string from the set of 11 digits including both zero and oh. Even though talkers were requested to speak each string fluently with no pauses, a few strings had internal pauses.

The first data base, DB50, was used for tests in a speaker-trained and a multispeaker mode, whereas the second database, TI Set, was used for speaker-independent tests. To contrast the spoken material in the two sets, Figure 7.32 shows a plot of the average speaking rate (measured in words per minute) of the two datasets as a function of the number of digits in the string. It can be seen that the speaking rate of the TI talkers is somewhat lower than that of the DB50 talkers (by from 10-20 wpm).



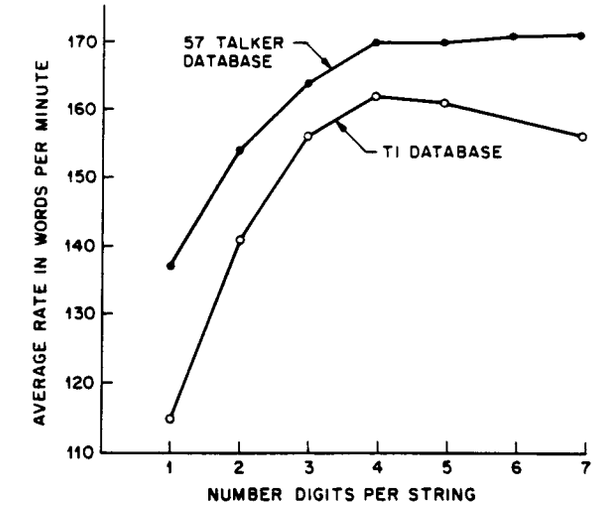



**Figure 7.32** Average speaking rate of talkers in the two connected digit databases as a function of the number digits per string (after Rabiner et al. [13]).

The conditions used in the performance evaluations were as follows (when used for training HMMs):

1. **Speaker-Trained Mode (50 Talkers)**—For each talker, half the strings (randomly selected) were used for training, the other half for testing. A single HMM per digit was used. (For this test the observation vector did not use either energy or delta-delta cepstral components.)
2. **Multispeaker Mode (50 Talkers)**—One-fourth of the training set, as in the speaker-trained mode, was used for training 6 models per digit (using clustering techniques); the same testing set was used as in the speaker trained mode. (Again the delta-delta cepstral components or energy were not used in the observation vectors.)
3. **Speaker-Independent Mode (225 Talkers)**—The specified training and testing sets were used to create a single HMM per digit (based on the 38 parameter observation vectors). No silence model was created to account for pauses within digit strings.

The resulting performance of the connected digit recognizer is shown in Table 7.1. For each of the three testing conditions, results are given for self-test (namely, testing on the training data) and for the true testing set, in terms of average *string* error rates (%) for unknown length (UL) strings (i.e., allowing insertions and deletions as well as substitutions), as well as for known length (KL) strings (i.e., only allowing strings of the correct length). Performance in all three modes is comparable with string error rates around 1–3% for unknown length strings, and 0.4–1.7% for known length strings. (Performance

**TABLE 7.1.** Average String Error Rates (%) for Connected Digit Recognition Tests

| Mode | Training Data | | Testing Data | |
|---|---|---|---|---|
| | UL | KL | UL | KL |
| Speaker Trained | 0.4 | 0.16 | 0.8 | 0.35 |
| Multi-Speaker | 1.7 | 1.0 | 2.85 | 1.65 |
| Speaker Independent | 0.3 | 0.05 | 1.4 | 0.8 |

scores for speaker-trained and multispeaker modes are underbounds, since the tests were not performed with the extended observation vector used in the speaker-independent mode tests.)

## 7.11 SUMMARY

In this chapter we have shown how the results presented in earlier chapters on time alignment of single words and phrases can be extended to the problem of aligning an input consisting of a sequence of words to a concatenation of individual word patterns. We have shown how an optimal matching can be achieved using one of several different procedures, and we have also shown how the basic ideas can be applied to either templates or statistical models. By way of example, we showed how the techniques described have been applied to the problem of recognizing a fluently spoken string of digits with high performance.

## REFERENCES

[1] H. Sakoe, "Two-Level DP Matching—A Dynamic Programming-Based Pattern Matching Algorithm for Connected Word Recognition," *IEEE Trans. Acoustics, Speech, Signal Proc.*, ASSP-27 (6): 588–595, December 1979.

[2] H. Sakoe, "A Generalized Two-Level DP-Matching Algorithm for Continuous Speech Recognition," *Trans. of the IEE of Japan*, E65 (11): 649–656, November 1982.

[3] C.S. Myers and L.R. Rabiner, "A Level Building Dynamic Time Warping Algorithm for Connected Word Recognition," *IEEE Trans. Acoustics, Speech, Signal Proc.*, ASSP-29 (2): 284–297, April 1981.

[4] C.S. Myers and L.R. Rabiner, "Connected Digit Recognition Using a Level-Building DTW Algorithm," *IEEE Trans. Acoustics, Speech, Signal Proc.*, ASSP-29 (3): 351–363, June 1981.

[5] T.K. Vintsyuk, "Element-Wise Recognition of Continuous Speech Consisting of Words From a Specified Vocabulary," *Kibernetika* (Cybernetics), No. 2: 133–143, March–April 1971.

[6] J.S. Bridle, M.D. Brown, and R.M. Chamberlain, "An Algorithm for Connected Word Recognition," *Proc. ICASSP 82*, Paris, 899–902, May 1982.



[7] J.S. Bridle, M.D. Brown, and R.M. Chamberlain, "Continuous Connected Word Recognition Using Whole Word Templates," *The Radio and Electronic Engineer*, 53 (4): 167–175, April 1983.

[8] H. Ney, "The Use of a One-Stage Dynamic Programming Algorithm for Connected Word Recognition," *IEEE Trans. Acoustics, Speech, Signal Proc.*, ASSP-32 (2): 263–271, April 1984.

[9] C.H. Lee and L.R. Rabiner, "A Frame-Synchronous Network Search Algorithm for Connected Word Recognition," *IEEE Trans. Acoustics, Speech, Signal Proc.*, 37 (11): 1649–1658, November 1989.

[10] C.S. Myers and L.R. Rabiner, "A Comparative Study of Several Dynamic Time-Warping Algorithms for Connected-Word Recognition," *Bell System Tech. J.*, 60 (7): 1389-1409, September 1981.

[11] B.H. Juang and L.R. Rabiner, "The Segmental *K*-Means Algorithm for Estimating Parameters of Hidden Markov Models," *IEEE Trans. Acoustics, Speech, Signal Proc.*, 38 (9): 1639–1641, September 1990.

[12] L.R. Rabiner, J.G. Wilpon, and B.H. Juang, "A Segmental *K*-Means Training Procedure for Connected Word Recognition," *AT&T Tech. J.*, 64 (3): 21–40, May 1986.

[13] L.R. Rabiner, J.G. Wilpon, and B.H. Juang, "A Model-Based Connected Digit Recognition System Using Either Hidden Markov Models or Templates," *Computer Speech, and Language*, 1 (2): 167–197, December 1986.

[14] L.R. Rabiner, J.G. Wilpon., and F.K. Soong, "High Performance Connected Digit Recognition Using Hidden Markov Models," *IEEE Trans. Acoustics, Speech, Signal Proc.*, 37 (8): 1214–1225, August 1989.

[15] J.G. Wilpon, C.H. Lee, and L.R. Rabiner, "Improvements in Connected Digit Recognition Using Higher Order Spectral and Energy Features," *Proc. ICASSP 91*, Toronto, Ontario, Canada, May 1991.

Chapter 8

# LARGE VOCABULARY CONTINUOUS SPEECH RECOGNITION

## 8.1 INTRODUCTION

Throughout this book we have developed a wide range of tools, techniques, and algorithms for attacking several fundamental problems in speech recognition. In the previous chapter we saw how the different techniques came together to solve the connected word recognition problem. In this chapter we extend the concepts to include issues needed to solve the large vocabulary, continuous speech recognition problem. We will see that the fundamental ideas need modification because of the use of subword speech units; however, a great deal of the formalism for recognition, based on word units, is still preserved.

The standard approach to large vocabulary continuous speech recognition is to assume a simple probabilistic model of speech production whereby a specified word sequence, $W$, produces an acoustic observation sequence $Y$, with probability $P(W, Y)$. The goal is then to decode the word string, based on the acoustic observation sequence, so that the decoded string has the maximum a posteriori (MAP) probability, i.e.,

$$\hat{W} \ni P(\hat{W}|Y) = \max_W P(W|Y). \tag{8.1}$$

Using Bayes' Rule, Equation (8.1) can be written as

$$P(W|Y) = \frac{P(Y|W)P(W)}{P(Y)}. \tag{8.2}$$





Since $P(Y)$ is independent of $W$, the MAP decoding rule of Eq. (8.1) is

$$\hat{W} = \arg\max_W P(Y|W)P(W). \tag{8.3}$$

The first term in Eq. (8.3), $P(Y|W)$, is generally called the acoustic model, as it estimates the probability of a sequence of acoustic observations, conditioned on the word string. The way in which we compute $P(Y|W)$, for large vocabulary speech recognition, is to build statistical models for subword speech units, build up word models from these subword speech unit models (using a lexicon to describe the composition of words), and then postulate word sequences and evaluate the acoustic model probabilities via standard concatenation methods. Such methods are discussed in Sections 8.2–8.4 of this chapter.

The second term in Eq. (8.3), $P(W)$, is generally called the language model, as it describes the probability associated with a postulated sequence of words. Such language models can incorporate both syntactic and semantic constraints of the language and the recognition task. Often, when only syntactic constraints are used, the language model is called a grammar and may be of the form of a formal parser and syntax analyzer, an $N$-gram word model ($N = 2, 3, \ldots$), or a word pair grammar of some type. Generally such language models are represented in a finite state network so as to be integrated into the acoustic model in a straightforward manner. We discuss language models further in Section 8.5 of this chapter.

We begin the chapter with a discussion of subword speech units. We formally define subword units and discuss their relative advantages (and disadvantages) as compared to whole-word models. We next show how we use standard statistical modeling techniques (i.e., hidden Markov models) to model subword units based on either discrete or continuous densities. We then show how such units can be trained automatically from continuous speech, without the need for a bootstrap model of each of the subword units. Next we discuss the problem of creating and implementing word lexicons (dictionaries) for use in both training and recognition phases. To evaluate the ideas discussed in this chapter we use a specified database access task, called the DARPA Resource Management (RM) task, in which there is a word vocabulary of 991 words (plus a silence or background word), and any one of several word grammars can be used. Using such a system, we show how a basic set of subword units performs on this task. Several directions for creating subword units which are more specialized are described, and several of these techniques are evaluated on the RM task. Finally we conclude the chapter with a discussion of how task semantics can be applied to further constrain the recognizer and improve overall performance.

## 8.2 SUBWORD SPEECH UNITS

We began Chapter 2 with a discussion of the basic phonetic units of language and discussed the acoustic properties of the phonemes in different speech contexts. We then argued that the acoustic variability of the phonemes due to context was sufficiently large and not well understood, that such units would not be useful as the basis for speech models for recognition. Instead, we have used whole-word models as the basic speech unit, both for

isolated word recognition systems and for connected word recognition systems, because whole words have the property that their acoustic representation is well defined, and the acoustic variability occurs mainly in the region of the beginning and the end of the word. Another advantage of using whole-word speech models is that it obviates the need for a word lexicon, thereby making the recognition structure inherently simple.

The disadvantages of using whole-word speech models for continuous speech recognition are twofold. First, to obtain reliable whole-word models, the number of word utterances in the training set needs to be sufficiently large, i.e., each word in the vocabulary should appear in each possible phonetic context several times in the training set. In this way the acoustic variability at the beginning and at the end of each word can be modeled appropriately. For word vocabularies like the digits, we know that each digit can be preceded and followed by every other digit; hence for an 11-digit vocabulary (zero to nine plus oh), there are exactly 121 phonetic contexts (some of which are essentially identical). Thus with a training set of several thousand digit strings, it is both realistic and practical to see every digit in every phonetic context several times. Now consider a vocabulary of 1000 words with an average of 100 phonetic contexts for both the beginning and end of each word. To see each word in each phonetic context exactly once requires $100 \times 1000 \times 100 = 10$ million carefully designed sentences. To see each combination 10 times requires 100 million such sentences. Clearly, the recording and processing of such homogeneous amounts of speech data is both impractical and unthinkable. Second, with a large vocabulary the phonetic content of the individual words will inevitably overlap. Thus storing and comparing whole-word patterns would be unduly redundant because the constituent sounds of individual words are treated independently, regardless of their identifiable similarities. Hence some more efficient speech representation is required for such large vocabulary systems. This is essentially the reason we use subword speech units.

There are several possible choices for subword units that can be used to model speech. These include the following:

- **Phonelike units** (PLUs) in which we use the basic phoneme set (or some appropriately modified set) of sounds but recognize that the acoustic properties of these units could be considerably different than the acoustic properties of the "basic" phonemes [1–7]. This is because we define the units based on linguistic similarity but model the unit based on acoustic similarity. In cases in which the acoustic and phonetic similarities are roughly the same (e.g., stressed vowels) then the phoneme and the PLU will be essentially identical. In other cases there can be large differences and a simple one-to-one correspondence may be inadequate in terms of modeling accuracy. Typically there are about 50 PLUs for English.
- **Syllable-like units** in which we again use the linguistic definition of a syllable (namely a vowel nucleus plus the optional initial and final consonants or consonant clusters) to initially define these units, and then model the unit based on acoustic similarity. In English there are approximately 10,000 syllables.
- **Dyad or demisyllable-like units** consisting of either the initial (optional) consonant cluster and some part of the vowel nucleus, or the remaining part of the vowel nucleus and the final (optional) consonant cluster [8]. For English there is something on the order of 2000 demisyllable-like units.
- **Acoustic units**, which are defined on the basis of clustering speech segments from a segmentation of fluent, unlabeled speech using a specified objective criterion (e.g., maximum likelihood) [9]. Literally a codebook of speech units is created whose interpretation, in terms of classical linguistic units, is at best vague and at worst totally nonexistent. It has been shown that a set of 256–512 acoustic units is appropriate for modeling a wide range of speech vocabularies.



Consider the English word *segmentation*. Its representation according to each of the above subword unit sets is

- **PLUs**: /s/ /ɛ/ /g/ /m/ /ə/ /n/ /t/ /eʸ/ /sh/ /ə/ /n/ (11 units)
- **syllables**: /seg/ /men/ /ta/ /tion/ (4 syllables)
- **demisyllables**: /sɛ/ /ɛg/ /mə/ /ən/ /teʸ/ /eʸsh/ /shə/ /ən/ (8 demisyllables)
- **acoustic units**: 17 111 37 3 241 121 99 171 37 (9 acoustic units).

We see, from the above example, that the number of subword units for this word can be as small as 4 (from a set of 10,000 units) or as large as 11 (from a set of 50 units).

Since each of the above subword unit sets is capable of representing any word in the English language, the issues in the choice of subword unit sets are the context sensitivity and the ease of training the unit from fluent speech. (In addition, for acoustic units, an issue is the creation of a word lexicon since the units themselves have no inherent linguistic interpretation.) It should be clear that there is no ideal (perfect) set of subword units. The PLU set is extremely context sensitive because each unit is potentially affected by its predecessors (one or more) and its followers. However, there is only a small number of PLUs and they are relatively easy to train. On the other extreme are the syllables which are longest units and are the least context sensitive. However, there are so many of them that they are almost as difficult to train as whole-word models.

For simplicity we will initially assume that we use PLUs as the basic speech units. In particular we use the set of 47 PLUs shown in Table 8.1 (which includes an explicit symbol for silence –h#). For each PLU we show an orthographic symbol (e.g., aa) and a word associated with the symbol (e.g., father). (These symbols are essentially identical to the ARPAPET alphabet of Table 2.1; lowercase symbols are used throughout this chapter for consistency with the DARPA community.) Table 8.2 shows typical pronunciations for several words from the DARPA RM task in terms of the PLUs in Table 8.1. A strong advantage of using PLUs is the ease of creating word lexicons of the type shown in Table 8.2 from standard (electronic) dictionaries. We will see later in this chapter how we exploit the advantages of PLUs, while reducing the context dependencies, by going to more specialized PLUs which take into consideration either the left or right (or both) contexts in which the PLU appears.

One problem with word lexicons of the type shown in Table 8.2 is that they don't easily account for variations in word pronunciation across different dialects and in the context of a sentence. Hence a simple word like "a" is often pronounced as /ey/ in isolation (e.g., the

**TABLE 8.1.** Set of basic PLUs for speech.

| Number | Symbol | Word | Number | Symbol | Word |
|---|---|---|---|---|---|
| 1 | h# | silence | 26 | k | *k*ick |
| 2 | aa | father | 27 | l | *l*ed |
| 3 | ae | b*a*t | 28 | m | *m*om |
| 4 | ah | b*u*tt | 29 | n | *n*o |
| 5 | ao | b*ough*t | 30 | ng | si*ng* |
| 6 | aw | b*ough* | 31 | ow | b*oa*t |
| 7 | ax | *a*gain | 32 | oy | b*oy* |
| 8 | axr | din*er* | 33 | p | *p*op |
| 9 | ay | b*i*te | 34 | r | *r*ed |
| 10 | b | *b*ob | 35 | s | *s*is |
| 11 | ch | *ch*urch | 36 | sh | *sh*oe |
| 12 | d | *d*ad | 37 | t | *t*ot |
| 13 | dh | *th*ey | 38 | th | *th*ief |
| 14 | eh | b*e*t | 49 | uh | b*oo*k |
| 15 | el | bott*le* | 40 | uw | b*oo*t |
| 16 | en | butt*on* | 41 | v | *v*ery |
| 17 | er | b*ir*d | 42 | w | *w*et |
| 18 | ey | b*ai*t | 43 | y | *y*et |
| 19 | f | *f*ief | 44 | z | *z*oo |
| 20 | g | *g*ag | 45 | zh | mea*s*ure |
| 21 | hh | *h*ag | 46 | dx | b*u*tter |
| 22 | ih | b*i*t | 47 | nx | ce*n*ter |
| 23 | ix | ros*es* | | | |
| 24 | iy | b*ea*t | | | |
| 25 | jh | *j*udge | | | |

**TABLE 8.2.** Typical word pronunciations (word lexicon) based on context-independent PLUs.

| Word | Number of phones | Transcription | | | | |
|---|---|---|---|---|---|---|
| a | 1 | ax | | | | |
| above | 4 | ax | b | ah | v | |
| bad | 3 | b | ae | d | | |
| carry | 4 | k | ae | r | iy | |
| define | 5 | d | iy | f | ay | n |
| end | 3 | eh | n | d | | |
| gone | 3 | g | ao | n | | |
| hours | 4 | aw | w | axr | z | |



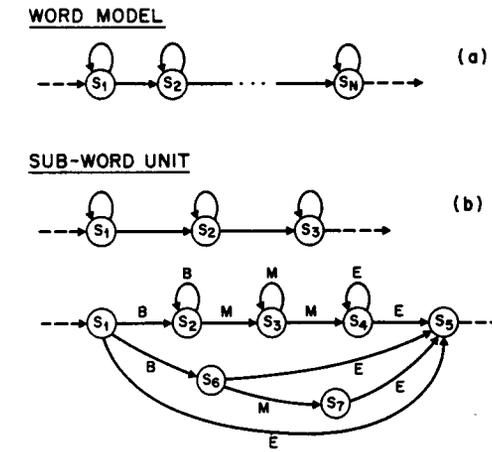



**Figure 8.1** HMM representations of a word (a) and a subword unit (b).

letter A), but is pronounced as /ax/ in context. Another example is a word like "data," which can be pronounced as /d ey t ax/ or /d ae t ax/ depending on the speaker's dialect. Finally words like "you" are normally pronounced as /y uw/ but in context often are pronounced as /jh ax/ or /jh uh/. There are several ways of accounting for word pronunciation variability, including multiple entries in the word lexicon, use of phonological rules in the recognition grammar, and use of context dependent PLUs. We will discuss these options later in this chapter.

## 8.3 SUBWORD UNIT MODELS BASED ON HMMS

As we have shown several times in this book, the most popular way in which speech is modeled is as a left-to-right hidden Markov model. As shown in Figure 8.1a, a whole-word model typically uses a left-to-right HMM with $N$ states, where $N$ can be a fixed value (e.g., 5–10 for each word), or can be variable with the number of sounds (phonemes) in the word, or can be set equal to the average number of frames in the word. For subword units, typically, the number of states in the HMM is set to a fixed value, as shown in Figure 8.1b where a three-state model is used. This means that the shortest tokens of each subword unit must last at least three frames, a restriction that seems reasonable in practice. (Models that use jumps to eliminate this restriction have been studied [2].)

To represent the spectral density associated with the states of each subword unit, one of three approaches can be used. These approaches are illustrated in Figure 8.2. Perhaps the simplest approach is to design a VQ-based codebook for all speech sounds (as shown in part a of the figure). For this approach the probability density of the observed

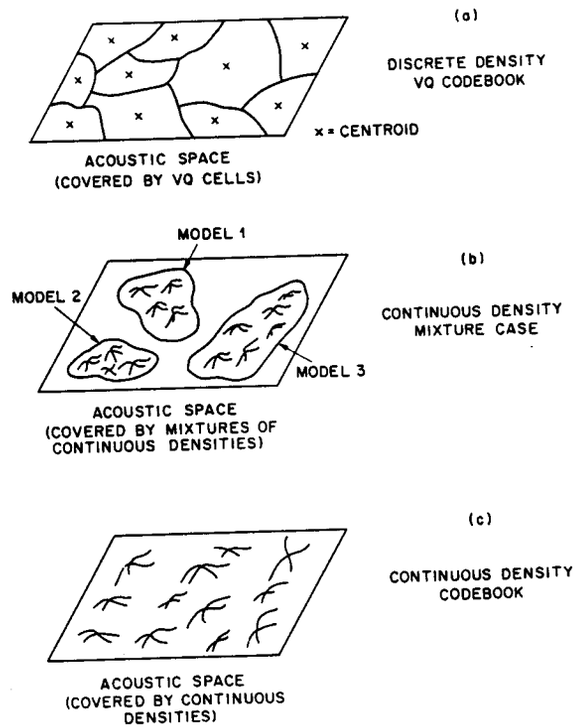



**Figure 8.2** Representations of the acoustic space of speech by (a) partitioned VQ cells, (b) sets of continuous mixture Gaussian densities, and (c) a continuous-density codebook (after Lee et al. [7]).

spectral sequence within each state of each PLU is simply a discrete density defined over the codebook vectors. The interpretation of the discrete density within a state is that of implicitly isolating the part of the acoustic space in which the spectral vectors occur and assigning the appropriate codebook vector (over that part of the space) a fixed probability for spectral vectors within each isolated region regardless of its proximity to the corresponding codebook vector. A second alternative, illustrated in part b of Figure 8.2, is to represent the continuous probability density in each subword unit state by a mixture density that explicitly defines the part of the acoustic space in which the spectral vectors occur. Each mixture component has a spectral mean and variance that is highly dependent on the spectral characteristics of the subword unit (i.e., highly localized in the acoustic space). Hence the models for different subword units usually do not have substantial overlap in the acoustic space. Finally, a third alternative is to design a type of continuous density codebook over the entire acoustic space, as illustrated in part c of Figure 8.2. Basically the entire acoustic



space is covered by a set of independent Gaussian densities, derived in much the same way as the discrete VQ codebook, with the resulting set of means and covariances stored in a codebook. This alternative is a compromise between the previous two possibilities. It differs from the discrete density case in the way the probability of an observation vector is computed; instead of assigning a fixed probability to any observation vector that falls within an isolated region, it actually determines the probability according to the closeness of the observation vector to the codebook vector (i.e., it calculates the exponents of the Gaussian distributions). For each state of each subword unit, the density is assumed to be a mixture of the fixed codebook densities. Hence, even though each state is characterized by a continuous mixture density, one need only estimate the set of mixture gains to specify the continuous density completely. Furthermore, since the codebook set of Gaussian densities is common for all states of all subword models, one can precompute the likelihoods associated with an input spectral vector for each of the codebook vectors, and ultimately determine state likelihoods using only a simple dot product with the state mixture gains. This represents a significant computational reduction over the full mixture continuous density case. This mixed density method has been called the tied mixture approach [10, 28] as well as the semicontinuous modeling method [11] and has been applied to the entire acoustic space as well as to pieces of the acoustic space for detailed PLU modeling. This method can be further extended to the case in which a set of continuous density codebooks is designed, one for each state of each basic (context independent) speech unit. One can then estimate sets of mixture gains appropriate to context dependent versions of each basic speech unit and use them appropriately for recognition. We will return to this issue later in this chapter.

## 8.4 TRAINING OF SUBWORD UNITS

Implicitly it would seem that training of the models for subword units would be extremely difficult, because there is no simple way to create a bootstrap model of such short, imprecisely defined, speech sounds. Fortunately, this is not the case. The reason for this is because of the inherent tying of subword units across words and sentences—that is, every subword unit occurs a large number of times in any reasonable size training set. Hence estimation algorithms like the forward-backward procedure, or the segmental $k$-means algorithm, can start with a uniform segmentation (flat or random initial models) and rapidly converge to the best model estimates in just a few iterations.

To illustrate how models of subword units are estimated, assume we have a labeled training set of speech sentences, where each sentence consists of the speech waveform and its transcription into words. (We assume that waveform segmentation into words is not available.) We further assume a word lexicon is available that provides a transcription of every word in the training set strings in terms of the set of subword units being trained. We assume that silence can (but needn't) precede or follow any word within a sentence (i.e., we allow pauses in speaking), with silence at the beginning and end of each sentence the most likely situation. Based on the above assumptions, a typical sentence in the training set can be transcribed as

$$S_W: \; W_1 \; W_2 \; W_3 \; \ldots \; W_I,$$

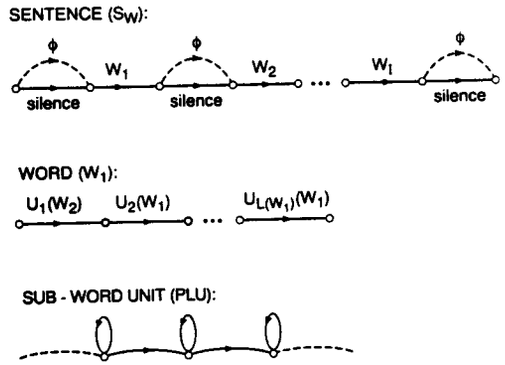



**Figure 8.3** Representation of a sentence, word, and subword unit in terms of FSNs.

in which each $W_i$, $1 \leq i \leq I$, is a word in the lexicon. Hence the sentence "show all alerts" is a three-word sentence with $W_1$ = show, $W_2$ = all, and $W_3$ = alerts. Each word can be looked up in the lexicon to find its transcription in terms of subword units. Hence the sentence $S$ can be written in terms of subword units as

$$\begin{aligned} S_U: \quad & U_1(W_1)U_2(W_1)\ldots U_{L(W_1)}(W_1) \oplus U_1(W_2)U_2(W_2)\ldots U_{L(W_2)}(W_2) \oplus \\ & U_1(W_3)U_2(W_3)\ldots U_{L(W_3)}(W_3) \oplus \cdots \oplus U_1(W_I)U_2(W_I)\ldots U_{L(W_I)}(W_I), \end{aligned}$$

where $L(W_1)$ is the length (in units) of word $W_1$, etc. Finally we replace each subword unit by its HMM (the three-state models shown in Figure 8.1) and incorporate the assumptions about silence between words to give an extended HMM for each sentence.

