XUEDONG HUANG | ALEX ACERO | HSIAO-WUEN HON

SPOKEN LANGUAGE PROCESSING

A Guide to Theory, Algorithm, and System Development



Foreword by Dr. Raj Reddy
Carnegie Mellon University

EX-1020
U.S. Patent 7,818,176

Spoken Language Processing

ISBN 0-13-022616-5

90000

9 780130 226167

Spoken Language Processing

A Guide to Theory, Algorithm, and System Development

Xuedong Huang

Alex Acero

Hsiao-Wuen Hon

Microsoft Research



Prentice Hall PTR
Upper Saddle River, New Jersey 07458
www.phptr.com

Library of Congress Cataloging-in-Publication Data

Huang, Xuedong.
Spoken language processing: a guide to theory, algorithm, and system development/
Xuedong Huang, Alex Acero, Hsiao-Wuen Hon.
p. cm.
Includes bibliographical references and index.
ISBN 0-13-022616-5
1. Natural language processing (Computer science) I. Acero, Alex. II. Hon,
Hsiao-Wuen. III. Title.

QA76.9.N38 H83 2001
006.3'5—dc21 00-050196

Editorial/production supervision: *Jane Bonnell*
Cover design director: *Jerry Votta*
Cover design: *Anthony Gemmellaro*
Manufacturing buyer: *Maura Zaldivar*
Development editor: *Russ Hall*
Acquisitions editor: *Tim Moore*
Editorial assistant: *Allyson Kloss*
Marketing manager: *Debby van Dijk*



© 2001 by Prentice Hall PTR
Prentice-Hall, Inc.
Upper Saddle River, New Jersey 07458

Prentice Hall books are widely used by corporations and government agencies for training, marketing, and resale.
The publisher offers discounts on this book when ordered in bulk quantities. For more information, contact Corporate Sales Department, Phone: 800-382-3419; FAX: 201-236-7141;
E-mail: corpsales@prenhall.com
Or write: Prentice Hall PTR, Corporate Sales Dept., One Lake Street, Upper Saddle River, NJ 07458.

Company and product names mentioned herein are the trademarks or registered trademarks of their respective owners.

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

ISBN 0-13-022616-5

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*
Pearson Education Asia Pte. Ltd.
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

To Yingzhi, Angela, Christina, and Derek

To Donna and Nicolas

To Phen, Stephanie, and Jacqueline

Contents

FOREWORD xxi

PREFACE xxv

1. INTRODUCTION 1

1.1. MOTIVATIONS 2
1.1.1. Spoken Language Interface 2
1.1.2. Speech-to-Speech Translation 3
1.1.3. Knowledge Partners 3
1.2. SPOKEN LANGUAGE SYSTEM ARCHITECTURE 4
1.2.1. Automatic Speech Recognition 4
1.2.2. Text-to-Speech Conversion 6
1.2.3. Spoken Language Understanding 7
1.3. BOOK ORGANIZATION 8
1.3.1. Part I: Fundamental Theory 9
1.3.2. Part II: Speech Processing 9
1.3.3. Part III: Speech Recognition 9
1.3.4. Part IV: Text-to-Speech Systems 10
1.3.5. Part V: Spoken Language Systems 10
1.4. TARGET AUDIENCES 10
1.5. HISTORICAL PERSPECTIVE AND FURTHER READING 11

PART I: FUNDAMENTAL THEORY

2. SPOKEN LANGUAGE STRUCTURE 19

2.1. SOUND AND HUMAN SPEECH SYSTEMS 21
2.1.1. Sound 21
2.1.2. Speech Production 24
2.1.3. Speech Perception 29

2.2. PHONETICS AND PHONOLOGY 36
2.2.1. *Phonemes* 36
2.2.2. *The Allophone: Sound and Context* 47
2.2.3. *Speech Rate and Coarticulation* 49
2.3. SYLLABLES AND WORDS 51
2.3.1. *Syllables* 51
2.3.2. *Words* 53
2.4. SYNTAX AND SEMANTICS 58
2.4.1. *Syntactic Constituents* 58
2.4.2. *Semantic Roles* 63
2.4.3. *Lexical Semantics* 64
2.4.4. *Logical Form* 67
2.5. HISTORICAL PERSPECTIVE AND FURTHER READING 68

3. PROBABILITY, STATISTICS, AND INFORMATION THEORY 73
3.1. PROBABILITY THEORY 74
3.1.1. *Conditional Probability and Bayes' Rule* 75
3.1.2. *Random Variables* 77
3.1.3. *Mean and Variance* 79
3.1.4. *Covariance and Correlation* 82
3.1.5. *Random Vectors and Multivariate Distributions* 83
3.1.6. *Some Useful Distributions* 85
3.1.7. *Gaussian Distributions* 92
3.2. ESTIMATION THEORY 98
3.2.1. *Minimum/Least Mean Squared Error Estimation* 99
3.2.2. *Maximum Likelihood Estimation* 104
3.2.3. *Bayesian Estimation and MAP Estimation* 107
3.3. SIGNIFICANCE TESTING 113
3.3.1. *Level of Significance* 114
3.3.2. *Normal Test (Z-Test)* 115
3.3.3. *χ^2 Goodness-of-Fit Test* 116
3.3.4. *Matched-Pairs Test* 118
3.4. INFORMATION THEORY 120
3.4.1. *Entropy* 120
3.4.2. *Conditional Entropy* 123
3.4.3. *The Source Coding Theorem* 124
3.4.4. *Mutual Information and Channel Coding* 126
3.5. HISTORICAL PERSPECTIVE AND FURTHER READING 128

4. PATTERN RECOGNITION 133
4.1. BAYES' DECISION THEORY 134
4.1.1. *Minimum-Error-Rate Decision Rules* 135

4.1.2. *Discriminant Functions* 138
4.2. HOW TO CONSTRUCT CLASSIFIERS 140
4.2.1. *Gaussian Classifiers* 142
4.2.2. *The Curse of Dimensionality* 144
4.2.3. *Estimating the Error Rate* 146
4.2.4. *Comparing Classifiers* 148
4.3. DISCRIMINATIVE TRAINING 150
4.3.1. *Maximum Mutual Information Estimation* 150
4.3.2. *Minimum-Error-Rate Estimation* 156
4.3.3. *Neural Networks* 158
4.4. UNSUPERVISED ESTIMATION METHODS 163
4.4.1. *Vector Quantization* 163
4.4.2. *The EM Algorithm* 170
4.4.3. *Multivariate Gaussian Mixture Density Estimation* 172
4.5. CLASSIFICATION AND REGRESSION TREES 175
4.5.1. *Choice of Question Set* 177
4.5.2. *Splitting Criteria* 178
4.5.3. *Growing the Tree* 181
4.5.4. *Missing Values and Conflict Resolution* 182
4.5.5. *Complex Questions* 182
4.5.6. *The Right-Sized Tree* 184
4.6. HISTORICAL PERSPECTIVE AND FURTHER READING 190

PART II: SPEECH PROCESSING

5. DIGITAL SIGNAL PROCESSING 201

5.1. DIGITAL SIGNALS AND SYSTEMS 202
5.1.1. *Sinusoidal Signals* 203
5.1.2. *Other Digital Signals* 206
5.1.3. *Digital Systems* 206
5.2. CONTINUOUS-FREQUENCY TRANSFORMS 208
5.2.1. *The Fourier Transform* 208
5.2.2. *Z-Transform* 211
5.2.3. *Z-Transforms of Elementary Functions* 212
5.2.4. *Properties of the Z- and Fourier Transforms* 215
5.3. DISCRETE-FREQUENCY TRANSFORMS 216
5.3.1. *The Discrete Fourier Transform (DFT)* 218
5.3.2. *Fourier Transforms of Periodic Signals* 219
5.3.3. *The Fast Fourier Transform (FFT)* 222
5.3.4. *Circular Convolution* 227
5.3.5. *The Discrete Cosine Transform (DCT)* 228

5.4. DIGITAL FILTERS AND WINDOWS 229
5.4.1. *The Ideal Low-Pass Filter* 229
5.4.2. *Window Functions* 230
5.4.3. *FIR Filters* 232
5.4.4. *IIR Filters* 238
5.5. DIGITAL PROCESSING OF ANALOG SIGNALS 242
5.5.1. *Fourier Transform of Analog Signals* 243
5.5.2. *The Sampling Theorem* 243
5.5.3. *Analog-to-Digital Conversion* 245
5.5.4. *Digital-to-Analog Conversion* 246
5.6. MULTIRATE SIGNAL PROCESSING 248
5.6.1. *Decimation* 248
5.6.2. *Interpolation* 249
5.6.3. *Resampling* 250
5.7. FILTERBANKS 251
5.7.1. *Two-Band Conjugate Quadrature Filters* 251
5.7.2. *Multiresolution Filterbanks* 254
5.7.3. *The DFT as a Filterbank* 255
5.7.4. *Modulated Lapped Transforms* 258
5.8. STOCHASTIC PROCESSES 260
5.8.1. *Statistics of Stochastic Processes* 261
5.8.2. *Stationary Processes* 264
5.8.3. *LTI Systems with Stochastic Inputs* 267
5.8.4. *Power Spectral Density* 268
5.8.5. *Noise* 269
5.9. HISTORICAL PERSPECTIVE AND FURTHER READING 270

6. SPEECH SIGNAL REPRESENTATIONS 275

6.1. SHORT-TIME FOURIER ANALYSIS 276
6.1.1. *Spectrograms* 276
6.1.2. *Pitch-Synchronous Analysis* 281
6.2. ACOUSTICAL MODEL OF SPEECH PRODUCTION 283
6.2.1. *Glottal Excitation* 283
6.2.2. *Lossless Tube Concatenation* 284
6.2.3. *Source-Filter Models of Speech Production* 284
6.3. LINEAR PREDICTIVE CODING 288
6.3.1. *The Orthogonality Principle* 290
6.3.2. *Solution of the LPC Equations* 291
6.3.3. *Spectral Analysis via LPC* 292
6.3.4. *The Prediction Error* 300
6.3.5. *Equivalent Representations* 301
303

6.4. CEPSTRAL PROCESSING 306
6.4.1. *The Real and Complex Cepstrum* 307
6.4.2. *Cepstrum of Pole-Zero Filters* 308
6.4.3. *Cepstrum of Periodic Signals* 311
6.4.4. *Cepstrum of Speech Signals* 312
6.4.5. *Source-Filter Separation via the Cepstrum* 314
6.5. PERCEPTUALLY MOTIVATED REPRESENTATIONS 315
6.5.1. *The Bilinear Transform* 315
6.5.2. *Mel-Frequency Cepstrum* 316
6.5.3. *Perceptual Linear Prediction (PLP)* 318
6.6. FORMANT FREQUENCIES 319
6.6.1. *Statistical Formant Tracking* 320
6.7. THE ROLE OF PITCH 324
6.7.1. *Autocorrelation Method* 324
6.7.2. *Normalized Cross-Correlation Method* 327
6.7.3. *Signal Conditioning* 329
6.7.4. *Pitch Tracking* 330
6.8. HISTORICAL PERSPECTIVE AND FURTHER READING 332

7. SPEECH CODING 337

7.1. SPEECH CODERS ATTRIBUTES 338
7.2. SCALAR WAVEFORM CODERS 340
7.2.1. *Linear Pulse Code Modulation (PCM)* 340
7.2.2. *μ-law and A-law PCM* 342
7.2.3. *Adaptive PCM* 344
7.2.4. *Differential Quantization* 345
7.3. SCALAR FREQUENCY DOMAIN CODERS 348
7.3.1. *Benefits of Masking* 349
7.3.2. *Transform Coders* 350
7.3.3. *Consumer Audio* 351
7.3.4. *Digital Audio Broadcasting (DAB)* 352
7.4. CODE EXCITED LINEAR PREDICTION (CELP) 353
7.4.1. *LPC Vocoder* 353
7.4.2. *Analysis by Synthesis* 353
7.4.3. *Pitch Prediction: Adaptive Codebook* 356
7.4.4. *Perceptual Weighting and Postfiltering* 357
7.4.5. *Parameter Quantization* 358
7.4.6. *CELP Standards* 359
7.5. LOW-BIT RATE SPEECH CODERS 361
7.5.1. *Mixed-Excitation LPC Vocoder* 362
7.5.2. *Harmonic Coding* 363
7.5.3. *Waveform Interpolation* 367
7.6. HISTORICAL PERSPECTIVE AND FURTHER READING 371

PART III: SPEECH RECOGNITION

8. HIDDEN MARKOV MODELS 377

8.1. THE MARKOV CHAIN 378
8.2. DEFINITION OF THE HIDDEN MARKOV MODEL 380
8.2.1. Dynamic Programming and DTW 383
8.2.2. How to Evaluate an HMM—The Forward Algorithm 385
8.2.3. How to Decode an HMM—The Viterbi Algorithm 387
8.2.4. How to Estimate HMM Parameters—Baum-Welch Algorithm 389
8.3. CONTINUOUS AND SEMICONTINUOUS HMMS 394
8.3.1. Continuous Mixture Density HMMs 394
8.3.2. Semicontinuous HMMs 396
8.4. PRACTICAL ISSUES IN USING HMMS 398
8.4.1. Initial Estimates 398
8.4.2. Model Topology 399
8.4.3. Training Criteria 401
8.4.4. Deleted Interpolation 401
8.4.5. Parameter Smoothing 403
8.4.6. Probability Representations 404
8.5. HMM LIMITATIONS 405
8.5.1. Duration Modeling 406
8.5.2. First-Order Assumption 408
8.5.3. Conditional Independence Assumption 409
8.6. HISTORICAL PERSPECTIVE AND FURTHER READING 409

9. ACOUSTIC MODELING 415

9.1. VARIABILITY IN THE SPEECH SIGNAL 416
9.1.1. Context Variability 417
9.1.2. Style Variability 418
9.1.3. Speaker Variability 418
9.1.4. Environment Variability 419
9.2. HOW TO MEASURE SPEECH RECOGNITION ERRORS 419
9.3. SIGNAL PROCESSING—EXTRACTING FEATURES 421
9.3.1. Signal Acquisition 422
9.3.2. End-Point Detection 422
9.3.3. MFCC and Its Dynamic Features 424
9.3.4. Feature Transformation 426
9.4. PHONETIC MODELING—SELECTING APPROPRIATE UNITS 428
9.4.1. Comparison of Different Units 429
9.4.2. Context Dependency 430
9.4.3. Clustered Acoustic-Phonetic Units 432
9.4.4. Lexical Baseforms 436

9.5. ACOUSTIC MODELING—SCORING ACOUSTIC FEATURES 439
9.5.1. Choice of HMM Output Distributions 439
9.5.2. Isolated vs. Continuous Speech Training 441
9.6. ADAPTIVE TECHNIQUES—MINIMIZING MISMATCHES 444
9.6.1. Maximum a Posteriori (MAP) 445
9.6.2. Maximum Likelihood Linear Regression (MLLR) 447
9.6.3. MLLR and MAP Comparison 450
9.6.4. Clustered Models 452
9.7. CONFIDENCE MEASURES: MEASURING THE RELIABILITY 453
9.7.1. Filler Models 453
9.7.2. Transformation Models 454
9.7.3. Combination Models 456
9.8. OTHER TECHNIQUES 457
9.8.1. Neural Networks 457
9.8.2. Segment Models 459
9.9. CASE STUDY: WHISPER 464
9.10. HISTORICAL PERSPECTIVE AND FURTHER READING 465

10. ENVIRONMENTAL ROBUSTNESS 477

10.1. THE ACOUSTICAL ENVIRONMENT 478
10.1.1. Additive Noise 478
10.1.2. Reverberation 480
10.1.3. A Model of the Environment 482
10.2. ACOUSTICAL TRANSDUCERS 486
10.2.1. The Condenser Microphone 486
10.2.2. Directionality Patterns 489
10.2.3. Other Transduction Categories 496
10.3. ADAPTIVE ECHO CANCELLATION (AEC) 497
10.3.1. The LMS Algorithm 499
10.3.2. Convergence Properties of the LMS Algorithm 500
10.3.3. Normalized LMS Algorithm 501
10.3.4. Transform-Domain LMS Algorithm 502
10.3.5. The RLS Algorithm 503
10.4. MULTIMICROPHONE SPEECH ENHANCEMENT 504
10.4.1. Microphone Arrays 505
10.4.2. Blind Source Separation 510
10.5. ENVIRONMENT COMPENSATION PREPROCESSING 515
10.5.1. Spectral Subtraction 516
10.5.2. Frequency-Domain MMSE from Stereo Data 519
10.5.3. Wiener Filtering 520
10.5.4. Cepstral Mean Normalization (CMN) 522
10.5.5. Real-Time Cepstral Normalization 525
10.5.6. The Use of Gaussian Mixture Models 525

10.6. ENVIRONMENTAL MODEL ADAPTATION 528
10.6.1. *Retraining on Corrupted Speech* 528
10.6.2. *Model Adaptation* 530
10.6.3. *Parallel Model Combination* 531
10.6.4. *Vector Taylor Series* 535
10.6.5. *Retraining on Compensated Features* 537
10.7. MODELING NONSTATIONARY NOISE 538
10.8. HISTORICAL PERSPECTIVE AND FURTHER READING 540

11. LANGUAGE MODELING 545
11.1. FORMAL LANGUAGE THEORY 546
11.1.1. *Chomsky Hierarchy* 547
11.1.2. *Chart Parsing for Context-Free Grammars* 549
11.2. STOCHASTIC LANGUAGE MODELS 554
11.2.1. *Probabilistic Context-Free Grammars* 554
11.2.2. *N-gram Language Models* 558
11.3. COMPLEXITY MEASURE OF LANGUAGE MODELS 560
11.4. N-GRAM SMOOTHING 562
11.4.1. *Deleted Interpolation Smoothing* 564
11.4.2. *Backoff Smoothing* 565
11.4.3. *Class N-grams* 570
11.4.4. *Performance of N-gram Smoothing* 573
11.5. ADAPTIVE LANGUAGE MODELS 574
11.5.1. *Cache Language Models* 574
11.5.2. *Topic-Adaptive Models* 575
11.5.3. *Maximum Entropy Models* 576
11.6. PRACTICAL ISSUES 578
11.6.1. *Vocabulary Selection* 578
11.6.2. *N-gram Pruning* 580
11.6.3. *CFG vs. N-gram Models* 581
11.7. HISTORICAL PERSPECTIVE AND FURTHER READING 584

12. BASIC SEARCH ALGORITHMS 591
12.1. BASIC SEARCH ALGORITHMS 592
12.1.1. *General Graph Searching Procedures* 593
12.1.2. *Blind Graph Search Algorithms* 597
12.1.3. *Heuristic Graph Search* 601
12.2. SEARCH ALGORITHMS FOR SPEECH RECOGNITION 608
12.2.1. *Decoder Basics* 609
12.2.2. *Combining Acoustic and Language Models* 610
12.2.3. *Isolated Word Recognition* 610
12.2.4. *Continuous Speech Recognition* 611

12.3. LANGUAGE MODEL STATES 613
12.3.1. Search Space with FSM and CFG 613
12.3.2. Search Space with the Unigram 616
12.3.3. Search Space with Bigrams 617
12.3.4. Search Space with Trigrams 619
12.3.5. How to Handle Silences Between Words 621
12.4. TIME-SYNCHRONOUS VITERBI BEAM SEARCH 622
12.4.1. The Use of Beam 624
12.4.2. Viterbi Beam Search 625
12.5. STACK DECODING (A* SEARCH) 626
12.5.1. Admissible Heuristics for Remaining Path 630
12.5.2. When to Extend New Words 631
12.5.3. Fast Match 634
12.5.4. Stack Pruning 638
12.5.5. Multistack Search 639
12.6. HISTORICAL PERSPECTIVE AND FURTHER READING 640

13. LARGE-VOCABULARY SEARCH ALGORITHMS 645
13.1. EFFICIENT MANIPULATION OF A TREE LEXICON 646
13.1.1. Lexical Tree 646
13.1.2. Multiple Copies of Pronunciation Trees 648
13.1.3. Factored Language Probabilities 650
13.1.4. Optimization of Lexical Trees 653
13.1.5. Exploiting Subtree Polymorphism 656
13.1.6. Context-Dependent Units and Inter-Word Triphones 658
13.2. OTHER EFFICIENT SEARCH TECHNIQUES 659
13.2.1. Using Entire HMM as a State in Search 659
13.2.2. Different Layers of Beams 660
13.2.3. Fast Match 661
13.3. N-BEST AND MULTIPASS SEARCH STRATEGIES 663
13.3.1. N-best Lists and Word Lattices 664
13.3.2. The Exact N-best Algorithm 666
13.3.3. Word-Dependent N-best and Word-Lattice Algorithm 667
13.3.4. The Forward-Backward Search Algorithm 670
13.3.5. One-Pass vs. Multipass Search 673
13.4. SEARCH-ALGORITHM EVALUATION 674
13.5. CASE STUDY—MICROSOFT WHISPER 676
13.5.1. The CFG Search Architecture 676
13.5.2. The N-gram Search Architecture 677
13.6. HISTORICAL PERSPECTIVE AND FURTHER READING 681

PART IV: TEXT-TO-SPEECH SYSTEMS

14. TEXT AND PHONETIC ANALYSIS 689

14.1. MODULES AND DATA FLOW 690
14.1.1. Modules 692
14.1.2. Data Flows 694
14.1.3. Localization Issues 696
14.2. LEXICON 697
14.3. DOCUMENT STRUCTURE DETECTION 699
14.3.1. Chapter and Section Headers 700
14.3.2. Lists 701
14.3.3. Paragraphs 702
14.3.4. Sentences 702
14.3.5. Email 704
14.3.6. Web Pages 705
14.3.7. Dialog Turns and Speech Acts 705
14.4. TEXT NORMALIZATION 706
14.4.1. Abbreviations and Acronyms 709
14.4.2. Number Formats 712
14.4.3. Domain-Specific Tags 718
14.4.4. Miscellaneous Formats 719
14.5. LINGUISTIC ANALYSIS 720
14.6. HOMOGRAPH DISAMBIGUATION 724
14.7. MORPHOLOGICAL ANALYSIS 725
14.8. LETTER-TO-SOUND CONVERSION 728
14.9. EVALUATION 730
14.10. CASE STUDY: FESTIVAL 732
14.10.1. Lexicon 733
14.10.2. Text Analysis 733
14.10.3. Phonetic Analysis 735
14.11. HISTORICAL PERSPECTIVE AND FURTHER READING 735

15. PROSODY 739

15.1. THE ROLE OF UNDERSTANDING 740
15.2. PROSODY GENERATION SCHEMATIC 743
15.3. SPEAKING STYLE 744
15.3.1. Character 744
15.3.2. Emotion 744
15.4. SYMBOLIC PROSODY 744
15.4.1. Pauses 745
15.4.2. Prosodic Phrases 747
749

15.4.3. Accent 751
15.4.4. Tone 753
15.4.5. Tune 757
15.4.6. Prosodic Transcription Systems 759
15.5. DURATION ASSIGNMENT 761
15.5.1. Rule-Based Methods 762
15.5.2. CART-Based Durations 763
15.6. PITCH GENERATION 763
15.6.1. Attributes of Pitch Contours 764
15.6.2. Baseline F0 Contour Generation 768
15.6.3. Parametric F0 Generation 774
15.6.4. Corpus-Based F0 Generation 778
15.7. PROSODY MARKUP LANGUAGES 783
15.8. PROSODY EVALUATION 784
15.9. HISTORICAL PERSPECTIVE AND FURTHER READING 785

16. SPEECH SYNTHESIS 793

16.1. ATTRIBUTES OF SPEECH SYNTHESIS 794
16.2. FORMANT SPEECH SYNTHESIS 796
16.2.1. Waveform Generation from Formant Values 797
16.2.2. Formant Generation by Rule 800
16.2.3. Data-Driven Formant Generation 803
16.2.4. Articulatory Synthesis 803
16.3. CONCATENATIVE SPEECH SYNTHESIS 804
16.3.1. Choice of Unit 805
16.3.2. Optimal Unit String: The Decoding Process 810
16.3.3. Unit Inventory Design 817
16.4. PROSODIC MODIFICATION OF SPEECH 818
16.4.1. Synchronous Overlap and Add (SOLA) 818
16.4.2. Pitch Synchronous Overlap and Add (PSOLA) 820
16.4.3. Spectral Behavior of PSOLA 822
16.4.4. Synthesis Epoch Calculation 823
16.4.5. Pitch-Scale Modification Epoch Calculation 825
16.4.6. Time-Scale Modification Epoch Calculation 826
16.4.7. Pitch-Scale Time-Scale Epoch Calculation 827
16.4.8. Waveform Mapping 827
16.4.9. Epoch Detection 828
16.4.10. Problems with PSOLA 829
16.5. SOURCE-FILTER MODELS FOR PROSODY MODIFICATION 831
16.5.1. Prosody Modification of the LPC Residual 832
16.5.2. Mixed Excitation Models 832
16.5.3. Voice Effects 834

16.6. EVALUATION OF TTS SYSTEMS 834
16.6.1. *Intelligibility Tests* 837
16.6.2. *Overall Quality Tests* 840
16.6.3. *Preference Tests* 842
16.6.4. *Functional Tests* 842
16.6.5. *Automated Tests* 843
16.7. HISTORICAL PERSPECTIVE AND FURTHER READING 844

PART V: SPOKEN LANGUAGE SYSTEMS

17. SPOKEN LANGUAGE UNDERSTANDING 853

17.1. WRITTEN VS. SPOKEN LANGUAGES 855
17.1.1. *Style* 856
17.1.2. *Disfluency* 857
17.1.3. *Communicative Prosody* 858
17.2. DIALOG STRUCTURE 859
17.2.1. *Units of Dialog* 860
17.2.2. *Dialog (Speech) Acts* 861
17.2.3. *Dialog Control* 866
17.3. SEMANTIC REPRESENTATION 867
17.3.1. *Semantic Frames* 867
17.3.2. *Conceptual Graphs* 872
17.4. SENTENCE INTERPRETATION 873
17.4.1. *Robust Parsing* 874
17.4.2. *Statistical Pattern Matching* 878
17.5. DISCOURSE ANALYSIS 881
17.5.1. *Resolution of Relative Expression* 882
17.5.2. *Automatic Inference and Inconsistency Detection* 885
17.6. DIALOG MANAGEMENT 886
17.6.1. *Dialog Grammars* 887
17.6.2. *Plan-Based Systems* 888
17.6.3. *Dialog Behavior* 892
17.7. RESPONSE GENERATION AND RENDITION 894
17.7.1. *Response Content Generation* 895
17.7.2. *Concept-to-Speech Rendition* 899
17.7.3. *Other Renditions* 901
17.8. EVALUATION 901
17.8.1. *Evaluation in the ATIS Task* 901
17.8.2. *PARADISE Framework* 903
17.9. CASE STUDY—DR. WHO 906
17.9.1. *Semantic Representation* 906
17.9.2. *Semantic Parser (Sentence Interpretation)* 908

17.9.3. Discourse Analysis 909
17.9.4. Dialog Manager 910
17.10. HISTORICAL PERSPECTIVE AND FURTHER READING 913

18. APPLICATIONS AND USER INTERFACES 919

18.1. APPLICATION ARCHITECTURE 920
18.2. TYPICAL APPLICATIONS 921
18.2.1. Computer Command and Control 921
18.2.2. Telephony Applications 924
18.2.3. Dictation 926
18.2.4. Accessibility 929
18.2.5. Handheld Devices 930
18.2.6. Automobile Applications 930
18.2.7. Speaker Recognition 931
18.3. SPEECH INTERFACE DESIGN 931
18.3.1. General Principles 931
18.3.2. Handling Errors 937
18.3.3. Other Considerations 941
18.3.4. Dialog Flow 942
18.4. INTERNATIONALIZATION 943
18.5. CASE STUDY—MIPAD 945
18.5.1. Specifying the Application 946
18.5.2. Rapid Prototyping 948
18.5.3. Evaluation 949
18.5.4. Iterations 951
18.6. HISTORICAL PERSPECTIVE AND FURTHER READING 952

INDEX 957

Foreword

Recognition and understanding of spontaneous unrehearsed speech remains an elusive goal. To understand speech, a human considers not only the specific information conveyed to the ear, but also the context in which the information is being discussed. For this reason, people can understand spoken language even when the speech signal is corrupted by noise. However, understanding the context of speech is, in turn, based on a broad knowledge of the world. And this has been the source of the difficulty and over forty years of research.

It is difficult to develop computer programs that are sufficiently sophisticated to understand continuous speech by a random speaker. Only when programmers simplify the problem—by isolating words, limiting the vocabulary or number of speakers, or constraining the way in which sentences may be formed—is speech recognition by computer possible.

Since the early 1970s, researchers at AT&T, BBN, CMU, IBM, Lincoln Labs, MIT, and SRI have made major contributions in Spoken Language Understanding Research. In 1971, the Defense Advanced Research Projects Agency (DARPA) initiated an ambitious five-year, $15 million, multisite effort to develop speech understanding systems. The goals were to develop systems that would accept continuous speech from many speakers, with minimal speaker adaptation, and operate on a 1000-word vocabulary, artificial syntax, and a

constrained task domain. Two of the systems, Harpy and Hearsay-II, both developed at Carnegie Mellon University, achieved the original goals and in some instances surpassed them.

During the last three decades I have been at Carnegie Mellon, I have been very fortunate to be able to work with many brilliant students and researchers. Xuedong Huang, Alex Acero, and Hsiao-Wuen Hon were arguably among the outstanding researchers in the speech group at CMU. Since then, they have moved to Microsoft and have put together a world-class team at Microsoft Research. Over the years, they have contributed standards for building spoken language understanding systems with Microsoft's SAPI/SDK family of products and pushed the technologies forward with the rest of the community. Today, they continue to play a premier leadership role in both the research community and in industry.

This new book, *Spoken Language Processing*, represents a welcome addition to the technical literature on this increasingly important emerging area of Information Technology. As we move from desktop PCs to personal digital assistants (PDAs), wearable computers, and Internet cell phones, speech becomes a central, if not the only, means of communication between the human and machine! Huang, Acero, and Hon have undertaken a commendable task of creating a comprehensive reference that covers theoretical, algorithmic, and systems aspects of the spoken language tasks of recognition, synthesis, and understanding.

The task of spoken language communication requires a system to recognize, interpret, execute, and respond to a spoken query. This task is complicated by the fact that the speech signal is corrupted by many sources: noise in the background, characteristics of the microphone, vocal tract characteristics of the speakers, and differences in pronunciation. In addition, the system has to cope with non-grammaticality of spoken communication and ambiguity of language. An effective system must strive to utilize all the available sources of knowledge—acoustics, phonetics and phonology, lexical, syntactic, and semantic structure of language, and task-specific context-dependent information.

Speech is based on a sequence of discrete sound segments that are linked in time. These segments, called phonemes, are assumed to have unique articulatory and acoustic characteristics. While the human vocal apparatus can produce an almost infinite number of articulatory gestures, the number of phonemes is limited. English as spoken in the United States, for example, contains 16 vowel and 24 consonant sounds. Each phoneme has distinguishable acoustic characteristics and, in combination with other phonemes, forms larger units such as syllables and words. Knowledge about the acoustic differences among these sound units is essential to distinguish one word from another, say, *bit* from *pit*.

When speech sounds are connected to form larger linguistic units, the acoustic characteristics of a given phoneme will change as a function of its immediate phonetic environment because of the interaction among various anatomical structures (such as the tongue, lips, and vocal chords) and their different degrees of sluggishness. The result is an overlap of phonemic information in the acoustic signal from one segment to the other. For example, the same underlying phoneme *t* can have drastically different acoustic characteristics in different words, say, in *tea*, *tree*, *city*, *beaten*, and *steep*. This effect, known as coarticulation, can occur within a given word or across a word boundary. Thus, the word *this* will have very different acoustic properties in phrases such as *this car* and *this ship*.

This book is self-contained for those who wish to familiarize themselves with the current state of spoken language systems technology. However, a researcher or a professional in the field will benefit from a thorough grounding in a number of disciplines, including:

- *Signal processing*: Fourier Transforms, DFT, and FFT
- *Acoustics*: physics of sounds and speech, models of vocal tract
- *Pattern recognition*: clustering and pattern matching techniques
- *Artificial intelligence*: knowledge representation and search, natural language processing
- *Computer science*: hardware, parallel systems, algorithm optimization
- *Statistics*: probability theory, hidden Markov models, dynamic programming
- *Linguistics*: acoustic phonetics, lexical representation, syntax, and semantics

A newcomer to this field, easily overwhelmed by the vast number of different algorithms scattered across many conference proceedings, can find in this book a set of techniques that Huang, Acero, and Hon have found to work well in practice. This book is unique in that it includes both the theory and implementation details necessary to build spoken language systems. If you were able to assemble all the individual material that is covered in the book and put it on a shelf, it would be several times larger than this volume and yet you would be missing vital information. You would not have the material that is in this book that threads it all into one story, one context. If you need additional resources, the authors include extensive references to get that additional detail. *Spoken Language Processing* is very appealing both as a textbook and as a reference book for practicing engineers. Some readers familiar with a specific topic may decide to skip a few chapters; others may want to focus in other chapters. This is not a book that you will pick up and read once from cover to cover, but one you will keep near you for reference as long as you work in this field.

Raj Reddy
Dean, School of Computer Science
Carnegie Mellon University

Preface

Our primary motivation in writing this book is to share our working experience to bridge the gap between the knowledge of industry gurus and newcomers to the spoken language processing community. Many powerful techniques hide in conference proceedings and academic papers for years before becoming widely recognized by the research community or the industry. We spent many years pursuing spoken language technology research at Carnegie Mellon University before we started spoken language R&D at Microsoft. We fully understand that it is by no means a small undertaking to transfer a state-of-the-art spoken language research system into a commercially viable product that can truly help people improve their productivity. Our experience in both industry and academia is reflected in the context of this book, which presents a contemporary and comprehensive description of both theoretic and practical issues in spoken language processing. This book is intended for people of diverse academic and practical backgrounds. Speech scientists, computer scientists, linguists, engineers, physicists, and psychologists all have a unique perspective on spoken language processing. This book will be useful to all of these special interest groups.

Spoken language processing is a diverse subject that relies on knowledge of many levels, including acoustics, phonology, phonetics, linguistics, semantics, pragmatics, and discourse. The diverse nature of spoken language processing requires knowledge in computer science, electrical engineering, mathematics, syntax, and psychology. There are a number of excellent books on the subfields of spoken language processing, including speech recognition, text-to-speech conversion, and spoken language understanding, but there is no single book that covers both theoretical and practical aspects of these subfields and spoken language interface design. We devote many chapters systematically introducing fundamental

theories needed to understand how speech recognition, text-to-speech synthesis, and spoken language understanding work. Even more important is the fact that the book highlights what works well in practice, which is invaluable if you want to build a practical speech recognizer, a practical text-to-speech synthesizer, or a practical spoken language system. Using numerous real examples in developing Microsoft's spoken language systems, we concentrate on showing how the fundamental theories can be applied to solve real problems in spoken language processing.

We would like to thank many people who helped us during our spoken language processing R&D careers. We are particularly indebted to Professor Raj Reddy at the School of Computer Science, Carnegie Mellon University. Under his leadership, Carnegie Mellon University has become a center of research excellence on spoken language processing. Today's computer industry and academia benefit tremendously from his leadership and contributions.

Special thanks are due to Microsoft for its encouragement of spoken language R&D. The management team at Microsoft has been extremely generous to the speech technology group. We are particularly grateful to Bill Gates, Nathan Myhrvold, Rick Rashid, Dan Ling, and Jack Breese for the great environment they have created for us at Microsoft Research. We would also like to thank Bob Muglia and Kai-Fu Lee for their leadership role in Microsoft's speech product development.

Scott Meredith helped us write a number of chapters in this book and deserves to be a co-author. His insight and experience in text-to-speech synthesis enriched this book a great deal. We also owe gratitude to many colleagues we worked with in the speech technology group of Microsoft Research. In alphabetic order, Jim Adcock, Bruno Alabiso, Fil Alleva, Eric Bidstrup, Antonio Bigazzi, Ciprian Chelba, Li Deng, James Droppo, Doug Duchene, Joshua Goodman, Mei-Yuh Hwang, Larry Israel, Derek Jacoby, Li Jiang, Yun-Cheng Ju, David Larson, Kevin Larson, Jingsong Liu, Ricky Loynd, Milind Mahajan, Peter Mau, John Merrill, Yunus Mohammed, Salman Mughal, Mike Plumpe, Scott Quinn, Bill Rockenbeck, Mike Rozak, Kevin Schofield, Roxana Teodorescu, Gina Venolia, Kuansan Wang, Ye-Yi Wang, and Shenzhi Zhang.

In addition, we want to thank Les Atlas, Jeff Bilmes, Alan Black, David Caulton, Eric Chang, Phil Chou, Dinei Florencio, Allen Gersho, Francisco Gimenez-Galanes, Hynek Hermansky, Henrique Malvar, Julian Odell, Mari Ostendorf, Joseph Pentheroudakis, Tandy Trower, and Charles Wayne. They provided us with many wonderful comments to refine this book. Tim Moore, Russ Hall, and Jane Bonnell at Prentice Hall helped us finish this book in a finite amount of time.

Finally, writing this book was a marathon that could not have been finished without the support of our spouses, Yingzhi, Donna, and Phen, during the many evenings and weekends we spent on this project.

Xuedong Huang
Alex Acero
Hsiao-Wuen Hon
Redmond, WA

CHAPTER 1

Introduction

From human prehistory to the new media of the future, speech communication has been and will be the dominant mode of human social bonding and information exchange. The spoken word is now extended, through technological mediation such as telephony, movies, radio, television, and the Internet. This trend reflects the primacy of spoken communication in human psychology.

In addition to human-human interaction, this human preference for spoken language communication finds a reflection in human-machine interaction as well. Most computers currently utilize a *graphical user interface* (GUI), based on graphically represented interface objects and functions such as windows, icons, menus, and pointers. Most computer operating systems and applications also depend on a user's keyboard strokes and mouse clicks, with a display monitor for feedback. Today's computers lack the fundamental human abilities to speak, listen, understand, and learn. Speech, supported by other natural modalities, will be one of the primary means of interfacing with computers. And, even before speech-based interaction reaches full maturity, applications in home, mobile, and office segments are incorporating spoken language technology to change the way we live and work.

A spoken language system needs to have both speech recognition and speech synthesis capabilities. However, those two components by themselves are not sufficient to build a useful spoken language system. An understanding and dialog component is required to manage interactions with the user; and domain knowledge must be provided to guide the system's interpretation of speech and allow it to determine the appropriate action. For all these components, significant challenges exist, including robustness, flexibility, ease of integration, and engineering efficiency. The goal of building commercially viable spoken language systems has long attracted the attention of scientists and engineers all over the world. The purpose of this book is to share our working experience in developing advanced spoken language processing systems with both our colleagues and newcomers. We devote many chapters to systematically introducing fundamental theories and to highlighting what works well based on numerous lessons we learned in developing Microsoft's spoken language systems.