The above process is illustrated (in general) in Figure 8.3. We see that a sentence is represented as a finite-state network (FSN) where the arcs are either words or silence or null arcs (where a null ($\phi$) transition is required to skip the alternative silence). Each word is represented as an FSN of subword units and each subword unit is represented as a three-state HMM.

Figure 8.4 shows the process of creating the composite FSN for the sentence "Show all alerts," based on a single-word pronunciation lexicon. One feature of this implementation is the use of a single-state HMM for the silence word. This is used (rather than the three-state HMMs used for each PLU), since silence is generally stationary and has no temporal structure to exploit.

When there are multiple representations of words in the lexicon (e.g., for two or more distinct pronunciations) it is easy to modify the FSN of Figures 8.3 and 8.4 to add parallel paths for the word arcs. (We will see that only one path is chosen in training, namely the best representation of the actual word pronunciation in the context of the spoken sentence.) Furthermore, multiple models of each subword unit can be used by introducing parallel paths in the word FSNs and then choosing the best version of each subword unit in the decoding process.



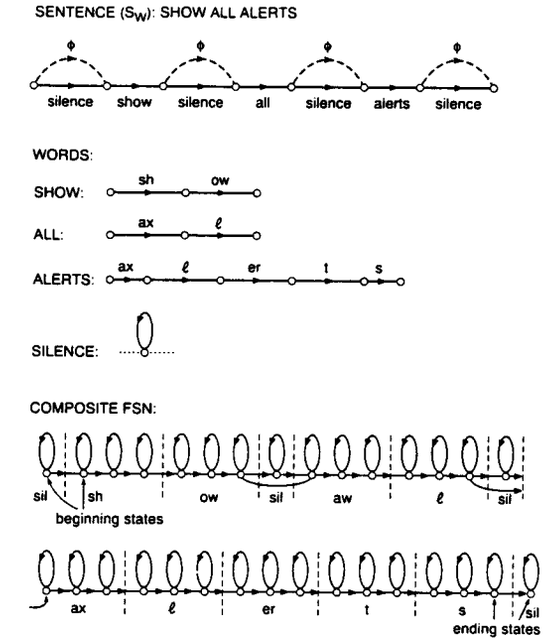



**Figure 8.4** Creation of composite FSN for sentence "Show all alerts."

Once a composite sentence FSN is created for each sentence in the training set, the training problem becomes one of estimating the subword unit model parameters which maximize the likelihood of the models for all the given training data. The maximum likelihood parameters can be solved for using either the forward-backward procedure (see Ref. [2] for example) or the segmental $k$-means training algorithm. The way in which we use the segmental $k$-means training procedure to estimate the set of model parameters (based on using a mixture density with $M$ mixtures/state) is as follows:

1. **Initialization**: Linearly segment each training utterance into units and HMM states assuming no silence between words (i.e., silence only at the beginning and end of each sentence), a single lexical pronunciation of each word, and a single model for each subword unit. Figure 8.5, iteration 0, illustrates this step for the first few units of one training sentence. Literally we assume every unit is of equal duration initially.
2. **Clustering**: All feature vectors from all segments corresponding to a given state ($i$) of a given subword unit are partitioned into $M$ clusters using the $k$-means algorithm. (This step is iterated for all states of all subword units.)
3. **Estimation**: The mean vectors, $\boldsymbol{\mu}_{ik}$, the (diagonal) covariance matrices, $\mathbf{U}_{ik}$, and the

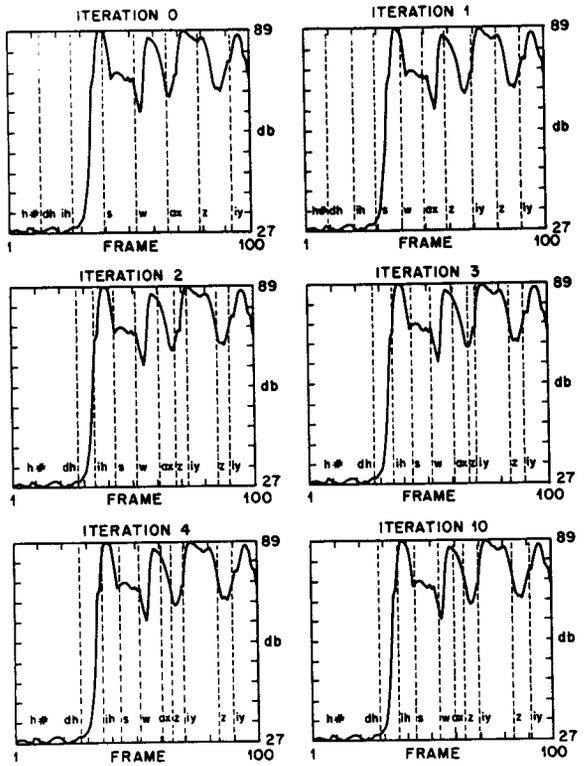



**Figure 8.5** Segmentations of a training utterance resulting from the segmental $k$-means training for the first several iterations (after Lee et al. [7]).

mixture weights, $c_{ik}$, are estimated for each cluster $k$ in state $i$. (This step is iterated for all states of all subword units.)

4. **Segmentation**: The updated set of subword unit models (based on the estimation of step 3) is used to resegment each training utterance into units and states (via Viterbi decoding). At this point multiple lexical entries can be used for any word in the vocabulary. Figure 8.5 shows the result of this resegmentation step for iterations 1–4 and 10 for one training utterance. It can be seen that by iteration 2 the segmentation into subword units is remarkably stable.
5. **Iteration**: Steps 2–4 are iterated until convergence (i.e., until the overall likelihoods stop increasing).

Figure 8.6 illustrates the resulting segmentation of the first few units of the utterance



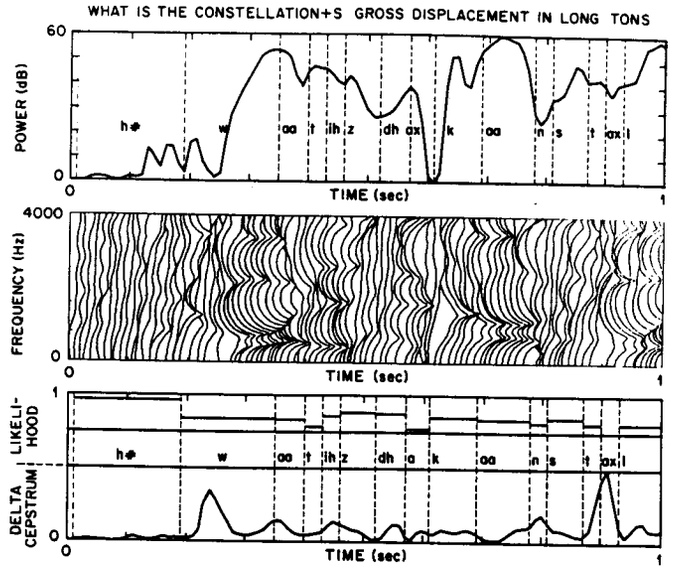



**Figure 8.6** Segmentation of an utterance into PLUs (after Lee et al. [7]).

"What is the constellation. . . ." Shown in this figure are the power contour in dB (upper panel), the running LPC spectral slices (the middle panel), and the likelihood scores and delta-cepstral values (lower panel) for the first second of the sentence. The resulting segmentations are generally remarkably consistent with those one might manually choose based on acoustic-phonetic criteria. Since we use an acoustic criterion for choice of segmentation points, the closeness of PLU units to true phonetic units is often remarkable, especially in light of the phonetic variability in word pronunciation discussed previously.

In summary we have shown how one can use a training set of speech sentences that have only word transcriptions associated with each sentence and optimally determine the parameters of a set of subword unit HMMs. The resulting parameter estimates are extremely robust to the training material as well as to details of word pronunciation as obtained from the word lexicon. The reason for this is that a common word lexicon (with associated word pronunciation errors) is used for both training and recognition; hence errors in associating proper subword units to words are consistent throughout the process and are less harmful than they would be in alternative methods of estimating parameters of subword models.

The results of applying the segmental $k$-means training procedure to a set of 3990 training sentences from 109 different talkers, in terms of PLU counts and PLU likelihood scores are shown in Table 8.3. A total of 155,000 PLUs occurred in the 3990 sentences with silence (h#) having the most occurrences (10,638 or 6.9% of the total) and nx (flapped

**TABLE 8.3.** PLU statistics on count and average likelihood score.

| PLU | Count | % | Average likelihood | (Rank) |
|---|---|---|---|---|
| h# | 10638 | 6.9 | 18.5 | (1) |
| r | 8997 | 5.8 | 8.4 | (45) |
| t | 8777 | 5.7 | 9.7 | (37) |
| ax | 8715 | 5.6 | 7.1 | (47) |
| s | 8625 | 5.6 | 15.4 | (3) |
| n | 8478 | 5.5 | 8.3 | (46) |
| ih | 6542 | 4.2 | 9.9 | (35) |
| iy | 5816 | 3.7 | 12.0 | (17) |
| d | 5391 | 3.5 | 8.5 | (44) |
| ae | 4873 | 3.1 | 13.3 | (10) |
| ℓ | 4857 | 3.1 | 8.9 | (41) |
| z | 4733 | 3.0 | 12.4 | (14) |
| eh | 4604 | 3.0 | 11.2 | (21) |
| k | 4286 | 2.8 | 10.6 | (27) |
| p | 3793 | 2.4 | 14.3 | (6) |
| m | 3625 | 2.3 | 8.5 | (43) |
| ao | 3489 | 2.2 | 10.4 | (32) |
| f | 3276 | 2.1 | 17.7 | (2) |
| ey | 3271 | 2.1 | 14.5 | (5) |
| w | 3188 | 2.1 | 10.2 | (34) |
| ix | 3079 | 2.0 | 8.7 | (42) |
| dh | 2984 | 1.9 | 11.8 | (18) |
| v | 2979 | 1.9 | 12.0 | (16) |
| aa | 2738 | 1.8 | 10.3 | (33) |
| b | 2138 | 1.4 | 10.7 | (25) |
| y | 2137 | 1.4 | 13.1 | (11) |
| uw | 2032 | 1.3 | 10.6 | (26) |
| sh | 1875 | 1.2 | 13.1 | (12) |
| ow | 1875 | 1.2 | 10.9 | (24) |
| axr | 1825 | 1.2 | 9.5 | (38) |
| ah | 1566 | 1.0 | 11.3 | (20) |
| dx | 1548 | 1.0 | 10.4 | (31) |
| ay | 1527 | 1.0 | 13.9 | (8) |
| en | 1478 | 0.9 | 9.1 | (40) |
| g | 1416 | 0.9 | 9.8 | (36) |
| hh | 1276 | 0.8 | 11.4 | (19) |
| th | 924 | 0.6 | 14.1 | (7) |
| ng | 903 | 0.6 | 9.1 | (39) |
| ch | 885 | 0.6 | 12.5 | (13) |
| el | 863 | 0.6 | 11.0 | (23) |
| er | 852 | 0.5 | 10.6 | (29) |
| jh | 816 | 0.5 | 10.6 | (28) |
| aw | 682 | 0.4 | 13.6 | (9) |
| uh | 242 | 0.2 | 11.0 | (22) |
| zh | 198 | 0.1 | 12.2 | (15) |
| oy | 130 | 0.1 | 15.3 | (4) |
| nx | 57 | 0.04 | 10.4 | (30) |



n) having the fewest occurrences (5 or 0.04% of the total). In terms of average likelihood scores, silence (h#) had the highest score (18.5) followed by f (17.7) and s (15.4), while ax had the lowest score (7.1), followed by n (8.3) and r (8.4). (Note that, in this case, a higher average likelihood implies less variation among different renditions of the particular sound.) It is interesting to note that the PLUs with the three lowest average likelihood scores (ax, n, and r) were among the most frequently occurring sounds (r was second, n sixth, and ax fourth in frequency of occurrence). Similarly, some of the sounds with the highest likelihood scores were among the least occurring sounds (e.g., oy was fourth according to likelihood score but 21st according to frequency of occurrence).

## 8.5 LANGUAGE MODELS FOR LARGE VOCABULARY SPEECH RECOGNITION

Small vocabulary speech-recognition systems are used primarily for command-and-control applications where the vocabulary words are essentially acoustic control signals that the system has to respond to. (See Chapter 9 for a discussion of command-and-control applications of speech recognition.) As such, these systems generally do not rely heavily on language models to accomplish their selected tasks. A large vocabulary speech-recognition system, however, is generally critically dependent on linguistic knowledge embedded in the input speech. Therefore, for large vocabulary speech recognition, incorporation of knowledge of the language, in the form of a "language" model, is essential. In this section we discuss a statistically motivated framework for language modeling.

The goal of the (statistical) language model is to provide an estimate of the probability of a word sequence $W$ for the given recognition task. If we assume that $W$ is a specified sequence of words, i.e.,

$$W = w_1 w_2 \ldots w_Q, \tag{8.4}$$

then it would seem reasonable that $P(W)$ can be computed as

$$P(W) = P(w_1 w_2 \ldots w_Q) = P(w_1)P(w_2|w_1)P(w_3|w_1 w_2)\ldots P(w_Q|w_1 w_2 \ldots w_{Q-1}). \tag{8.5}$$

Unfortunately, it is essentially impossible to reliably estimate the conditional word probabilities, $P(w_j|w_1 \ldots w_{j-1})$ for all words and all sequence lengths in a given language. Hence, in practice, it is convenient to use an $N$-gram word model, where we approximate the term $P(w_j|w_1 \ldots w_{j-1})$ as

$$P(w_j|w_1 w_2 \ldots w_{j-1}) \approx P(w_j|w_{j-N+1} \ldots w_{j-1}), \tag{8.6}$$

i.e., based only on the preceding $N-1$ words. Even $N$-gram probabilities are difficult to estimate reliably for all but $N = 2$ or possibly 3. Hence, in practice, it is often convenient to use a word pair model that specifies which word pairs are valid in the language through the use of a binary indicator function, i.e.,

$$P(w_j|w_k) = \begin{cases} 1 & \text{if } w_k w_j \text{ is valid} \\ 0 & \text{otherwise} \end{cases} \tag{8.7}$$

Another simple language model, often called the no-grammar model, assumes $P(w_j|w_k) = 1$ for all $j$ and $k$, so that every word is assumed capable of being followed by every other word in the language. In the next section we show how the word pair and the no-grammar models can be implemented as finite state networks so as to be integrated simply into a recognition decoding algorithm.

Alternative language models include formal grammars (e.g., context free or context dependent grammar), $N$-grams of word classes (rather than words) etc. These types of grammars provide more realistic models for natural language input to machines than the artificial $N$-grams or words, or the word pair grammars. However, they are somewhat more difficult to integrate with the acoustic decoding and hence will not be discussed here.

## 8.6 STATISTICAL LANGUAGE MODELING

In large vocabulary speech recognition, in which word sequences $W$ are uttered to convey some message, the language model $P(W)$ is of critical importance to the recognition accuracy as shown in Eq. (8.3). In most cases, the language model $P(W)$ has to be estimated from a given (large) text corpus. In this section we discuss how to construct such a statistical language model from a (textual) training corpus.

For practical reasons, the word sequence probability $P(W)$ is approximated by

$$P_N(W) = \prod_{i=1}^{Q} P(w_i|w_{i-1}, w_{i-2}, \ldots, w_{i-N+1}), \tag{8.8}$$

which is called an $N$-gram language model. The conditional probabilities $P(w_i|w_{i-1}, \ldots, w_{i-N+1})$ can be estimated by the simple relative frequency approach,

$$\hat{P}(w_i|w_{i-1}, \ldots, w_{i-N+1}) = \frac{F(w_i, w_{i-1}, \ldots, w_{i-N+1})}{F(w_{i-1}, \ldots, w_{i-N+1})}, \tag{8.9}$$

in which $F$ is the number of occurrences of the string in its argument in the given training corpus. Obviously, in order for the estimate in Eq. (8.9) to be reliable, $F(w_i, w_{i-1}, \ldots, w_{i-N+1})$ has to be substantial in the given corpus. The implications of this are that the size of the training corpus may be prohibitively large and that $F(w_i, w_{i-1}, \ldots, w_{i-N+1}) = 0$ for many possible word strings, $w_i, w_{i-1}, \ldots, w_{i-N+1}$, due to the limited size of the corpus.

One way to circumvent this problem is to smooth the $N$-gram frequencies as suggested by Jelinek et al. [12]. Consider $N = 3$, the trigram model. The smoothing is done by interpolating trigram, bigram and unigram relative frequencies

$$\hat{P}(w_3|w_1, w_2) = p_1 \frac{F(w_1, w_2, w_3)}{F(w_1, w_2)} + p_2 \frac{F(w_1, w_2)}{F(w_1)} + p_3 \frac{F(w_1)}{\sum F(w_i)}, \tag{8.10}$$

in which the nonnegative weights satisfy $p_1 + p_2 + p_3 = 1$ and $\sum F(w_i)$ is the size of the corpus. The weights depend on the values of $F(w_1, w_2)$ and $F(w_1)$ and can be obtained by applying the principle of cross-validation [12].



## 8.7 PERPLEXITY OF THE LANGUAGE MODEL

Having constructed a language model from a training corpus, one may ask how well the language model will perform in the context of speech recognition. This can be answered based on the concept of source of information in information theory. To provide such a measure of performance, we must first discuss several concepts, including entropy, estimated entropy, and perplexity.

Consider an information source that puts out sequences of words (symbols) $w_1, w_2, \ldots, w_Q$, each of which is chosen from a vocabulary $\bar{V}$ with size $|\bar{V}|$, according to some stochastic law. The entropy of the source can be defined as

$$H = -\lim_{Q\to\infty} \left(\frac{1}{Q}\right) \left\{\sum P(w_1, w_2, \ldots, w_Q) \log P(w_1, w_2, \ldots, w_Q)\right\}, \tag{8.11}$$

in which $P(\,)$ is the probability of the argument string the source will put out according to the stochastic law and the summation is over all string sequences $w_1, w_2, \ldots, w_Q$. If the words in the string sequence are generated by the source in an independent manner

$$P(w_1, w_2, \ldots, w_Q) = P(w_1)P(w_2)\ldots P(w_Q), \tag{8.12}$$

then

$$H = -\sum_{w\in\bar{V}} P(w) \log P(w), \tag{8.13}$$

which is sometimes referred to as the first-order entropy of the source (even if Eq. (8.12) is not true).

The quantity $H$ of Eq. (8.11) can be considered as the average information of the source when it puts out a word $w$. Equivalently, a source of entropy $H$ is one that has as much information content as a source which puts out words equiprobably from a vocabulary of size $2^H$.

If the source is ergodic (meaning its statistical properties can be completely characterized in a sufficiently long sequence that the source puts out), the entropy of Eq. (8.11) is equivalent to

$$H = -\lim_{Q\to\infty} \left(\frac{1}{Q}\right) \log P(w_1, w_2, \ldots, w_Q). \tag{8.14}$$

In other words, we can compute the entropy from a "typical" (long) sequence of words generated by the source. The length of this typical sequence (i.e., the corpus) has to approach infinity, which is of course unattainable. We often compute $H$ based on a finite but sufficiently large $Q$; i.e.,

$$H = -\left(\frac{1}{Q}\right) \log P(w_1, w_2, \ldots, w_Q). \tag{8.15}$$

An interesting interpretation of $H$ from the perspective of speech recognition is that it is the degree of difficulty that the recognizer encounters, on average, when it is to determine a word from the same source. This difficulty, or uncertainty, is based on the actual probability $P(w_1, w_2, \ldots, w_Q)$ which, for natural languages, is usually not known

beforehand and thus has to be estimated. (We do not include acoustic uncertainty in the present context of language modeling.)

One way to estimate $H$ is to use $P(W) = P(w_1, w_2, \ldots, w_Q)$ from the language model. For example, if the $N$-gram language model $P_N(W)$ (Eq. (8.8)) is used, an estimate of $H$ of Eq. (8.15) is thus

$$H_p = -\frac{1}{Q}\sum_{i=1}^{Q} \log P(w_i | w_{i-1}, w_{i-2}, \ldots, w_{i-N+1}). \tag{8.16}$$

In general,

$$H_p = -\frac{1}{Q} \log \hat{P}(w_1, w_2, \ldots, w_Q), \tag{8.17}$$

where $\hat{P}(w_1, w_2, \ldots, w_Q)$ is an estimate of $P(w_1, w_2, \ldots, w_Q)$. The quantity $H_p$ is an estimated entropy as calculated from a sufficiently long sequence based on a language model. If the source is ergodic and $Q \to \infty$, $H_p \geq H$. Intuitively, this can be easily verified by the fact that knowledge of the true probability $P(w_1, w_2, \ldots, w_Q)$ is the best a recognizer can use and any other probability estimate or language model can never make the task easier for the recognizer. Since $H_p$ is an indication of the recognition difficulty lower-bounded by $H$, a language model that achieves a lower $H_p$ (i.e., closer to $H$) is therefore considered a better model than another language model which leads to a higher $H_p$.

Associated with $H_p$ is a quantity called perplexity (often called the average word branching factor of the language model) defined as

$$B = 2^{H_p} = \hat{P}(w_1, w_2, \ldots, w_Q)^{-1/Q}. \tag{8.18}$$

Note that $H_p$ is the average difficulty or uncertainty in each word based on the language model. When the recognizer uses this language model for the task, the difficulty it faces is equivalent to that of recognizing a text generated by a source that chooses words from a vocabulary size of $B$ independently of each other and with equal probability. Another way to view perplexity is to consider it as the average number of possible words following any string of $(N-1)$ words in a large corpus based on an $N$-gram language model. Perplexity is an important parameter in specifying the degree of sophistication in a recognition task, from the source uncertainty to the quality of the language model.

## 8.8 OVERALL RECOGNITION SYSTEM BASED ON SUBWORD UNITS

A block diagram of the overall continuous speech-recognition system based on subword speech units is shown in Figure 8.7. The first step in the processing is spectral analysis to derive the feature vector used to characterize the spectral properties of the speech input. For the most part, we will consider spectral vectors with 38 components consisting of 12 cepstral components, 12 delta cepstral components, 12 delta-delta cepstral components, delta log energy, and delta-delta log energy. (Systems with the first 12 and the first 24



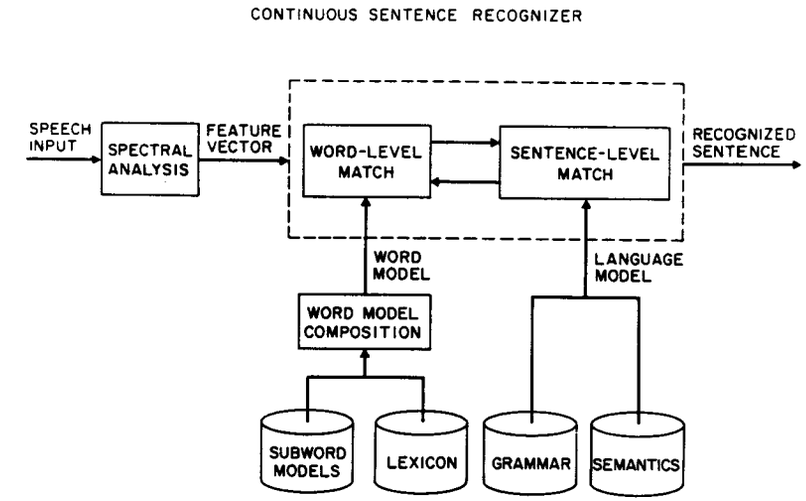



**Figure 8.7** Overall block diagram of subword unit based continuous speech recognizer.

features were also studied, but results on such systems will not be presented here.)

The second step in the recognizer is a combined word-level/sentence-level match. The way this is accomplished is as follows. Using the set of subword HMMs and the word lexicon, a set of word models (HMMs) is created by concatenating each of the subword unit HMMs as specified in the word lexicon. At this point, the system is very similar to the connected word recognizers of Chapter 7. The way in which the sentence-level match is done is via an FSN realization of the word grammar (the syntax of the system) and the semantics as expressed in a composite FSN language model. The implementation of the combined word-level match/sentence-level match is via any of the structures described in Chapter 7. In particular, most systems use structures similar to the frame synchronous level-building method (usually with some type of beam search to restrict the range of paths) to solve for the "best" recognition sentence.

Consider using the recognizer of Figure 8.7 for a database management task called the Naval Resource (Battleship) Management Task—as popularly defined within the DARPA community [13]. This task, which has a 991-word vocabulary (plus a separate silence word), can be used to query a database as to locations, attributes, constraints, history, and other information about ships within the database. Typical examples of sentences used to query the database include

- what is mishawaka's percent fuel
- total the ships that will arrive in diego-garcia by next month

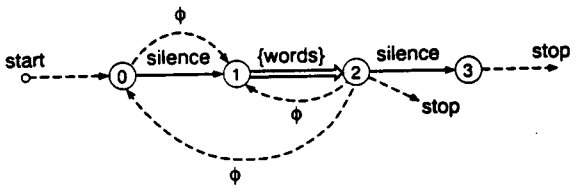



**Figure 8.8** FSN for the NG syntax.

- do any vessels that are in gulf of tonkin have asw mission area of m4
- show the names of any submarines in yellow sea on twenty eight october
- list all the alerts
- what's jason's m-rating on mob
- give t-lam vessels that weren't deployed in november.

The vocabulary thus includes many jargon words, such as m4, m-rating, mob, and t-lam, and several long-content words, such as mishawaka's, diego-garcia, submarines, november, etc., and many short-function words, such as is, the, by, do, in, of, and on.

A wide range of sentences can be constructed from the 991-word vocabulary to query this database. It is possible to construct a finite-state network representation of the full grammar associated with all such sentences. The perplexity (average word branching factor) (see Section 8.7) of the full grammar network is computed to be about 9. However, such a network is rather large (because of the high degree of constraint among words within the vocabulary which form syntactically valid and semantically meaningful sentences) with upward of 50,000 arcs and 20,000 nodes, and cannot easily be implemented as a practical system. Instead, several types of FSN approximations to the full grammar have been constructed.

Perhaps the least constraining grammar (and the simplest to implement) is the no grammar (NG) case, in which any word in the vocabulary is allowed to follow any word in the vocabulary. Such an FSN has the property that, although its coverage of valid sentences is perfect, its overcoverage of the language (i.e., the ratio of sentences generated by the grammar to valid sentences within the task language) is extremely large. The perplexity of the FSN for the NG case is 991, since each word can follow every word in the grammar (assuming all words are essentially equiprobable). The FSN for the NG case is shown in Figure 8.8. (Note that the FSN of Figure 8.8 allows arbitrary phrasing, i.e., groups of words spoken together followed by a pause, because of the silence model and the null arcs.)