1.1. MOTIVATIONS

What motivates the integration of spoken language as the primary interface modality? We present a number of scenarios, roughly in order of expected degree of technical challenges and expected time to full deployment.

1.1.1. Spoken Language Interface

There are generally two categories of users who can benefit from adoption of speech as a control modality in parallel with others, such as the mouse, keyboard, touch-screen, and joystick. For novice users, functions that are conceptually simple should be directly accessible. For example, raising the voice output volume under software control on the desktop speakers, a conceptually simple operation, in some GUI systems of today requires opening one or more windows or menus, and manipulating sliders, check-boxes, or other graphical elements. This requires some knowledge of the system's interface conventions and structures. For the novice user, to be able to say *raise the volume* would be more direct and natural. For expert users, the GUI paradigm is sometimes perceived as an obstacle or nuisance and shortcuts are sought. Frequently these shortcuts allow the power user's hands to remain on the keyboard or mouse while mixing content creation with system commands. For example, an operator of a graphic design system for CAD/CAM might wish to specify a text formatting command while keeping the pointer device in position over a selected screen element.

Speech has the potential to accomplish these functions more powerfully than keyboard and mouse clicks. Speech becomes more powerful when supplemented by information streams encoding other dynamic aspects of user and system status, which can be resolved by the semantic component of a complete multimodal interface. We expect such multimodal interactions to proceed based on more complete user modeling, including speech, visual orientation, natural and device-based gestures, and facial expression, and these will be coordinated with detailed system profiles of typical user tasks and activity patterns.

In some situations you must rely on speech as an input or output medium. For example, with wearable computers, it may be impossible to incorporate a large keyboard. When driving, safety is compromised by any visual distraction, and hands are required for controlling the vehicle. The ultimate speech-only device, the telephone, is far more widespread than the PC. Certain manual tasks may also require full visual attention to the focus of the work. Finally, spoken language interfaces offer obvious benefits for individuals challenged with a variety of physical disabilities, such as loss of sight or limitations in physical motion and motor skills. Chapter 18 contains a detailed discussion on spoken language applications.

1.1.2. Speech-to-Speech Translation

Speech-to-speech translation has been depicted for decades in science fiction stories. Imagine questioning a Chinese-speaking conversational partner by speaking English into an unobtrusive device, and hearing real-time replies you can understand. This scenario, like the spoken language interface, requires both speech recognition and speech synthesis technology. In addition, sophisticated multilingual spoken language understanding is needed. This highlights the need for tightly coupled advances in speech recognition, synthesis, and understanding systems, a point emphasized throughout this book.

1.1.3. Knowledge Partners

The ability of computers to process spoken language as proficient as humans will be a landmark to signal the arrival of truly intelligent machines. Alan Turing [29] introduced his famous *Turing test*. He suggested a game, in which a computer's use of language would form the criterion for intelligence. If the machine could win the game, it would be judged intelligent. In Turing's game, you play the role of an interrogator. By asking a series of questions via a teletype, you must determine the identity of the other two participants: a machine and a person. The task of the machine is to fool you into believing it is a person by responding as a person to your questions. The task of the other person is to convince you the other participant is the machine. The critical issue for Turing was that using language as humans do is sufficient as an operational test for intelligence.

The ultimate use of spoken language is to pass the Turing test in allowing future extremely intelligent systems to interact with human beings as knowledge partners in all aspects of life. This has been a staple of science fiction, but its day will come. Such systems require reasoning capabilities and extensive world knowledge embedded in sophisticated search, communication, and inference tools that are beyond the scope of this book. We expect that spoken language technologies described in this book will form the essential enabling mechanism to pass the Turing test.

1.2. SPOKEN LANGUAGE SYSTEM ARCHITECTURE

Spoken language processing refers to technologies related to speech recognition, text-to-speech, and spoken language understanding. A spoken language system has at least one of the following three subsystems: a speech recognition system that converts speech into words, a text-to-speech system that conveys spoken information, and a spoken language understanding system that maps words into actions and that plans system-initiated actions.

There is considerable overlap in the fundamental technologies for these three subareas. Manually created rules have been developed for spoken language systems with limited success. But, in recent decades, data-driven statistical approaches have achieved encouraging results, which are usually based on modeling the speech signal using well-defined statistical algorithms that can automatically extract knowledge from the data. The data-driven approach can be viewed fundamentally as a pattern recognition problem. In fact, speech recognition, text-to-speech conversion, and spoken language understanding can all be regarded as pattern recognition problems. The patterns are either recognized during the runtime operation of the system or identified during system construction to form the basis of runtime generative models such as prosodic templates needed for text-to-speech synthesis. While we use and advocate the statistical approach, we by no means exclude the knowledge engineering approach from consideration. If we have a good set of rules in a given problem area, there is no need to use the statistical approach at all. The problem is that, at time of this writing, we do not have enough knowledge to produce a complete set of high-quality rules. As scientific and theoretical generalizations are made from data collected to construct data-driven systems, better rules may be constructed. Therefore, the rule-based and statistical approaches are best viewed as complementary.

1.2.1. Automatic Speech Recognition

A source-channel mathematical model described in Chapter 3 is often used to formulate speech recognition problems. As illustrated in Figure 1.1, the speaker's mind decides the source word sequence $\mathbf{W}$ that is delivered through his/her text generator. The source is passed through a noisy communication channel that consists of the speaker's vocal apparatus to produce the speech waveform and the speech signal processing component of the speech recognizer. Finally, the speech decoder aims to decode the acoustic signal $\mathbf{X}$ into a word sequence $\hat{\mathbf{W}}$, which is hopefully close to the original word sequence $\mathbf{W}$.

A typical practical speech recognition system consists of basic components shown in the dotted box of Figure 1.2. Applications interface with the decoder to get recognition results that may be used to adapt other components in the system. *Acoustic models* include the representation of knowledge about acoustics, phonetics, microphone and environment variability, gender and dialect differences among speakers, etc. *Language models* refer to a system's knowledge of what constitutes a possible word, what words are likely to co-occur, and in what sequence. The semantics and functions related to an operation a user may wish to perform may also be necessary for the language model. Many uncertainties exist in these areas, associated with speaker characteristics, speech style and rate, recognition of basic

speech segments, possible words, likely words, unknown words, grammatical variation, noise interference, nonnative accents, and confidence scoring of results. A successful speech recognition system must contend with all of these uncertainties. But that is only the beginning. The acoustic uncertainties of the different accents and speaking styles of individual speakers are compounded by the lexical and grammatical complexity and variations of spoken language, which are all represented in the language model.

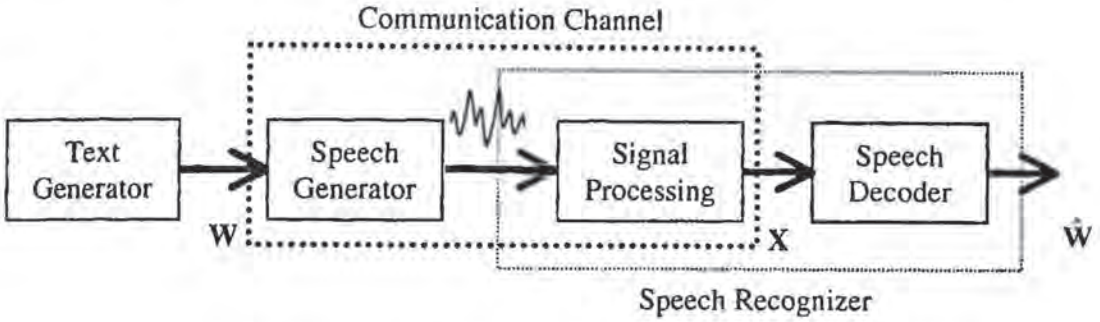


Figure 1.1 A source-channel model for a speech recognition system [15].

The speech signal is processed in the signal processing module that extracts salient feature vectors for the decoder. The decoder uses both acoustic and language models to generate the word sequence that has the maximum posterior probability for the input feature vectors. It can also provide information needed for the adaptation component to modify either the acoustic or language models so that improved performance can be obtained.

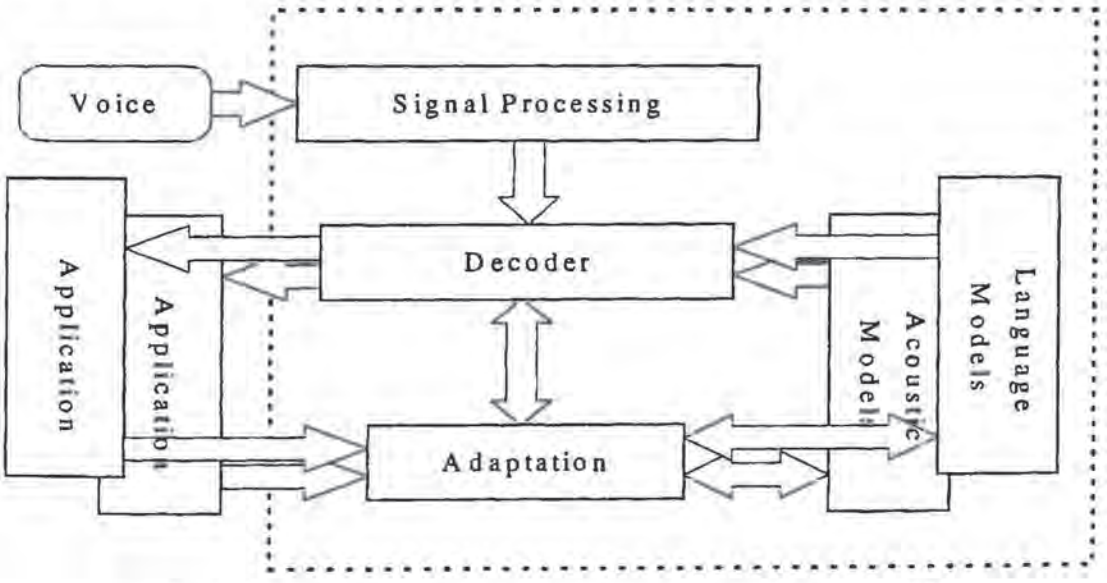


Figure 1.2 Basic system architecture of a speech recognition system [12].

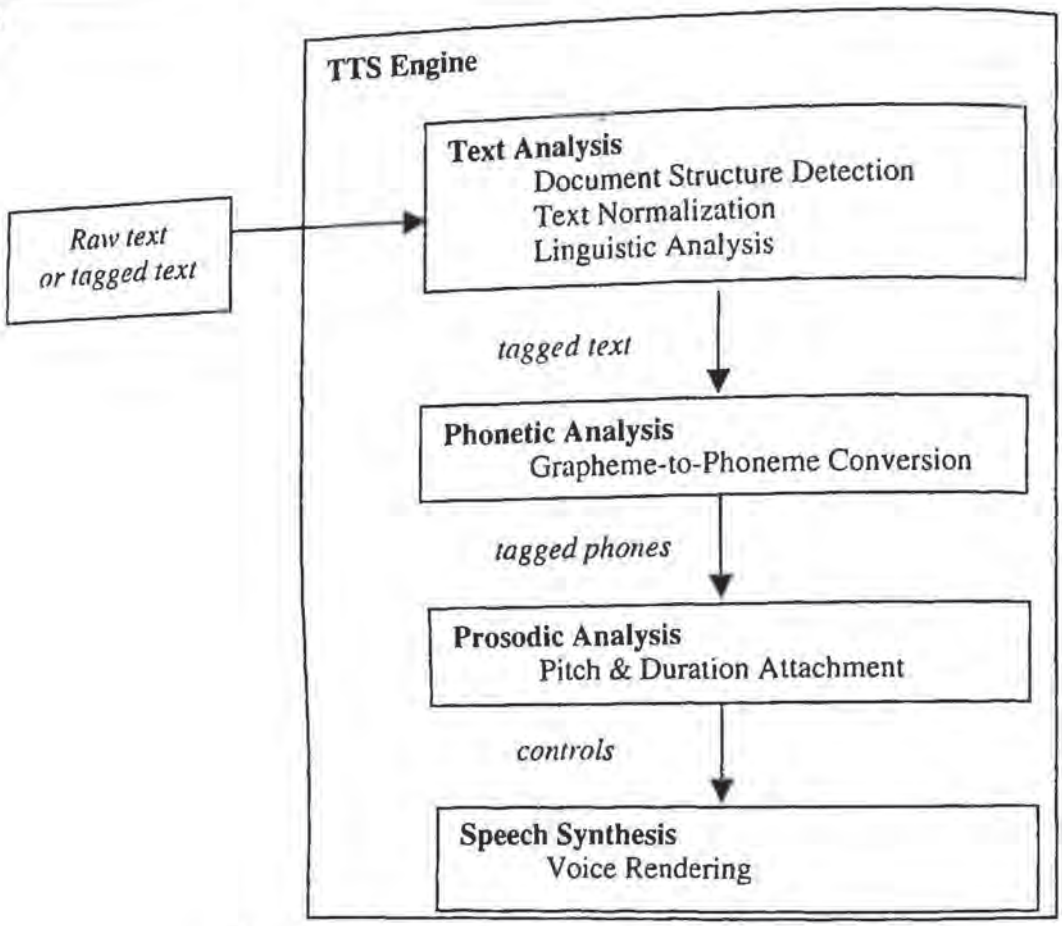


Figure 1.3 Basic system architecture of a TTS system.

1.2.2. Text-to-Speech Conversion

The term *text-to-speech*, often abbreviated as TTS, is easily understood. The task of a text-to-speech system can be viewed as speech recognition in reverse – a process of building a machinery system that can generate human-like speech from any text input to mimic human speakers. TTS is sometimes called *speech synthesis*, particularly in the engineering community.

The conversion of words in written form into speech is nontrivial. Even if we can store a huge dictionary for most common words in English; the TTS system still needs to deal with millions of names and acronyms. Moreover, in order to sound natural, the intonation of the sentences must be appropriately generated.

The development of TTS synthesis can be traced back to the 1930s when Dudley's *Voder*, developed by Bell Laboratories, was demonstrated at the World's Fair [18]. Taking advantage of increasing computation power and storage technology, TTS researchers have been able to generate high-quality commercial multilingual text-to-speech systems, although the quality is inferior to human speech for general-purpose applications.

The basic TTS components are shown in Figure 1.3. The text analysis component normalizes the text to the appropriate form so that it becomes speakable. The input can be

either raw text or tagged. These tags can be used to assist text, phonetic, and prosodic analysis. The phonetic analysis component converts the processed text into the corresponding phonetic sequence, which is followed by prosodic analysis to attach appropriate pitch and duration information to the phonetic sequence. Finally, the speech synthesis component takes the parameters from the fully tagged phonetic sequence to generate the corresponding speech waveform.

Various applications have different degrees of knowledge about the structure and content of the text that they wish to speak so some of the basic components shown in Figure 1.3 can be skipped. For example, some applications may have certain broad requirements such as rate and pitch. These requirements can be indicated with simple command tags appropriately located in the text. Many TTS systems provide a set of markups (tags), so the text producer can better express their semantic intention. An application may know a lot about the structure and content of the text to be spoken to greatly improve speech output quality. For engines providing such support, the *text analysis* phase can be skipped, in whole or in part. If the system developer knows the phonetic form, the phonetic analysis module can be skipped as well. The prosodic analysis module assigns a numeric duration to every phonetic symbol and calculates an appropriate pitch contour for the utterance or paragraph. In some cases, an application may have prosodic contours precalculated by some other process. This situation might arise when TTS is being used primarily for compression, or the prosody is *transplanted* from a real speaker's utterance. In these cases, the quantitative prosodic controls can be treated as special tagged field and sent directly along with the phonetic stream to speech synthesis for voice rendition.

1.2.3. Spoken Language Understanding

Whether a speaker is inquiring about flights to Seattle, reserving a table at a Pittsburgh restaurant, dictating an article in Chinese, or making a stock trade, a spoken language understanding system is needed to interpret utterances in context and carry out appropriate actions. Lexical, syntactic, and semantic knowledge must be applied in a manner that permits cooperative interaction among the various levels of acoustic, phonetic, linguistic, and application knowledge in minimizing uncertainty. Knowledge of the characteristic vocabulary, typical syntactic patterns, and possible actions in any given application context for both interpretation of user utterances and planning system activity are the heart and soul of any spoken language understanding system.

A schematic of a typical spoken language understanding system is shown in Figure 1.4. Such a system typically has a speech recognizer and a speech synthesizer for basic speech input and output, and a *sentence interpretation* component to parse the speech recognition results into semantic forms, which often need *discourse analysis* to track context and resolve ambiguities. The *Dialog Manager* is the central component that communicates with applications and the spoken language understanding modules such as discourse analysis, sentence interpretation, and response generation.

While most components of the system may be partly or wholly generic, the dialog manager controls the flow of conversation tied to the action. The dialog manager is respon-

sible for providing status needed for formulating responses, and maintaining the system's idea of the state of the discourse. The discourse state records the current transaction, dialog goals that motivated the current transaction, current objects in focus (temporary center of attention), the object history list for resolving dependent references, and other status information. The discourse information is crucial for sentence interpretation to interpret utterances in context. Various systems may alter the flow of information implied in Figure 1.4. For example, the dialog manager may be able to supply contextual discourse information or pragmatic inferences, as feedback to guide the recognizer's evaluation of hypotheses at the earliest level of search.

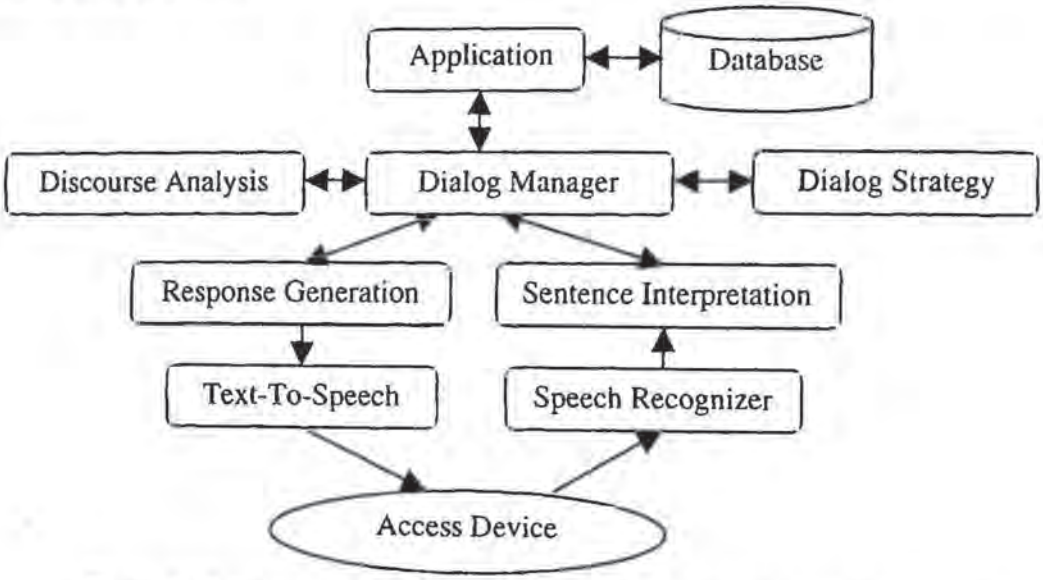


Figure 1.4 Basic system architecture of a spoken language understanding system.

1.3. BOOK ORGANIZATION

We attempt to present a comprehensive introduction to spoken language processing, which includes not only fundamentals but also a practical guide to build a working system that requires knowledge in speech signal processing, recognition, text-to-speech, spoken language understanding, and application integration. Since there is considerable overlap in the fundamental spoken language processing technologies, we have devoted Part I to the foundations needed. Part I contains background on speech production and perception, probability and information theory, and pattern recognition. Parts II, III, IV, and V include chapters on speech processing, speech recognition, speech synthesis, and spoken language systems, respectively. A reader with sufficient background can skip Part I, referring back to it later as needed. For example, the discussion of speech recognition in Part III relies on the pattern recognition algorithms presented in Part I. Algorithms that are used in several chapters

within Part III are also included in Parts I and II. Since the field is still evolving, at the end of each chapter we provide a historical perspective and list further readings to facilitate future research.

1.3.1. Part I: Fundamental Theory

Chapters 2 to 4 provide you with a basic theoretic foundation to better understand techniques that are widely used in modern spoken language systems. These theories include the essence of linguistics, phonetics, probability theory, information theory, and pattern recognition. These chapters prepare you fully to understand the rest of the book.

Chapter 2 discusses the basic structure of spoken language including speech science, phonetics, and linguistics. Chapter 3 covers probability theory and information theory, which form the foundation of modern pattern recognition. Many important algorithms and principles in pattern recognition and speech coding are derived based on these theories. Chapter 4 introduces basic pattern recognition, including decision theory, estimation theory, and a number of algorithms widely used in speech recognition. Pattern recognition forms the core of most of the algorithms used in spoken language processing.

1.3.2. Part II: Speech Processing

Part II provides you with necessary speech signal processing knowledge that is critical to spoken language processing. Most of what discuss here is traditionally the subject of electrical engineering.

Chapters 5 and 6 focus on how to extract useful information from the speech signal. The basic principles of digital signal processing are reviewed and a number of useful representations for the speech signal are discussed. Chapter 7 covers how to compress these representations for efficient transmission and storage.

1.3.3. Part III: Speech Recognition

Chapters 8 to 13 provide you with an in-depth look at modern speech recognition systems. We highlight techniques that have been proven to work well in real systems and explain in detail how and why these techniques work from both theoretic and practical perspectives.

Chapter 8 introduces hidden Markov models, the most prominent technique used in modern speech recognition systems. Chapters 9 and 11 deal with acoustic modeling and language modeling respectively. Because environment robustness is critical to the success of practical systems, we devote Chapter 10 to discussing how to make systems less affected by environment noises. Chapters 12 and 13 deal in detail with how to efficiently implement the decoder for speech recognition. Chapter 12 discusses a number of basic search algorithms, and Chapter 13 covers large vocabulary speech recognition. Throughout our discussion, Microsoft's Whisper speech recognizer is used as a case study to illustrate the methods introduced in these chapters.

1.3.4. Part IV: Text-to-Speech Systems

In Chapters 14 through 16, we discuss proven techniques in building text-to-speech systems. The synthesis system consists of major components found in speech recognition systems, except that they are in the reverse order.

Chapter 14 covers the analysis of written documents and the text needed to support spoken rendition, including the interpretation of audio markup commands, interpretation of numbers and other symbols, and conversion from orthographic to phonetic symbols. Chapter 15 focuses on the generation of pitch and duration controls for linguistic and emotional effect. Chapter 16 discusses the implementation of the synthetic voice, and presents algorithms to manipulate a limited voice data set to support a wide variety of pitch and duration controls required by the text analysis. We highlight the importance of trainable synthesis, with Microsoft's Whistler TTS system as an example.

1.3.5. Part V: Spoken Language Systems

As discussed in Section 1.1, spoken language applications motivate spoken language R&D. The central component is the spoken language understanding system. Since it is closely related to applications, we group it together with application and interface design.

Chapter 17 covers spoken language understanding. The output of the recognizer requires interpretation and action in a particular application context. This chapter details useful strategies for dialog management, and the coordination of all the speech and system resources to accomplish a task for a user. Chapter 18 concludes the book with a discussion of important principles for building spoken language interfaces and applications, including general human interface design goals, and interaction with other modalities in specific application contexts. Microsoft's MiPad is used as a case study to illustrate a number of issues in developing spoken language and multimodal applications.

1.4. TARGET AUDIENCES

This book can serve a variety of audiences:

Integration engineers: Software engineers who want to build spoken language systems, but who do not want to learn detailed speech technology internals, will find plentiful relevant material, including application design and software interfaces. Anyone with a professional interest in aspects of speech applications, integration, and interfaces can also achieve enough understanding of how the core technologies work, to allow them to take full advantage of state-of-the-art capabilities.

Speech technology engineers: Engineers and researchers working on various subspecialties within the speech field will find this book a useful guide to understanding related technologies in sufficient depth to help them gain insight on where their own approaches overlap with, or diverge from, their neighbors' common practice.

Graduate students: This book can serve as a primary textbook in a graduate or advanced undergraduate speech analysis or language engineering course. It can serve as a sup-

plementary textbook in some applied linguistics, digital signal processing, computer science, artificial intelligence, and possibly psycholinguistics course.

Linguists: As the practice of linguistics increasingly shifts to empirical analysis of real-world data, students and professional practitioners alike should find a comprehensive introduction to the technical foundations of computer processing of spoken language helpful. The book can be read at different levels and through different paths, for readers with differing technical skills and background knowledge.

Speech scientists: Researchers engaged in professional work on issues related to normal or pathological speech may find this complete exposition of the state-of-the-art in computer modeling of generation and perception of speech interesting.

Business planners: Increasingly, business and management functions require some level of insight into the vocabulary and common practices of technology development. While not the primary audience, managers, marketers, and others with planning responsibilities and sufficient technical background will find portions of this book useful in evaluating competing proposals, and in making business decisions related to the speech technology components.

1.5. HISTORICAL PERSPECTIVE AND FURTHER READING

Spoken language processing is a diverse field that relies on knowledge of language at the levels of signal processing, acoustics, phonology, phonetics, syntax, semantics, pragmatics, and discourse. The foundations of spoken language processing lie in computer science, electrical engineering, linguistics, and psychology. In the 1970s an ambitious speech understanding project was funded by DARPA, which led to many seminal systems and technologies [17]. A number of human language technology projects funded by DARPA in the 1980s and 1990s further accelerated the progress, as evidenced by many papers published in *The Proceedings of the DARPA Speech and Natural Language/Human Language Workshop*. The field is still rapidly progressing and there are a number of excellent review articles and introductory books. We provide a brief list here. More detailed references can be found within each chapter of this book. Gold and Morgan's *Speech and Audio Signal Processing* [10] also has a strong historical perspective on spoken language processing.

Hyde [14] and Reddy [24] provided an excellent review of early speech recognition work in the 1970s. Some of the principles are still applicable to today's speech recognition research. Waibel and Lee assembled many seminal papers in *Readings in Speech Recognition Speech Recognition* [31]. There are a number of excellent books on modern speech recognition [1, 13, 15, 22, 23].

Where does the state of the art speech recognition system stand today? A number of different recognition tasks can be used to compare the recognition error rate of people vs. machines. Table 1.1 shows five typical recognition tasks with vocabularies ranging from 10 to 5000 words speaker-independent continuous speech recognition. The Wall Street Journal Dictation (WSJ) Task has a 5000-word vocabulary as a continuous dictation application for the WSJ articles. In Table 1.1, the error rate for machines is based on state of the art speech

recognizers such as systems described in Chapter 9, and the error rate of humans is based on a range of subjects tested on the similar task. We can see the error rate of humans is at least 5 times smaller than machines except for the sentences that are generated from a trigram language model, where the sentences have the perfect match between humans and machines so humans cannot use high-level knowledge that is not used in machines.[1]

Table 1.1 Word error rate comparisons between human and machines on similar tasks.

Tasks	Vocabulary	Humans	Machines
Connected digits	10	0.009%	0.72%
Alphabet letters	26	1%	5%
Spontaneous telephone speech	2000	3.8%	36.7%
WSJ with clean speech	5000	0.9%	4.5%
WSJ with noisy speech (10-db SNR)	5000	1.1%	8.6%
Clean speech based on trigram sentences	20,000	7.6%	4.4%

We can see that humans are far more robust than machines for normal tasks. The error rate for machine spontaneous conversational telephone speech recognition is above 35%, more than a factor 10 higher than humans on the similar task. In addition, the error rate of humans does not increase as dramatically as machines when the environment becomes noisy (from quiet to 10-db SNR environments on the WSJ task). The relative error rate of humans increases from 0.9% to 1.1% (1.2 times), while the error rate of CSR systems increases from 4.5% to 8.6% (1.9 times). One interesting experiment is that when we generated sentences using the WSJ trigram language model (cf. Chapter 11), the difference between humans and machines disappears (the last row in Table 1.1). In fact, the error rate of humans is even higher than machines. This is because both humans and machines have the same high-level syntactic and semantic models. The test sentences are somewhat random to humans but perfect to machines that used the same trigram model for decoding. This experiment indicates humans make more effective use of semantic and syntactic constraints for improved speech recognition in meaningful conversation. In addition, machines don't have attention problems as humans do on random sentences.

Fant [7] gave an excellent introduction to speech production. Early reviews of text-to-speech synthesis can be found in [3, 8, 9]. Sagisaka [26] and Carlson [6] provide more recent reviews of progress in speech synthesis. A more detailed treatment can be found in [19, 30].

Where does the state of the art text to speech system stand today? Unfortunately, like speech recognition, this is not a solved problem either. Although machine storage capabilities are improving, the quality remains a challenge for many researchers if we want to pass the Turing test.

[1] Some of these experiments were conducted at Microsoft with only a small number of human subjects (3-5 people), which is not statistically significant. Nevertheless, the experiments give some interesting insight on the performance of humans and machines.

Spoken language understanding is deeply rooted in speech recognition research. There are a number of good books on spoken language understanding [2, 5, 16]. Manning and Schutze [20] focuses on statistical methods for language understanding. Like Waibel and Lee, Grosz et al. assembled many foundational papers in *Readings in Natural Language Processing* [11]. More recent reviews of progress in spoken language understanding can be found in [25, 28]. Related spoken language interface design issues can be found in [4, 21, 27, 32].

In comparison to speech recognition and text to speech, spoken language understanding is further away from approaching the level of humans, especially for general-purpose spoken language applications.

A number of good conference proceedings and journals report the latest progress in the field. Major results on spoken language processing are presented at the *International Conference on Acoustics, Speech and Signal Processing (ICASSP), International Conference on Spoken Language Processing (ICSLP), Eurospeech Conference,* the *DARPA Speech and Human Language Technology Workshops,* and many workshops organized by the *European Speech Communications Associations (ESCA)* and *IEEE Signal Processing Society.* Journals include *IEEE Transactions on Speech and Audio Processing, IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), Computer Speech and Language, Speech Communication,* and *Journal of Acoustical Society of America (JASA).* Research results can also be found at computational linguistics conferences such as the *Association for Computational Linguistics (ACL), International Conference on Computational Linguistics (COLING),* and *Applied Natural Language Processing (ANLP).* The journals *Computational Linguistics* and *Natural Language Engineering* cover both theoretical and practical applications of language research. *Speech Recognition Update* published by TMA Associates is an excellent industry newsletter on spoken language applications.

REFERENCES

[1] Acero, A., *Acoustical and Environmental Robustness in Automatic Speech Recognition*, 1993, Boston, MA, Kluwer Academic Publishers.

[2] Allen, J., *Natural Language Understanding*, 2nd ed., 1995, Menlo Park, CA, The Benjamin/Cummings Publishing Company.

[3] Allen, J., M.S. Hunnicutt, and D.H. Klatt, *From Text to Speech: The MITalk System*, 1987, Cambridge, UK, University Press.

[4] Balentine, B., and D. Morgan, *How to Build a Speech Recognition Application*, 1999, Enterprise Integration Group.

[5] Bernsen, N., H. Dybkjar, and L. Dybkjar, *Designing Interactive Speech Systems*, 1998, Springer.

[6] Carlson, R., "Models of Speech Synthesis" in *Voice Communications Between Humans and Machines. National Academy of Sciences*, D.B. Roe and J.G. Wilpon, eds., 1994, Washington, D.C., National Academy of Sciences.

[7] Fant, G., *Acoustic Theory of Speech Production*, 1970, The Hague, NL, Mouton.

[8] Flanagan, J., *Speech Analysis Synthesis and Perception*, 1972, New York, Springer-Verlag.

[9] Flanagan, J., "Voices Of Men And Machines," *Journal of Acoustical Society of America*, 1972, **51**, p. 1375.

[10] Gold, B. and N. Morgan, *Speech and Audio Signal Processing: Processing and Perception of Speech and Music*, 2000, John Wiley and Sons.

[11] Grosz, B., F.S. Jones, and B.L. Webber, *Readings in Natural Language Processing*, 1986, Los Altos, CA, Morgan Kaufmann.

[12] Huang, X., *et al.*, "From Sphinx-II to Whisper -- Make Speech Recognition Usable" in *Automatic Speech and Speaker Recognition*, C.H. Lee, F.K. Soong, and K.K. Paliwal, eds. 1996, Norwell, MA, Kluwer Academic Publishers.

[13] Huang, X.D., Y. Ariki, and M.A. Jack, *Hidden Markov Models for Speech Recognition*, 1990, Edinburgh, U.K., Edinburgh University Press.

[14] Hyde, S.R., "Automatic Speech Recognition: Literature, Survey, and Discussion" in *Human Communication, A Unified Approach*, E.E. David and P.B. Denes, eds. 1972, New York, McGraw Hill.

[15] Jelinek, F., *Statistical Methods for Speech Recognition*, Language, Speech, and Communication, 1998, Cambridge, MA, MIT Press.

[16] Jurafsky, D. and J. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, 2000, Upper Saddle River, NJ, Prentice Hall.

[17] Klatt, D., "Review of the ARPA Speech Understanding Project," *Journal of Acoustical Society of America*, 1977, **62**(6), pp. 1324-1366.

[18] Klatt, D., "Review of Text-to-Speech Conversion for English," *Journal of Acoustical Society of America*, 1987, **82**, pp. 737-793.

[19] Kleijn, W.B. and K.K. Paliwal, *Speech Coding and Synthesis*, 1995, Amsterdam, Netherlands, Elsevier.

[20] Manning, C. and H. Schutze, *Foundations of Statistical Natural Language Processing*, 1999, MIT Press, Cambridge, USA.

[21] Markowitz, J., *Using Speech Recognition*, 1996, Prentice Hall.

[22] Mori, R.D., *Spoken Dialogues with Computers*, 1998, London, UK, Academic Press.

[23] Rabiner, L.R. and B.H. Juang, *Fundamentals of Speech Recognition*, May, 1993, Prentice-Hall.

[24] Reddy, D.R., "Speech Recognition by Machine: A Review," *IEEE Proc.*, 1976, **64**(4), pp. 502-531.

[25] Sadek, D. and R.D. Mori, "Dialogue Systems" in *Spoken Dialogues with Computers*, R.D. Mori, Editor 1998, London, UK, pp. 523-561, Academic Press.

[26] Sagisaka, Y., "Speech Synthesis from Text," *IEEE Communication Magazine*, 1990(1).

[27] Schmandt, C., *Voice Communication with Computers*, 1994, New York, NY, Van Nostrand Reinhold.

[28] Seneff, S., "The Use of Linguistic Hierarchies in Speech Understanding," *Int. Conf. on Spoken Language Processing*, 1998, Sydney, Australia.

[29] Turing, A.M., "Computing Machinery and Intelligence," *Mind*, 1950, **LIX**(236), pp. 433-460.

[30] van Santen, J., *et al.*, *Progress in Speech Synthesis*, 1997, New York, Springer-Verlag.

[31] Waibel, A.H. and K.F. Lee, *Readings in Speech Recognition*, 1990, San Mateo, CA, Morgan Kaufman Publishers.

[32] Weinschenk, S. and D. Barker, *Designing Effective Speech Interfaces*, 2000, John Wiley & Sons, Inc.

PART I

FUNDAMENTAL THEORY

CHAPTER 2

Spoken Language Structure

Spoken language is used to communicate information from a speaker to a listener. Speech production and perception are both important components of the speech chain. Speech begins with a thought and intent to communicate in the brain, which activates muscular movements to produce speech sounds. A listener receives it in the auditory system, processing it for conversion to neurological signals the brain can understand. The speaker continuously monitors and controls the vocal organs by receiving his or her own speech as feedback.

Considering the universal components of speech communication as shown in Figure 2.1, the fabric of spoken interaction is woven from many distinct elements. The speech production process starts with the semantic message in a person's mind to be transmitted to the listener via speech. The computer counterpart to the process of message formulation is the application semantics that creates the concept to be expressed. After the message is created,

the next step is to convert the message into a sequence of words. Each word consists of a sequence of phonemes that corresponds to the pronunciation of the words. Each sentence also contains a prosodic pattern that denotes the duration of each phoneme, intonation of the sentence, and loudness of the sounds. Once the language system finishes the mapping, the talker executes a series of neuromuscular signals. The neuromuscular commands perform articulatory mapping to control the vocal cords, lips, jaw, tongue, and velum, thereby producing the sound sequence as the final output. The speech understanding process works in reverse order. First the signal is passed to the cochlea in the inner ear, which performs frequency analysis as a filter bank. A neural transduction process follows and converts the spectral signal into activity signals on the auditory nerve, corresponding roughly to a feature extraction component. Currently, it is unclear how neural activity is mapped into the language system and how message comprehension is achieved in the brain.

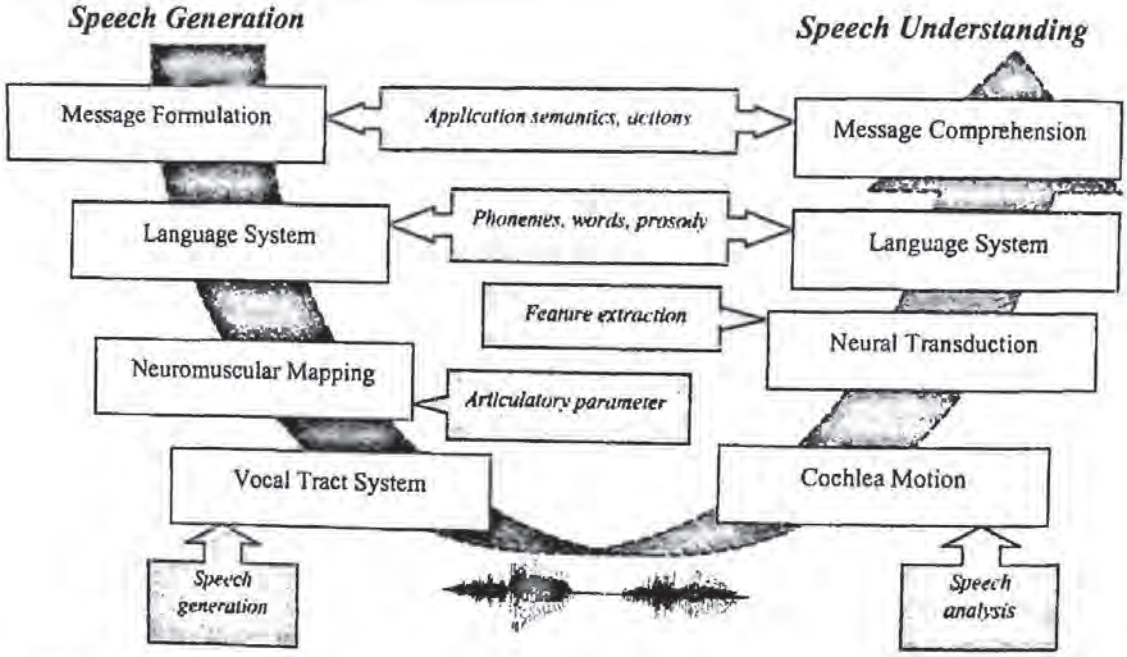


Figure 2.1 The underlying determinants of speech generation and understanding. The gray boxes indicate the corresponding computer system components for spoken language processing.