A second FSN form of the task syntax is to create a word pair (WP) grammar that specifies explicitly which words can follow each of the 991 words in the vocabulary. The perplexity of this grammar is about 60, and the overcoverage, while significantly below that of the NG case, is still very high. Although the network of Figure 8.8 could be used for the WP grammar (by explicitly including the word pair information at node 2), a somewhat more efficient structure exploits the fact that only a subset of the vocabulary occurs as the first word in a sentence ($B$ or beginning words), and only a subset of the vocabulary occurs



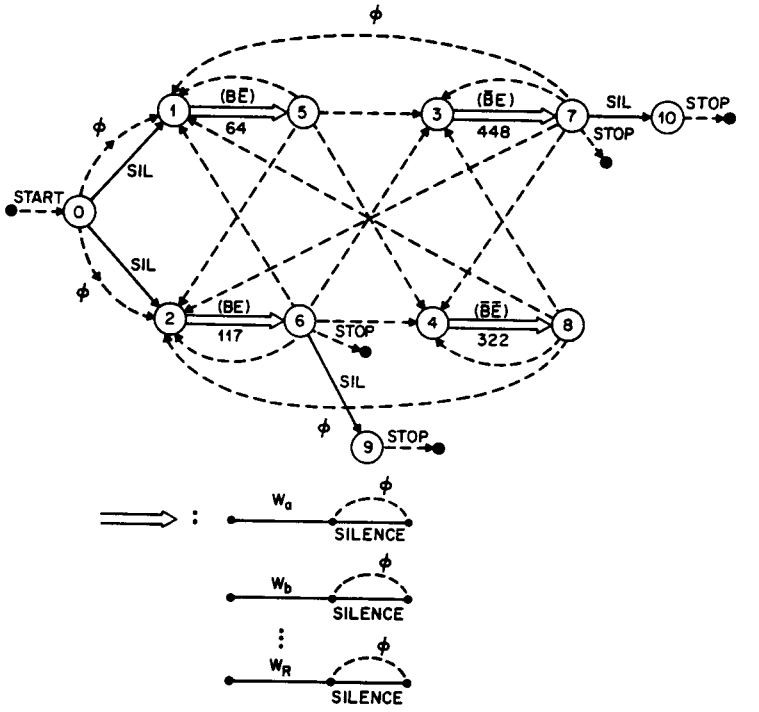



**Figure 8.9** FSN of the WP syntax.

as the last word in a sentence ($E$ or ending words); hence we can partition the vocabulary into four nonoverlapping sets of words, namely

$\{BE\}$=set of words that can either begin or end a sentence, $|BE| = 117$
$\{B\bar{E}\}$=set of words that can begin a sentence but cannot end a sentence, $|B\bar{E}| = 64$
$\{\bar{B}E\}$=set of words that cannot begin a sentence but can end a sentence, $|\bar{B}E| = 488$
$\{\bar{B}\bar{E}\}$=set of words that cannot begin or end a sentence, $|\bar{B}\bar{E}| = 322$.

The resulting FSN, based on this partitioning scheme, is shown in Figure 8.9. This network has 995 real arcs and 18 null arcs. To account for silence between words (which is optional), each word arc bundle (e.g., nodes 1 to 4) is expanded to individual words followed by optional silence, as shown at the bottom of Figure 8.9. Hence the overall FSN allows recognition of sentences of the form

$$S: \text{(silence)} - \{B\bar{E}, BE\} - \text{(silence)} - (\{W\}) \ldots (\{W\}) - \text{(silence)} - \{\bar{B}E, BE\} - \text{(silence)}.$$

Finally, one could construct a task syntax based on statistical word bigram (or even

trigram) probabilities—that is, we assign a probability, $p_{ij}$, to each word pair $(W_i, W_j)$ where $p_{ij}$ is the probability that $W_i$ is followed immediately by $W_j$. That is, if $W_n$ is the $n^{\text{th}}$ word in a string of words, then $p_{ij} = P(W_n = W_j | W_{n-1} = W_i)$ is the language model according to Eq. (8.6). The advantage of the word bigram (WB) approach is that the perplexity is reduced considerably (to 20) for the Resource Management task, with essentially no increase in complexity of the implementation.

### 8.8.1 Control of Word Insertion/Word Deletion Rate

Using a structure of the type shown in Figure 8.9, there is no control on the sentence length. That is, it is possible to generate sentences that are arbitrarily long by inserting a large number of short-function words. To prevent this from occurring, it is a simple matter to incorporate a word insertion penalty into the Viterbi decoding, such that a fixed negative quantity is added to the likelihood score at the end of *each* word arc (i.e., at nodes 5–8 in Figure 8.9). By adjusting the word penalty, we can control the rate of word insertion and word deletion; a very large word penalty will reduce the word insertion rate and increase the word deletion rate, and a very small penalty will have the opposite effect. A value for word penalty is usually experimentally determined to balance these adverse effects.

### 8.8.2 Task Semantics

We have discussed how task syntax can be incorporated into the overall recognition structure. At the end of this chapter we will briefly describe a general procedure for integrating a semantic component into the recognizer.

### 8.8.3 System Performance on the Resource Management Task

Using the segmental $k$-means training algorithm, the set of 47 PLUs of Table 8.1 were trained using a set of 4360 sentences from 109 talkers. The likelihood scores were essentially unchanged after two iterations of the $k$-means loop. The number of mixtures per state was varied from 1 to 256 in multiples of 2 to investigate the effects of higher acoustic resolution on performance.

To evaluate the recognizer performance, five different sets of test data were used, including:

| | |
|---|---|
| train 109 | A randomly selected set of 2 sentences from each of the 109 training talkers; this set was used to evaluate the ability of the algorithm to recognize the training material |
| feb 89 | A set of 30 sentences from each of 10 talkers, none of whom was in the training set; this set was distributed by DARPA in February of 1989 to evaluate performance |
| oct 89 | A second set of 30 sentences from each of 10 additional talkers, none of whom was in the training set; this set was distributed by DARPA in October of 1989 |
| jun 90 | A set of 120 sentences from each of 4 new talkers, none of whom was in the training set (distributed by DARPA in June of 1990) |
| feb 91 | A set of 30 sentences from each of 10 new talkers, none of whom was in the training set (distributed by DARPA in February of 1991). |



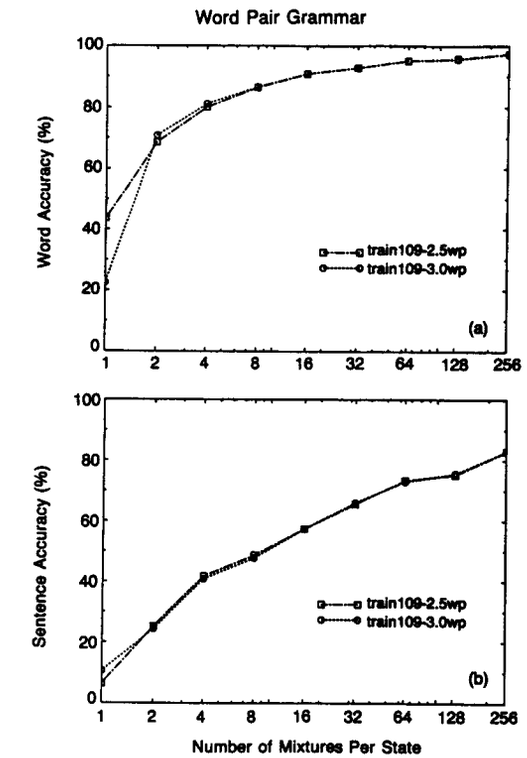



**Figure 8.10** Word and sentence accuracies versus number of mixtures per state for the training subset using the WP syntax.

The recognizer performance was evaluated for each of the test sets, using both WP and NG syntax, and with different word penalties. For all cases, evaluations were made using models with from 1 to 256 mixtures per state for each PLU.

The recognition results are presented in terms of word accuracy (percentage words correct minus percentage word insertions) and sentence accuracy as a function of the number of mixtures per state for each PLU model. The alignment of the text of the recognized string with the text of the spoken string was performed using a dynamic programming alignment method as specified by DARPA.

The recognition results on the training subset (train 109) are given in Figures 8.10 (for the WP syntax) and 8.11 (for the NG syntax). The upper curves show word accuracy (in percentage) versus number of mixtures per state (on a logarithmic scale) for two different values of the word penalty, and the lower curves show sentence accuracy for the same

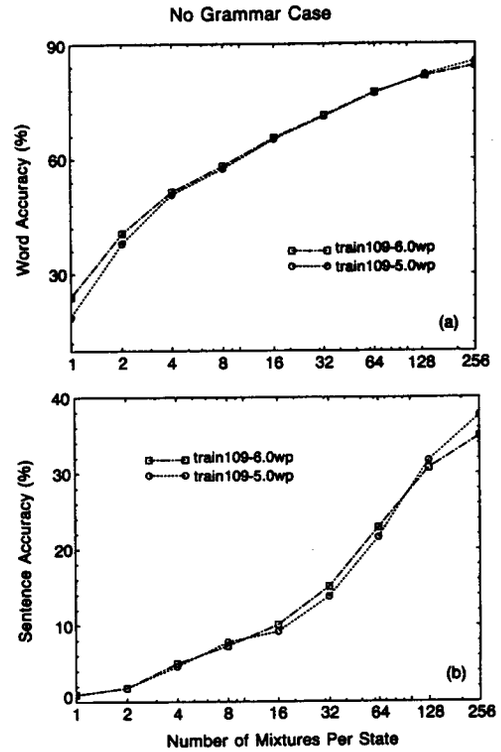



**Figure 8.11** Word and sentence accuracies versus number of mixtures per state for the training subset using the NG syntax.

parameters. A sharp and steady increase in accuracy is obtained as the number of mixtures per state increases, going from about 43.6% word accuracy (10.6% sentence accuracy) for 1 mixture per state to 97.3% word accuracy (83% sentence accuracy) for 256 mixtures per state for the WP syntax using a word penalty of 2.5. For the NG syntax (using a word penalty of 6.0), the comparable results were 24% word accuracy (0.9% sentence accuracy) for 1 mixture per state and 84.2% word accuracy (34.9% sentence accuracy) for 256 mixtures per state.

The recognition results on the independent test sets are given in Figures 8.12 (for the WP syntax) and 8.13 (for the NG syntax). Although there are detailed differences in performance among the different test sets (especially for small numbers of mixtures per state), the performance trends are essentially the same for all the test sets. In particular we see that for the WP syntax, the range of word accuracies for 1 mixture per state is 42.9%



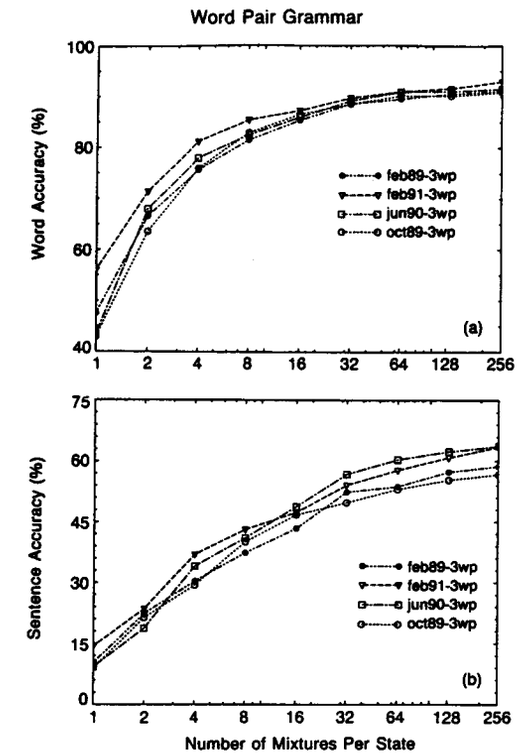



**Figure 8.12** Word and sentence accuracies versus number of mixtures per state for the four test sets using the WP syntax.

(for feb 89) to 56.0% (for jun 90), whereas for 256 mixtures per state the range is 90.9% (for feb 89) to 93.0% (for jun 90). For the NG syntax, the range of word accuracies for 1 mixture per state is 20.1% (for feb 91) to 28.5% (for jun 90) and for 256 mixtures per state it is 68.5% (for oct 89) to 70.0% (for feb 91).

Perhaps the most significant aspect of the performance is the difference in accuracies between the test sets and the training subset. Thus there is a gap of 4–7% in word accuracy for the WP syntax at 256 mixtures per state, and a gap of 14.2–15.7% for the NG syntax at 256 mixtures per state. Such gaps are indicative of the ability of the training procedure to overtrain (learn details) on the training set, thereby achieving significantly higher recognition accuracy on this set than on any other representative test set.

The results presented in this section show that a simple set of context-independent PLUs can be trained for a continuous speech large vocabulary recognition task, using

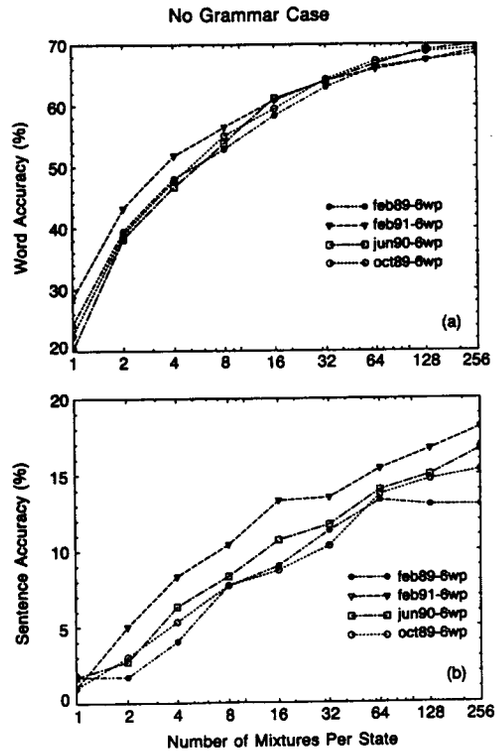



**Figure 8.13** Word and sentence accuracies versus number of mixtures per state for the four test sets using the NG syntax.

standard Viterbi training procedures, and be used to provide reasonably good recognition accuracy for a moderately complex task. The key issue now is what can be done, in meaningful ways to improve recognizer performance. To answer this question, we will examine several possible extensions of the basic recognition system in the next few sections.

## 8.9 CONTEXT-DEPENDENT SUBWORD UNITS

There are several advantages to using a small basic set of context-independent subword units for large vocabulary speech recognition. First of all we have shown that the models of these subword units are easily trained from fluent speech, with essentially no human decisions as to segmentation and labeling of individual sections of speech. Second, the



resulting units are generalizable to new contexts (word vocabularies, tasks with different syntax and semantics) with no extra effort. Finally, the resulting models are relatively insensitive to the details of the context from which the training tokens are extracted. By this we mean that, in theory, we can derive the subword unit model parameters from two arbitrary but sufficiently large training sets of fluent speech (hopefully of the same size and general linguistic content but not necessarily the same vocabulary words and sentences) to obtain essentially the same parameter estimates for each model. In practice, this is almost the case.

However, there are situations in which the subword unit model parameters are extracted from a training set whose linguistic content matches the test set precisely—that is, when the training set is a set of sentences drawn from the recognition task (with the same vocabulary, syntax, and semantics). In such a case, the resulting subword units are somewhat "word sensitive" (showing higher likelihood scores than in the general case) and typically provide higher recognition performance than equivalent model sets derived from arbitrary input speech. In particular, for the Resource Management task discussed in the previous section, "word-sensitive" subword unit models, trained on task-specific training sentences, give about 10% higher word recognition accuracy than the same set of subword unit models trained on arbitrary sentences of comparable size. If the training set is increased in size by a factor of about 3, the word accuracy of the text-independent models approaches that of the word-sensitive models.

Obviously, this performance difference can be attributed to the fact that context-independent subword unit models are not adequate in representing the spectral and temporal properties of the speech unit in all contexts. (By context we mean the effects of the preceding and following sounds as well as the sound stress and information, and even the word in which the sound occurs.) The ultimate effect is a decrease in performance in word and sentence accuracy on speech-recognition tasks.

The solution to this problem is basically a simple, straightforward one—namely, to extend the set of subword units to include context-dependent units (either in addition to or as a replacement for context-independent units) in the recognition system. In theory, the only change necessary in either training or recognition is to modify the word lexicon to be consistent with the final set of subword units. Consider the word "above." Based on using (1) context-independent units, (2) triphone (left and right context) units, (3) multiple-phone models, and (4) word-dependent units, we could have the following lexical representations:

| | | | | | |
|---|---|---|---|---|---|
| (1) above: | ax | b | ah | v | Context-Independent Units |
| (2) above: | $-ax-b | ax-b-ah | b-ah-v | ah-v-$ | Triphones (Context Dependent) |
| (3) above: | ax2 | b2 | ah1 | v1 | Multiple Phone Units |
| (4) above: | ax (above) | b (above) | ah (above) | v (above) | Word-Dependent Units. |

In representation (2), using triphone units, the number of units needed for all sounds in all words is very large (on the order of 10–20,000). In practice, only a small percentage of such triphone units are used, since most units are seen rarely, if at all, in a finite training set. (We discuss this issue below in more detail.) In representation (3), using multiple models of each subword unit, the idea is to cluster common contexts together so as to reduce the

number of context-dependent models. This leads to problems in defining lexical entries for words. (We discuss this issue further in a later section of this chapter.) Finally, the use of word-dependent units is most effective for modeling short-function words (like a, the, in, of, an, and, or) whose spectral variability is significantly greater than that of long-content words like aboard and battleship. (We discuss the modeling of function words in a later section of this chapter.) Finally, it is both reasonable and meaningful to combine all four types of units in a common structure. In theory, as well as in practice, the training and recognition architectures can handle subword unit sets of arbitrary size and complexity. We now discuss each of these issues in more detail.

### 8.9.1 Creation of Context-Dependent Diphones and Triphones

Consider the basic set of context-independent PLUs in which we use the symbol $p$ to denote an arbitrary PLU. We can define a set of context-dependent (CD) diphones as

$$p_L - p - \$ \quad \text{left context (LC) diphone}$$
$$\$ - p - p_R \quad \text{right context (RC) diphone,}$$

in which $p_L$ is the PLU immediately preceding $p$ (the left context sound), $p_R$ is the PLU immediately following $p$ (the right context sound), and \$ denotes a don't care (or don't know) condition.

Similarly we can define a set of context-dependent triphones as

$$p_L - p - p_R \quad \text{left-right context (LRC) triphone.}$$

In theory, the potential number of left (or right) context diphones is $46 \times 45$ (for a basic set of 47 PLUs and excluding silence) or about 2070 left context diphone units. The potential number of left-right context triphone units is $45 \times 46 \times 45$ or 93,150 units. In practice, the actual number of context-dependent PLUs actually seen in a finite training set of sentences is significantly smaller than these upper bounds.

To better understand these concepts, consider the RM task (991 word vocabulary) with a training set of 3990 sentences. To use diphone and triphone context-dependent units, we first convert the lexicon to such units using the rule that the initial sound becomes a right context diphone, the middle sounds become left-right context diphones, and the final sound becomes a left context diphone. Hence the word "above" is converted to the set of units \$–ax–b, ax–b–ah, b–ah–v, ah–v–\$. (We must use diphone units at the beginnings and ends of words because we do not know the preceding or following words.) The above rule is modified to eliminate triphone middles for words with only two PLUs (e.g., in, or) and to revert to the context-independent PLU for words with only one PLU (e.g., a). Using the above method of creating the lexicon, one can count the number of left-right context-dependent (LRC) units (1778), the number of left-context (LC) units (279), the number of right-context (RC) units (280), and the number of context-independent (CI) units (3) for a total of 2340 PLUs in the training set. This number of units, although significantly smaller than the maximum possible number of context-dependent units, is deceiving because many of the units occur only a small number of times in the training set, and therefore it would be difficult to reliably estimate model parameters for such models.



**TABLE 8.4.** Number of intra-word CD units as a function of count threshold, $T$.

| Count Threshold ($T$) | Number of LRC PLUs | Number of LC PLUs | Number of RC PLUs | Number of CI PLUs | Total Number of PLUs |
|---|---|---|---|---|---|
| 50 | 378 | 158 | 171 | 47 | 754 |
| 40 | 461 | 172 | 188 | 47 | 868 |
| 30 | 639 | 199 | 205 | 47 | 1090 |
| 20 | 952 | 212 | 234 | 46 | 1444 |
| 10 | 1302 | 243 | 258 | 44 | 1847 |
| 5 | 1608 | 265 | 270 | 32 | 2175 |
| 1 | 1778 | 279 | 280 | 3 | 2340 |

To combat the difficulties due to the small number of occurrences of some context-dependent units, one can use one of three strategies. Perhaps the simplest approach is to eliminate all models that don't occur sufficiently often in the training set. More formally we define $c(\cdot)$ as the occurrence count for a given unit. Then, given a threshold $T$ on the required number of occurrences of a unit (for reliable model estimation), a reasonable Unit Reduction Rule is

If $c(p_L - p - p_R) < T$, then

1. $p_L - p - p_R \rightarrow \$ - p - p_R$ if $c(\$ - p - p_R) > T$
2. $p_L - p - p_R \rightarrow p_L - p - \$$ if $c(p_L - p - \$) > T$
3. $p_L - p - p_R \rightarrow \$ - p - \$$ otherwise.

The tests above are made sequentially until one passes and the procedure terminates. To illustrate the sensitivity of the CD PLU set to the threshold $T$, Table 8.4 shows the counts of LRC PLUs, LC PLUs, RC PLUs, CI PLUs, and the total PLU count for the 3990 sentence training set. For a threshold of 50, which is generally adequate for estimating model parameters, there are only 378 LRC PLUs (almost a 5-to-1 reduction over the number with a count threshold of 1) and a total of 754 PLUs. We will see later that although such CD PLU sets do provide improvements in recognition performance over CI PLU sets, the amount of context dependency achieved is small and alternative techniques are required to create CD PLU sets.

### 8.9.2 Using Interword Training to Create CD Units

Although the lexical entry for each word uses right or left context diphone units for the first and last sound of each word, both in training and in scoring, one can utilize the known (or postulated) sequence of words to replace these diphone units with the triphone unit appropriate to the words actually (or assumed) spoken. Hence the sentence "Show all ships" would be represented as

\$–sh–ow sh–ow–\$ \$–aw–ℓ aw–ℓ–\$ \$–sh–i sh–i–p i–p–s p–s–\$

using only intraword units, whereas the sentence would be represented as

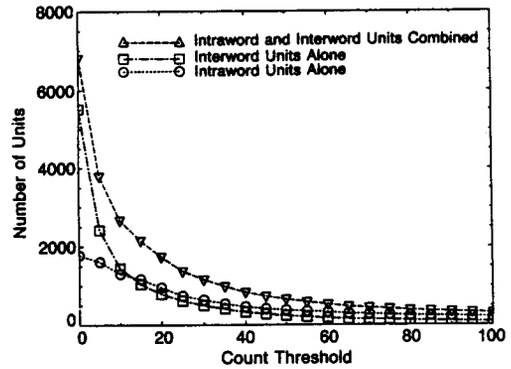



**Figure 8.14** Plots of the number of intraword units, interword units, and combined units as a function of the count threshold.

$-sh–ow sh–ow–aw ow–aw–ℓ aw–ℓ–sh ℓ–sh–i sh–i–p i–p–s p–s–$

using both intraword and interword units. From this simple example we see that, whereas there were only two triphones based on intraword units, there are six triphones based on intraword and interword units—that is, a threefold increase in context-dependent triphone units. (We are assuming no silence between words; it is straightforward to handle the cases when silence actually occurs between words.) To illustrate this effect, Figure 8.14 shows a plot of the number of intraword units, the number of interword units, and the combined count, as a function of the count threshold, for the 1990 sentence DARPA training set. More than 5000 interword triphone units occur one or more times versus less than 2000 intraword units for the same count threshold.

Even when using interword units, the problems associated with estimating model parameters from a small number of occurrences of the units is the major issue. In the next sections we discuss various ways of smoothing and interpolating context dependent models, created from small numbers of occurrences in the training set, with context-independent models, created from large numbers of occurrences in the training set.

### 8.9.3 Smoothing and Interpolation of CD PLU Models

As shown above, we are faced with the following problem. For a training set of reasonable size, there is sufficient data to reliably train context-independent unit models. However, as the number of units becomes larger (by including more context dependencies) the amount of data available for each unit decreases and the model estimates become less reliable. Although there is no ideal solution to this problem (short of increasing the amount of training data ad infinitum), a reasonable compromise is to exploit the reliability of the estimates of the higher level (e.g., CI) unit models to smooth or interpolate the estimates of the lower level (CD) unit models. There are many ways in which such smoothing or



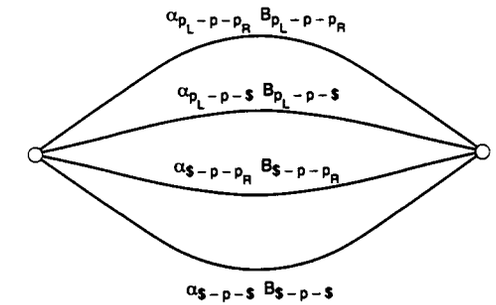



**Figure 8.15** Deleted interpolation model for smoothing discrete density models.

interpolation can be achieved.

The simplest way to smooth the parameter estimates for the CD models is to interpolate the spectral parameters with all higher (less context dependency) models that are consistent with the model [12]. By this we mean that the model for the CD unit $p_L - p - p_R$ (call this $\lambda_{p_L-p-p_R}$) should be interpolated with the models for the units $\$-p-p_R$ ($\lambda_{\$-p-p_R}$), $p_L - p - \$$ ($\lambda_{p_L-p-\$}$) and $\$ - p - \$$ ($\lambda_{\$-p-\$}$). Such an interpolation of model parameters is meaningful only for discrete densities, within states of the HMM, based on a common codebook. Thus if each model $\lambda$ is of the form $(A, B, \pi)$ where $B$ is a discrete density over a common codebook, then we can formulate the interpolation as:

$$\hat{B}_{p_L-p-p_R} = \alpha_{p_L-p-p_R} B_{p_L-p-p_R} + \alpha_{p_L-p-\$} B_{p_L-p-\$} + \alpha_{\$-p-p_R} B_{\$-p-p_R} + \alpha_{\$-p-\$} B_{\$-p-\$}, \tag{8.19}$$

where $\hat{B}_{p_L-p-p_R}$ is the interpolated density. We constrain the $\alpha$s to add up to 1; hence

$$\alpha_{p_L-p-p_R} + \alpha_{p_L-p-\$} + \alpha_{\$-p-p_R} + \alpha_{\$-p-\$} = 1. \tag{8.20}$$

The way in which the $\alpha$s are determined is according to the deleted interpolation algorithm discussed in Section 6.13. We review the ideas, as they apply to these speech unit models, here. Each of the discrete densities, $B_{p_L-p-p_R}$, $B_{p_L-p-\$}$, $B_{\$-p-p_R}$, and $B_{\$-p-\$}$, is estimated from the training data where a small percentage (e.g., 20%) is withheld (deleted). Using the withheld data, the $\alpha$s are estimated using a standard forward-backward approach based on the HMM shown in Figure 8.15. The interpretation of the $\alpha$s is essentially the probability weighted percentage of new data (unseen in training) that favors each of the distributions over the others. Hence, for well-trained detailed models we get $\alpha_{p_L-p-p_R} \to 1$, whereas for poorly trained models we get $\alpha_{p_L-p-p_R} \to 0$ (i.e., the LRC model is essentially obtained from interpolating higher-level, lower context dependency models that are better trained than the detailed CD model).

Other smoothing methods include empirical estimates of the $\alpha$s based on occurrence counts, co-occurrence smoothing based on joint probabilities of pairs of codebook symbols [14], and use of fuzzy VQs in which an input spectral vector is coded into two or more

codebook symbols.