Speech signals are composed of analog sound patterns that serve as the basis for a discrete, symbolic representation of the spoken language – phonemes, syllables, and words. The production and interpretation of these sounds are governed by the syntax and semantics of the language spoken. In this chapter, we take a bottom up approach to introduce the basic concepts from sound to phonetics and phonology. Syllables and words are followed by syntax and semantics, which form the structure of spoken language processing. The examples in this book are drawn primarily from English, though they are relevant to other languages.

2.1. SOUND AND HUMAN SPEECH SYSTEMS

In this section, we briefly review human speech production and perception systems. We hope spoken language research will enable us to build a computer system that is as good as or better than our own speech production and understanding system.

2.1.1. Sound

Sound is a longitudinal pressure wave formed of compressions and rarefactions of air molecules, in a direction parallel to that of the application of energy. Compressions are zones where air molecules have been forced by the application of energy into a tighter-than-usual configuration, and rarefactions are zones where air molecules are less tightly packed. The alternating configurations of compression and rarefaction of air molecules along the path of an energy source are sometimes described by the graph of a sine wave as shown in Figure 2.2. In this representation, crests of the sine curve correspond to moments of maximal compression and troughs to moments of maximal rarefaction.

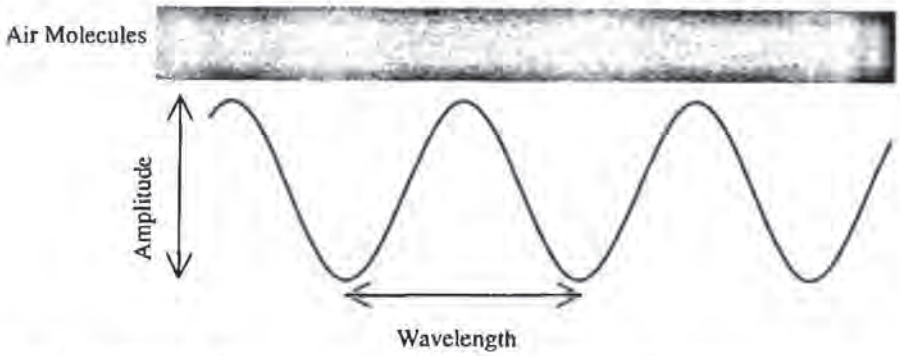


Figure 2.2 Application of sound energy causes alternating compression/rarefaction of air molecules, described by a sine wave. There are two important parameters, amplitude and wavelength, to describe a sine wave. Frequency [cycles/second measured in Hertz (Hz)] is also used to measure of the waveform.

The use of the sine graph in Figure 2.2 is only a notational convenience for charting local pressure variations over time, since sound does not form a transverse wave, and the air particles are just *oscillating in place* along the line of application of energy. The speed of a sound pressure wave in air is approximately $331.5 + 0.6T_c m/s$, where T_c is the Celsius temperature.

The amount of work done to generate the energy that sets the air molecules in motion is reflected in the amount of displacement of the molecules from their resting position. This *degree of displacement* is measured as the amplitude of a sound as shown in Figure 2.2. Because of the wide range, it is convenient to measure sound amplitude on a logarithmic scale in *decibels* (dB). A decibel scale is a means for comparing the intensity of two sounds:

$$10 \log_{10} (I / I_0) \tag{2.1}$$

where I and I_0 are the two intensity levels, with intensity being proportional to the square of the sound pressure P.

Sound pressure level (SPL) is a measure of absolute sound pressure P in dB:

$$SPL(dB) = 20 \log_{10} \left(\frac{P}{P_0} \right) \tag{2.2}$$

where the reference 0 dB corresponds to the threshold of hearing, which is $P_0 = 0.0002 \mu bar$ for a tone of 1kHz. The speech conversation level at 3 feet is about 60 dB SPL, and a jackhammer's level is about 120 dB SPL. Alternatively, watts/meter^2 units are often used to indicate intensity. We can bracket the limits of human hearing as shown in Table 2.1. On the low end, the human ear is quite sensitive. A typical person can detect sound waves having an intensity of 10^{-12} W/m^2 (the *threshold of hearing* or TOH). This intensity corresponds to a pressure wave affecting a given region by only one-billionth of a centimeter of molecular motion. On the other end, the most intense sound that can be safely detected without suffering physical damage is one trillion times more intense than the TOH. 0 dB begins with the TOH and advances logarithmically. The faintest audible sound is arbitrarily assigned a value of 0 dB, and the loudest sounds that the human ear can tolerate are about 120 dB.

Table 2.1 Intensity and decibel levels of various sounds.

Sound	dB Level	Times > TOH
Threshold of hearing (TOH: $10^{-12} W/m^2$)	0	10^0
Light whisper	10	10^1
Quiet living room	20	10^2
Quiet conversation	40	10^4
Average office	50	10^5
Normal conversation	60	10^6
Busy city street	70	10^7
Acoustic guitar – 1 ft. away	80	10^8
Heavy truck traffic	90	10^9
Subway from platform	100	10^{10}
Power tools	110	10^{11}
Pain threshold of ear	120	10^{12}
Airport runway	130	10^{13}
Sonic boom	140	10^{14}
Permanent damage to hearing	150	10^{15}
Jet engine, close up	160	10^{16}
Rocket engine	180	10^{18}
Twelve ft. from artillery cannon muzzle ($10^{10} W/m^2$)	220	10^{22}

The absolute threshold of hearing is the maximum amount of energy of a pure tone that cannot be detected by a listener in a noise free environment. The absolute threshold of hearing is a function of frequency that can be approximated by

$$T_q(f) = 3.64(f/1000)^{-0.8} - 6.5e^{-0.6(f/1000-3.3)^2} + 10^{-3}(f/1000)^4 \quad (dB\ SPL) \tag{2.3}$$

and is plotted in Figure 2.3.

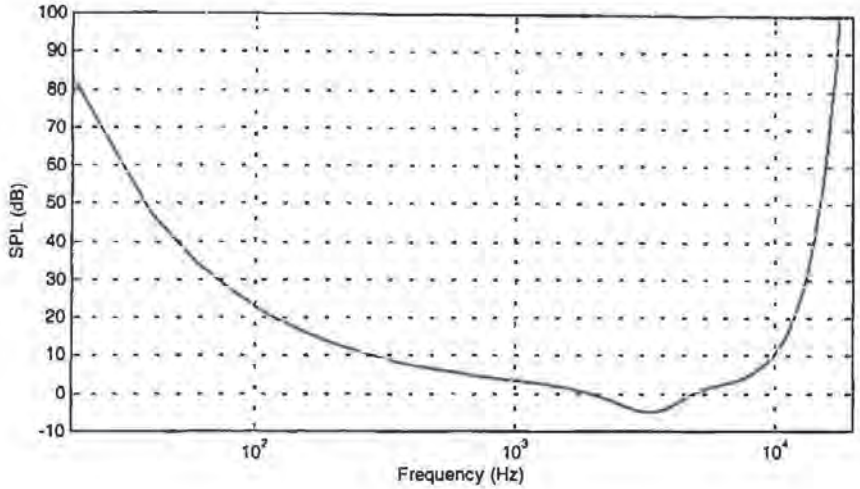


Figure 2.3 The sound pressure level (SPL) level in dB of the absolute threshold of hearing as a function of frequency. Sounds below this level are inaudible. Note that below 100 Hz and above 10 kHz this level rises very rapidly. Frequency goes from 20 Hz to 20 kHz and is plotted in a logarithmic scale from Eq. (2.3).

Let's compute how the pressure level varies with distance for a sound wave emitted by a point source located a distance r away. Assuming no energy absorption or reflection, the sound wave of a point source is propagated in a spherical front, such that the energy is the same for the sphere's surface at all radius r. Since the surface of a sphere of radius r is $4\pi r^2$, the sound's energy is inversely proportional to r^2, so that every time the distance is doubled, the sound pressure level decreases by 6 dB. For the point sound source, the energy (E) transported by a wave is proportional to the square of the amplitude (A) of the wave and the distance (r) between the sound source and the listener:

$$E \propto \frac{A^2}{r^2} \tag{2.4}$$

The typical sound intensity of a speech signal one inch away (close-talking microphone) from the talker is 1 Pascal = 10μbar, which corresponds to 94 dB SPL. The typical sound intensity 10 inches away from a talker is 0.1 Pascal = 1μbar, which corresponds to 74 dB SPL.

2.1.2. Speech Production

We review here basic human speech production systems, which have influenced research on speech coding, synthesis, and recognition.

2.1.2.1. Articulators

Speech is produced by air-pressure waves emanating from the mouth and the nostrils of a speaker. In most of the world's languages, the inventory of *phonemes,* as discussed in Section 2.2.1, can be split into two basic classes:

- consonants – articulated in the presence of constrictions in the throat or obstructions in the mouth (tongue, teeth, lips) as we speak.
- vowels – articulated without major constrictions and obstructions.

The sounds can be further partitioned into subgroups based on certain articulatory properties. These properties derive from the anatomy of a handful of important articulators and the places where they touch the boundaries of the human vocal tract. Additionally, a large number of muscles contribute to articulator positioning and motion. We restrict ourselves to a schematic view of only the major articulators, as diagrammed in Figure 2.4. The

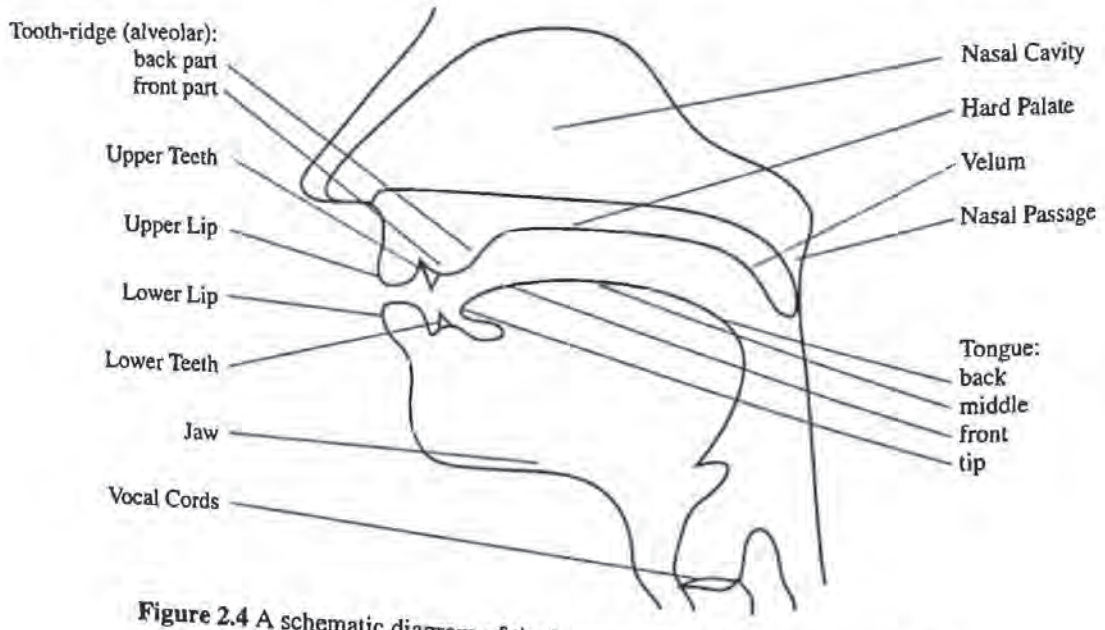


Figure 2.4 A schematic diagram of the human speech production apparatus.

gross components of the speech production apparatus are the lungs, trachea, larynx (organ of voice production), pharyngeal cavity (throat), oral and nasal cavity. The pharyngeal and oral cavities are typically referred to as the vocal tract, and the nasal cavity as the nasal tract. As illustrated in Figure 2.4, the human speech production apparatus consists of:

- *Lungs*: source of air during speech.
- *Vocal cords (larynx):* when the vocal folds are held close together and oscillate against one another during a speech sound, the sound is said to be *voiced.* When the folds are too slack or tense to vibrate periodically, the sound is said to be *unvoiced.* The place where the vocal folds come together is called the *glottis.*
- *Velum (soft palate)*: operates as a *valve*, opening to allow passage of air (and thus resonance) through the nasal cavity. Sounds produced with the flap open include *m* and *n*.
- *Hard palate*: a long relatively hard surface at the roof inside the mouth, which, when the tongue is placed against it, enables consonant articulation.
- *Tongue*: flexible articulator, shaped away from the palate for vowels, placed close to or on the palate or other hard surfaces for consonant articulation.
- *Teeth*: another place of articulation used to brace the tongue for certain consonants.
- *Lips*: can be rounded or spread to affect vowel quality, and closed completely to stop the oral air flow in certain consonants (*p, b, m*).

2.1.2.2. The Voicing Mechanism

The most fundamental distinction between sound types in speech is the voiced/voiceless distinction. Voiced sounds, including vowels, have in their time and frequency structure a roughly regular pattern that voiceless sounds, such as consonants like *s*, lack. Voiced sounds typically have more energy as shown in Figure 2.5. We see here the waveform of the word *sees*, which consists of three phonemes: an unvoiced consonant */s/*, a vowel */iy/*, and a voiced consonant */z/*.

What in the speech production mechanism creates this fundamental distinction? When the vocal folds vibrate during phoneme articulation, the phoneme is considered voiced; otherwise it is unvoiced. Vowels are voiced throughout their duration. The distinct vowel *timbres* are created by using the tongue and lips to shape the main oral resonance cavity in different ways. The vocal folds vibrate at slower or faster rates, from as low as 60 cycles per second (Hz) for a large man, to as high as 300 Hz or higher for a small woman or child. The rate of cycling (opening and closing) of the vocal folds in the larynx during phonation of voiced sounds is called the *fundamental frequency*. This is because it sets the periodic baseline for all higher-frequency harmonics contributed by the pharyngeal and oral resonance

cavities above. The fundamental frequency also contributes more than any other single factor to the perception of *pitch* (the semi-musical rising and falling of voice tones) in speech.

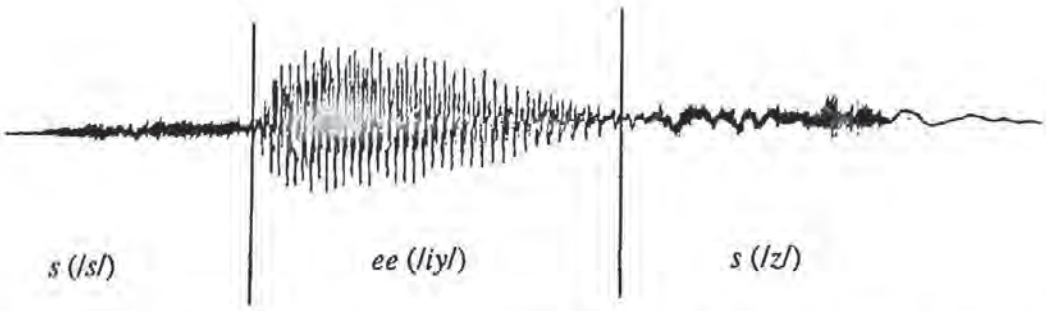


Figure 2.5 Waveform of *sees*, showing a voiceless phoneme /s/, followed by a voiced sound, the vowel /iy/. The final sound, /z/, is a type of voiced consonant.

The glottal cycle is illustrated in Figure 2.6. At stage (a), the vocal folds are closed and the air stream from the lungs is indicated by the arrow. At some point, the air pressure on the underside of the barrier formed by the vocal folds increases until it overcomes the resistance of the vocal fold closure and the higher air pressure below blows them apart (b). However, the tissues and muscles of the larynx and the vocal folds have a natural elasticity which tends to make them fall back into place rapidly, once air pressure is temporarily equalized (c). The successive airbursts resulting from this process are the source of energy for all voiced sounds. The time for a single open-close cycle depends on the stiffness and size of the vocal folds and the amount of subglottal air pressure. These factors can be controlled by a speaker to raise and lower the perceived frequency or pitch of a voiced sound.

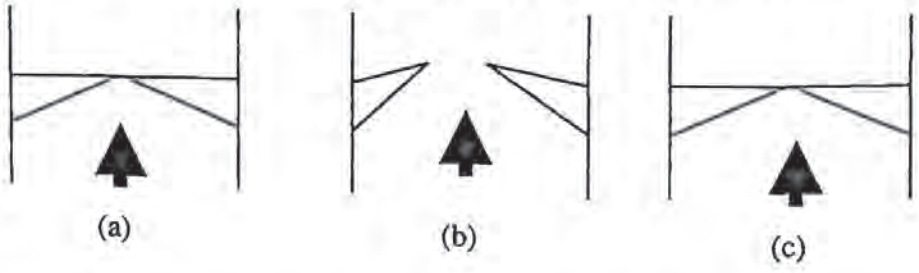


Figure 2.6 Vocal fold cycling at the larynx. (a) Closed with sub-glottal pressure buildup; (b) trans-glottal pressure differential causing folds to blow apart; (c) pressure equalization and tissue elasticity forcing temporary reclosure of vocal folds, ready to begin next cycle.

The waveform of air pressure variations created by this process can be described as a periodic flow, in cubic centimeters per second (after [15]). As shown in Figure 2.7, during the time bracketed as *one cycle*, there is no air flow during the initial closed portion. Then as

the glottis opens (open phase), the volume of air flow becomes greater. After a short peak, the folds begin to resume their original position and the air flow declines until complete closure is attained, beginning the next cycle. A common measure is the number of such cycles per second (Hz), or the fundamental frequency (*F0*). Thus the fundamental frequency for the waveform in Figure 2.7 is about 120 Hz.

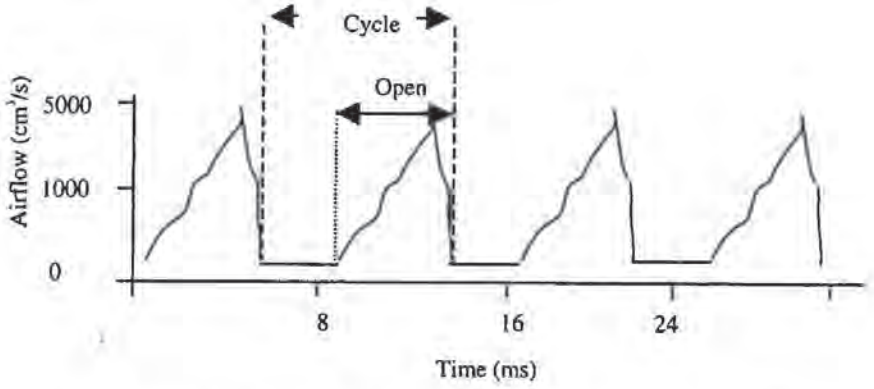


Figure 2.7 Waveform showing air flow during laryngeal cycle.

2.1.2.3. Spectrograms and Formants

Since the glottal wave is periodic, consisting of fundamental frequency (*F0*) and a number of harmonics (integral multiples of *F0*), it can be analyzed as a sum of sine waves as discussed in Chapter 5. The resonances of the vocal tract (above the glottis) are excited by the glottal energy. Suppose, for simplicity, we regard the vocal tract as a straight tube of uniform cross-sectional area, closed at the glottal end, open at the lips. When the shape of the vocal tract changes, the resonances change also. Harmonics near the resonances are emphasized, and, in speech, the resonances of the cavities that are typical of particular articulator configurations (e.g., the different vowel timbres) are called *formants*. The vowels in an actual speech waveform can be viewed from a number of different perspectives, emphasizing either a *cross-sectional* view of the harmonic responses at a single moment, or a longer-term view of the formant track evolution over time. The actual spectral analysis of a vowel at a single time-point, as shown in Figure 2.8, gives an idea of the uneven distribution of energy in resonances for the vowel */iy/* in the waveform for *see*, which is shown in Figure 2.5.

Another view of *sees* of Figure 2.5, called a spectrogram, is displayed in the lower part of Figure 2.9. It shows a long-term frequency analysis, comparable to a complete series of single time-point *cross sections* (such as that in Figure 2.8) ranged alongside one another in time and viewed from *above*.

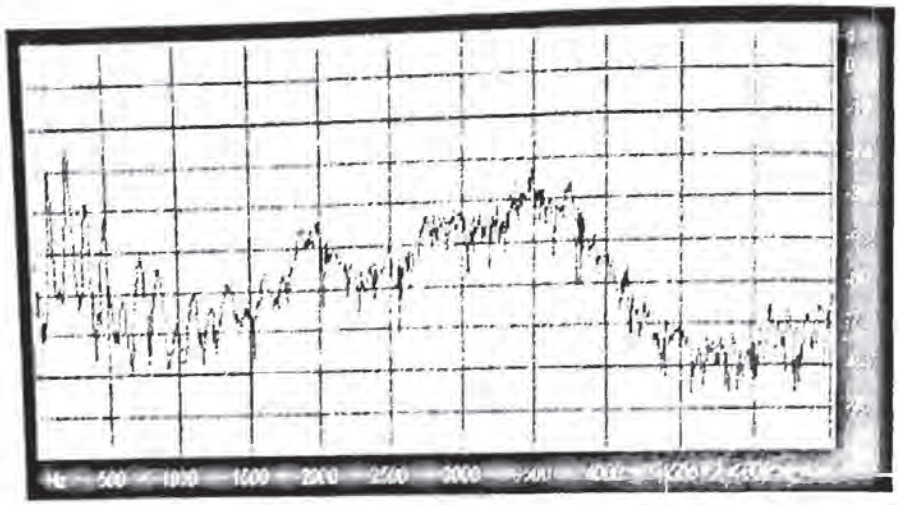


Figure 2.8 A spectral analysis of the vowel */iy/*, showing characteristically uneven distribution of energy at different frequencies.

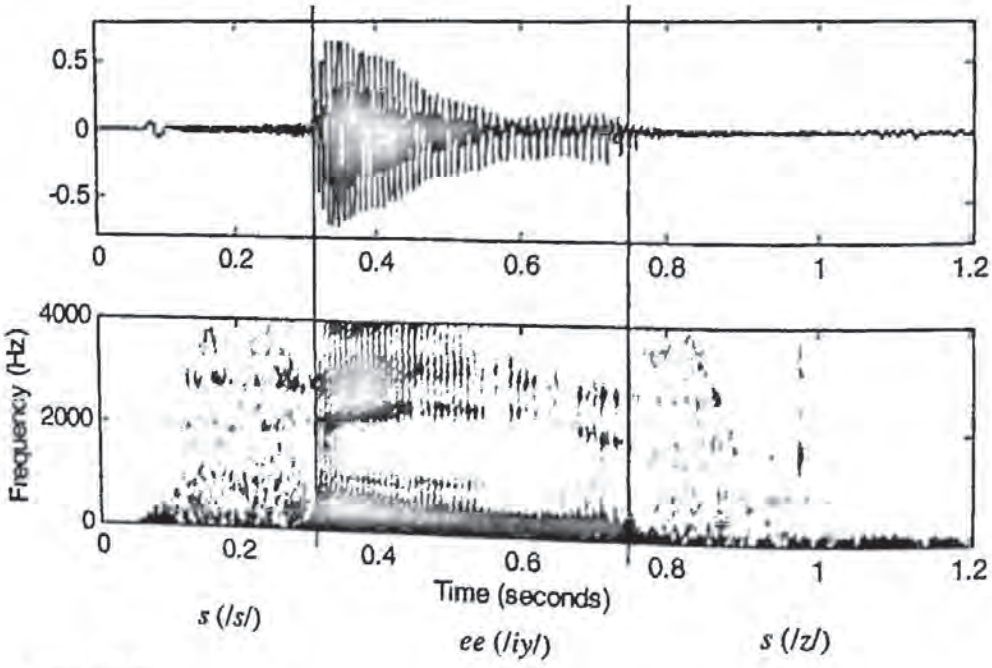


Figure 2.9 The spectrogram representation of the speech waveform *sees* (approximate phone boundaries are indicated with heavy vertical lines).

In the spectrogram of Figure 2.9, the darkness or lightness of a band indicates the relative amplitude or energy present at a given frequency. The dark horizontal bands show the formants, which are harmonics of the fundamental at natural resonances of the vocal tract cavity position for the vowel */iy/* in *see*. The mathematical methods for deriving analyses and representations such as those illustrated above are covered in Chapters 5 and 6.

2.1.3. Speech Perception

There are two major components in the auditory perception system: the peripheral auditory organs (ears) and the auditory nervous system (brain). The ear processes an acoustic pressure signal by first transforming it into a mechanical vibration pattern on the basilar membrane, and then representing the pattern by a series of pulses to be transmitted by the auditory nerve. Perceptual information is extracted at various stages of the auditory nervous system. In this section we focus mainly on the auditory organs.

2.1.3.1. Physiology of the Ear

The human ear, as shown in Figure 2.10, has three sections: the outer ear, the middle ear, and the inner ear. The outer ear consists of the external visible part and the external auditory canal that forms a tube along which sound travels. This tube is about 2.5 cm long and is covered by the eardrum at the far end. When air pressure variations reach the eardrum from the outside, it vibrates, and transmits the vibrations to bones adjacent to its opposite side. The vibration of the eardrum is at the same frequency (alternating compression and rarefaction) as the incoming sound pressure wave. The middle ear is an air-filled space or cavity about 1.3 cm across, and about 6 cm^3 volume. The air travels to the middle ear cavity along the tube (when opened) that connects the cavity with the nose and throat. The oval window shown in Figure 2.10 is a small membrane at the bony interface to the inner ear (cochlea). Since the cochlear walls are bony, the energy is transferred by mechanical action of the stapes into an impression on the membrane stretching over the oval window.

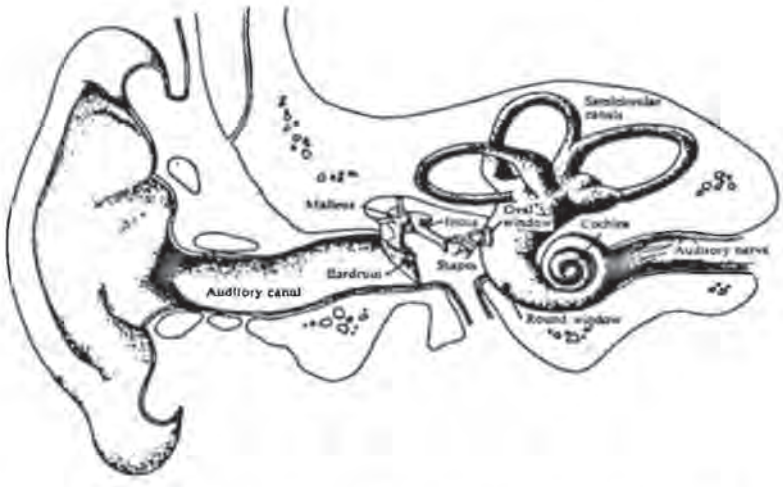


Figure 2.10 The structure of the peripheral auditory system with the outer, middle, and inner ear.

The relevant structure of the inner ear for sound perception is the cochlea, which communicates directly with the auditory nerve, conducting a representation of sound to the brain. The cochlea is a spiral tube about 3.5 cm long, which coils about 2.6 times. The spiral is divided, primarily by the basilar membrane running lengthwise, into two fluid-filled chambers. The cochlea can be roughly regarded as a filter bank, whose outputs are ordered by location, so that a frequency-to-place transformation is accomplished. The filters closest to the cochlear base respond to the higher frequencies, and those closest to its apex respond to the lower.

2.1.3.2. Physical vs. Perceptual Attributes

In psychoacoustics, a basic distinction is made between the perceptual attributes of a sound, especially a speech sound, and the measurable physical properties that characterize it. Each of the perceptual attributes, as listed in Table 2.2, seems to have a strong correlation with one main physical property, but the connection is complex, because other physical properties of the sound may affect perception in complex ways.

Table 2.2 Relation between perceptual and physical attributes of sound.

Physical Quantity	Perceptual Quality
Intensity	Loudness
Fundamental frequency	Pitch
Spectral shape	Timbre
Onset/offset time	Timing
Phase difference in binaural hearing	Location

Although sounds with a greater intensity level usually sound louder, the sensitivity of the ear varies with the frequency and the quality of the sound. One fundamental divergence between physical and perceptual qualities is the phenomenon of non-uniform *equal loudness* perception of tones of varying frequencies. In general, tones of differing pitch have different inherent *perceived loudness*. The sensitivity of the ear varies with the frequency and the quality of the sound. The graph of equal loudness contours adopted by ISO is shown in Figure 2.11. These curves demonstrate the relative insensitivity of the ear to sounds of low frequency at moderate to low intensity levels. Hearing sensitivity reaches a maximum around 4000 Hz, which is near the first resonance frequency of the outer ear canal, and peaks again around 13 kHz, the frequency of the second resonance [38].

Pitch is indeed most closely related to the fundamental frequency. The higher the fundamental frequency, the higher the pitch we perceive. However, discrimination between two pitches depends on the frequency of the lower pitch. Perceived pitch will change as intensity is increased and frequency is kept constant.

In another example of the non-identity of acoustic and perceptual effects, it has been observed experimentally that when the ear is exposed to two or more different tones, it is a common experience that one tone may *mask* the others. Masking is probably best explained

as an upward shift in the hearing threshold of the weaker tone by the louder tone. Pure tones, complex sounds, narrow and broad bands of noise all show differences in their ability to mask other sounds. In general, pure tones close together in frequency mask each other more than tones widely separated in frequency. A pure tone masks tones of higher frequency more effectively than tones of lower frequency. The greater the intensity of the masking tone, the broader the range of the frequencies it can mask [18, 31].

Binaural listening greatly enhances our ability to sense the direction of the sound source. The sense of localization attention is mostly focused on side-to-side discrimination or *lateralization*. Time and intensity cues have different impacts for low frequency and high frequency, respectively. Low-frequency sounds are lateralized mainly on the basis of interaural time difference, whereas high-frequency sounds are localized mainly on the basis of interaural intensity differences [5].

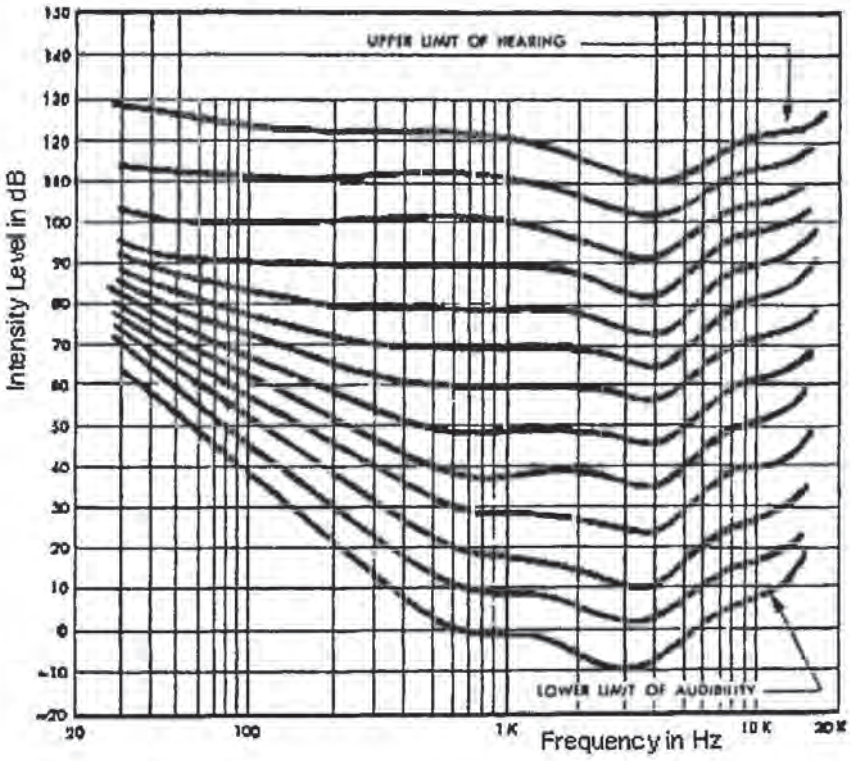


Figure 2.11 Equal-loudness curves indicate that the response of the human hearing mechanism is a function of frequency and loudness levels. This relationship again illustrates the difference between physical dimensions and psychological experience (after ISO 226).

Finally, an interesting perceptual issue is the question of distinctive voice quality. Speech from different people sounds different. Partially this is due to obvious factors, such as differences in characteristic fundamental frequency caused by, for example, the greater mass and length of adult male vocal folds as opposed to female. But there are more subtle

effects as well. In psychoacoustics, the concept of *timbre* (of a sound or instrument) is defined as that attribute of auditory sensation by which a subject can judge that two sounds similarly presented and having the same loudness and pitch are dissimilar. In other words, when all the easily measured differences are controlled, the remaining perception of difference is ascribed to timbre. This is heard most easily in music, where the same note in the same octave played for the same duration on a violin sounds different from a flute. The timbre of a sound depends on many physical variables including a sound's spectral power distribution, its temporal envelope, rate and depth of amplitude or frequency modulation, and the degree of inharmonicity of its harmonics.

2.1.3.3. Frequency Analysis

Researchers have undertaken psychoacoustic experimental work to derive frequency scales that attempt to model the natural response of the human perceptual system, since the cochlea of the inner ear acts as a spectrum analyzer. The complex mechanism of the inner ear and auditory nerve implies that the perceptual attributes of sounds at different frequencies may not be entirely simple or linear in nature. It is well known that the western musical pitch is described in *octaves*[1] and *semi-tones*.[2] The perceived musical pitch of complex tones is basically proportional to the logarithm of frequency. For complex tones, the just noticeable difference for frequency is essentially constant on the octave/semi-tone scale. Musical pitch scales are used in prosodic research (on speech intonation contour generation).

AT&T Bell Labs has contributed many influential discoveries in hearing, such as critical band and articulation index, since the turn of the 20th century [3]. Fletcher's work [14] pointed to the existence of critical bands in the cochlear response. Critical bands are of great importance in understanding many auditory phenomena such as perception of loudness, pitch, and timbre. The auditory system performs frequency analysis of sounds into their component frequencies. The cochlea acts as if it were made up of overlapping filters having bandwidths equal to the critical bandwidth. One class of critical band scales is called *Bark frequency scale*. It is hoped that by treating spectral energy over the Bark scale, a more natural fit with spectral information processing in the ear can be achieved. The Bark scale ranges from 1 to 24 Barks, corresponding to 24 critical bands of hearing as shown in Table 2.3. As shown in Figure 2.12, the perceptual resolution is finer in the lower frequencies. It should be noted that the ear's critical bands are continuous, and a tone of any audible frequency always finds a critical band centered on it. The Bark frequency b can be expressed in terms of the linear frequency (in Hz) by

$$b(f) = 13\arctan(0.00076f) + 3.5 * \arctan\left((f/7500)^2\right) \quad (Bark) \tag{2.5}$$

[1] A tone of frequency f_1 is said to be an octave above a tone with frequency f_2 if and only if $f_1 = 2f_2$.
[2] There are 12 semitones in one octave, so a tone of frequency f_1 is said to be a semitone above a tone with frequency f_2 if and only if $f_1 = 2^{1/12} f_2 = 1.05946 f_2$.

Table 2.3 The Bark frequency scale.

Bark Band #	Edge (Hz)	Center (Hz)
1	100	50
2	200	150
3	300	250
4	400	350
5	510	450
6	630	570
7	770	700
8	920	840
9	1080	1000
10	1270	1170
11	1480	1370
12	1720	1600
13	2000	1850
14	2320	2150
15	2700	2500
16	3150	2900
17	3700	3400
18	4400	4000
19	5300	4800
20	6400	5800
21	7700	7000
22	9500	8500
23	12000	10500
24	15500	13500

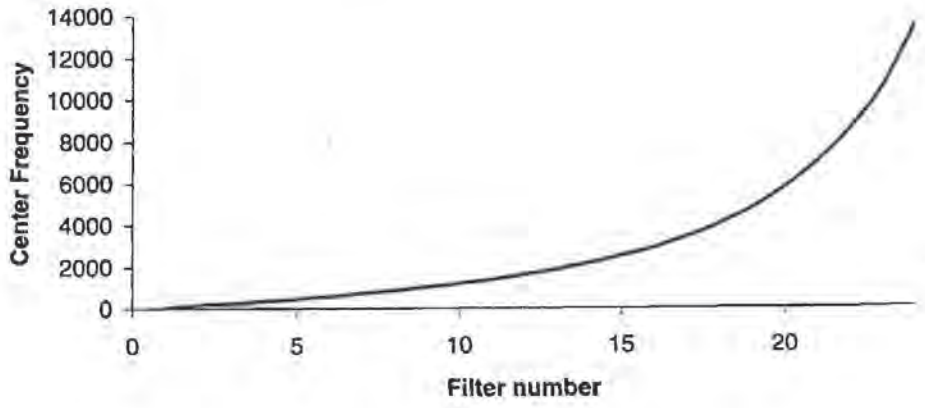


Figure 2.12 The center frequency of 24 Bark frequency filters as illustrated in Table 2.3.

Another such perceptually motivated scale is the mel frequency scale [41], which is linear below 1 kHz, and logarithmic above, with equal numbers of samples taken below and above 1 kHz. The mel scale is based on experiments with simple tones (sinusoids) in which subjects were required to divide given frequency ranges into four perceptually equal intervals or to adjust the frequency of a stimulus tone to be half as high as that of a comparison tone. One mel is defined as one thousandth of the pitch of a 1 kHz tone. As with all such attempts, it is hoped that the mel scale more closely models the sensitivity of the human ear than a purely linear scale and provides for greater discriminatory capability between speech segments. Mel-scale frequency analysis has been widely used in modern speech recognition systems. It can be approximated by:

$$B(f) = 1125 \ln(1 + f/700) \tag{2.6}$$

The mel scale is plotted in Figure 2.13 together with the Bark scale and the bilinear transform (see Chapter 6).

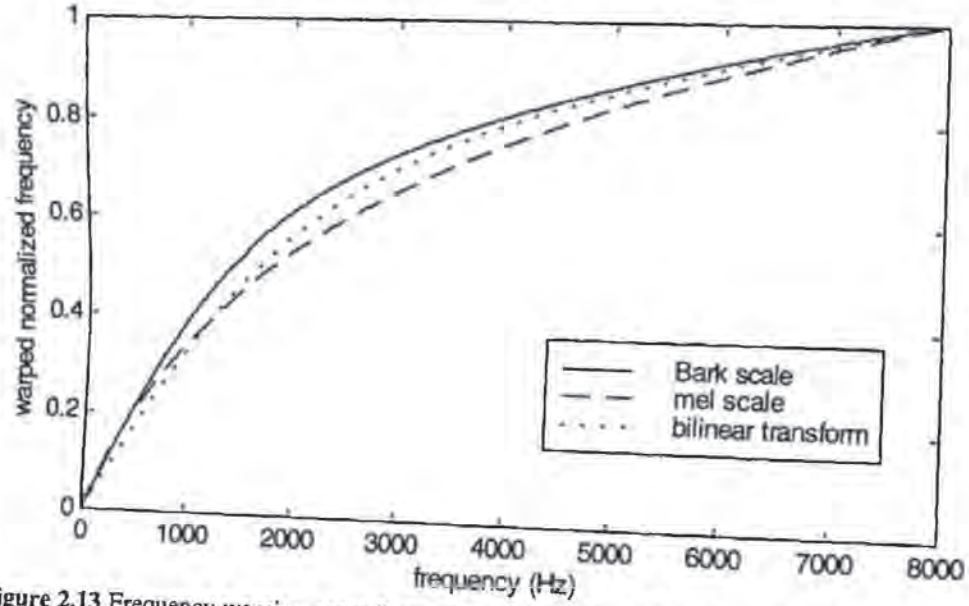


Figure 2.13 Frequency warping according to the Bark scale, ERB scale, mel-scale, and bilinear transform for $\alpha = 0.6$: linear frequency in the x-axis and normalized frequency in the y-axis.