### 8.9.4 Smoothing and Interpolation of Continuous Densities

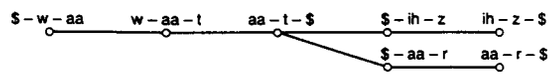
When one uses continuous density modeling of PLUs it is very difficult to devise a good smoothing or interpolation algorithm because the acoustic space of different units is inherently different. There are two reasonable ways to handle this problem. One is to exploit the so called semicontinuous or tied mixture modeling approach discussed earlier in which each PLU uses a fixed set (a codebook) of mixture means and variances, and the only variables are the mixture gains for each model. In this case it is trivial to exploit the method of deleted interpolation on the mixture gains in a manner virtually identical to the one discussed in the previous section.

An alternative modeling approach, and one more in line with independent continuous density modeling of different sounds, is to use a tied-mixture approach on the CI unit level; that is, we design a separate (large) codebook of densities for each CI PLU and then constrain each derived CD unit to use the same mixture means and variances but with independent mixture gains. Again we can use the method of deleted interpolation to smooth mixture gains in an optimal manner.

### 8.9.5 Implementation Issues Using CD Units

The FSN structure of Figure 8.9 is used to implement the continuous speech-recognition algorithm based on a given vocabulary and task syntax ([15–19]). The structure is straightforward to implement when using strictly intraword units because there is no effect of context at word boundaries. Hence the models (HMMs) for each word can be constructed independently and concatenated at the appropriate point of the processing. This is illustrated below for the recognition of the string "what {is, are}" based on intraword units, where the individual words are represented in the lexicon as

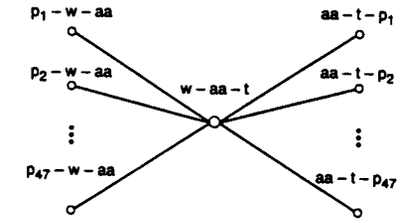
what – {$–w–aa, w–aa–t, aa–t–$}
is – {$–ih–z, ih–z–$}
are – {$–aa–r, aa–r–$}



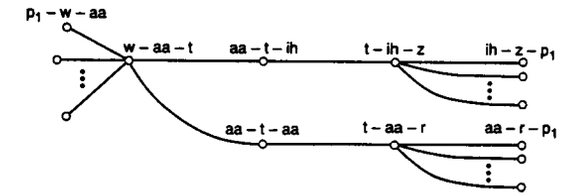

(If we allow silence between words there is a trivial modification to include a silence node after the word /what/.) When we include interword units in the recognition stage the FSN becomes considerably more complicated because the first unit of each word (which we call the head unit) is variable depending on the last unit of each possible preceding word; similarly, the last unit of each word (which we call the tail unit) is variable depending on the first unit of each possible following word. (We call the set of units between the head unit and the tail unit, the body units.) Thus, in theory, a word like "what" consists of (up



to) 47 head units and (up to) 47 tail units, and would be represented as




In practice many, if not most, of the head as well as tail units don't exist; hence, the structure is generally considerably less complex. Thus the FSN network of the strings "what {is, are}" becomes




that is, a considerably more complex network results. (Interestingly, the inclusion of silence adds only a single extra path to each branch of the network.) The bookkeeping associated with such networks can easily get out of hand and dominate the overall computation. Fortunately, several network architectures have been devised for efficiently handling the bookkeeping associated with such networks [15]. Interesting special cases occur when the number of units within a word falls below three. When there are exactly two units in a word, there are no body units so the variable head units merge with the variable tail units. When there is only a single unit within a word, there are no body units nor is there a tail unit. Effectively, the bookkeeping must look at both the preceding word set of tail units and the following word set of head units to handle this case. The three cases described above—namely, implementations of words with three or more units, words with two units, and words with one unit—are illustrated in Figure 8.16.

#### 8.9.5.1 Word Junction Effects

The assumption that is made when training interword units is that in continuous speech, words are pronounced similar to the way they are pronounced in isolation. In most cases this assumption is reasonable in that the coarticulation phenomena at word boundaries only lead to small (soft) changes in the word pronunciation and therefore can readily be modeled by interword units based on the concatenation of the tail unit from one word with the appropriate head unit from the following word. However, in some cases, the pronunciation changes are radical (hard) changes in which a boundary sound (tail or head)

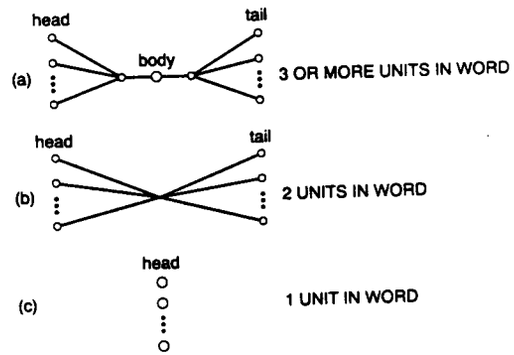



**Figure 8.16** FSN representation of words with three or more units (a), two units (b), and one unit (c).

is completely deleted or replaced with a totally different sound. Examples of such hard changes include the strings "what time" and "did you," among others. In the string "what time," the double stop consonant (t followed by t) between words is replaced with a single occurrence of t; hence one of the *t*s is deleted. In the string "did you," the standard phonetic transcription would be /d ih d y uw/; however, in continuous speech the actual transcription would be /d ih j h uw/ (or even /d ih jh ax/) where the /d y/ boundary phones are changed to the single sound /jh/.

Phonologically predictable changes of the type shown above (the so-called hard sound changes) cannot easily be learned from the training procedure because they occur infrequently and they lead to radically different sounds than would be predicted from the concatenation of boundary sounds of the relevant words as spoken in isolation. To handle these hard changes correctly, a set of phonological rules has to be superimposed on both the training and recognition networks (in a relatively straightforward (brute force) manner). There are about 11 such rules that handle most of the known phonological changes in English [16].

Some typical phonological rules include the following:

Rule 1 Geminate deletion. If a word ends in a consonant and the following word begins with the same consonant, the ending consonant is deleted; e.g., the final /t/ is deleted in the pair "what time."

Rule 2 Palatization. If a word ends in a /d/, and the following word begins with a /y/, then (optionally), the final sound can be converted to a /jh/, and the initial sound of the following word is deleted. Thus the words "did you" can be spoken as either /d ih d y uw/ or /d ih jh uw/.

Rule 3 Plosive deletion. If a word ends in the nasal /n/ followed by a plosive sound, and the following word begins in a plosive sound, then the final plosive in the initial word is deleted. Thus the words "went down" can be spoken as /w eh n d aw n/.



The complete set of rules is available in Reference [15].

#### 8.9.5.2 Variance Estimation Problems

Perhaps the most difficult problem, in training, when using continuous density HMMs, is the estimation of mixture variances when the amount of training data is small (as is almost always the case when using context-dependent units). The problem is that to maximize the likelihood on the training data, the estimation procedure often tries to make the variance very small (i.e., choosing data samples that are very close to each other in value). Although this leads to good training likelihood scores, it often provides poor matches to independent test data. Hence, some protection against variances getting too small in training is required.

Several proposals have been made as to how to realistically control the variance of the estimates to prevent such effects from occurring. One simple one is to tie variances across units, states, and even words, ultimately leading to a grand variance for each spectral component that is independent of the unit, state, and word. This idea is reasonable and has been shown to work well in practice [20]. An alternative is to set a floor on the variance of each spectral component that is based on a statistical analysis of the range of values of the variance component for different sounds, states, etc. Thus rather than using a grand variance, the concept of setting a variance clipping threshold at an appropriate point of the distribution (e.g., 2 sigma below the mean) preserves the (reasonable) range of variance estimates while at the same time preventing the variance from getting unreasonably small. (One could also argue that a high clipping threshold would prevent the variance from getting unreasonably large; in practice, the process of maximizing likelihood prevents this from happening. See Section 6.5 for a more complete discussion of these concepts.)

### 8.9.6 Recognition Results Using CD Units

A key issue in continuous speech recognition is the total number of subword units used in the system. We have already discussed several different types of subword units, including context-independent units, intraword context-dependent units, interword context-dependent units, and various combinations of these. In later sections of this chapter we will extend the unit classes to include position-dependent units, function word-dependent units, and even function phase-dependent units.

On the one hand, it seems clear that as we add more units with greater context dependency, the performance of the recognition should continue to improve. On the other hand, for a fixed training set, the amount of training data available for estimating model parameters of context-dependent units becomes smaller as the number of units increases. Hence the reliability of the estimates of model parameters decreases and therefore recognition performance falls. (To combat this second effort, various smoothing and interpolation procedures have been devised.) The overall result is that recognition performance is maximized for a finite size subword unit set whose size depends on the training data, the recognition vocabulary, the task syntax, and the method for creating the context-dependent units. In this section we present several results illustrating this trade-off between number of subword units and overall word accuracy of the recognizer.

**TABLE 8.5.** Word error rates as a function of occurrence threshold for the feb 89 test set using intraword units with a 38 component/vector analysis.

| Threshold | ∞ | 30[a] | 25[a] | 20[a] | 15[a] | 10[a] |
|---|---|---|---|---|---|---|
| Number of Units | 47 | 1090 | 1215 | 1444 | 1694 | 1874 |
| WP Word Error (%) | 14.0 | 6.7 | 7.0 | 7.1 | 7.4 | 7.6 |
| NG Word Error (%) | 40.0 | 25.0 | 24.8 | 25.0 | 25.2 | 25.6 |

[a]Sixteen mixtures per state were used for these model sets.

**TABLE 8.6.** Word error rates as a function of occurrence threshold for the feb 89 test set using both intraword and interword units (independently) with a 38 component/vector analysis.

| Threshold | ∞ | 30[a] | 25[a] | 20[a] | 15[a] | 10[a] |
|---|---|---|---|---|---|---|
| Number of Units | 47 | 1769 | 2125 | 2534 | 2985 | 3863 |
| WP Word Error (%) | 9.1 | 4.6 | 4.7 | 4.6 | 4.7 | 5.3 |
| NG Word Error (%) | 32.7 | 20.8 | 19.8 | 19.4 | 20.6 | 20.9 |

[a]Sixteen mixtures per state were used for these model sets.

For evaluating speech-recognition performance, we use the feb 89 test set of 300 sentences spoken by 10 adult male and female talkers (30 sentences per talker). Using, as a baseline system, the recognizer based on the 47 context-independent units with 256 mixtures per state, the unit reduction rule was used at several thresholds to generate unit sets with up to 1874 intraword units (no interword units were used here), and the tests were done using cepstral plus differential cepstral (delta and delta-delta) with differential energy (first and second order) parameters (38/vector) with both the word pair (WP) and no grammar (NG) syntaxes. The word recognition accuracies for these systems are given in Table 8.5. A significant improvement in performance is achieved when adding context-dependent intraword units (e.g., from 14% to 9.2% error rate for the WP case); however, increasing the total number of units from 638 to 915 or 1759 or 2340 does not reduce word error rate but instead increases it slightly. This is the tradeoff referred to above. For the NG syntax a similar trend is observed, although the recognizer performance is relatively flat for a large range of units.

The results when using both intraword and interword units (independently, see next section) are shown in Table 8.6. For this test, we again used the feb 89 test set; however, we used the full 38 component/vector analysis frame (including delta-delta cepstral values, delta energy, and delta-delta energy). The effects of the enhanced analysis frame are seen in the improved performance of the 47 PLU set where the error rate falls from 14% to 9.1% for the WP syntax, and from 40% to 32.7% for the NG syntax. Similarly the use of interword units (along with the enhanced analysis) reduced the error rate to 4.6% for the WP case (using 1769 units) and to 19.4% for the NG case (using 2534 units). Again



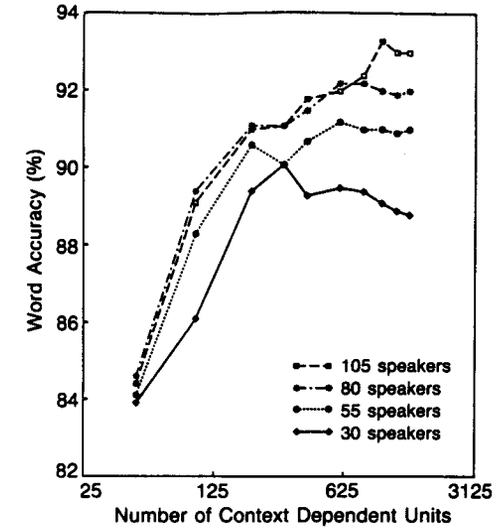



**Figure 8.17** Word accuracy (%) as a function of the number of generalized triphone models for several training set sizes (after Lee [2]).

we clearly see the saturation in performance as the number of units increases due to the problems in reliably estimating parameters of the context-dependent models from a finite training set.

To illustrate the effects of training set size on recognition performance even more dramatically, Figure 8.17 (due to K. F. Lee) shows a plot of word accuracy versus number of units (called generalized triphone models in Lee's notation) for different size training sets (measured in terms of the number of speakers in the set). For the smallest training set (30 speakers), the word accuracy peaks at around 300 units and then falls dramatically beyond this point. For the 55-speaker training set the word accuracy peaks at around 625 units and then falls slightly. For the 80 and 105 speaker sets the performance peaks at about 1000 and 2000 units. These results dramatically illustrate the difficulties in creating subword unit sets with a large number of context-dependent units.

### 8.9.7 Position Dependent Units

When using both intraword and interword units, it is natural and reasonable to combine occurrences of the same unit independent of whether they occurred within the word or across words. It has been observed that phones within words are significantly more stable, acoustically, than phones occurring at word boundaries. Thus it seems plausible that the spectral behavior of the same intraword and interword unit could be considerably different.

To illustrate this point, two sets of context-dependent subword units were created using the same unit reduction rule [21]. In one set common occurrences of intraword and interword units were combined; in the other set they were modeled independently according to their positions within the words or across words (thus the name position dependent). Using a threshold of 30, there were 1282 combined units, including 1101 left-right context units, 99 left-context units, 35 right-context units, and 47 CI units, and 1769 separate position-dependent units including 913 intraword units and 856 interword units.

To show that the spectral properties of these two sets were different, the histograms of unit separation distances for the two sets were computed as follows. For each unit, $\lambda_p$, in each set, we computed the minimum distance (likelihood separation) as

$$D(p) = \min_{q \neq p}\{L(Y_p|\lambda_p) - L(Y_p|\lambda_q)\}$$

where $Y_p$ represents the training data segments used to estimate $\lambda_p$. $D(p)$ represents the smallest likelihood score difference when using any other model than the one created from $Y_p$. (In practice, the computation is performed only for models, $\lambda_q$, which had the same base unit as $\lambda_p$ since all other models gave significantly larger difference scores.) The histograms of unit separation distance for the 1282 PLU set of combined intraword and interword units, and for the 1769 PLU set of position dependent units are shown in Figure 8.18. For the 1769 PLU set almost *all* of the unit separation distances are larger than 2.0 (difference in log likelihoods), including the cases where the same unit occurred in both intraword and interword contexts. The average unit separation distance for this set is about 9.0. For the 1282 PLU set the histogram is skewed to the left, showing many small unit separations, with an average distance on the order of 4–5. The results clearly show that the spectral properties of context-dependent units are often significantly different within words than when they occur at word boundaries.

### 8.9.8 Unit Splitting and Clustering

We have shown in previous sections that it is relatively simple to train models for a small set of context-independent units from a training set of a reasonable size. The problem is that the recognizer performance is not good enough for large vocabulary continuous speech-recognition tasks. We also discussed one simple way to train models for a large set of context-dependent units from the same training set. Here the problem is the inadequacy of training data, which leads to poor estimates of model parameters for all but a small subset of the units observed in a typical training set. The result of the poor model estimates is that recognition performance saturates for about 1000–2000 context-dependent units and either remains constant or decreases as the number of units trained increases.

Thus a key issue in the design and implementation of large vocabulary continuous speech recognizers is how to efficiently determine the number and character of the context-dependent units that give best recognition performance for a given training set. Unfortunately, there is no simple answer to this question. In this section we discuss several proposed methods based on the concepts of either starting from a small set of context-independent units and iteratively splitting the units, or of starting from a large set



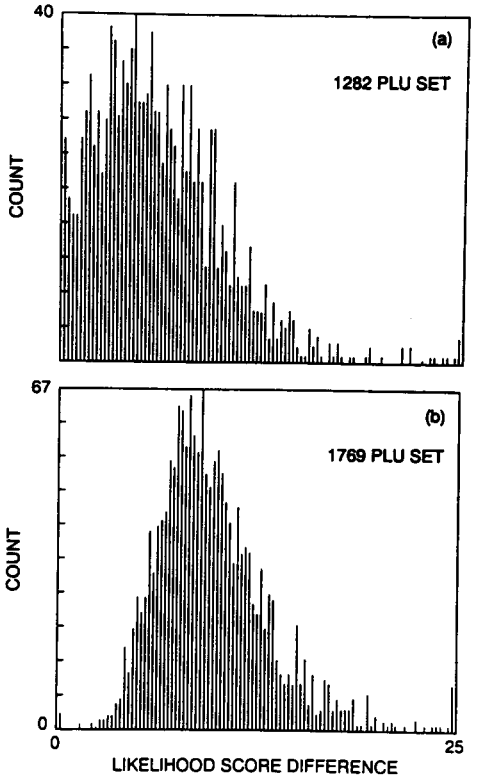



**Figure 8.18** Histograms of the unit separation distance for (a) combined intraword and interword units (1282 PLUs) and (b) separate intraword and interword units (1769 PLUs) (after Lee et al. [19]).

of context-dependent units and merging similar units to reduce the number of units based on some type of clustering procedure.

#### 8.9.8.1 Splitting of Subword Units

The basic idea of subword unit splitting is illustrated in Figure 8.19. We assume that for each subword unit $p_i$ (with model $\lambda_i$), representing a context-independent unit, there is some inherent internal distribution of training tokens that naturally clusters into two or more clusters. (Within the figure we show three clusters, namely $p_i^1, p_i^2$, and $p_i^3$.) The clusters represent classes of sounds that are all labeled as $p_i$, but which have different spectral

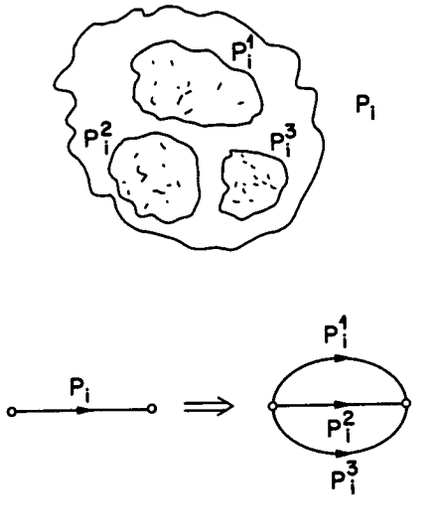



**Figure 8.19** Splitting of subword unit $p_i$ into three clusters (after Lee et al. [7]).

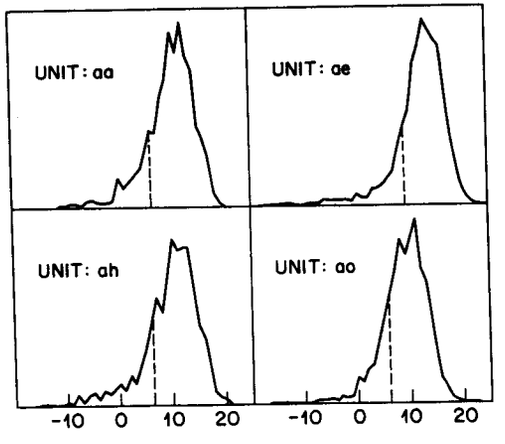



**Figure 8.20** Histograms of likelihood score for four context-independent units (vowels) (after Lee et al. [7]).



properties depending on the context in which they occur. Once the separation of $p_i$ into clusters is achieved, we have effectively created multiple models of the context-independent subword unit, as shown at the bottom of Figure 8.19.

There are several ways to create the clusters for each unit, but a particularly simple (and meaningful) one is based on the following argument. If we examine the histogram of (log) likelihood scores for each training token which is labeled $p_i$ we get curves similar to those shown in Figure 8.20. The likelihood score histograms show that for a large percentage of the training tokens, good scores are obtained using the context-independent unit. These training tokens are (relatively) well represented by $\lambda_i$ and do not need to be split off. Instead the low tail of the histogram (below the dashed lines) represents training tokens whose likelihood scores are relatively low and these tokens need an alternative representation (model) to be well represented.

Based on the above discussion, a simple procedure for splitting off training tokens with low likelihood scores and creating a new model from these tokens is as follows:

1. For each subword unit, $p_i$, which is to be split (not every unit need be split), all training tokens whose likelihood scores fall below a threshold are split off and used to estimate an additional model for that unit.
2. The segmental $k$-means training procedure is iterated on the split-off tokens until the new model reaches convergence.
3. The above procedure (steps 1 and 2) is iterated until the desired number of models, for each subword unit, is obtained.

The results (in terms of average likelihood score over the entire set of units) of applying the above splitting procedure to the 47 PLU set of context-independent units is shown in Figure 8.21. The results are shown, as a function of iteration number, for splitting each of the 47 models into 2, 3, and 4 models. The procedure converges rapidly and provides small but consistent increases in average likelihood scores.

The above model splitting procedure leads to one major difficulty, namely, How do we modify the word pronunciation dictionary to account for the presence of multiple versions of each subword unit? The inherent problem is illustrated in Figure 8.22, which shows the networks for a complete set of word models assuming every version of each sound in the word can follow every version of every other sound in the word (part a), or that instead we determine one or two best representations of each word via some type of word learning procedure (part b). The problem with the network of part a is that a word with $N$ sounds (e.g., "often" has three sounds, /ao, f, en/) has $2^N$ representations when we use the complete network (e.g., eight versions of "often") with two models for each sound. This means not only more computation, but even worse, more chances to cause word insertion or substitution in the recognition phase because there are so many more ways in which the words can occur. The network of part b, in which we explicitly enumerate the version of each unit used for each word, is far more viable; however, the problem is how to estimate the best sequence of units for each word in the lexicon. To do this properly we need occurrences, within the training set, of each word in the vocabulary from which we use the network of part a and backtrack to get the best sequences of the type shown in part b.

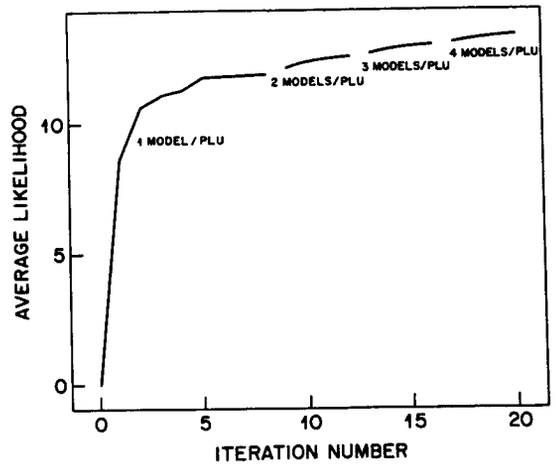



**Figure 8.21** Average likelihood scores for sets of PLU models obtained from model splitting (after Lee et al. [7]).

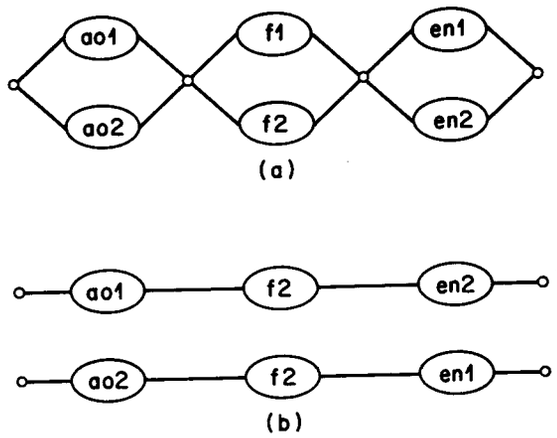



**Figure 8.22** Word networks based on all combinations of units (a), or selected combinations of units (b) (after Lee et al. [7]).



For words that don't occur in the training set, some canonic representation must be relied on (e.g., use the primary model for each sound as a default). The necessity of having each word occur in the training makes this type of splitting method less viable than alternative procedures.

#### 8.9.8.2 Clustering of Context Dependent Units

The alternative to model splitting (a top-down approach) is model clustering (a bottom-up approach) in which we initially start with the complete set of context-dependent units (as many as exist in the training set based on a count threshold of 1) and then sequentially merge units (actually the training tokens associated with the units) so that the decrease in likelihood score is minimized at each step. (In practice, we merge only units whose contexts are comparable, e.g., unit $p_i - p_j - p_k$ could be merged with units $p_i - p_j - p_\ell$, $\ell \neq k$, or $p_\ell - p_j - p_k$, or $\$ - p_j - p_k$, or $p_i - p_j - \$$, or $\$ - p_j - \$$.) This procedure is iterated either until a desired number of units is reached or until the resulting decrease in likelihoods gets too large.

A key advantage to model clustering is that it is trivial to modify the word lexicon to account for the decrease in units from merging. The procedure is basically to change each occurrence of both $p_i - p_j - p_k$ and $p_\ell - p_j - p_n$ to the merged unit, call it $\hat{p}_i - p_j - p_k$, whenever they occur in the lexicon. Thus model clustering is inherently simpler to implement than model splitting and therefore has been used more widely in practical systems.

Many variations on model clustering have been proposed, including knowledge-based allophonic clustering [22], in which specific knowledge of the vowel and consonant contexts is explicitly used to guide the clustering procedure, and CART-based phonetic clusters in which a decision tree is used to choose the most reasonable clustering sequence based on phonetic considerations.

### 8.9.9 Other Factors for Creating Additional Subword Units

In practice, the training methods for creating robust, complete sets of subword unit models for representing continuous speech are up against hard physical limits, including amount of training data and ability to reliably estimate model parameters from insufficient training. To obtain improvements in recognition performance, subject to the above constraints, several ideas have emerged for creating specialized units and models. For completeness, we briefly outline several interesting proposals that have been advanced along these lines.

A key source of difficulty in continuous speech recognition are the so-called function words, which include words like a, and, for, in, and is. These function words have the following properties:

1. They are generally unstressed in speech.
2. They are poorly articulated in continuous speech.
3. They are highly variable in pronunciation depending on context.

4. They account for a large percentage of the word recognition errors in continuous speech (upward of 50–70% in some tests).

To combat these problems, one simple idea is to represent function words independently of the rest of the training set, using either whole-word models, multiple pronunciations in the lexicon (e.g., the, thee), or special subword units, called function word dependent units, trained directly from occurrences of the function words within the training set. Experience shows small but consistent improvements in recognition performance when function word dependent units are added to the standard set of subword units.

The idea of representing function words can be extended to the representation of function phrases such as "in the," or "what is." Thus, specialized units can be created for these combinations in much the same way as for individual function words. Again there are small performance gains that are achieved when using function phrase units.

Another interesting idea is to create separate sets of units for both male and female talkers. The idea is that the spectral properties of the units are distinct for males and females. The problem is that by separating male from female talkers, the amount of training data for each separate gender set is reduced. Hence the reliability of the estimates of both sets of models is reduced even further. Experience again shows small, consistent gains in recognition performance using gender-specific models; hence this method is worth considering for practical implementations.

Finally it has been proposed that a combination of word models and subword unit models might give the best performance for specific tasks. The idea is that for words that do occur often in the training set (e.g., function words), creation of whole-word models provides the highest recognition performance. For all other words in the lexicon, some type of subword units is required. Hence a combination of word and subword units would probably lead to the best implementation for many applications. This idea has yet to be evaluated in a practical application.

### 8.9.10 Acoustic Segment Units

In this chapter we have shown that large vocabulary continuous speech recognition systems use a combination of ideas from phonetics and acoustics to define subword units and to create a "consistent" framework for training the units and implementing the overall recognition structure. The resulting system is neither phonetically nor acoustically consistent, but is instead a hybrid of the two methodologies. This is why the resulting subword units are called phonelike units (PLUs) rather than phones or allophones.

In an attempt to create a consistent acoustic framework (devoid, in theory, of the phonetic basis), it is possible to define a set of acoustic segment units (ASUs) that can be trained from continuous (unlabeled) speech, and which form a basis for representing any spoken input. In concept, all one need do is to have a procedure that automatically segments fluent speech into unlabeled sections [9, 23] (based on a maximum likelihood procedure using some type of spectral similarity measure), and then cluster the resulting segments to create a codebook of ASUs.



The problem now becomes one of creating an acoustic lexicon that represents words in the recognition vocabulary in terms of the appropriate sequence of ASUs. For systems in which every vocabulary word is seen in the training set, techniques for creating the acoustic lexicon exist and appear to work well [9, 24]. However, for large vocabulary systems the problem of automatically creating the acoustic lexicon remains a major obstacle to the practical use of ASUs.

## 8.10 CREATION OF VOCABULARY-INDEPENDENT UNITS

A major limitation in the training procedures discussed in this chapter is that the resulting subword unit models are not truly vocabulary independent. This is because the unit models are generally trained from tokens that occur in only a small subset of the possible contexts, and this subset is from the same words as used in the recognition tests. As such, the resulting units are word/vocabulary dependent and do not perform well for tasks in which different vocabularies and task syntaxes are used.

To alleviate this problem of vocabulary dependence, the "ideal" training procedure would be to use a training set that is completely independent of the test material, both in vocabulary and in syntax. If a sufficiently large training set is available, the units models will eventually converge so that the resulting recognition performance is virtually independent of the vocabulary and task.

To evaluate this idea, two experiments were run at CMU [25]. Using a vocabulary-independent training set of 15,000 sentences, subword unit models were created from subsets of 5000 (VI–5000), 10,000 (VI–10000), and 15,000 (VI–15000) sentences and tested against two tasks, namely a 122-word office correspondence task, and the 911 RM task. The results of these two experiments are given in Tables 8.7 and 8.8. For comparison, in Table 8.7, results based on training models from 1000 sentences from the office correspondence task (VD–1000) are also given. For 5 times the size training set the error rate from VI–5000 models is more than twice that of the VD–1000 models. Even with 15 times as much training data, the error rate is still somewhat larger for the VI–15000 model than for the VD–1000 model.

An even worse performance is seen for the RM task in which a VI–15000 training set led to almost twice the error rate of the RM–4200 (sentence) training set. (A final test was run at CMU in which new test sentences were recorded at CMU under the same recording conditions as those of the VI–15000 set, and the resulting recognition performance of both the VI–15000 set and the RM–4200 set were comparable. Thus, some of the large differences in performance result from differences in recording conditions.)

The results of these tests show that robust techniques for creating truly vocabulary-independent units are yet to be devised. Until such methods are available, truly continuous speech recognition for unlimited vocabularies and tasks will be out of range. The interim solution is to use VI models and bootstrap them to VD models for specific applications. Experimental evidence exists that such procedures are viable for many applications.

**TABLE 8.7.** Recognition performance on 122-word, office correspondence task with both VI and VD models (after Hon & Lee [25]).

| Training Set | Word Coverage (%) | Triphone Coverage (%) | Word Error Rate (%) |
|---|---|---|---|
| VI–5000 | 44.3 | 63.7 | 23.9 |
| VI–10000 | 63.9 | 95.3 | 15.2 |
| VI–15000 | 70.5 | 99.2 | 13.3 |
| VD–1000 | 100 | 100 | 11.4 |

**TABLE 8.8.** Recognition performance on 991-word, RM task, with both VI and VD models (after Hon & Lee [25]).

| Training Set | Word Coverage (%) | Triphone Coverage (%) | Word Error Rate (%) |
|---|---|---|---|
| VI–15000 | 57.0 | 90 | 15.4 |
| RM–4200 | 100 | 100 | 8.3 |

## 8.11 SEMANTIC POSTPROCESSOR FOR RECOGNITION

The final stage of processing in most speech recognizers is a semantic processor whose job is to eliminate from consideration all semantically meaningless sentences. In a sense, the semantic processor exploits the fact that the syntax used in recognition has a great deal of overcoverage; that is, it allows meaningless sentences to be passed to the semantic analyzer. The semantic processor can use the actual perplexity of the task (generally much lower than the perplexity of the syntax) to convert the recognized output to a semantically valid string.

In theory, the semantic processor should be able to communicate back to the recognizer to request a new string whenever the resulting string is deemed invalid. In practice, one of two simple strategies can be used; either the recognizer can generate a list of the best $N$ sentences ($N = 500 - 1000$) that the semantic processor can search until a valid one is found, or it can assume that the best (recognized) string is semantically "close" to the correct string and therefore process it appropriately to determine a valid approximation.

Rather than discussing the details of how such semantic processing is done in practice, Figure 8.23 shows plots of improvements in word and sentence accuracy for different sets of subword units due to the use of a simple semantic postprocessor for the RM task [26]. Improvements in word accuracy of up 10% and improvements in sentence accuracy of over 20% are achieved, even with simple processing.

## 8.12 SUMMARY

The framework of large vocabulary, continuous speech recognition is well established.



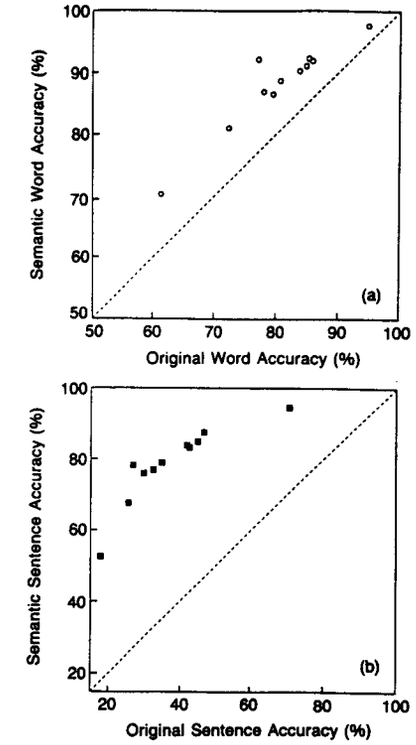



**Figure 8.23** Word and sentence accuracy improvements in RM after semantic processing (after Pieraccini and Lee [26]).

Techniques for training subword models have been developed and work well in practice. Recognition systems have been developed and these, too, work well in practice. Recognition systems have been implemented with upward of 1000–20,000 word vocabularies using upward of 1000–2000 subword units. Many unanswered questions remain. A key one is how to efficiently choose and design context-dependent, vocabulary-independent units from training sets of reasonable (but finite) size. Other issues concern effectiveness of different spectral representations, including codebooks and tied-mixture densities, efficiency of implementation of search strategies, and efficient implementations of task syntax. Finally, the issues involved with task semantics are yet to be fully understood or resolved. Large vocabulary recognition has come a long way, but a great deal remains to be done before such systems will be used for practical applications.

## REFERENCES

[1] F. Jelinek, "The Development of an Experimental Discrete Dictation Recognizer," *Proc. IEEE*, 73: 1616–1624, November 1985.

[2] K.F. Lee, *Automatic Speech Recognition—The Development of the SPHINX System*, Kluwer Academic Publishers, Boston, 1989.

[3] D.B. Paul, "The Lincoln Robust Continuous Speech Recognizer," *Proc. ICASSP 89*, Glasgow, Scotland, pp. 449–452, May 1989.

[4] R. Schwartz et al., "The BBN BYBLOS Continuous Speech Recognition System," *Proc. DARPA Speech and Natural Language Workshop*, pp. 94–99, February 1989.

[5] M. Weintraub et al., "Linguistic Constraints in Hidden Markov Model Based Speech Recognition," *Proc. ICASSP 89*, Glasgow, Scotland, pp. 699–702, May 1989.

[6] V. Zue, J. Glass, M. Phillips, and S. Seneff, "The MIT Summit Speech Recognition System: A Progress Report," *Proc. DARPA Speech and Natural Language Workshop*, pp. 179–189, February 1989.

[7] C.H. Lee, L.R. Rabiner, R. Pieraccini, and J.G. Wilpon, "Acoustic Modeling for Large Vocabulary Speech Recognition," *Computer Speech and Language*, 4: 1237–165, January 1990.

[8] A.E. Rosenberg, L.R. Rabiner, J.G. Wilpon, and D. Kahn, "Demisyllable Based Isolated Word Recognition," *IEEE Trans. on Acoustics, Speech, and Signal Proc.*, ASSP–31: 713–726, June 1983.

[9] C.H. Lee, F.K. Soong, and B.H. Juang, "A Segment Model Based Approach to Speech Recognition," *Proc. ICASSP 88*, New York, pp. 501–504, April 1988.

[10] J.R. Bellegarda and D. Nahamoo, "Tied Mixture Continuous Parameter Models for Large Vocabulary Isolated Speech Recognition," *Proc. ICASSP 89*, Glasgow, Scotland, pp. 13–16, May 1989.

[11] X.D. Huang and M.A. Jack, "Semi-Continuous Hidden Markov Models for Speech Signals," *Computer Speech and Language*, 3: 239–251, 1989.

[12] F. Jelinek and R.L. Mercer, "Interpolated Estimation of Markov Source Parameters From Sparse Data," *Pattern Recognition in Practice*, E.S. Gelsema and L.N. Kanal, Eds., North-Holland Pub. Co., Amsterdam, pp. 381–397, 1980.

[13] P.J. Price, W. Fischer, J. Bernstein, and D. Pallett, "A Database for Continuous Speech Recognition in a 1000-Word Domain," *Proc. ICASSP 88*, New York, pp. 651–654, April 1988.

[14] R. Schwartz et al., "Robust Smoothing Methods for Discrete Hidden Markov Models," *Proc. ICASSP 89*, Glasgow, Scotland, pp. 548–551, May 1989.

[15] E. Giachin, C.H. Lee, L.R. Rabiner, A.E. Rosenberg, and R. Pieraccini, "On the Use of Interword Context-Dependent Units for Word Juncture Modeling," *Computer Speech and Language*, 6: 197–213, 1992.

[16] C.H. Lee, E.P. Giachin, and A.E. Rosenberg, "Word Juncture Modeling Using Phonological Rules for HMM Based Continuous Speech Recognition," *CSELT Tech. Reports*, 18 (3): 189–194, June 1990.

[17] C.H. Lee, L.R. Rabiner, and R. Pieraccini, "Speaker Independent Continuous Speech Recognition Using Continuous Density Hidden Markov Models," Proc. NATO-ASI, Speech Recognition and Understanding: Recent Advances, Trends and Applications, P. Laface and R. DeMori, Eds., Springer-Verlag, Cetraro, Italy, pp. 135–163, 1992.



[18] R. Pieraccini and A.E. Rosenberg, "Automatic Generation of Phonetic Units for Continuous Speech Recognition," *Proc. ICASSP 89*, Glasgow, UK, pp. 623–626, May 1989.

[19] R. Pieraccini, C.H. Lee, E. Giachin, and L.R. Rabiner, "An Efficient Structure for Continuous Speech Recognition," Proc. NATO-ASI, Speech Recognition and Understanding: Recent Advances, Trends and Applications, P. Laface and R. DeMori, Eds., Springer-Verlag, Cetraro, Italy, pp. 211–216, 1992.

[20] D. Paul, "A Speaker Stress Resistant Isolated Word Recognizer," *Proc. ICASSP 87*, Dallas, TX, pp. 713–716, April 1987.

[21] C.H. Lee, E. Giachin, L.R. Rabiner, R. Pieraccini, and A.E. Rosenberg, "Improved Acoustic Modeling for Large Vocabulary Continuous Speech Recognition," *Computer Speech and Language*, 6: 103–127, 1992.

[22] L. Deng et al., "Acoustic Recognition Component of an 86,000 Word Speech Recognizer," *Proc. ICASSP 90*, Albuquerque, NM, pp. 741–744, April 1990.

[23] T. Svendsen and F.K. Soong, "On the Automatic Segmentation of Speech Signals," *Proc. ICASSP 87*, Dallas, TX, pp. 77–80, April 1987.

[24] L.R. Bahl, P.F. Brown, P.V. de Souza, R.L. Mercer, and M.A. Picheny, "Automatic Construction of Acoustic Markov Models for Words," *Proc. Int. Symposium on Signal Processing Applications*, (Brisbane, Australia), 565–569, 1987.

[25] H.W. Hon and K.F. Lee, "On Vocabulary Independent Speech Modeling," *Proc. ICASSP 90*, Albuquerque, NM, pp. 725–728, April 1990.

[26] R. Pieraccini and C.H. Lee, "Factorization of Language Constraints in Speech Recognition," *Proc. 29th Annual Meeting of the Association for Computational Linguistics, Berkeley, CA, June 1991.*

Chapter 9

# TASK-ORIENTED APPLICATIONS OF AUTOMATIC SPEECH RECOGNITION

## 9.1 INTRODUCTION

Throughout this book we have been concerned primarily with the basic principles behind the design and implementation of systems for speech recognition by machine. Although we have discussed several canonic recognition problems (e.g., isolated digit recognition, connected digit recognition, fluent speech recognition on the Resource Management task), we have avoided any discussion of two key aspects of speech recognition, namely, issues in real-time hardware implementations of speech recognizers and task-specific applications of speech recognition. For several reasons, we choose not to discuss the problem of how to build real-time hardware for speech recognition. First, the field of digital hardware is rapidly changing, and any specific implementation would be out-of-date and would not be state-of-the-art over time. Second, there are often many ways of achieving real-time implementations, and the choice of hardware architectures is often dominated by considerations other than that needed for speech recognition. Finally, an understanding of real-time implementations of speech-recognition systems contributes little to an improved understanding of the basic speech-recognition technology.

Thus, in this chapter, we will discuss the problem of how to integrate a speech-recognition system into a task-specific application to perform a useful task. To better understand this problem, consider the "Task Specific Voice Control and Dialog" system of Figure 9.1. The overall system consists of a speech recognizer, a language analyzer, an expert system, a physical system being controlled by the voice commands, and a text-





TASK-SPECIFIC VOICE CONTROL AND DIALOG

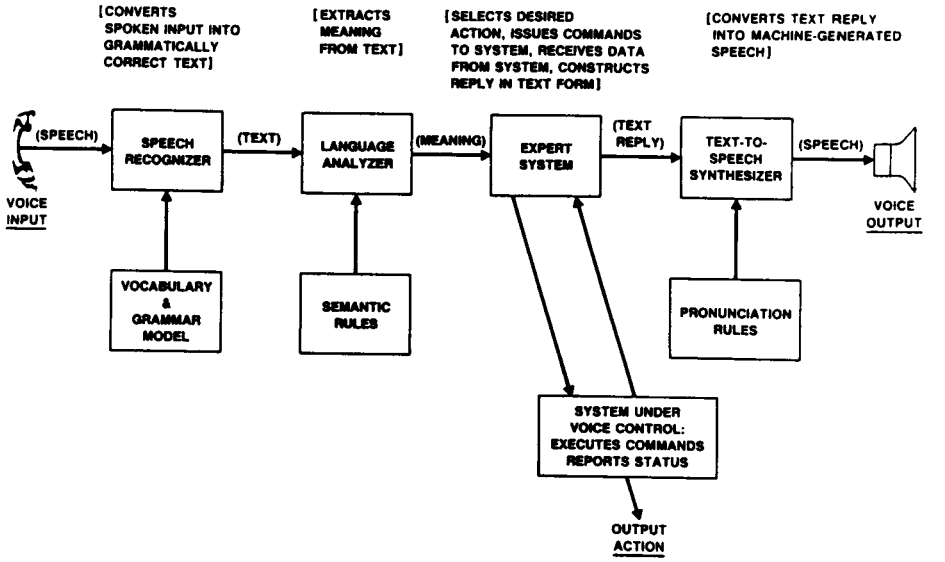



**Figure 9.1** Block diagram of a task-specific voice control and dialog system.

to-speech synthesizer. The broad function of the speech recognizer is to convert spoken input into grammatically correct text as constrained by the recognizer vocabulary and grammar model. In theory we would like the speech input, to the recognizer, to be totally unconstrained—that is, natural language; in practice we have seen that we have to restrict the input speech somewhat for it to be recognized correctly based on the techniques described throughout this book. The output of the speech recognizer is the text string that is most likely to have been spoken based on the recognizer's vocabulary and grammar. (It should be noted that, as shown throughout this book, the recognizer vocabulary and grammar can vary over a wide range. For example, simple systems may use small vocabulary sizes ("yes"—"no," or "off hook," "dial," "hangup") without an explicit grammar, whereas more sophisticated systems may be required to recognize tens of thousands of words with constrained word grammars.) The text string is next sent to a language analyzer which, using a set of semantic rules appropriate to the task being performed, extracts the meaning from the text. The language analyzer can be implemented as a semantic concept spotter using statistical methods, or as a rule-based system. The decoded meaning of the input speech is sent to an expert system, which first selects a desired action, then issues appropriate commands to the physical system under voice control to carry out the action, then receives data on the

**TABLE 9.1.** Performance scores for several types of speech-recognition systems as measured under laboratory conditions.

| Technology | Task | Syntax | Mode | Vocabulary | Word Error Rate (%) |
|---|---|---|---|---|---|
| Isolated Words | None | None | SD | 10 Digits | 0 |
| | | | | 39 Alphadigits | 4.5 |
| | | | | 1109 Basic English | 4.3 |
| | | | SI | 10 Digits | 0.1 |
| | | | | 39 Alphadigits | 7.0 |
| | | | | 129 Airline Words | 2.9 |
| Connected Words | Digit Strings | Known Length | SD | 10 Digits | 0.1 |
| | | String | SI | 11 Digits | 0.2 |
| | Airline Reservations | Finite-State Grammer (Perplexity = 4) | SD | 129 Airline Words | 0.1 |
| Fluent Speech | Naval Resource Management | Finite-State Grammer (Perplexity = 60) | SI | 991 Words | 4.5 |

command status (e.g., "command carried out successfully," or "command cannot be carried out due to . . ."), and constructs a textual reply that is converted into a speech message (using a text-to-speech synthesizer with appropriate word pronunciation rules) and played back to the user. The entire system of Figure 9.1 is a voice dialog control system based on speech input and output that performs a specified task of interest. It should be clear from Figure 9.1 that the speech-recognition module, although significant in a computational sense, is often only a small piece of an overall task-oriented application. In this chapter we will discuss several such systems as used in a broad range of applications. Before going into specifics of individual task applications, we first provide some generic comments about speech recognition applications.

## 9.2 SPEECH-RECOGNIZER PERFORMANCE SCORES

To appreciate the range of tasks to which speech recognition has been applied successfully, it is worthwhile to review the state-of-the-art in laboratory benchmarks of speech-recognition systems, as discussed in earlier chapters. Table 9.1 shows the performance of a wide range of speech-recognition systems, as measured under laboratory conditions (i.e., high-quality microphone, low noise environment). The performance of these systems is given in terms of a word error rate score (in %) as measured for a specified technology, for a given task, with a specified task syntax, in a specified mode, and for a specified word vocabulary. For tasks in which the technology of isolated word recognition is suitable, without specification of either task or task syntax, word error rates of below 5% in speaker-dependent (SD) mode are obtained; for speaker-independent (SI) mode the word error rate is at or below 7% for a wide range of word vocabularies. These word error rates are rather low, and when well-defined tasks with their appropriate syntax are specified (providing additional



constraints on the recognized sequence of isolated words), performance often improves by an order of magnitude or more. (We will see examples of this later in this chapter.)

For tasks based on connected word recognition, such as connected digit strings, word error rates on the order of 0.1–0.2% are obtained in both speaker-dependent and speaker-independent modes for known length (prescribed number of digits) strings. Such high accuracy makes recognition of personal identification numbers (PIN) codes, telephone numbers, credit card numbers, and catalog codes feasible, especially when the additional syntax imposed by each of these applications is taken into account.

Finally, for fluent speech recognition of a 991-word vocabulary for the Naval Resource Management task, with a task perplexity (average word branching factor) of 60, a word error rate on the order of 4% is state-of-the-art performance in a speaker-independent mode. In this case, a word error rate of 4% is equivalent to sentence error rates on the order of 20%; hence, this technology is not yet suitable for real-world task applications.

All the performance scores in Table 9.1 were obtained under laboratory testing conditions. Typically, word error rates increase by a factor of 2 to 5 when tested in realistic environments with a broad range of speaker accents, noise conditions, speaker errors including hesitations, *ah*s and *uhm*s, and other variables. Fortunately, task constraints serve to reduce error rates by equivalent factors of from 2 to 10; hence the quoted word error rates of Table 9.1 are often realistic for actual experimental evaluations in real-world tests.

## 9.3 CHARACTERISTICS OF SPEECH-RECOGNITION APPLICATIONS

To decide whether a proposed task is suitable for speech-recognition deployment, several requirements are essential. These include the following:

- The proposed recognition system must provide a real (and hopefully measurable) benefit to the user in the form of increased productivity, ease of use, better human-machine interface, or a more natural mode of communication. Many proposed applications have tried, generally unsuccessfully, to exploit the novelty of voice recognition to attract attention or to get increased sales. Without the true (and lasting) benefit to the user, such applications do not succeed over time.
- The proposed recognition system must be "user friendly"; that is, it must make the user feel comfortable with the voice dialogue, it must provide friendly and helpful voice prompts, and it must provide an effective means of communications (i.e., a fall-back mode) when the recognizer fails to understand properly the spoken commands. The concept of a user-friendly system is essential to the maintenance of a voice dialogue between the user and the machine.
- The proposed recognition system must be accurate; that is, it must achieve, at least, a specified level of performance on the task associated with the recognition decision. Interestingly, there appears to be a nonlinear perception of the effectiveness of a recognizer in that the absolute level of performance is relatively unimportant so long as the accuracy exceeds some specified level. For example, users have a

great difficulty discerning differences between isolated word recognition systems that provide 95% word accuracy, and those which provide 99% word accuracy. This is because the 95% word accuracy system only makes an error, on average, once in 20 tries, whereas the 99% word accuracy system only makes an error, on average, once in 100 tries. Both systems appear, to the user, to be highly accurate and rarely make errors; hence when an occasional error is made, the user tends to attribute it to an improper speaking mode on his (or her) part, rather than to an inadequacy in the recognizer. Similarly when the performance of the recognizer falls below some lower threshold (e.g., 90% word accuracy), the perception of the user is that the system makes too many errors and is therefore unreliable. The actual word error is again almost irrelevant; the user perceives the system to be unusable.

- The proposed recognition system must respond in real time. It is essential that the user feel that the response to the query comes essentially immediately (within 250 msec after the end of the spoken input) so that a voice dialog can be maintained between the user and the system.

Every one of the above requirements is essential and mandatory for a proposed use of speech recognition to a specified task to be successful.

### 9.3.1 Methods of Handling Recognition Errors

Given the fact that a speech recognition system will make errors in recognition of the spoken input, a key question is how do we handle such errors in a way in which communication between the user and the machine doesn't break down. There are at least four ways to deal with recognition errors, any one or more of which can generally be applied in various specified tasks. These include

- *Fail soft methods*; in this case, the "cost" of a recognition error, as measured in terms of either user annoyance, or loss in revenue as a result of the misrecognition, is low. Hence the error is essentially accepted with the assumption that it will be detected and corrected at a later stage of interaction with the machine. Hence if a command word is misrecognized and, as a result, the next piece of the dialogue is inappropriate for the actually spoken command, the user enters a correction mode (by spelling a correction command) to backtrack to the point where the error was made. Such a transaction costs the user a small amount of time but has little other effect on the user.
- *Self-detection/correction of errors*; in this case the recognition system utilizes known task constraints to automatically detect and correct recognition errors. Thus, in spelling a name from a finite list of names, it is generally easy to detect and correct recognition errors in the spelled letters because the recognized name is constrained to the set of names within the given list. For applications using digit strings (e.g., catalog ordering, inventory control), the digit strings corresponding to individual items can be chosen to take advantage of known error correcting codes (e.g., Reed-Solomon codes). Thus, within the error correction capability of such codes, digit recognition errors can be detected and corrected.
- *Verification or multilevel decision before proceeding*; in this case the recognition system asks the user for help whenever there are two or more recognition candidates whose likelihood scores are all high and it is difficult resolving small differences in the strings. The recognizer asks the user to verify the first choice decision; if it is not verified, the recognizer asks the user to verify the second choice etc. Alternatively, the recognition system can list the confusing candidates and ask the user to choose the correct string based on the string index number in the list. In this manner, the confusing words are transformed into list numbers (i.e., digits) for which there is considerably less ambiguity than for the originally spoken words.
- *Rejection/pass on to operator*; in this case the recognition system defers making a decision on inputs where either two or more recognition candidates have high likelihood scores, or whenever the highest likelihood score is too low for a reliable recognition decision. By recording all spoken inputs in digital format, the system can reduce the error rate by rejecting a small but finite percentage of the spoken strings, and passing on such strings to a human operator (attendant) who makes the final recognition decision based on listening to the spoken input.

By utilizing any (or all) of the above techniques, the effective accuracy of the speech recognizer approaches 100%; hence the proposed applications can be successfully carried out.



## 9.4 BROAD CLASSES OF SPEECH-RECOGNITION APPLICATIONS

There are five broad classes of applications to which speech recognition has been applied, namely:

1. Office or business systems. Typical applications include data entry onto forms, database management and control, keyboard enhancement, and others.
2. Manufacturing. Typically speech recognition is used to provide "eyes-free, hands-free" monitoring of manufacturing processes (e.g., parts inspection) for quality control.
3. Telephone or telecommunications. A wide range of applications are feasible over dialed-up telephones, including automation of operator assisted services, inbound and outbound telemarketing, call distribution by voice, expanded utility of a rotary phone, repertory dialing, and catalog ordering.
4. Medical. The primary application is voice creation and editing of specialized medical reports.
5. Other. Including voice controlled and operated games and toys, voice recognition aids for the handicapped, and voice control of nonstrategic functions in a moving vehicle (e.g., climate control, the audio system).

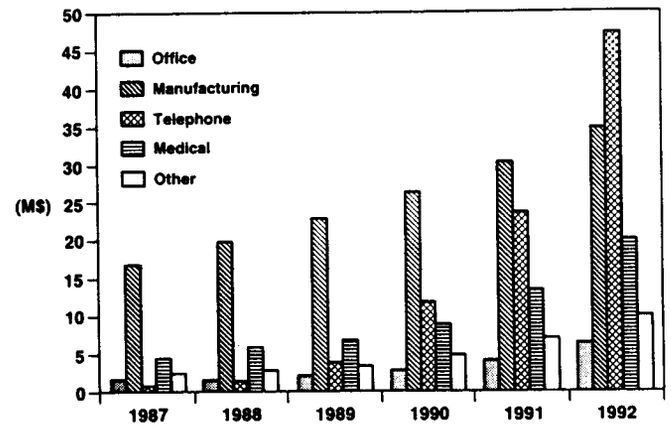



**Figure 9.2** Market sales for speech-recognition hardware over time, in each of five market segments. (Data estimated for 1990–1992 sales.)