A number of techniques in the modern spoken language system, such as cepstral analysis, and dynamic feature, have benefited tremendously from perceptual research as discussed throughout this book.

2.1.3.4. Masking

Frequency masking is a phenomenon under which one sound cannot be perceived if another sound close in frequency has a high enough level. The first sound *masks* the other one. Fre-

quency-masking levels have been determined empirically, with complicated models that take into account whether the masker is a tone or noise, the masker's level, and other considerations.

We now describe a phenomenon known as *tone-masking noise*. It has been determined empirically that noise with energy E_N (dB) at Bark frequency g masks a tone at Bark frequency b if the tone's energy is below the threshold

$$T_T(b) = E_N - 6.025 - 0.275g + S_m(b-g) \quad (dB\ SPL) \tag{2.7}$$

where the *spread-of-masking* function $S_m(b)$ is given by

$$S_m(b) = 15.81 + 7.5(b+0.474) - 17.5\sqrt{1+(b+0.474)^2} \quad (dB) \tag{2.8}$$

We now describe a phenomenon known as *noise-masking tone*. It has been determined empirically that a tone at Bark frequency g with energy E_T (dB) masks noise at Bark frequency b if the noise energy is below the threshold

$$T_N(b) = E_T - 2.025 - 0.175g + S_m(b-g) \quad (dB\ SPL) \tag{2.9}$$

Masking thresholds are commonly referred to in the literature as Bark scale functions of *just noticeable distortion* (JND). Equation (2.8) can be approximated by a triangular spreading function that has slopes of +25 and –10 dB per Bark, as shown in Figure 2.14.

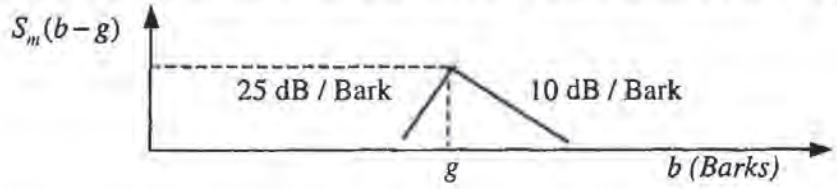


Figure 2.14 Contribution of Bark frequency g to the masked threshold $S_m(b)$.

In Figure 2.15 we show both the threshold of hearing and the masked threshold of a tone at 1 kHz with a 69 dB SPL. The combined masked threshold is the sum of the two in the linear domain

$$T(f) = 10\log_{10}\left(10^{0.1T_h(f)} + 10^{0.1T_T(f)}\right) \tag{2.10}$$

which is approximately the largest of the two.

In addition to frequency masking, there is a phenomenon called temporal masking by which a sound too close in time to another sound cannot be perceived. Whereas premasking tends to last about 5 ms, postmasking can last from 50 to 300 ms. Temporal masking level of a masker with a uniform level starting at 0 ms and lasting 200 ms is shown in Figure 2.16.

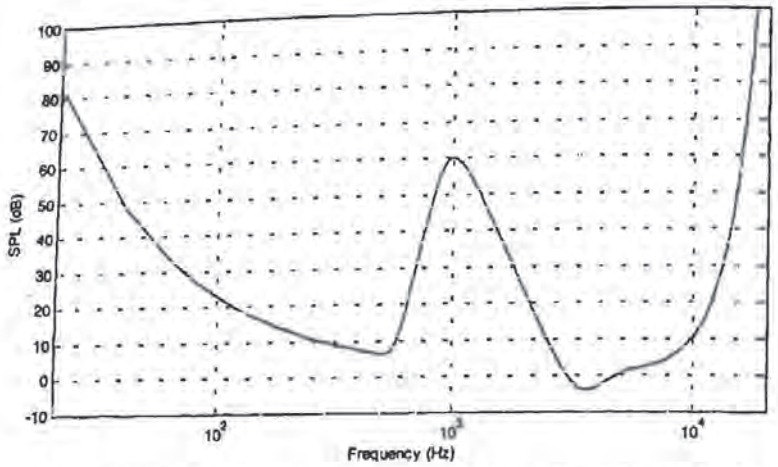


Figure 2.15 Absolute threshold of hearing and spread of masking threshold for a 1 kHz sine-wave masker with a 69 dB SPL. The overall masked threshold is approximately the largest of the two thresholds.

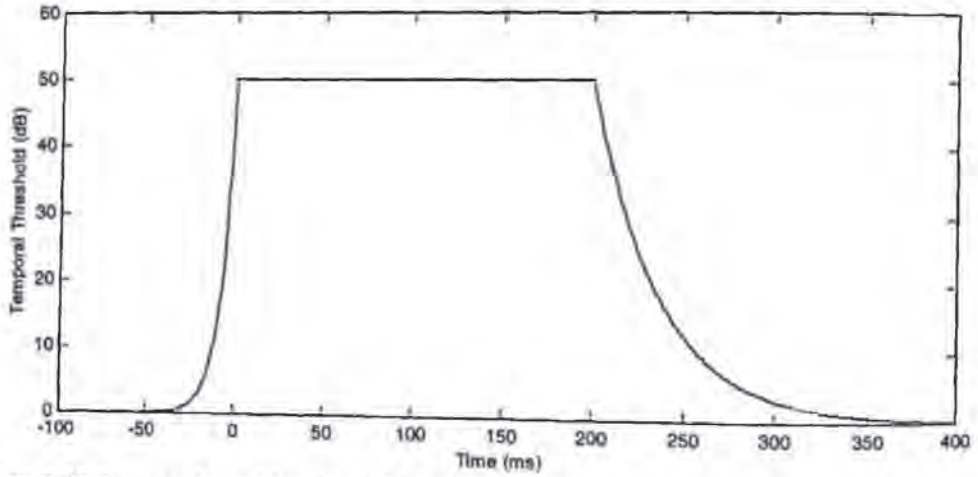


Figure 2.16 Temporal masking level of a masker with a uniform level starting at 0 ms and lasting 200 ms.

2.2. PHONETICS AND PHONOLOGY

We now discuss basic phonetics and phonology needed for spoken language processing. *Phonetics* refers to the study of speech sounds and their production, classification, and transcription. *Phonology* is the study of the distribution and patterning of speech sounds in a language and of the tacit rules governing pronunciation.

2.2.1. Phonemes

Linguist Ferdinand de Saussere (1857-1913) is credited with the observation that the relation between a sign and the object signified by it is arbitrary. The same concept, a certain yellow and black flying social insect, has the sign *honeybee* in English and *mitsubachi* in Japanese.

There is no particular relation between the various pronunciations and the meaning, nor do these pronunciations per se describe the bee's characteristics in any detail. For phonetics, this means that the speech sounds described in this chapter have no inherent meaning, and should be randomly distributed across the lexicon, except as affected by extraneous historical or etymological considerations. The sounds are just a set of arbitrary effects made available by human vocal anatomy. You might wonder about this theory when you observe, for example, the number of words beginning with *sn* that have to do with nasal functions in English: *sneeze, snort, sniff, snot, snore, snuffle*, etc. But Saussere's observation is generally true, except for obvious onomatopoetic (sound) words like *buzz*.

Like fingerprints, every speaker's vocal anatomy is unique, and this makes for unique vocalizations of speech sounds. Yet language communication is based on commonality of form at the perceptual level. To allow discussion of the commonalities, researchers have identified certain gross characteristics of speech sounds that are adequate for description and classification of words in dictionaries. They have also adopted various systems of notation to represent the subset of phonetic phenomena that are crucial for meaning.

As an analogy, consider the system of computer coding of text characters. In such systems, the *character* is an abstraction, e.g. the Unicode character U+0041. The identifying property of this character is its Unicode name *LATIN CAPITAL LETTER A*. This is a genuine abstraction; no particular realization is necessarily specified. As the Unicode 2.1 standard [1] states:

The Unicode Standard does not define glyph images. The standard defines how characters are interpreted, not how glyphs are rendered. The software or hardware-rendering engine of a computer is responsible for the appearance of the characters on the screen. The Unicode Standard does not specify the size, shape, nor orientation of on-screen characters.

Thus, the U+0041 character can be realized differently for different purposes, and in different sizes with different fonts:

U+0041 → A, *A*, A, A, A, ...

The realizations of the character U+0041 are called glyphs, and there is no distinguished uniquely correct glyph for U+0041. In speech science, the term *phoneme* is used to denote any of the minimal units of speech sound in a language that can serve to distinguish one word from another. We conventionally use the term *phone* to denote a phoneme's acoustic realization. In the example given above, U+0041 corresponds to a phoneme and the various fonts correspond to the phone. For example, English phoneme */t/* have two very different acoustic realizations in the words *sat* and *meter*. You had better treat them as two different phones if you want to build a spoken language system. We will use the terms *phone* or *phoneme* interchangeably to refer to the speaker-independent and context-independent units of meaningful sound contrast. Table 2.4 shows a complete list of phonemes used in American English. The set of phonemes will differ in realization across individual speakers. But phonemes will always function systematically to differentiate meaning in words, just as the phoneme */p/* signals the word *pat* as opposed to the similar-sounding but distinct *bat*. The important contrast distinguishing this pair of words is */p/* vs. */b/*.

In this section we concentrate on the basic qualities that *define and differentiate abstract phonemes*. In Section 2.2.1.3 below we consider *why and how phonemes vary* in their actual realizations by different speakers and in different contexts.

Table 2.4 English phonemes used for typical spoken language systems.

Phonemes	Word Examples	Description
iy	*feel, eve, me*	front close unrounded
ih	*fill, hit, lid*	front close unrounded (lax)
ae	*at, carry, gas*	front open unrounded (tense)
aa	*father, ah, car*	back open unrounded
ah	*cut, bud, up*	open-mid back unrounded
ao	*dog, lawn, caught*	open-mid back round
ay	*tie, ice, bite*	diphthong with quality: aa + ih
ax	*ago, comply*	central close mid (schwa)
ey	*ate, day, tape*	front close-mid unrounded (tense)
eh	*pet, berry, ten*	front open-mid unrounded
er	*turn, fur, meter*	central open-mid unrounded rhoti-
ow	*go, own, tone*	back close-mid rounded
aw	*foul, how, our*	diphthong with quality: aa + uh
oy	*toy, coin, oil*	diphthong with quality: ao + ih
uh	*book, pull, good*	back close-mid unrounded (lax)
uw	*tool, crew, moo*	back close round
b	*big, able, tab*	voiced bilabial plosive
p	*put, open, tap*	voiceless bilabial plosive
d	*dig, idea, wad*	voiced alveolar plosive
t	*talk, sat*	voiceless alveolar plosive &
t	*meter*	alveolar flap
g	*gut, angle, tag*	voiced velar plosive
k	*cut, ken, take*	voiceless velar plosive
f	*fork, after, if*	voiceless labiodental fricative
v	*vat, over, have*	voiced labiodental fricative
s	*sit, cast, toss*	voiceless alveolar fricative
z	*zap, lazy, haze*	voiced alveolar fricative
th	*thin, nothing, truth*	voiceless dental fricative
dh	*then, father, scythe*	voiced dental fricative
sh	*she, cushion, wash*	voiceless postalveolar fricative
zh	*genre, azure*	voiced postalveolar fricative
l	*lid*	alveolar lateral approximant
l	*elbow, sail*	velar lateral approximant
r	*red, part, far*	retroflex approximant
y	*yacht, yard*	palatal sonorant glide
w	*with, away*	labiovelar sonorant glide
hh	*help, ahead, hotel*	voiceless glottal fricative
m	*mat, amid, aim*	bilabial nasal
n	*no, end, pan*	alveolar nasal
ng	*sing, anger*	velar nasal
ch	*chin, archer, march*	voiceless alveolar affricate: t + sh
jh	*joy, agile, edge*	voiced alveolar affricate: d + zh

2.2.1.1. Vowels

The tongue shape and positioning in the oral cavity do not form a major constriction of air flow during vowel articulation. However, variations of tongue placement give each vowel its distinct character by changing the resonance, just as different sizes and shapes of bottles give rise to different acoustic effects when struck. The primary energy entering the pharyngeal and oral cavities in vowel production vibrates at the fundamental frequency. The major resonances of the oral and pharyngeal cavities for vowels are called F1 and F2 – the first and second formants, respectively. They are determined by tongue placement and oral tract shape in vowels, and they determine the characteristic timbre or quality of the vowel.

The relationship of F1 and F2 to one another can be used to describe the English vowels. While the shape of the complete vocal tract determines the spectral outcome in a complex, nonlinear fashion, generally F1 corresponds to the back or pharyngeal portion of the cavity, while F2 is determined more by the size and shape of the oral portion, forward of the major tongue extrusion. This makes intuitive sense – the cavity from the glottis to the tongue extrusion is longer than the forward part of the oral cavity, thus we would expect its resonance to be lower. In the vowel of *see*, for example, the tongue extrusion is far forward in the mouth, creating an exceptionally long rear cavity, and correspondingly low F1. The forward part of the oral cavity, at the same time, is extremely short, contributing to higher F2. This accounts for the wide separation of the two lowest dark horizontal bands in Figure 2.9, corresponding to F1 and F2, respectively. Rounding the lips has the effect of extending the front-of-tongue cavity, thus lowering F2. Typical values of F1 and F2 of American English vowels are listed in Table 2.5.

Table 2.5 Phoneme labels and typical formant values for vowels of English.

Vowel Labels	Mean F1 (Hz)	Mean F2 (Hz)
iy (feel)	300	2300
ih (fill)	360	2100
ae (gas)	750	1750
aa (father)	680	1100
ah (cut)	720	1240
ao (dog)	600	900
ax (comply)	720	1240
eh (pet)	570	1970
er (turn)	580	1380
ow (tone)	600	900
uh (good)	380	950
uw (tool)	300	940

The characteristic F1 and F2 values for vowels are sometimes called formant targets, which are ideal locations for perception. Sometimes, due to fast speaking or other limitations on performance, the speaker cannot quite attain an ideal target before the articulators begin shifting to targets for the following phoneme, which is phonetic context dependent. Additionally, there is a special class of vowels that combine two distinct sets of F1/F2 targets.

These are called *diphthongs*. As the articulators move, the initial vowel targets glide smoothly to the final configuration. Since the articulators are working faster in production of a diphthong, sometimes the *ideal* formant target values of the component values are not fully attained. Typical diphthongs of American English are listed in Table 2.6.

Table 2.6 The diphthongs of English.

Diphthong Labels	Components
ay (tie)	/aa/ → /iy/
ey (ate)	/eh/ → /iy/
oy (coin)	/ao/ → /iy/
aw (foul)	/aa/ → /uw/

Figure 2.17 shows the first two formants for a number of typical vowels.

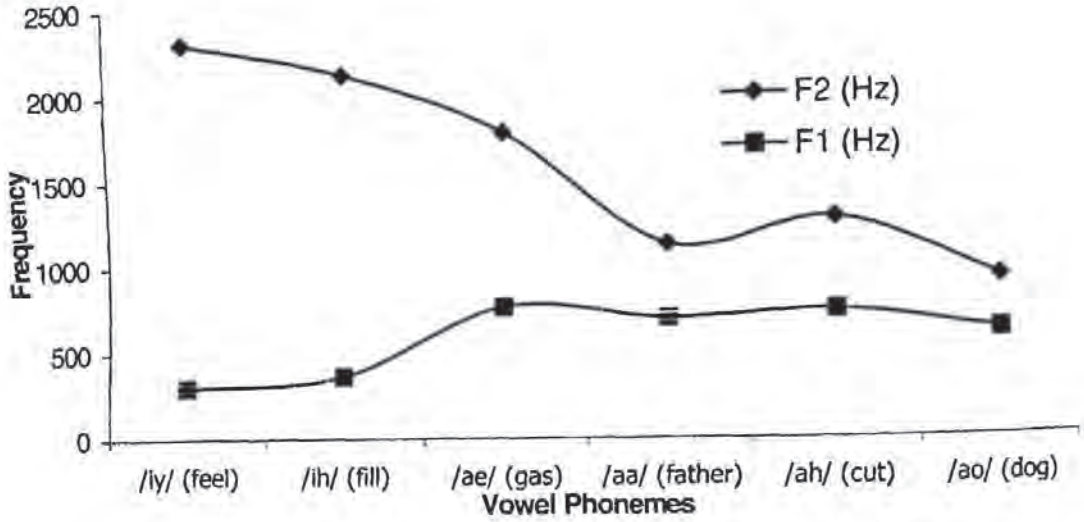


Figure 2.17 F1 and F2 values for articulations of some English vowels.

The major articulator for English vowels is the middle to rear portion of the tongue. The position of the tongue's surface is manipulated by large and powerful muscles in its root, which move it as a whole within the mouth. The linguistically important dimensions of movement are generally the ranges [front ⇔ back] and [high ⇔ low]. You can feel this movement easily. Say mentally, or whisper, the sound */iy/* (as in *see*) and then */aa/* (as in *father*). Do it repeatedly, and you will get a clear perception of the tongue movement from high to low. Now try */iy/* and then */uw/* (as in *blue*), repeating a few times. You will get a clear perception of place of articulation from front */iy/* to back */uw/*. Figure 2.18 shows a schematic characterization of English vowels in terms of relative tongue positions. There are two kinds of vowels: those in which tongue height is represented as a point and those in which it is represented as a vector.

Though the tongue hump is the major actor in vowel articulation, other articulators come into play as well. The most important secondary vowel mechanism for English and many other languages is lip rounding. Repeat the exercise above, moving from the */iy/* (*see*)

to the */uw/* (*blue*) position. Now rather than noticing the tongue movement, pay attention to your lip shape. When you say */iy/*, your lips will be flat, slightly open, and somewhat spread. As you move to */uw/*, they begin to *round out*, ending in a more puckered position. This lengthens the oral cavity during */uw/*, and affects the spectrum in other ways.

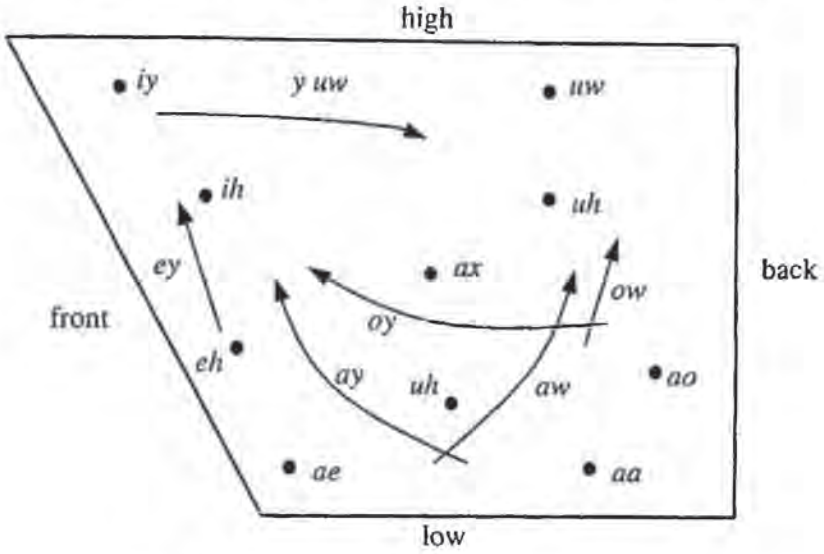


Figure 2.18 Relative tongue positions of English vowels [24].