To get an idea of the size of the speech-recognition hardware markets in each of these areas, Figure 9.2 shows a plot of dollar volume over time from 1987 to 1992. (The plot was created in 1990 with estimates of sales in the last three years.) Although the manufacturing segment was, by far, the largest piece of the market in 1987, in 1992 the telephone segment was projected to pass the manufacturing segment by a substantial margin. What is even more impressive than this exponential growth in sales in the telephone segment is the projected revenue (not shown) for the telephone services associated with the hardware sales. Many of the proposed recognition applications have service revenue (either via usage or via reduction in costs from automation) of upward of several hundred millions of dollars.

In the remainder of this chapter we will discuss typical speech-recognition applications with the goal of illustrating the range of ways in which speech recognition has been successfully applied. For most cases our discussion will be brief, serving merely to illustrate the range of vocabulary and task syntax used in real-world systems.

## 9.5 COMMAND-AND-CONTROL APPLICATIONS

The canonic term "command and control" applications has been applied to any type of speech recognition system where the user speaks a single command (either an isolated word or phrase, or possibly a connected sequence of words) and the machine, upon correctly recognizing the command, acts appropriately. In this manner the user exercises control over the machine using simple commands. We begin this section with one simple command-and-control application, namely, a voice repertory dialer.



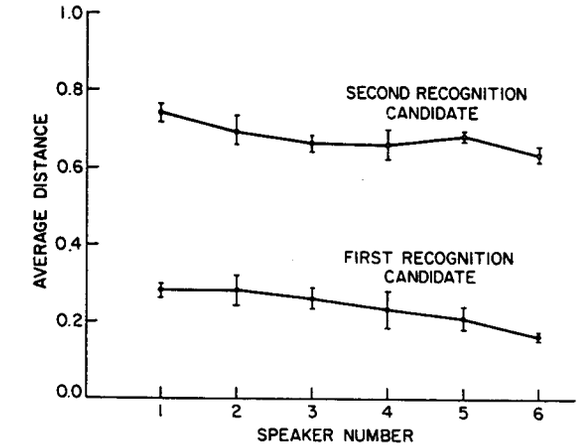



**Figure 9.3** Average LPC distance scores for first- and second-recognition candidates for each of the six test speakers (after Rabiner et al. [1]).

### 9.5.1 Voice Repertory Dialer

A voice repertory dialer is a telephone adjunct that allows a caller to place calls by speaking the name of someone in the repertory, rather than dialing the digit codes associated with the repertory name. Although voice repertory dialers could be of value in a home or business environment, the potentially most valuable place for such a system is in the wireless network (i.e., a cellular phone within a car, or a portable wireless terminal), where the advantages of hands-free, eyes-free, dialing of telephone numbers are clear.

In terms of the recognition system requirements, a typical repertory dialer would need a speaker-trained set of vocabulary patterns corresponding to repertory names (and their associated telephone numbers), as well as a speaker-independent set of vocabulary patterns corresponding to the digits (for all digit voice dialing) and a set of command words for controlling normal telephone features (e.g., off-hook, dial, error, repeat, hangup). Typically about 10 to 20 repertory names, 10 digits, and about 5 to 10 command words are sufficient for most users. To provide the voice dialogue to guide the user and to provide feedback as to the transaction status, a voice-response system with a vocabulary of about 100 to 200 words is sufficient for this simple transaction.

A formal evaluation of such a voice repertory dialer was made [1] using a template-based system with a 37 word-recognition vocabulary and a simple control grammar. With six test talkers, each one training the system, a word error rate of 0% was obtained. Figure 9.3 shows a plot of the average LPC distance score, for each speaker in the test, for both the first recognition candidate (the correct word) and the second recognition candidate (the nearest incorrect word). The spacing of these curves is indicative of the robustness of this simple system.

### 9.5.2 Automated Call-Type Recognition

An interesting and novel telephone application of speech recognition is the automation of operator-assisted toll calls, i.e., calls made from a pay phone that normally require operator assistance, including collect calls, person-to-person calls, third-party-billing calls, operator-assisted calls, and credit (or calling) card calls. Since there are only five possible options for this type of service, a simple five word vocabulary, consisting of the following words is adequate:

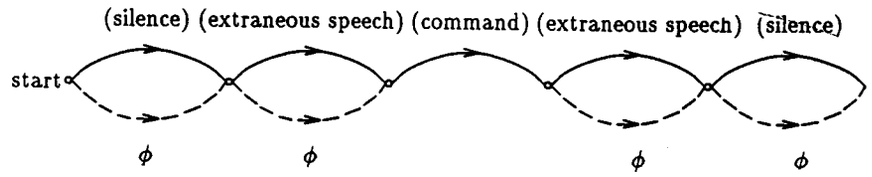
- "collect" to make collect calls
- "person" to make person-to-person calls
- "third number" to make third-party-billing calls
- "operator" to make operator-assisted calls
- "calling card" to make calling card calls.

The system is inherently speaker independent and, of course, is intended to work over the standard dialed-up telephone network.

A simple voice-response system is adequate to provide the voice prompt necessary to elicit the spoken command, as well as to provide feedback to the user in the case of difficulties. A real-world trial [2] of this system was run (within an AT&T switching office in Haywood California) and had the following results:

- When customers obeyed the voice prompt and spoke one of the command words, the word accuracy exceeded 99% consistently.
- About 20% of the time, customers didn't obey the prompts and embedded the command word (or phrase) in some type of carrier sentence, e.g.,
  - *collect* call please
  - um? Gee, ok I'd like to place a *calling card* call
  - collect from Tom

For such cases it was necessary to use some type of keyword spotting technique to find the command words embedded within the sentence. A very powerful keyword spotter was devised [3] based on the concept of training an "extraneous speech" pattern from the spoken input with extraneous speech, and on recognizing the input using connected word recognition techniques (as discussed in Chapter 7) based on a sentence grammar of the form:






i.e., an isolated command uses the appropriate set of null arcs, whereas an embedded command uses the extraneous speech pattern as part of the sequence to match the input speech. Using the keyword spotter, the overall command word accuracy remained at about 99% on this task.

### 9.5.3 Call Distribution by Voice Commands

Another interesting telephony-based task to which voice recognition has been successfully applied is the area of call distribution by voice commands. For this application a call is placed that is normally answered by an attendant or operator, who then distributes the call to the appropriate location (or person) based on user responses to the questions asked by the attendant. In this application, the attendant function is automated via voice processing in that a voice-response system poses a series of menu-based questions, and based on the user responses, routes the call appropriately.

By way of example consider calling AMTRAK, and receiving the response:

> Welcome to AMTRAK. If you want information about train departures say the digit 1; if you want information about train arrivals say the digit 2; if you want to make a reservation, say the digit 3; if you want information on the Metroliner, say the digit 4.

In this manner, the call can be routed to an appropriate location, via voice control, without the need for an attendant. (Of course, the recognition system could use a command vocabulary consisting of the words "departures," "arrivals," "reservations," and "Metroliner" rather than the digits. This would make remembering the menu selections a considerably easier task for the user.)

There are numerous applications of such a call-distribution system, including hotels, department stores, and large corporations.

### 9.5.4 Directory Listing Retrieval

An interesting and useful application of speech recognition is to provide access to directory information from spoken spelled names. A block diagram of such a system is shown in Figure 9.4. To access directory information for a name in the directory, the user spells the name (either as a sequence of isolated letters, with a distinct pause between individual letters, or as a connected sequence of letters) using the word "stop" between the last name and the initials, as well as after the initials, as in

"RABINER—stop—LR—stop."

The speech recognizer determines the name in the given directory which best matches the spoken input and then speaks the directory information for that name to the user.

Perhaps the most interesting feature of this system, from a speech-recognition point of view, is that although the recognition of spoken spelled letters is highly error prone (because of acoustic confusabilities among similar sounding letters), the telephone directory provides a powerful form of task syntax that automatically detects and corrects improperly

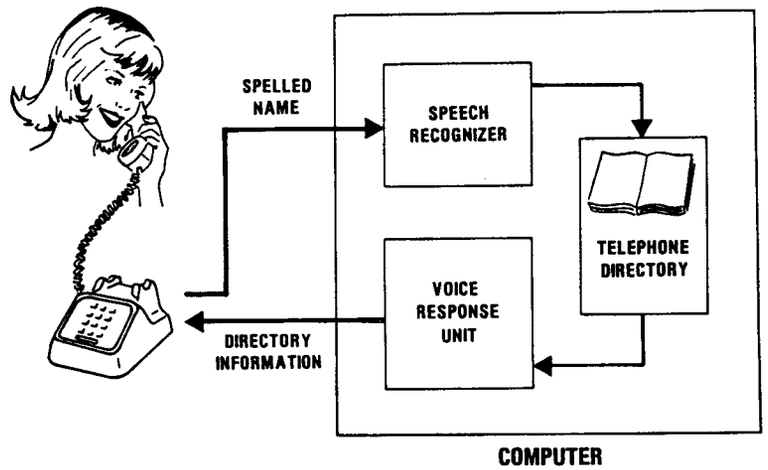



**Figure 9.4** Block diagram of a directory listing retrieval system based on spelled spoken names (after Rabiner et al. [1]).

recognized letters. In response to the spelled input given above, a string that is very likely to be recognized is

"R J V Y M Z R—stop—LR—stop."

Without the list syntax imposed by the telephone directory, this string would be perfectly acceptable; however, since the recognized string must both match the acoustic properties of the spelled letters and be a valid string in the directory, the system generally has no problem decoding the spelled input into the correct string.

Using isolated letters for spelling the name, with an 18,000-name directory, in a speaker-independent system operating over a dialed-up telephone line, a *name* accuracy of over 98% was obtained [4]. Not only did the system work well with correctly spelled input, it could also handle common misspellings of names with a single insertion or deletion of a letter, or a single-letter substitution or a single letter inversion with another letter in the string.

### 9.5.5 Credit Card Sales Validation

Another interesting application in the area of telecommunications is the problem of automated validation of credit card sales. The application is again of the class in which a service provided by an attendant is automated by voice-recognition processing. For this application, whenever a credit card sales transaction occurs where the merchant needs validation and does not have use of an automatic card reader/modem dialer, the merchant



must call an 800 number and provide an attendant with a 10-digit merchant identification number, a 15-digit credit card number, and the dollar amount of the transaction. In this case, the speech-recognition system uses a connected digit recognizer to recognize the merchant identification number and the credit card number, and a connected word recognizer for the transactions amount.

This application is an interesting one, from the point of view of speech recognition, for several reasons. First of all, there is a very interesting syntax associated with merchant IDs and credit card numbers—namely, there is a check digit (a nonlinear combination of all preceding digits) associated with each of these codes that enables the recognition system to detect single-digit errors (by far the most likely failure mode), and there are several simple and straightforward methods to correct such errors automatically. Using this built-in syntax, the string recognition error is reduced by almost an order of magnitude (to below 1%) under real-world operating conditions.

A second interesting aspect of this task is the "natural number" mode of speaking the dollar amount of the credit card transactions, i.e.,

- one thirty seven dollars
- one hundred and three seven dollars
- one three seven dollars
- one hundred three seven dollars.

in which the same string can be spoken in a variety of ways. Because of this mode of speaking, the vocabulary size for recognition is considerably larger than that needed for the credit card, i.e., about 40 words versus 11 words for digit strings. The words are also much more similar (e.g., "six," "sixteen," "sixty"), and hence the recognition task is quite a bit more difficult. In spite of these problems, fairly high string recognition scores have been obtained using many of the discriminative techniques discussed in Chapter 5.

## 9.6 PROJECTIONS FOR SPEECH RECOGNITION

It is almost impossible to predict accurately the rate of progress in any scientific field. However, based on the rate of progress over the past decade, it seems reasonable to make some broad projections as to where speech recognition is headed in the next decade. Such a set of projections is shown in Table 9.2. We see that in the current time frame (1990–1995) there is demonstrated recognition capability in connected word recognition, for small vocabularies, for tasks such as voice dialing, credit card entry, catalog ordering, and inventory inquiry. Such systems can usually be implemented using a handful (2–4) of DSP chips, each capable of delivering about 25 Mips.In the same time frame, we have a demonstrated capability in medium vocabulary (100–1000 words), continuous speech recognition of fluent speech for highly constrained tasks such as transaction processing, robotic control, and elementary database management such as the Resource Management problem discussed in Chapter 8. Such tasks require 4 to 10 DSPs, each capable of providing

**TABLE 9.2.** Projections for speech recognition.

| Year | 1900–95 | 1995–2000 | 2000+ |
|---|---|---|---|
| Recognition Capability | Connected Words | Continuous Speech; Whole Word Models, Finite-State Grammars, Constrained Tasks | Continuous Speech; Subword Recognition Elements, Language Models Representative of Natural English; Task-Specific Semantics | Continuous Speech; Natural-Language Grammar, Syntax, Semantics; Neural Adaptation, Learning |
| Vocabulary Size | 10–30 | 100–1,000 | 5,000–20,000 | Unrestricted |
| Processor Requirements | 2–4 DSPs (25 Mips/Chip) | 4–10 DSPs (50 Mips/Chip) | 5–50 DSPs (200 Mips/Chip) | 20–60 DSPs (1,000 Mips/Chip) |
| Applications | Voice Dialing, Credit-Card Entry, Catalog Ordering, Inventory Inquiry | Transaction Processing, Robot Control, Resource Management | Dictation Machines, Computer-Based Secretarial Assistants, Data-Base Access | Natural Language Interaction, Translating Telephony |



50 Mips.

In the 1995–2000 time frame, we foresee growth in the area of large vocabulary (5000-20,000 words) continuous speech recognition for tasks as complicated as voice dictation of first-draft quality, computer-based secretarial assistants, and full database access and management. Computational requirements will be increased to from 5 to 50 DSPs, each capable of delivering 200 Mips.

Finally, in the 2000+ time frame, we foresee natural language interaction with machines with totally unrestricted vocabulary, syntax, and semantics, including the capability of translating telephony, i.e., speaking in one language and having the sentence meaning converted to another language. The computation for such capability is large, requiring on the order of 20 to 60 DSP chips, each one providing 1000 Mips, or higher.

For the predictions of Table 9.2 to come about, we need continued research in virtually every aspect of speech recognition and natural language interaction with machine, as well as a steady stream of new ideas leading to more and more powerful algorithms. We look forward to seeing the promise of speech recognition become a reality.

## REFERENCES


[1] L.R. Rabiner, J.G. Wilpon, and A.E. Rosenberg, "A Voice-Controlled, Repertory-Dialer System," *Bell System Tech. J.*, 59 (7): 1153–1163, September 1980.

[2] J.G. Wilpon, P. Mikkilineni, D.B. Roe, and S. Gokcen, "Speech Recognition: From the Laboratory to the Real World," *AT&T Tech. J.*, 69 (5): 14–24, Sept./Oct. 1990.

[3] J.G. Wilpon, L.R. Rabiner, C.H. Lee, and E. Goldman, "Automatic Recognition of Keywords in Unconstrained Speech Using Hidden Markov Models," *IEEE Trans. on Acoustics, Speech, and Signal Proc.*, 38 (11): 1870–1878, November 1990.

[4] B. Aldefeld, L.R. Rabiner, A.E. Rosenberg, and J.G. Wilpon, "Automated Directory Listing Retrieval System Based on Isolated Word Recognition," *Proc. IEEE*, 68 (11): 1364–1379, November 1980.

# INDEX


**A posteriori probability, 338**
A priori probability, 363
Absolute loudness, 281
Accelerometer, 312
Acoustic:
  knowledge, 52
  lexicon, 477
  model, 435
  segment units, 476–7
  similarity, 436
  units, 437
  waveform, 14
Acoustic-phonetic:
  approach, 42, 45–50
  labeling, 48–9
  model, 70
  vowel classifier, 46
Adaptation, 54
  parameter, 376
Adaptive:
  clustering, 285, 288
  discriminative training, 304
  learning, 62
  level equalization, 149
  preemphasis network, 113
  noise cancellation, 310
Adverse environments, 305–17
  adaptive models, 312–13
  articulation effects, 309
  distortion, 307–9
  noise masking, 312–13
  noise, 306–7
  robust distortion measures, 314–5
  spectrum, 307
  stress compensation, 313
  transducer arrangements, 311–12
Affricates, 21
All-pole model, 72, 98, 316
  minimum phase, 156, 164
  spectrum, 112, 156
All-zero filter, 353
Ambient noise, 306
Amplitude compression, 76
Analysis:
  order, 112
  window, 19, 102
Antiresonances, 30, 31
Anvil, 133
Area ratio, 190
  log, 191
ARPABET, 20, 437
Articulation:
  effect, 309
  index, 186
Articulators, 14
Articulatory motion, 12
Artifacts, 144
  heavy breathing, 144
  lip smacks, 144
  mouth clicks, 144
  pops, 144
Artificial intelligence approach, 4, 42, 52–3
Artificial neural networks, 2, 57 (See also neural networks)
Aspiration, 18, 35
Associative memory, 60
Asynchronous sequential decision, 204
Attractors, 60
Audiogram, 185
Auditory:
  analysis, 54
  discriminability, 153
  masking, 176
  models, 132–9
  nerve, 12, 133, 134

Auditory (cont.):
nerve firing, 316
perception, 78
physiological models, 132
sensitivity, 151
system, 4, 70, 316
Autocorrelation:
analysis, 114
function, 105
matrix, 120
method, 103–6, 117
residual normalized, 174, 249
sequence, 154
short term, 155
Automated call-type recognition, 490–1
Autoregression process, 352
Autoregressive Gaussian source, 352
Auxiliary function, 344
Average:
cluster center, 274
speaking rate, 430
word branching factor, 450

**Back cavity, 31**
Back propagation learning, 61
Background silence, 18
Backtracking, 340, 385, 394
Backward:
procedure, 337
recursion, 367
variable, 337, 346, 366
Bahl, 376
Baker, 322
Bakis model, 348, 379
Balashek, 6
Bank-of-filters:
model, 70
processor, 73–97
Bark scale, 78, 187
warping, 186, 276
Barkhausen, 185
Barks, 185
Barney, 26
Barred I, 21
Basilar membrane, 12, 133, 185
Baum, 322
Baum-Welch method, 342, 372, 374
Bayes:
risk, 376
rule, 434
statistics, 371
Bayesian adaptation, 372–5
Beam search, 451
Beginning range reduction, 410–11
Belar, 6
Bellman, 205
Bessel filter, 93
Biddulph, 6
Binary:
decision tree, 50
indicator function, 447
search trees, 129
split algorithm, 126, 244
Blackboard approach, 53
Block quantizer, 130
Body units, 464
Boll, 311
Bottom-up processor, 53
Brain, 12
Bridle, 8, 313
Broad phonetic features, 11
Broadband signals, 70
Bronchi, 14
Brown, 8
BYBLOS system, 9

**Call distribution by voice commands, 491**
CART-based phonetic clusters, 475
Cartesian product, 149
Catalog ordering, 493
Casual training, 265–6
Cellular phone, 489
Center clipper, 93
Centroid, 125, 128, 271
computation, 124, 246, 247
formant, 28
update, 125
Cepstral:
coefficients, 115, 163
coefficients weighted, 117
derivative, 116–7
liftering, 166, 169, 314
projection measure, 315
trajectory, 196
weighting, 169, 199
Cepstral distance, 125, 163–71, 250
index weighting, 169, 171
liftering, 166–71
mel-frequency, 190, 276
truncated, 164, 275, 281
warped, 186, 188
weighting, 166–71, 275, 314
Cepstrum:
complex, 163, 164
LPC, 163
mel-scale, 189
Chainmap, 267
Check digit, 493
Chen, 313
Chiba, 7, 209, 214, 216
Cholesky decomposition, 107
Circular convolution, 73
City block distance, 247
Class-reference templates, 296
Classification and regression trees (CART), 50
Cluster center, 271, 273
Cluster splitting, 272
Clustering, 267–74, 433
templates, 279
bandpass liftered cepstral distance, 275
compactness, 270
distance matrix, 268
distortion measures, 274–80
energy usage, 280–2
kNN decision rule, 277–9
likelihood ratio measure, 275
outliers, 270
weighted cepstral distance, 275
weighted likelihood ratio, 275
Co-occurrence smoothing, 463
Coarticulation, 38, 43, 465
Cochlea, 133
model, 70
Cochlear filters, 133, 134
Code word, 244
Codebook, 122, 124, 244, 245
generation, 246
multiple, 129
universal, 288
Coding:
efficiency, 244
techniques, 13
Coin-toss models, 326–8
Command-and-control, 390, 447, 488–93
Composite:
ending range, 403
language model, 451
model spectrum, 311
Computer-based secretarial assistant, 495
Concatenated reference model, 393
Concept spotter, 483
Confusion sets, 291
Conjugate prior, 374
Connected digit recognition, 425–32, 482
grammar networks, 425–7
performance evaluation, 430–2
postprocessing, 428
spectral analysis, 428
word pattern matching, 428
Connected word models, 390–432
grammar networks, 414–6
multiple candidate strings, 420–3
segmental k-means training, 427
Connected word:
pattern matching, 428
recognition, 392, 485
Connectionist network, 54



Consonant clusters, 38
Consonants, 20
nasal, 21, 30
stop, 33
unvoiced fricatives, 31–2
unvoiced stop, 21
voiced fricatives, 32–3
voiced stop, 21
Content addressable memory, 60
Context-dependent grammar, 448
Context-dependent units, 458–75
diphones, 460–1
triphones, 460–1
interpolation, 462–4
interword unit, 468
smoothing, 462–4
subword unit, 458–9
triphone, 460, 462
unit clustering, 470, 475
unit splitting, 470–5
variance estimation, 467
Context-free grammar, 448
Context-independent:
PLU, 457
subword unit, 459
Context sensitive unit, 377
Context sensitivity, 437
Continuous observation density, 350
Control strategies, 46
Convergence rate, 376
Convolution representation, 73, 90
Corrective training, 372, 376–7
COSH distortion, 178
Cost prediction, 238
Covariance:
function, 105
weighted spectral difference, 125
Covariance matrix:
diagonal, 380
full, 380
Covariance method, 106–7
window, 106
Coverage, 270, 279, 452
Credit card sales validation, 492–3
Critical band, 184
bandwidth, 184
filter, 78
scale, 78
spectrum, 192
Critical point of likelihood, 343–4
Cross entropy, 365
Cross talk, 146
Cross-validation, 448

**Dal Degan, 312**
DARPA:
community, 437, 451
Resource Management task, 435, 451
Database query, 8
Dautrich, 305
Davis, 6, 276
Decimation, 92
Decision:
logic, 51, 70, 257
tree, 50
Defense Advance Research Projects Agency (DARPA), 8
Deleted interpolation, 371, 372–3, 463
Delta:
cepstral distance, 283
cepstrum, 49, 445
Delta-delta:
cepstrum, 429
log energy, 429
Demisyllable-like units, 436
Denes, 6
Descent algorithms, 347
Dictionary, 437
Difference limen (DL), 150, 151
formant bandwidth, 152
formant frequency, 151
fundamental frequency, 152
LPC complex pole, 152
Differential:
spectral distance, 198, 199
threshold, 150, 151
Differentiated log magnitude spectrum, 116
Digit:
deletion, 426
insertion, 426
Digital signal processor (DSP), 493
Diphthongs, 18, 20, 21, 28–9
Direct form filter structure, 80
Directed divergence, 171
Directory listing retrieval, 491–2
Discrete:
observation probability, 350
Fourier transform (DFT), 51
symbols, 350
Discrete-time Markov processes, 322–25
Discriminant:
analysis, 215, 376
function, 376
Discrimination information, 171, 363, 365
Discriminative methods, 291–305
Discriminative training, 243, 294, 302, 376
minimum recognition error, 302–5
Discriminative weighting, 293, 297–301, 302
Disjoint clusters, 268
Dissimilarity measure, 142
Distance functions, 149–200
invariant, 149
positive definiteness, 150
symmetry, 150
triangle inequality, 150
Distance matrix, 267, 271, 398
Distance measures, 8, 125, 142, 149–200
cepstral, 125
likelihood, 125
likelihood ratio, 127
spectral, 143
Distortion, 305, 307, 308
Distortion measures, 148–200
asymmetrically weighted COSH, 180
auditory sensitivity, 150
cepstral projection, 315
computation, 193–4
COSH, 178, 250
Itakura, 174, 247, 249
Itakura-Saito, 171, 247, 276, 280, 311, 355
just-discriminable change, 150
likelihood, 247
likelihood ratio, 174, 247, 275, 282, 356
perceptual considerations, 150–4
positive definiteness of, 150
spectral, 143, 154
squared error, 150
subjective interpretation, 150
symmetry of, 150
triangular inequality of, 150
weighted likelihood, 179
weighted likelihood ratio, 181, 183, 275, 277
Distribution estimation, 376
Divergence, 365
directed, 171
symmetrized directed, 179
Doshita, 7
Durbin method, 106, 115, 117
Dyad-like units, 436
Dynamic pattern, 63
Dynamic programming, 7, 131, 204–8, 221, 396
algorithms, 219
paths, 206, 209
range reduction, 410
two level, 395
Dynamic time-warping, 4, 51, 221–6, 394–5
algorithm, 51
alignment path, 239
dynamic programming, 221
K-best paths, 232

Dynamic time-warping (cont.)
minimum partial accumulated distortion, 222
multiple paths, 232–8
optimal warping path, 239
parallel algorithm, 233–4
procedure, 287
range limited, 396
recursion, 222
relaxed endpoint constraints, 230
serial algorithm, 235–6
synchronous sequential decoding, 232
tree-trellis search, 236–8
Dynamical systems, 62

**Ear:**
inner, 12, 132
middle, 132
model, 54
outer, 132
Eardrum, 133
Elliptically symmetric density, 350
Endpoint constraints, 208–9
Endpoint detection, 143–9, 229, 390
explicit approach, 146–7
hybrid approach, 146–9
implicit approach, 146–8
Endpoint error, 143
Energy:
contour of speech, 49
distance, 281
pulses, 149
spectral density, 155
Ensemble interval histogram (EIH), 134–9, 316
auditory nerve fibers, 134
level crossing detectors, 134
noise, 138
reverberation, 138
Entropy, 377, 449–50
change, 378, 449
estimated, 449–50
first order, 449
relative, 171
Ephraim, 311
Equivalence relations between HMM parameters, 358
Ergodic, 329, 348, 379, 449
model, 329, 348, 379
sources, 245, 449
Error:
backpropagation training, 61
correcting code, 486
probability, 305
transfer function, 101
Error handling, 486
fail soft method, 486
multilevel decision, 486
rejection, 486
self error correction, 486
self error detection, 486
Esophagus, 14
Estimation maximum likelihood, 171
Euclidean distance, 191, 247, 383
Excitation, 100, 101
potential threshold, 57
source, 20
Exemplar, 51
Expectation-maximization method (EM), 342, 347
Expected number of transitions, 343
Expert system, 4, 44, 482
Exponential state duration density, 358
External canal, 132
Extraneous speech, 490–1

**Fail soft methods, 486**
Fall-back mode, 485
Fault tolerance, 62
Feature detection, 45, 53
Feature measurement, 45, 51
Feed forward connectionist network, 54
Ferguson, 322, 358
Filter bank, 4, 45, 51, 77–81
analyzer, 51
critical band, 94
implementations, 80–1
methods, 45
model, 69
nonuniform, 78, 89, 91
octave spaced, 91, 94
tree structure, 91
types, 77–9
uniform, 77, 87, 88
Final word candidate rule, 258–9
Finite alphabet, 350
Finite precision effect, 112
Finite-state:
automata (FSA), 414
network (FSN), 357, 358, 414, 435, 442, 452
vector quantizer, 254
First-order:
Markov chain, 255, 323
differential log spectrum, 196
Fischer's linear discriminant, 297, 298, 299
Fixed relaxation points, 59
Flanagan, 153
Forgie, 6
Formal:
grammar, 448
parser, 435
Formant, 20, 23, 26, 84
frequencies, 20, 27, 29, 45
synthesizer, 192
Forward:
procedure, 335–7
variable, 335, 346, 359, 361, 366
Forward-backward:
algorithm, 344, 372, 373, 376, 441, 443, 463
procedure, 441
Frame rate, 282
Frame-synchronous, 400
level building (FSLB), 416, 423, 451
French, 186
Frequency:
resolution, 81, 85, 94
response composite, 90, 93
smoothing, 96
Frequency scale:
Bark, 186
critical band, 184
mel, 183, 184
Frequency warping, 188
Bark scale, 276
Frication, 45
Fricative:
unvoiced, 21, 31
voiced, 21, 32
Front cavity, 31
Front-end processor, 72
Fry, 6
Full search, 128
Full-wave rectifier, 73
Fully connected HMM, 348
Function:
words, 475–6
models, 460
Fundamental frequency, 81, 158
Furui, 196
Fuzzy vector quantizers, 463

**Gain independent probability density function, 353**
Gamma family, 362
Gaussian:
mixture, 374
noise, 311
Gender-specific model, 476
Generalized:
Lloyd algorithm, 125–8, 245, 274, 287, 377
Rayleigh quotient, 301
triphone, 469
Geometric mean spectrum, 251
Germinate deletion, 466
Ghitza, 316
Glides, 21



Global:
codebook, 263
dissimilarity measure, 293
match, 394
pattern dissimilarity measure, 201, 202
range reduction, 411
time alignment procedure, 51
Global path constraints, 213–5
absolute time deviation, 215
maximum expansion, 213
minimum expansion, 213
parallelograms, 214
range-limiting, 215
Glottal:
air flow, 14
excitation, 30
stop, 20
waveform, 14
Glottis, 14, 32
Gradient:
descent method, 303
microphones, 312
techniques, 342, 347
Grammar:
context dependent, 448
context free, 448
formal, 448
networks, 2, 414–6, 420, 423, 424, 435
nodes, 424
word pair, 435
Grand variance, 467
Greedy growing algorithm, 377
Group delay spectrum, 168, 314

**Hair cell transduction, 134**
Half-wave rectifier, 73
Hammer, 133
Hamming window, 81, 114
Hard limiter, 57
Hard sound change, 466
Head units, 464
Henry, 312
Hidden Markov models, 131, 321–87
Bayesian adaptation, 373–5
autoregressive models, 352–4
choice of model, 371
continuous observation densities, 350–4
corrective training, 376–7
dynamic programming methods, 339–40
elements, 329–30
ergodic, 348, 379
evaluation problem, 333
explicit state duration density, 358–64
generator of observations, 330–1
initial parameter estimates, 370
initial state distribution, 330
insufficient training, 370–1
left-right, 348, 379, 439
model clustering, 377–8
model distance, 364–5
model parameters, 379–81
model splitting, 377–8
multiple, 377
multiple observation sequences, 369–70
observation symbol probability distribution, 330
observation symbols, 330
optimal state sequence, 337–40
parameter estimation, 342–8
parameter set of model, 330
probability evaluation, 334–5
Q function, 344
scaling, 365–8
semicontinuous, 441, 464
state duration, 384–5
state-transition probability distribution, 330
states, 329
stochastic constraints, 347
types, 348–50
Hidden:
control neural network (HCNN), 64–5
layers, 58
Hierarchical spectral clustering, 288–91
Higher level processes, 53
Holmes, 313
Hopfield network, 58–9
Human ear, 132–4
inner ear, 132–4
middle ear, 132–3
outer ear, 132–3
Hypertangent function, 302
Hypothesis-and-test paradigm, 53

**Imai, 314**
Impulse response, 80, 90
finite, 80
infinite, 80
Incremental paths, 210
Incus, 133
Indicator function, 303
Information:
rate of speech, 12
source, 244
Informative prior, 374
Initial:
state distribution, 330
state probabilities, 324
Inner hair cells (IHC), 133
Insufficient data, 357, 370
Intercluster distance, 267
Interconnection weights, 61
Interword:
context dependent unit, 461
training, 461–2
units, 461
Intracluster distance, 267, 271–2
Intraword context dependent unit, 462
Inventory inquiry, 493
Inverse filtering, 155, 281
ISODATA, 267
Isolated digit recognition, 6, 482
Isolated word recognition, 284, 390, 484
HMM, 378
multispeaker, 284
speaker independent, 278
speaker trained, 257
Itakura, 7, 171, 212, 314
Itakura distortion measure, 174, 247
Itakura-Saito distortion, 171, 247, 276, 280, 311

**Jaw, 12, 14**
Jelinek, 322, 372, 448
Juang, 315
Junqua, 309
Just-noticeable difference (JND), 150–1

**K-means clustering, 125–8, 377**
K-means iteration, 427
K-nearest neighbor rule, 243, 296
K-tuple quantizers, 130
Kaiser window, 93
Katagiri, 376
Keyword spotting, 490–1
Klatt, 192, 312
Knowledge:
acoustic, 52
automatic acquisition, 54
automatic adaptation, 54
based allophonic clustering, 475
lexical, 52
phonemic, 52
pragmatic, 52
semantic, 52
sources, 52
syntactic, 52
Knowledge-based allophonic clustering, 475
Kullback-Leibler information, 171

**L(p) norm, 158**
Labeling of speech, 43, 46, 49

Lagrange:
multipliers, 347
optimization, 347
Language:
analyzer, 482
code, 12
Language model, 53, 435, 447
bigram, 448
N-gram, 447, 448–50
no-grammar, 448, 452
trigram, 448
unigram, 448
word pair grammar, 447, 452, 468
Large vocabulary recognition, 434–79
language models, 447–50
Larynx, 14
Lattice structure, 337
Laurent expansion, 163
Least-squares estimator, 311
Lee, 8, 378, 469
Left context diphone, 460
Left-right:
context triphone, 460
model, 348, 379
Level building algorithm, 5, 400–16
beginning range reduction, 410–11
computation, 407–9
global range reduction, 411
implementation, 410–16
multiple levels, 405–7
reference pattern uncertainty region, 411
test pattern ending range, 411
Level building flaw, 421
Level crossing:
detector, 134
rate, 69
Level synchronous, 403
Lexical:
access, 43, 46
constraint, 46
decoding, 53
knowledge, 44, 52
Lexicon, 435, 473
Lifter bandpass, 116
Liftering, 169, 199
Likelihood:
distance measure, 125
distortions, 171–83
ratio distortion, 174, 247
ratio test, 171
Line spectral frequencies (LSF), 191
Linear discriminant analysis, 293, 297
Linear phase, 93
Linear predictive coding, 45 (See also LPC)
analysis, 4, 7, 51
model, 69, 97–112
Linear time:
alignment, 202
normalization, 201
Linguistic:
similarity, 436
unit, 357
Lippmann, 310
Lips, 12, 14
Liquids, 21
Lloyd algorithm, 254, 274
Local continuity constraints, 209
Itakura type, 212
Type I, 210
Type II, 210
Type III, 210
Local:
distance, 51, 222
likelihood, 424
warping distance, 401
Log area ratio, 115
Log spectral distance, 158–62, 173
mean absolute, 158
peak, 158
rms, 158, 166
Logarithmic:
encoding, 76
frequency scale, 78
Log-concave density, 350
Lombard effect, 309
Loudness, 12
Lower path constraint, 406
LPC model, 70, 100–122
analysis, 121
analysis equations, 101–7
analysis order, 72, 122, 282
analysis parameters, 121–2
analysis window, 282
cepstral coefficients, 115
cepstrum, 163
excitation term, 100
frame rate, 282
front end, 97, 112
gain of the excitation, 100
line spectral frequencies, 191
log area ratio coefficients, 115, 191
LPC coefficients, 115
mean-squared prediction error, 102
normalized excitation, 100
normalized prediction error, 112
parameters, 72, 115
PARCOR coefficients, 115
partial correlation coefficients, 190
prediction error, 101
prediction order, 112
predictor coefficients, 102
quasiperiodic pulse train, 100
random noise source, 100
reflection coefficients, 115
reflection coefficients, 190
RMS prediction error, 112
whitening characteristics, 109
LPC processor, 112–7
autocorrelation analysis, 114–5
frame blocking, 113
LPC analysis, 115
LPC parameter conversion, 115–6
parameter weighting, 116
preemphasis, 112–3
temporal cepstral derivative, 116–7
windowing, 114
Lungs, 14

**Mahalanobis distance, 247**
Malleus, 133
Mansour, 314, 315
MAP decoding rule, 435
Markov:
chain, 322
source, 322
Martin, 7
Masking:
audiogram, 185
effect, 188
Match:
sentence level, 451
word level, 451
Matched filter, 64
Matrix quantization, 130–1, 254
Matsumoto, 314
Maximum a posteriori (MAP):
decoding, 434–5
estimate, 371, 374
Maximum:
entropy principle, 181
expansion, 405
likelihood estimate, 342, 344, 355, 374
mutual information (MMI) criterion, 362–4, 376
McCullough, 57
Mean duration, 429
Mean-squared prediction error, 102
Median vector, 247
Mel:
frequency, 184
scale, 78, 183
Mel-frequency cepstrum, 189
distance, 190, 276
Memory:
associative, 60
constraint block, 254
constraint trellis, 256
content addressable, 60



Memoryless vector quantizer, 245
Mercer, 372
Mermelstein, 276
Message, 12
Metric, 150
Microphone:
carbon, 125
dynamic, 311
electret, 125
gradient, 312
noise cancelling, 311
pressure gradient noise cancelling, 312
transducer, 307
Mid-sagittal plane, 14
Minimax center, 268, 271, 273
Minimum:
Bayes risk, 374, 376
classification error rate, 376
cross-entropy, 364
discrimination information (MDI), 362–4
distortion labeling, 246
expansion, 405–6
expected distortion, 244
mean-squared error, 102, 103
phase, 134
recognition error, 243
residual energy, 281
Misclassification measure, 302
Misrecognition measure, 302
Mixture density, 350, 351, 427, 440
MKM algorithm, 279
Model:
initialization, 382
mismatch, 362
Modified k-means:
algorithm (MKM), 271, 279
clustering, 271–4
Monosyllabic word, 6
Monotonicity conditions, 209
Mouth, 14
cavity, 14
Mu-law encoding, 76
Multi-style training, 309
Multilayer perceptrons, 58
Multilevel decision strategy, 487
Multiple:
candidate string, 420
codebooks, 129
observation sequence, 369
phone models, 459
pronunciation, 476
Multiple time alignment, 232
parallel algorithm, 233, 234
serial algorithm, 235
tree-trellis search, 236
Multispeaker:
mode, 284
model, 431
Multistyle training, 309
Muscles, 14
Mutual information, 363
Myers, 8

**N-gram language model, 435, 447, 448**
Nagata, 7
Nakata, 7
Narrowband spectrogram, 19
Nasal:
cavity, 14
consonants, 21, 30–1
tract, 14, 30
Nasalized vowels, 30
Natural:
language, 9, 449, 495
number mode, 493
Naval resource management task, 451
Near-miss cases, 376
Nearest-neighbor:
principle, 288
rule, 243, 288, 377
search, 124, 125, 127
Nervous system, 57
Neural:
firing, 134
impulses, 133
transduction process, 12
tuning curves, 134
Neural networks, 44, 54–65
basics, 56–7
nonlinearity, 61
offset, 61
structures, 63–5
topologies, 57–61
Neuromuscular commands, 12
Neuron firing, 57
Neurons, 57
Next state function, 254
No-grammar model, 448
Nocerino, 275, 281, 283
Noise, 305, 306
adaptive bandwidth expansion, 314
coherence, 310
compensation, 312
criterion (NC), 306
elimination, 95
masking, 312
source, 100
Noise-cancelling microphone, 311
Noise-compensation schemes, 312
Noncontinuant sounds, 33
Noninformative prior, 374
Nonlinear:
amplification, 57
dynamic system, 62
Nonparametric mapping, 286
Nonrecursive filter, 92
Nonuniform acoustic tube, 190
Nonuniform filter bank, 78
FFT-based implementations, 91
implementations, 89–92
tree structures, 91–2
Normalized:
autocorrelation, 354
frequency, 70
prediction error, 112
Null:
arc, 442
transitions, 356–7, 373
Numeric floor, 371

**Observable Markov model, 323**
Observation probability, 330
Octave band spacing, 78
Olson, 6
One-pass (one-state) algorithm, 416–20
One-step prediction error, 156, 248
One-third octave band spacing, 78
Operator services, 9
Optimal:
path problem, 204
policy, 205
Oral cavity, 14, 30
Orthogonal polynomial, 117
Oval window, 133
Overtraining, 457

**Palatization, 466**
Parallel computation, 62
Parallel distributed processing model (PDP), 56
Parameter:
estimation, 357
scaling, 365, 368
thresholding, 371
tying, 357
weighting, 116
Parameter tying, 358
Parametric representation, 69
Parseval's theorem, 163
Parsing, 2
Partial:
correlation (PARCOR), 190
correlation coefficient (PARCOR), 115
observation sequence, 335, 337
word decision, 401
Partitioned cell, 126

Path:
backtracking, 394
contraction factor, 396
expansion, 396, 417
K-best, 232
locally optimal, 231
maximum likelihood, 378, 424
N-best, 420
Pattern:
classification, 44, 51
clustering, 267–74
comparison techniques, 141–240
comparison, 43, 70
dissimilarity measure, 221
matching, 263
measurement, 70
recognition approach, 4, 42, 51–2
recognition model, 70
training, 51
Pattern-to-class distance, 296
Perceived loudness, 158
Perceptrons, 58
convergence procedure, 61–2
multilayer, 58, 61
single layer, 58
Perceptual:
consistency, 153
masking, 162
Performance analysis, 274
Periodicity, 81
Perplexity, 449–50, 452, 478
Personal identification number (PIN), 425, 485
Peterson, 26
Pharynx, 14, 30
Phonelike units (PLU), 436
Phoneme, 20, 435
deletion, 48, 294
distance, 294
distance matrix, 297
insertion, 48, 294
lattice, 43, 46
recognizer, 6
sequences, 12
substitution, 48
Phonetic:
context, 436
decoding, 53
difference, 53
distance, 192
knowledge, 52
relevance, 150
transcription, 37
Phonological rules, 439, 466
geminate deletion, 466
palatization, 466
plosive deletion, 466
Phonology, 2
Physiological mechanism for speech, 14
Pinna, 132
Pisoni, 309
Pitch, 20
accent, 12
harmonics, 112
period, 103, 109
period estimation, 112
Pitts, 57
Plosive deletion, 466
Policy, 204
Porter, 311
Position dependent units, 469–70
Power spectral density, 155
Pragmatic knowledge, 2, 44, 52
Prati, 312
Prediction:
error, 101–102
error rms, 112
residual, 118, 171
residual minimum, 120
Predictor coefficient, 101, 103
Preemphasis network, 95, 112
Pressure gradient noise-cancelling microphone, 312
Principal components analysis, 95–6
Principle of optimality, 205–6, 221
Prior distribution, 374
conjugate, 374–5
informative, 374
noninformative, 374–5
Probabilistic:
density function, 135
functions of Markov chain, 322
lexical access, 46
Projection operation, 315
Projections for speech recognition, 494–5
Pronunciation, 439, 442, 443, 445
Prosody, 12
markers, 12
Pruning methods, 255
Pseudoaverage center, 273

**Quadrature mirror filter, 91**
Quantization error, 123
Quasiperiodic speech, 18, 75

**Rabiner, 8, 274, 279**
Radix-2 fast Fourier transform, 85
Raised sine function, 169
Range-limited algorithm, 396
Rate of convergence, 376
Real-time:
implementation, 482
response, 486
Rectifier:
full-wave, 73, 93
half-wave, 73, 93
Recurrent network, 58, 59
Recursive filter, 80, 92
Reddy, 7
Reduced-range level threshold, 411
Reestimation, 342, 369, 371
formulas, 343–8, 354, 360, 361, 382
transformation, 346
Reference adaptation, 243
Reference pattern, 51, 142, 264
uncertainly region, 411
Reflection coefficient, 115, 190
Rejection of input, 487
Relative:
entropy, 171
frequency approach, 448
Relaxation oscillator, 14
Repellers, 60
Reproduction vectors, 244
Residual:
energy, 353
minimization, 155
Resonance frequencies, 23
Reverberation, 4, 307
Right context diphone, 460
Robust:
distortion measure, 314
recognition decision, 420
training, 266–7
Robustness, 44
Roe, 313
Root power sums, 167

**Saito, 171**
Sakai, 7
Sakoe, 7, 8, 209, 214, 216
Sample mean, 375
Sampling:
frequency, 76
rate reduction, 76
Scalar quantization, 129
Scatter matrix, 298
Second differential spectral distance, 198
Sedgwick, 313
Segmental:
k-means algorithm, 382–3, 393, 427, 441, 443, 445, 473
k-means segmentation, 382–4
Segmentation, 46, 49, 444
manual, 370
maximum likelihood, 370, 380, 437
uniform, 441
Segmentation of speech, 43
Self-correction of errors, 486
Self-detection of errors, 486



Self-organizing feature map, 58, 60
Self transition, 358
Semantic:
constraints, 52, 435
knowledge, 44, 52
postprocessor, 478
rule, 483
Semi-Markov models, 358
Semivowels, 21, 29
Sensory information, 54
Sentence-level match, 451
Separation measure, 298, 301
Sequential adaptation, 285
Sequential:
decision, 204
training, 266
Short-term:
autocorrelation, 86, 155
covariance, 102
energy, 69
Fourier transform, 80–4
acoustic features, 201
modified coherence (SMC), 315
spectral envelope, 69
spectral magnitude, 84
spectral representation, 70
Sidelobe ripple, 94
Sigmoid function, 57, 302
Signal:
dynamic range, 76
enhancement, 2, 95
enhancement preprocessing, 310–11
Signal degradation:
cross-talk, 146
intermodulation distortion, 146
tonal interference, 146
preemphasis, 95
Signal processing:
auditory based, 70, 138
front end, 69
Signal-to-noise ratio (SNR), 305
Silence, 18, 442
Similarity:
measure, 51
score, 51
Single layer perceptrons, 58
Slope weighting, 215–9
Type a, 216
Type b, 216
Type c, 216
Type d, 216
global weighting, 216
Smooth error count, 303
Smoothed group delay:
spectrum, 314
weighting function, 314
Sondhi, 314
Sonorants, 47
Soong, 274, 314
Sound:
classes of American English, 21
classifier, 47–8, 50
coarticulation, 43, 392
field, 312
lexicon, 39
pressure, 14
pressure level (SPL), 306
spectrogram, 19
Source-coding techniques, 4, 244–64
Source entropy, 449
Source-tract separation, 98
Speaker:
adaptation, 285, 288, 374
conversion, 285
transformation, 285
Speaker-dependent mode, 284
Speaker-independent mode, 284, 431
Speaker-trained mode, 284, 431
Speaking:
environment, 52
rate variation, 51, 201
Spectral:
analysis auditory based, 132
analysis methods, 2, 45, 69
analysis models, 70–2
density, 154
density energy, 155
density power, 155
dissimilarity, 142
distortion, 123
distortion measures, 154–94
dynamic features, 194–200
envelope, 19, 84
harmonics, 20
representation, 19
resonances, 6
similarity, 123
slope, 167
tilt, 151, 309
transformation, 286–8
transitions, 196
Spectrogram, 4, 20
narrowband, 19
wideband, 19
Spectrum:
analyzer, 6
group delay, 167
Speech:
analysis system, 45
articulation, 78
detection, 115
enhancement, 310
features, 20–37
generation, 11
perception, 2, 11–37
production, 2, 11–37
recognition history, 6–9
sounds, 20–37
understanding system, 54
Speech signal:
discrete symbol rate, 12
information rate, 12
Spell mode, 291
Sphinx system, 9
Standing wave, 133
Stapes, 133
State duration:
density, 358, 429
density parametric, 361
probability, 384
State sequence:
most likely, 331
optimal, 337
State-transition:
matrix, 327, 349
probabilities, 323, 330
Static pattern, 63
Statistical:
consistency, 267
language model, 447, 448–50
model, 5, 393
pattern recognition approach, 51–2
word bigram, 453
Steinberg, 186
Stevens, 184
Stirrup, 133
Stochastic:
constraint, 345
convergence constraints, 303
gradient technique, 61
Stop:
voiced, 33
unvoiced, 33
Stress compensation, 313
String error rate, 431
known length, 431
unknown length, 431
String-length, 419
Subjective:
judgement, 150
pitch, 183, 185
spectrum, 185, 188
Subword speech units, 435–9
Subword units, 52
clustering, 470, 475
continuous density codebook, 440
models, 439–41
semicontinuous modeling method, 441
splitting, 470
tied mixture approach, 441
training, 441–7
Sufficient statistic, 322

Super-reference pattern, 393, 400
Suzuki, 7
Syllabic sounds, 20
Syllables, 6
Syllable-like units, 436
Synaptic connections, 133, 134
Synchronous sequential decision, 206, 232
Syntactic:
constraints, 52, 435
knowledge, 44, 52
Syntax, 2
analyzer, 435
System grammar, 6

**Tail units, 464**
Talker:
normalization, 28
population, 125
Talking style, 309
Tangora system, 8
Task, 52
perplexity, 485
semantics, 6, 52, 454
syntax, 52, 454
Task-oriented speech recognition system, 6
Task-specific applications, 482
Taylor series expansion, 163
Teeth, 33
Telephone:
channel, 125, 308
network, 9, 308
network loop survey, 308
quality signals, 70
Template, 51, 142, 243, 264, 383
Template adaptation, 285–91
background noise, 285
channel characteristics, 285
speaking environment, 285
talkers and accents, 285
transducers, 285
Template matching, 321
Template training, 264–74
casual, 265
clustering, 267
robust, 266
Temporal smoothing, 96
Temporal variability, 215
Test pattern, 51, 142, 393
ending range, 411
Throat cavity, 14
Tied mixture, 441, 464
Tied states, 356–7, 372
Time alignment, 51, 200–38, 390
optimal, 232
Time assignment speech interpolation (TASI), 143
Time delay neural network (TDNN), 63
Time differential log spectrum, 196
Time-frequency derivative, 200
Time normalization, 200–38
linear, 202
Time normalization constraint, 208–221
endpoint, 208, 230
global path, 208, 213
Itakura condition, 222
local continuity, 208, 209, 216
monotonicity, 208, 209
slope weighting, 208, 215, 222
Time scale:
contraction, 224
expansion, 224
Time spreading function, 64
Time synchronous, 403
level building, 5
Time waveform, 17
Time-sampled cepstral sequence, 196
Toeplitz:
matrix, 106
form, 187
Tonalness units, 185
Tongue, 12, 14
hump position, 21
Top-down processor, 53
Total constriction, 30
Trachea, 14
Training:
corpus, 448
discriminative, 243
problem, 243
procedure, 43
sequential, 266
Transducer, 307, 311
Transfer function, 20
zeros, 30
Transient sounds, 15, 33
Transition:
function, 254
pruning, 255
Translating telephony, 495
Transmission characteristics, 52
Trellis:
codes, 131
constraints, 256
structure, 206, 337
vector quantizers, 254
Trigram:
model, 448
phone models, 373
word probabilities, 373
Triphone units, 459
Truncated cepstral distance, 164
Turbulence, 31
Turbulent sounds, 15
Two-input noise cancelling, 310
Two-level dynamic programming algorithm, 395–8
combinatorics, 399
computation, 399–400
distance computation, 399
Two-sensor input, 312
Tympanic membrane, 133

**Umezaki, 314**
Uniform filter bank, 77
FFT implementation, 87–9
Uniform:
frequency spacing, 87
segmentation, 441
Unit:
acoustic, 437
body, 464
demisyllable-like, 436
dyad, 436
head, 464–5
phonelike, 436
phonetic, 435
reduction rule, 461, 468, 470
separation distance, 470
subword, 390, 435, 437, 439, 443, 445
syllable-like, 436
tail, 464–5
Universal codebook, 288
Unsupervised clustering, 268–74
without averaging (UWA), 268, 279
Unvoiced:
fricatives, 21, 31–2
sounds, 15, 18
stop consonants, 21, 33–6
Unweighted slope metric, 276
Upper path constraint, 406
Urn-and-ball models, 328–9
User friendly, 485
Utterance boundaries, 392
UWA algorithm, 279

**Valid candidate rule, 259**
Variance estimation, 467
Vector norm shrinkage, 315
Vector quantization, 4, 69–70, 122–32, 244, 427
distance measure, 125
efficient pattern matching, 263–4
extensions, 129–31
implementation, 123–4
memoryless, 245
preprocessor, 257
recognition preprocessor, 257–64
segmental, 256
similarity measure, 125



Vector quantization (cont.)
training set, 124–5
vector classification procedure, 128–9
with memory, 254
word based, 257
Vector quantization codebooks:
block memory constraints, 254
centroid computation, 246–7
centroid model spectrum, 248
cepstral distance, 250–1
COSH distortion, 250
likelihood distortions, 247–50
minimum distortion labeling, 246
segmental, 256–7
using memory, 254–6
Velichko, 7
Velum, 12, 14, 30, 33
Verification of response, 487
Vintsyuk, 7
Viswanathan, 312
Viterbi algorithm, 339–40, 368, 378, 382, 385
alternative implementation, 340
Viterbi:
decoding, 444
training, 458
Vocabulary independent units, 477
Vocal cords, 12, 14, 15, 18, 23
constriction, 15
relaxed, 15
tensed, 14
tensed, 18
Vocal tract, 12, 14, 15, 20, 33, 98
closure, 15
constriction, 31
cross sectional area, 14
length, 14
models, 190
natural frequencies, 20
resonances, 20
spectral envelope, 98
total constriction, 33
Voice:
bar, 33
command, 482
control, 483
dialogue, 485
dictation, 495
prompt, 490
repertory dialer, 488–9
response system, 490–1
Voiced:
fricatives, 21, 32–3
speech sounds, 15, 18
stop consonants, 21, 33–4
Volkmann, 184
Volume velocity, 14
Vowel, 20, 21–8
acute, 46
articulatory configuration, 24
back, 21, 24
combinations, 20
compact, 46
diffuse, 46
ellipses, 127
flat, 46
front, 21, 24
grave, 46
lax, 46
mid, 21, 24
nucleus, 436
plain, 46
recognition, 275
recognizer, 7
spectrograms, 24–7
tense, 46
tongue hump height, 24
tongue hump position, 24
triangle, 28
waveforms, 24–6

**Warped:**
cepstral distance, 186
frequency scale, 183–90
Warping function, 202
Waveform amplitude plot, 19
Weighted:
COSH distortion, 180
Euclidean distance, 247
cepstral distance, 171, 314
differential cepstral distance, 199
likelihood distortions, 179
likelihood ratio measure, 276, 277
slope measure, 192, 277
spectral distortion, 314
Whisper, 21
Whitening, 109
Whole word model, 435, 436
Wide-sense stationary process, 155
Wideband spectrogram, 19
Wilpon, 279
Window design method, 90
Windpipe, 14
Wireless network, 489
Within-class scatter, 298
Word:
arcs, 424, 441
boundary, 392
coarticulation, 412
deletion rate, 454
dependent unit, 459
dictionary, 288, 473
equivalence classes, 294–7
error probability, 53
error rate, 485
hypothesis, 53
initial, 38
insertion penalty, 454
insertion rate, 454
junction effects, 465–7
lexicon, 6, 435, 437, 439, 441, 451, 459
pair grammar, 435
pair model, 447
penalty, 454–456
pronunciation rule, 484
reference pattern, 393
sensitive model, 459
sequence probability, 448
sequence, 434
stress, 428
string, 434
syntax, 391
Word-by-word dissimilarity, 294
Word-dependent units, 459

**Zagoruyko, 7**
Zero-crossing:
analysis, 7
rate, 69

Chapter 1

# FUNDAMENTALS OF SPEECH RECOGNITION

## 1.1 INTRODUCTION

Automatic recognition of speech by machine has been a goal of research for more than four decades and has inspired such science fiction wonders as the computer HAL in Stanley Kubrick's famous movie *2001—A Space Odyssey* and the robot R2D2 in the George Lucas classic *Star Wars* series of movies. However, in spite of the glamour of designing an intelligent machine that can recognize the spoken word and comprehend its meaning, and in spite of the enormous research efforts spent in trying to create such a machine, we are far from achieving the desired goal of a machine that can understand spoken discourse on any subject by all speakers in all environments. Thus, an important question in this book is, What do we mean by "speech recognition by machine." Another important question is, How can we build a series of bridges that will enable us to advance both our knowledge as well as the capabilities of modern speech-recognition systems so that the "holy grail" of conversational speech recognition and understanding by machine is attained?

Because we do not know how to solve the ultimate challenge of speech recognition, our goal in this book is to give a series of presentations on the fundamental principles of most modern, successful speech-recognition systems so as to provide a framework from which researchers can expand the frontier. We will attempt to avoid making absolute judgments on the relative merits of various approaches to particular speech-recognition problems. Instead we will provide the theoretical background and justification for each topic discussed so that the reader is able to understand why the techniques have proved

valuable and how they can be used to advantage in practical situations.

One of the most difficult aspects of performing research in speech recognition by machine is its interdisciplinary nature, and the tendency of most researchers to apply a monolithic approach to individual problems. Consider the disciplines that have been applied to one or more speech-recognition problems:

1. **signal processing**—the process of extracting relevant information from the speech signal in an efficient, robust manner. Included in signal processing is the form of spectral analysis used to characterize the time-varying properties of the speech signal as well as various types of signal preprocessing (and postprocessing) to make the speech signal robust to the recording environment (signal enhancement).
2. **physics (acoustics)**—the science of understanding the relationship between the physical speech signal and the physiological mechanisms (the human vocal tract mechanism) that produced the speech and with which the speech is perceived (the human hearing mechanism).
3. **pattern recognition**—the set of algorithms used to cluster data to create one or more prototypical patterns of a data ensemble, and to match (compare) a pair of patterns on the basis of feature measurements of the patterns.
4. **communication and information theory**—the procedures for estimating parameters of statistical models; the methods for detecting the presence of particular speech patterns, the set of modern coding and decoding algorithms (including dynamic programming, stack algorithms, and Viterbi decoding) used to search a large but finite grid for a best path corresponding to a "best" recognized sequence of words.
5. **linguistics**—the relationships between sounds (phonology), words in a language (syntax), meaning of spoken words (semantics), and sense derived from meaning (pragmatics). Included within this discipline are the methodology of grammar and language parsing.
6. **physiology**—understanding of the higher-order mechanisms within the human central nervous system that account for speech production and perception in human beings. Many modern techniques try to embed this type of knowledge within the framework of artificial neural networks (which depend heavily on several of the above disciplines).
7. **computer science**—the study of efficient algorithms for implementing, in software or hardware, the various methods used in a practical speech-recognition system.
8. **psychology**—the science of understanding the factors that enable a technology to be used by human beings in practical tasks.

Successful speech-recognition systems require knowledge and expertise from a wide range of disciplines, a range far larger than any single person can possess. Therefore, it is especially important for a researcher to have a good understanding of the fundamentals of speech recognition (so that a range of techniques can be applied to a variety of problems), without necessarily having to be an expert in each aspect of the problem. It is the purpose of this book to provide this expertise by giving in-depth discussions of a number of



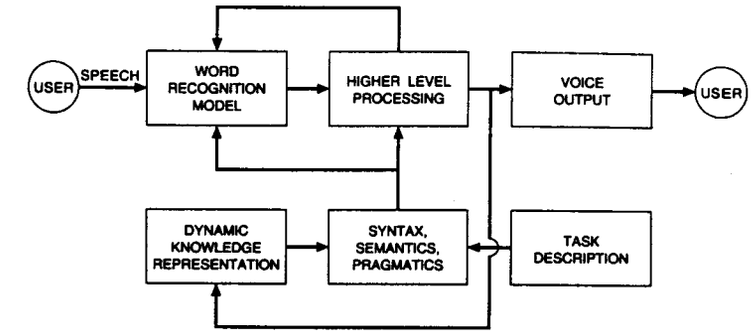



**Figure 1.1** General block diagram of a task-oriented speech-recognition system.

fundamental topics in speech-recognition research.

## 1.2 THE PARADIGM FOR SPEECH RECOGNITION

A general model for speech recognition, shown in Figure 1.1, is used throughout this book. The model begins with a user creating a speech signal (speaking) to accomplish a given task. The spoken output is first recognized in that the speech signal is decoded into a series of words that are meaningful according to the syntax, semantics, and pragmatics of the recognition task. The meaning of the recognized words is obtained by a higher-level processor that uses a dynamic knowledge representation to modify the syntax, semantics, and pragmatics according to the context of what it has previously recognized. In this manner, things such as non sequitors are omitted from consideration at the risk of misunderstanding, but at the gain of minimizing errors for sequentially meaningful inputs. The feedback from the higher-level processing box reduces the complexity of the recognition model by limiting the search for valid (acceptable) input sentences (speech) from the user. The recognition system responds to the user in the form of a voice output, or equivalently, in the form of the requested action being performed, with the user being prompted for more input.

## 1.3 OUTLINE

The material in this book is organized into nine chapters. Chapters 2 through 9 each deals with a basic concept or a fundamental technique used in various speech-recognition systems. The material discussed in these chapters is as follows.

**Chapter 2—The Speech Signal: Production, Perception, and Acoustic-Phonetic Characterization.** In this chapter we review the speech production/perception process in human beings. We show how different speech sounds can be characterized by a set of

spectral and temporal properties that depend on the acoustic-phonetic features of the sound and are manifest in the waveform, the sound spectrogram, or both. Included in the chapter is an overview of the three most common approaches to speech recognition, namely the acoustic-phonetic approach (which tries to directly exploit individual sound properties), the pattern recognition approach (which relies only on gross spectral and temporal properties of speech sounds and uses conventional as well as neural network pattern recognition technology to classify sounds), and the artificial intelligence (AI) approach in which an expert system or a self-organizing (learning) system, as implemented by neural networks, is used to classify sounds. We discuss the strengths and weaknesses of each approach and explain why the pattern-recognition approach is the one most heavily relied on in practical systems. We conclude the chapter with a discussion of the fundamental issues in speech recognition (i.e., those factors that most influence overall system performance), and with a brief overview of current applications.

**Chapter 3—Signal Processing and Analysis Methods for Speech Recognition.** In this chapter we present the two fundamental signal-processing approaches to speech spectral analysis: filter bank and linear predictive methods. We specialize the presentation of these two fundamental techniques to aspects related to speech analysis and compare and contrast the two methods in terms of robustness to speech sounds and required computation. For completeness we also discuss the popular source-coding technique referred to as vector quantization (VQ). Here, a codebook is created to represent the anticipated range of spectral vectors. This enables us to code an arbitrary continuous speech spectral vector into one of a fixed number of discrete codebook symbols at the cost of increased error in signal representation but with the benefit of significantly reduced computation in the recognition process. We conclude this chapter with a discussion of a spectral analysis model that attempts to mimic the processing in the human auditory system—the so-called ear model. Although our knowledge of the higher-order processing in the central nervous system is rudimentary, the importance of ear models is related to their robustness to noise, reverberation, and other environmental factors that often seriously degrade performance of current speech recognizers.

**Chapter 4—Pattern-Comparison Techniques.** In this chapter we discuss three fundamental aspects of comparing a pair of speech patterns. These are the basic concept of detecting speech (i.e., finding the speech signal in a background of noise or other acoustic interference), the idea of computing a measure of the local distance (or similarity) of a pair of spectral representations of a short-time piece of speech signal (a distance or distortion measure), and the concept of temporally aligning and globally comparing a pair of speech patterns corresponding to different speech utterances that may or may not be the same sequence of sounds or words (dynamic time warping algorithms). We show in this chapter how the basic pattern-comparison techniques can be combined in a uniform framework for speech-recognition applications.

**Chapter 5—Speech-Recognition System Design and Implementation Issues.** In this chapter we discuss the remaining pieces (after signal processing and pattern comparison) that enable us to build and study performance of a practical speech-recognition system.



In particular we discuss how speech recognizers are trained and how we can enhance the basic recognition procedure by adding features, by exploiting a preprocessor, by the use of methods of adaptation or by postprocessing the recognizer outputs using a set of pattern discriminators (as opposed to the pattern classifiers used in a conventional implementation). We conclude the chapter with a discussion of various ways of recognizing speech in adverse environments (e.g., noise, stress conditions, or mismatch between training and testing).

**Chapter 6—Theory and Implementation of Hidden Markov Models.** In this chapter we discuss aspects of the theory and implementation of the set of statistical modeling techniques collectively referred to as hidden Markov modeling. Included within these techniques are the algorithms for scoring a statistical (Markovian) model against a speech pattern, the techniques for aligning the model with the speech pattern so as to recover an estimate of the alignment path between different speech sounds and different model states, and the techniques for estimating parameters of the statistical models from a training set of utterances of the sounds being modeled. Also included is a discussion of the practical aspects of building hidden Markov models, including the issues of scaling of data, handling of multiple observation sequences, providing initial estimates of model parameters, and combating the problems of insufficient training data. We conclude the chapter with a practical example illustrating how a simple, isolated word recognizer would be implemented using hidden Markov models.

**Chapter 7—Speech Recognition Based on Connected Word Models.** In this chapter we show how the basic set of techniques developed for recognizing an isolated word or phrase can be readily extended to recognizing a sequence of words (e.g., a string of digits of a credit card number) spoken in a fluent or connected manner. We make the simplifying assumption that the connected word string is recognized by finding the optimal sequence of word models that best matches the spoken string. Hence we assume that the word is the basic recognition unit for these systems, and therefore the training problem is one of estimating the optimal parameters of word models on the basis of training data, which need not contain isolated versions of the words. We describe three "optimal" approaches to solving the recognition part of connected word-recognition problems: (1) the two-level dynamic programming method, (2) the level building method, and (3) the time synchronous level building (or the one-pass) method and discuss the properties, and the relative strengths and weaknesses of each method. We then show how we can optimally train connected word systems, even if isolated versions of the vocabulary words are not available. We conclude the chapter with a discussion of a connected digit recognizer implemented using the methods described in the chapter.

**Chapter 8—Large Vocabulary Continuous Speech Recognition.** In this chapter we discuss the issues in applying speech-recognition technology to the problem of recognizing fluently spoken speech with vocabulary sizes of 1000 or more words (with unlimited vocabularies as the ultimate goal). It is shown that a number of fundamental problems must be solved to implement such a system, including the choice of a basic subword speech unit (from which words, phrases, and sentences can be built up), an effective way

of modeling the basic speech unit, a way of deriving models of the unit, a way of designing and implementing a word lexicon (which provides a mapping between words and subword units), a way of implementing task syntax (the system grammar), a way of implementing the overall recognition part of the system (via some type of network search), and a way of imposing task semantics onto the solution. We concentrate primarily on the issues involved in building large vocabulary recognition systems. For illustrative purposes we describe one reasonable way of building such a system and discuss the resulting performance on a standard database management task.

**Chapter 9—Task Oriented Applications of Automatic Speech Recognition.** The final chapter of the book provides a brief overview of how one might apply the ideas discussed in the book to building a real, task-oriented, speech recognition system. It includes discussions of how one would evaluate recognizer performance and how one might decide whether a proposed task is viable for speech recognition. We also discuss a set of broad classes of applications, which appear to be the most promising ones at this time, along with typical examples of how recognizers have been successfully employed within these broad classes. The chapter concludes with some broad performance projections through the year 2000.

## 1.4 A BRIEF HISTORY OF SPEECH-RECOGNITION RESEARCH

Research in automatic speech recognition by machine has been done for almost four decades. To gain an appreciation for the amount of progress achieved over this period, it is worthwhile to briefly review some research highlights. The reader is cautioned that such a review is cursory, at best, and must therefore suffer from errors of judgment as well as omission.

The earliest attempts to devise systems for automatic speech recognition by machine were made in the 1950s, when various researchers tried to exploit the fundamental ideas of acoustic-phonetics. In 1952, at Bell Laboratories, Davis, Biddulph, and Balashek built a system for isolated digit recognition for a single speaker [1]. The system relied heavily on measuring spectral resonances during the vowel region of each digit. In an independent effort at RCA Laboratories in 1956, Olson and Belar tried to recognize 10 distinct syllables of a single talker, as embodied in 10 monosyllabic words [2]. The system again relied on spectral measurements (as provided by an analog filter bank) primarily during vowel regions. In 1959, at University College in England, Fry and Denes tried to build a phoneme recognizer to recognize four vowels and nine consonants [3]. They used a spectrum analyzer and a pattern matcher to make the recognition decision. A novel aspect of this research was the use of statistical information about allowable sequences of phonemes in English (a rudimentary form of language syntax) to improve overall phoneme accuracy for words consisting of two or more phonemes. Another effort of note in this period was the vowel recognizer of Forgie and Forgie, constructed at MIT Lincoln Laboratories in 1959, in which 10 vowels embedded in a /b/-vowel-/t/ format were recognized in a speaker-independent manner [4]. Again a filter bank analyzer was used to provide spectral information, and a



time varying estimate of the vocal tract resonances was made to decide which vowel was spoken.

In the 1960s several fundamental ideas in speech recognition surfaced and were published. However, the decade started with several Japanese laboratories entering the recognition arena and building special-purpose hardware as part of their systems. One early Japanese system, described by Suzuki and Nakata of the Radio Research Lab in Tokyo [5], was a hardware vowel recognizer. An elaborate filter bank spectrum analyzer was used along with logic that connected the outputs of each channel of the spectrum analyzer (in a weighted manner) to a vowel-decision circuit, and a majority decision logic scheme was used to choose the spoken vowel. Another hardware effort in Japan was the work of Sakai and Doshita of Kyoto University in 1962, who built a hardware phoneme recognizer [6]. A hardware speech segmenter was used along with a zero-crossing analysis of different regions of the spoken input to provide the recognition output. A third Japanese effort was the digit recognizer hardware of Nagata and coworkers at NEC Laboratories in 1963 [7]. This effort was perhaps most notable as the initial attempt at speech recognition at NEC and led to a long and highly productive research program.

In the 1960s three key research projects were initiated that have had major implications on the research and development of speech recognition for the past 20 years. The first of these projects was the efforts of Martin and his colleagues at RCA Laboratories, beginning in the late 1960s, to develop realistic solutions to the problems associated with nonuniformity of time scales in speech events. Martin developed a set cf elementary time-normalization methods, based on the ability to reliably detect speech starts and ends, that significantly reduced the variability of the recognition scores [8]. Martin ultimately developed the method and founded one of the first companies, Threshold Technology, which built, marketed, and sold speech-recognition products. At about the same time, in the Soviet Union, Vintsyuk proposed the use of dynamic programming methods for time aligning a pair of speech utterances [9]. Although the essence of the concepts of dynamic time warping, as well as rudimentary versions of the algorithms for connected word recognition, were embodied in Vintsyuk's work, it was largely unknown in the West and did not come to light until the early 1980s; this was long after the more formal methods were proposed and implemented by others.

A final achievement of note in the 1960s was the pioneering research of Reddy in the field of continuous speech recognition by dynamic tracking of phonemes [10]. Reddy's research eventually spawned a long and highly successful speech-recognition research program at Carnegie Mellon University (to which Reddy moved in the late 1960s) which, to this day, remains a world leader in continuous-speech-recognition systems.

In the 1970s speech-recognition research achieved a number of significant milestones. First the area of isolated word or discrete utterance recognition became a viable and usable technology based on fundamental studies by Velichko and Zagoruyko in Russia [11], Sakoe and Chiba in Japan [12], and Itakura in the United States [13]. The Russian studies helped advance the use of pattern-recognition ideas in speech recognition; the Japanese research showed how dynamic programming methods could be successfully applied; and Itakura's research showed how the ideas of linear predictive coding (LPC), which had already been successfully used in low-bit-rate speech coding, could be extended to speech-

recognition systems through the use of an appropriate distance measure based on LPC spectral parameters.

Another milestone of the 1970s was the beginning of a longstanding, highly successful group effort in large vocabulary speech recognition at IBM in which researchers studied three distinct tasks over a period of almost two decades, namely the New Raleigh language [14] for simple database queries, the laser patent text language [15] for transcribing laser patents, and the office correspondence task, called Tangora [16], for dictation of simple memos.

Finally, at AT&T Bell Labs, researchers began a series of experiments aimed at making speech-recognition systems that were truly speaker independent [17]. To achieve this goal a wide range of sophisticated clustering algorithms were used to determine the number of distinct patterns required to represent all variations of different words across a wide user population. This research has been refined over a decade so that the techniques for creating speaker-independent patterns are now well understood and widely used.

Just as isolated word recognition was a key focus of research in the 1970s, the problem of connected word recognition was a focus of research in the 1980s. Here the goal was to create a robust system capable of recognizing a fluently spoken string of words (e.g., digits) based on matching a concatenated pattern of individual words. A wide variety of connected word-recognition algorithms were formulated and implemented, including the two-level dynamic programming approach of Sakoe at Nippon Electric Corporation (NEC) [18], the one-pass method of Bridle and Brown at Joint Speech Research Unit (JSRU) in England [19], the level building approach of Myers and Rabiner at Bell Labs [20], and the frame synchronous level building approach of Lee and Rabiner at Bell Labs [21]. Each of these "optimal" matching procedures had its own implementational advantages, which were exploited for a wide range of tasks.

Speech research in the 1980s was characterized by a shift in technology from template-based approaches to statistical modeling methods—especially the hidden Markov model approach [22, 23]. Although the methodology of hidden Markov modeling (HMM) was well known and understood in a few laboratories (primarily IBM, Institute for Defense Analyses (IDA), and Dragon Systems), it was not until widespread publication of the methods and theory of HMMs, in the mid-1980s, that the technique became widely applied in virtually every speech-recognition research laboratory in the world.

Another "new" technology that was reintroduced in the late 1980s was the idea of applying neural networks to problems in speech recognition. Neural networks were first introduced in the 1950s, but they did not prove useful initially because they had many practical problems. In the 1980s, however, a deeper understanding of the strengths and limitations of the technology was obtained, as well as the relationships of the technology to classical signal classification methods. Several new ways of implementing systems were also proposed [24, 25].

Finally, the 1980s was a decade in which a major impetus was given to large vocabulary, continuous-speech-recognition systems by the Defense Advanced Research Projects Agency (DARPA) community, which sponsored a large research program aimed at achieving high word accuracy for a 1000-word, continuous-speech-recognition, database management task. Major research contributions resulted from efforts at CMU (notably the well-known SPHINX system) [26], BBN with the BYBLOS system [27], Lincoln Labs [28], SRI [29], MIT [30], and AT&T Bell Labs [31]. The DARPA program has continued into the 1990s, with emphasis shifting to natural language front ends to the recognizer, and the task shifting to retrieval of air travel information. At the same time, speech-recognition technology has been increasingly used within telephone networks to automate as well as enhance operator services.



## REFERENCES


[1] K. H. Davis, R. Biddulph, and S. Balashek, "Automatic Recognition of Spoken Digits," *J. Acoust. Soc. Am.*, 24 (6): 637–642, 1952.

[2] H. F. Olson and H. Belar, "Phonetic Typewriter," *J. Acoust. Soc. Am.*, 28 (6): 1072–1081, 1956.

[3] D. B. Fry, "Theoretical Aspects of Mechanical Speech Recognition"; and P. Denes, "The Design and Operation of the Mechanical Speech Recognizer at University College London," *J. British Inst. Radio Engr.*, 19: 4, 211–229, 1959.

[4] J. W. Forgie and C. D. Forgie, "Results Obtained From a Vowel Recognition Computer Program," *J. Acoust. Soc. Am.*, 31 (11): 1480–1489, 1959.

[5] J. Suzuki and K. Nakata, "Recognition of Japanese Vowels—Preliminary to the Recognition of Speech," *J. Radio Res. Lab*, 37 (8): 193–212, 1961.

[6] T. Sakai and S. Doshita, "The Phonetic Typewriter, Information Processing 1962," *Proc. IFIP Congress*, Munich, 1962.

[7] K. Nagata, Y. Kato, and S. Chiba, "Spoken Digit Recognizer for Japanese Language," *NEC Res. Develop.*, No. 6, 1963.

[8] T. B. Martin, A. L. Nelson, and H. J. Zadell, "Speech Recognition by Feature Abstraction Techniques," Tech. Report AL-TDR-64-176, Air Force Avionics Lab, 1964.

[9] T. K. Vintsyuk, "Speech Discrimination by Dynamic Programming," *Kibernetika*, 4 (2): 81–88, Jan.–Feb. 1968.

[10] D. R. Reddy, "An Approach to Computer Speech Recognition by Direct Analysis of the Speech Wave," Tech. Report No. C549, Computer Science Dept., Stanford Univ., September 1966.

[11] V. M. Velichko and N. G. Zagoruyko, "Automatic Recognition of 200 Words," *Int. J. Man-Machine Studies*, 2: 223, June 1970.

[12] H. Sakoe and S. Chiba, "Dynamic Programming Algorithm Optimization for Spoken Word Recognition," *IEEE Trans. Acoustics, Speech, Signal Proc.*, ASSP-26 (1): 43-49, February 1978.

[13] F. Itakura, "Minimum Prediction Residual Applied to Speech Recognition," *IEEE Trans. Acoustics, Speech, Signal Proc.*, ASSP-23 (1): 67-72, February 1975.

[14] C. C. Tappert, N. R. Dixon, A. S. Rabinowitz, and W. D. Chapman, "Automatic Recognition of Continuous Speech Utilizing Dynamic Segmentation, Dual Classification, Sequential Decoding and Error Recovery," Rome Air Dev. Cen, Rome, NY, Tech. Report TR-71-146, 1971.

[15] F. Jelinek, L. R. Bahl, and R. L. Mercer, "Design of a Linguistic Statistical Decoder for the

Recognition of Continuous Speech," *IEEE Trans. Information Theory*, IT-21: 250–256, 1975.

[16] F. Jelinek, "The Development of an Experimental Discrete Dictation Recognizer," *Proc. IEEE*, 73 (11): 1616–1624, 1985.

[17] L. R. Rabiner, S. E. Levinson, A. E. Rosenberg, and J. G. Wilpon, "Speaker Independent Recognition of Isolated Words Using Clustering Techniques," *IEEE Trans. Acoustics, Speech, Signal Proc.*, ASSP-27: 336–349, August 1979.

[18] H. Sakoe, "Two Level DP Matching—A Dynamic Programming Based Pattern Matching Algorithm for Connected Word Recognition," *IEEE Trans. Acoustics, Speech, Signal Proc.*, ASSP-27: 588–595, December 1979.

[19] J. S. Bridle and M. D. Brown, "Connected Word Recognition Using Whole Word Templates," *Proc. Inst. Acoust. Autumn Conf.*, 25–28, November 1979.

[20] C. S. Myers and L. R. Rabiner, "A Level Building Dynamic Time Warping Algorithm for Connected Word Recognition," *IEEE Trans. Acoustics, Speech, Signal Proc.*, ASSP-29: 284–297, April 1981.

[21] C. H. Lee and L. R. Rabiner, "A Frame Synchronous Network Search Algorithm for Connected Word Recognition," *IEEE Trans. Acoustics, Speech, Signal Proc.*, 37 (11): 1649–1658, November 1989.

[22] J. Ferguson, Ed., *Hidden Markov Models for Speech*, IDA, Princeton, NJ, 1980.

[23] L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proc. IEEE*, 77 (2): 257–286, February 1989.

[24] R. P. Lippmann, "An Introduction to Computing with Neural Nets," *IEEE ASSP Mag.*, 4 (2): 4–22, April 1987.

[25] A. Weibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang, "Phoneme Recognition Using Time-Delay Neural Networks," *IEEE Trans. Acoustics, Speech, Signal Proc.*, 37: 393–404, 1989.

[26] K. F. Lee, H. W. Hon, and D. R. Reddy, "An Overview of the SPHINX Speech Recognition System," *IEEE Trans. Acoustics, Speech, Signal Proc.*, 38: 600–610, 1990.

[27] Y. L. Chow, M. O. Dunham, O. A. Kimball, M. A. Krasner, G. F. Kubala, J. Makhoul, S. Roucos, and R. M. Schwartz, "BBYLOS: The BBN Continuous Speech Recognition System," *Proc. ICASSP 87*, 89–92, April 1987.

[28] D. B. Paul, "The Lincoln Robust Continuous Speech Recognizer," *Proc. ICASSP 89*, Glasgow, Scotland, 449–452, May 1989.

[29] M. Weintraub et al., "Linguistic Constraints in Hidden Markov Model Based Speech Recognition," *Proc. ICASSP 89*, Glasgow, Scotland, 699–702, May 1989.

[30] V. Zue, J. Glass, M. Phillips, and S. Seneff, "The MIT Summit Speech Recognition System: A Progress Report," *Proc. DARPA Speech and Natural Language Workshop*, 179–189, February 1989.

[31] C. H. Lee, L. R. Rabiner, R. Pieraccini, and J. G. Wilpon, "Acoustic Modeling for Large Vocabulary Speech Recognition," *Computer Speech and Language*, 4: 127–165, 1990.

Chapter 2

# THE SPEECH SIGNAL: PRODUCTION, PERCEPTION, AND ACOUSTIC-PHONETIC CHARACTERIZATION

## 2.1 INTRODUCTION

In this chapter we discuss the mechanics of producing and perceiving speech in human beings, and we show how an understanding of these processes leads naturally to several different approaches to speech recognition by machine. We begin by showing how the different classes of speech sounds, or phonetics, can each be characterized in terms of broad acoustic features whose properties are relatively invariant across words and speakers. The ideas of acoustic-phonetic characterization of sounds lead naturally to straightforward implementation of a speech-recognition algorithm based on sequential detection of sounds and sound classes. The strengths and weaknesses of such an approach are discussed. An alternative approach to speech recognition is to use standard pattern-recognition techniques in a framework in which all speech knowledge is "learned" via a training phase. We show that such a "blind" approach has some natural advantages for a wide range of speech-recognition systems. Finally we show how aspects of both the acoustic-phonetic approach and the pattern-recognition approach can be integrated into a hybrid method that includes techniques from artificial intelligence as well as neural network methods.

### 2.1.1 The Process of Speech Production and Perception in Human Beings

Figure 2.1 shows a schematic diagram of the speech-production/speech-perception process in human beings. The production (speech-generation) process begins when the talker