Though there is always some controversy, linguistic study of phonetic abstractions, called *phonology*, has largely converged on the five binary features: +/- high, +/- low, +/- front, +/- back, and +/- round, plus the phonetically ambiguous but phonologically useful feature +/- tense, as adequate to uniquely characterize the major vowel distinctions of Standard English (and many other languages). Obviously, such a system is a little bit too free with logically contradictory specifications, such as [+high, +low], but these are excluded from real-world use. These features can be seen in Table 2.7.

Table 2.7 Phonological (abstract) feature decomposition of basic English vowels.

Vowel	high	low	front	back	round	tense
iy	+	-	+	-	-	+
ih	+	-	+	-	-	-
ae	-	+	+	-	-	+
aa	-	+	-	-	-	+
ah	-	-	-	-	-	+
ao	-	+	-	+	+	+
ax	-	-	-	-	-	-
eh	-	-	+	-	-	-
ow	-	-	-	+	+	+
uh	+	-	-	+	-	-
uw	+	-	-	+	-	+

This kind of abstract analysis allows researchers to make convenient statements about classes of vowels that behave similarly under certain conditions. For example, one may speak simply of the high vowels to indicate the set */iy, ih, uh, uw/*.

2.2.1.2. Consonants

Consonants, as opposed to vowels, are characterized by significant constriction or obstruction in the pharyngeal and/or oral cavities. Some consonants are voiced; others are not. Many consonants occur in pairs, that is, they share the same configuration of articulators, and one member of the pair additionally has voicing which the other lacks. One such pair is */s, z/*, and the voicing property that distinguishes them shows up in the non-periodic noise of the initial segment */s/* in Figure 2.5 as opposed to the voiced consonant end-phone, */z/*. Manner of articulation refers to the articulation mechanism of a consonant. The major distinctions in manner of articulation are listed in Table 2.8.

Table 2.8 Consonant manner of articulation.

Manner	Sample Phone	Example Words	Mechanism
Plosive	*/p/*	tat, tap	Closure in oral cavity
Nasal	*/m/*	team, meet	Closure of nasal cavity
Fricative	*/s/*	sick, kiss	Turbulent airstream noise
Retroflex liquid	*/r/*	rat, tar	Vowel-like, tongue high and curled back
Lateral liquid	*/l/*	lean, kneel	Vowel-like, tongue central, side airstream
Glide	*/y/,/w/*	yes, well	Vowel-like

The English phones that typically have voicing without complete obstruction or narrowing of the vocal tract are called *semivowels* and include */l, r/*, the *liquid* group, and */y, w/*, the *glide* group. Liquids, glides, and vowels are all *sonorant*, meaning they have continuous voicing. Liquids */l/* and */r/* are quite vowel-like and in fact may become *syllabic* or act entirely as vowels in certain positions, such as the *l* at the end of *edible*. In */l/*, the airstream flows around the sides of the tongue, leading to the descriptive term *lateral*. In */r/*, the tip of the tongue is curled back slightly, leading to the descriptive term *retroflex*. Figure 2.19 shows some semivowels.

Glides */y, w/* are basically vowels */iy, uw/* whose initial position within the syllable require them to be a little shorter and to lack the ability to be stressed, rendering them just different enough from true vowels that they are classed as a special category of consonant. Pre-vocalic glides that share the syllable-initial position with another consonant, such as the */y/* in the second syllable of *computer /k uh m . p y uw . t er/*, or the */w/* in *quick /k w ih k/*, are sometimes called *on-glides*. The semivowels, as a class, are sometimes called *approximants*, meaning that the tongue approaches the top of the oral cavity, but does not completely contact so as to obstruct the air flow.

Even the non-sonorant consonants that require complete or close-to-complete obstruction may still maintain some voicing before or during the obstruction, until the pressure dif-

ferential across the glottis starts to disappear, due to the closure. Such voiced consonants include */b,d,g, z, zh, v/*. They have a set of counterparts that differ only in their characteristic lack of voicing: */p,t,k, s, sh, f/*.

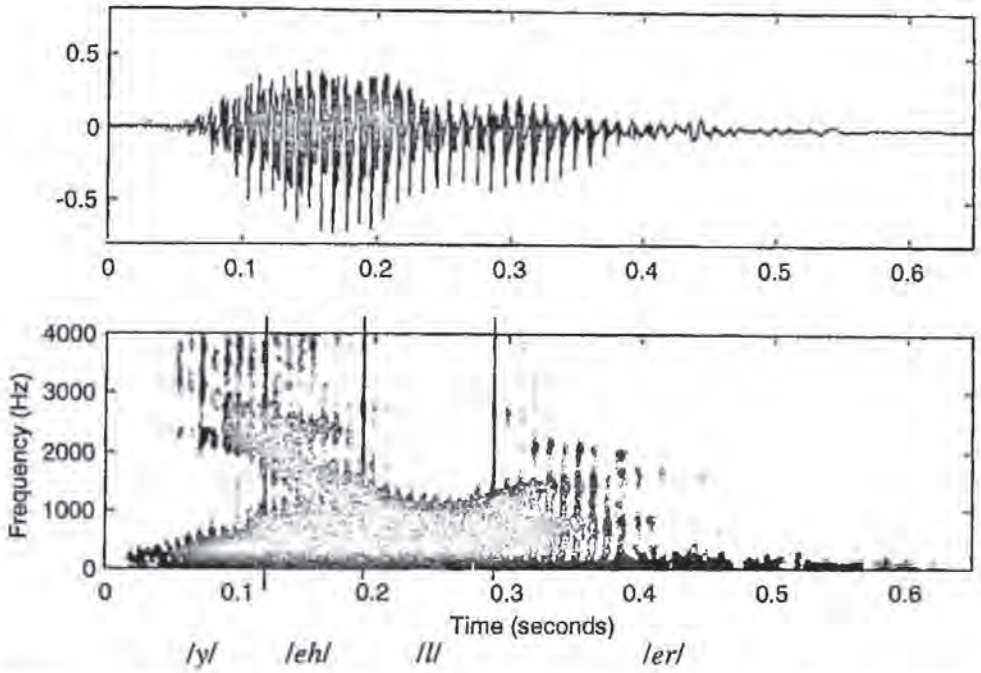


Figure 2.19 Spectrogram for the word *yeller*, showing semivowels */y/*, */l/*, */er/* (approximate phone boundaries shown with vertical lines).

Nasal consonants */m,n/* are a mixed bag: the oral cavity has significant constriction (by the tongue or lips), yet the voicing is continuous, like that of the sonorants, because, with the velar flap open, air passes freely through the nasal cavity, maintaining a pressure differential across the glottis.

A consonant that involves complete blockage of the oral cavity is called an obstruent stop, or plosive consonant. These may be voiced throughout if the trans-glottal pressure drop can be maintained long enough, perhaps through expansion of the wall of the oral cavity. In any case, there can be voicing for the early sections of stops. Voiced, unvoiced pairs of stops include: */b,p/*, */d,t/*, and */g,k/*. In viewing the waveform of a stop, a period of silence corresponding to the oral closure can generally be observed. When the closure is removed (by opening the constrictor, which may be lips or tongue), the trapped air rushes out in a more or less sudden manner. When the upper oral cavity is unimpeded, the closure of the vocal folds themselves can act as the initial blocking mechanism for a type of stop heard at the very beginning of vowel articulation in vowel-initial words like *atrophy*. This is called a *glottal stop*. Voiceless plosive consonants in particular exhibit a characteristic aperiodic *burst* of energy at the (articulatory) point of closure as shown in Figure 2.20 just prior to */i/*. By com-

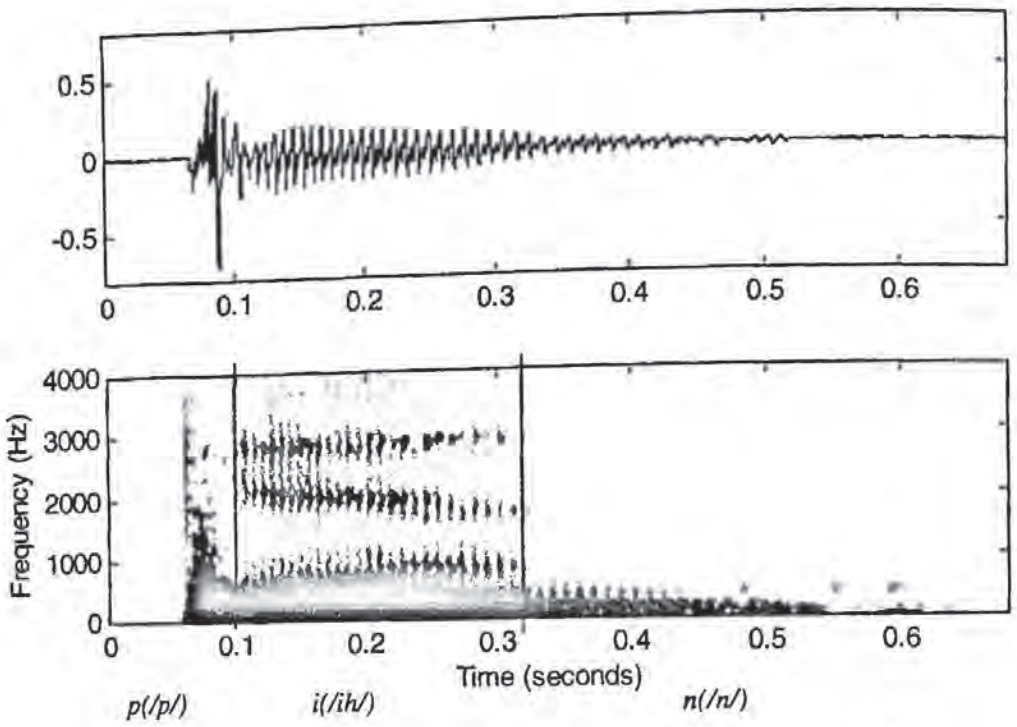


Figure 2.20 Spectrogram: stop release *burst* of */p/* in the word *pin*.

parison, the voicing of voiced plosive consonants may not always be obvious in a spectrogram.

A consonant that involves nearly complete blockage of some position in the oral cavity creates a narrow stream of turbulent air. The friction of this air stream creates a nonperiodic hiss-like effect. Sounds with this property are called fricatives and include */s, z/*. There is no voicing during the production of *s*, while there can be voicing (in addition to the frication noise), during the production of *z*, as discussed above. */s, z/* have a common place of articulation, as explained below, and thus form a natural similarity class. Though controversial, */h/* can also be thought of as a (glottal) fricative. */s/* in word-initial position and */z/* in word-final position are exemplified in Figure 2.5.

Some sounds are complex combinations of manners of articulation. For example, the *affricates* consist of a stop (e.g., */t/*), followed by a fricative [e.g., */sh/*) combining to make a unified sound with rapid phases of closure and continuancy (e.g., $\{t + sh\} = ch$ as in *church*). The affricates in English are the voiced/unvoiced pairs: */j/* $(d + zh)$ and */ch/* $(t + sh)$. The complete consonant inventory of English is shown in Table 2.9.

Consider the set */m/*, */n/*, */ng/* from Table 2.9. They are all voiced nasal consonants, yet they sound distinct to us. The difference lies in the location of the major constriction along the top of the oral cavity (from lips to velar area) that gives each consonant its unique quality. The articulator used to touch or approximate the given location is usually some spot along the length of the tongue. As shown in Figure 2.21, the combination of articulator and place of articulation gives each consonant its characteristic sound:

Table 2.9 Manner of articulation of English consonants.

Consonant Labels	Consonant Examples	Voiced?	Manner
b	**b**ig, able, ta**b**	+	plosive
p	**p**ut, open, ta**p**	-	plosive
d	**d**ig, **d**ea, wa**d**	+	plosive
t	**t**alk, sa**t**	-	plosive
g	**g**ut, angle, ta**g**	+	plosive
k	**c**ut, oa**k**en, ta**k**e	-	plosive
v	**v**at, over, ha**v**e	+	fricative
f	**f**ork, after, i**f**	-	fricative
z	zap, lazy, ha**z**e	+	fricative
s	**s**it, ca**s**t, to**ss**	-	fricative
dh	**th**en, fa**th**er, scy**th**e	+	fricative
th	**th**in, no**th**ing, tru**th**	-	fricative
zh	genre, azure, beige	+	fricative
sh	**sh**e, cu**sh**ion, wa**sh**	-	fricative
jh	**j**oy, agile, e**dg**e	+	affricate
ch	**ch**in, ar**ch**er, ma**rch**	-	affricate
l	**l**id, el**b**ow, sa**il**	+	lateral
r	**r**ed, pa**r**t, fa**r**	+	retroflex
y	**y**acht, o**n**ion, **y**ard	+	glide
w	**w**ith, a**w**ay	+	glide
hh	**h**elp, a**h**ead, **h**otel	+	fricative
m	**m**at, a**m**id, ai**m**	+	nasal
n	**n**o, e**n**d, pa**n**	+	nasal
ng	sing, a**n**ger, dri**nk**	+	nasal

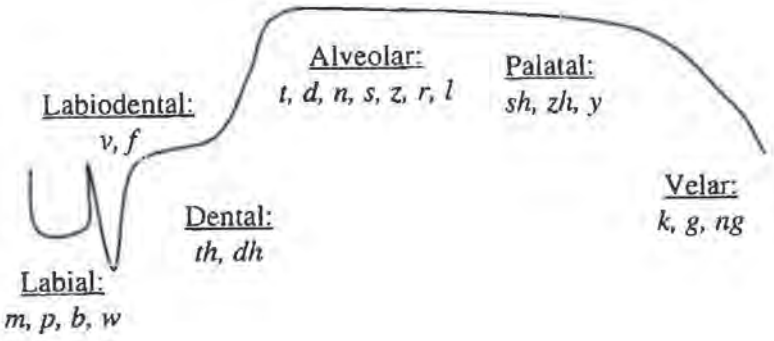


Figure 2.21 The major places of consonant articulation with respect to the human mouth.

- The *labial* consonants have their major constriction at the lips. This includes */p/, /b/* (these two differ only by manner of articulation) and */m/* and */w/*.
- The class of *dental or labio-dental* consonants includes */f, v/* and */th, dh/* (the members of these groups differ in manner, not place).
- *Alveolar* consonants bring the front part of the tongue, called the tip or the part behind the tip called the blade, into contact or approximation to the alveolar ridge, rising semi-vertically above and behind the teeth. These include */t, d, n, s, z, r, l/*. The members of this set again differ in manner of articulation (voicing, continuity, nasality), rather than place.
- *Palatal* consonants have approximation or constriction on or near the roof of the mouth, called the palate. The members include */sh, zh, y/*.
- *Velar* consonants bring the articulator (generally the back of the tongue), up to the rearmost top area of the oral cavity, near the velar flap. Velar consonants in English include */k, g/* (differing by voicing) and the nasal continuant */ng/*.

With the place terminology, we can complete the descriptive inventory of English consonants, arranged by manner (rows), place (columns), and voiceless/voiced (pairs in cells) as illustrated in Table 2.10.

Table 2.10 The consonants of English arranged by place (columns) and manner (rows).

	Labial	Labio-dental	Dental	Alveolar	Palatal	Velar	Glottal
Plosive	*p b*			*t d*		*k g*	*?*
Nasal	*m*			*n*		*ng*	
Fricative		*f v*	*th dh*	*s z*	*sh zh*		*h*
Retroflex sonorant				*r*			
Lateral sonorant				*l*			
Glide	*w*				*y*		

2.2.1.3. Phonetic Typology

The oral, nasal, pharyngeal, and glottal mechanisms actually make available a much wider range of effects than English happens to use. So, it is expected that other languages would utilize other vocal mechanisms, in an internally consistent but essentially arbitrary fashion, to represent their lexicons. In addition, often a vocal effect that is part of the systematic linguistic phonetics of one language is present in others in a less codified, but still perceptible, form. For example, Japanese vowels have a characteristic distinction of length that can be hard for non-natives to perceive and to use when learning the language. The words *kado* (*corner*) and *kaado* (*card*) are spectrally identical, differing only in that *kado* is much shorter

in all contexts. The existence of such minimally-contrasting pairs is taken as conclusive evidence that *length* is phonemically distinctive for Japanese. As noted above, what is linguistically distinctive in any one language is generally present as a less *meaningful* signaling dimension in other languages. Thus, vowel length can be manipulated in any English word as well, but this occurs either consciously for emphasis or humorous effect, or unconsciously and very predictably at clause and sentence end positions, rather than to signal lexical identity in all contexts, as in Japanese.

Other interesting sounds that the English language makes no linguistic use of include the trilled *r* sound and the implosive. The trilled *r* sound is found in Spanish, distinguishing (for example) the words *pero* (*but*) and *perro* (*dog*). This trill could be found in times past as a non-lexical sound used for emphasis and interest by American circus ringmasters and other showpersons.

While the world's languages have all the variety of manner of articulation exemplified above and a great deal more, the primary dimension lacking in English that is exploited by a large subset of the world's languages is pitch variation. Many of the huge language families of Asia and Africa are tonal, including all varieties of Chinese. A large number of other languages are not considered strictly tonal by linguistics, yet they make systematic use of pitch contrasts. These include Japanese and Swedish. To be considered tonal, a language should have lexical meaning contrasts cued by pitch, just as the lexical meaning contrast between *pig* and *big* is cued by a voicing distinction in English. For example, Mandarin Chinese has four primary tones (tones can have minor context-dependent variants just like ordinary phones, as well) as shown in Table 2.11.

Table 2.11 The contrastive tones of Mandarin Chinese.

Tone	Shape	Example	Chinese	Meaning
1	High level	*ma*	妈	mother
2	High rising	*ma*	麻	numb
3	Low rising	*ma*	马	horse
4	High falling	*ma*	骂	to scold

Though English does not make systematic use of pitch in its inventory of word contrasts, nevertheless, as we always see with any possible phonetic effect, pitch is systematically varied in English to signal a speaker's emotions, intentions, and attitudes, and it has some linguistic function in signaling grammatical structure as well. Pitch variation in English will be considered in more detail in Chapter 15.

2.2.2. The Allophone: Sound and Context

The vowel and consonant charts provide abstract symbols for the phonemes – major sound distinctions. Phonemic units should be correlated with potential meaning distinctions. For example, the change created by holding the tongue high and front (*/iy/*) vs. directly down

from the (frontal) position for */eh/*, in the consonant context */m _ n/*, corresponds to an important meaning distinction in the lexicon of English: *mean /m iy n/* vs. *men /m eh n/*. This meaning contrast, conditioned by a pair of rather similar sounds, in an identical context, justifies the inclusion of */iy/* and */eh/* as logically separate distinctions.

However, one of the fundamental, meaning-distinguishing sounds is often modified in some systematic way by its phonetic neighbors. The process by which neighboring sounds influence one another is called *coarticulation*. Sometimes, when the variations resulting from coarticulatory processes can be consciously perceived, the modified phonemes are called *allophones*. Allophonic differences are always *categorical*, that is, they can be understood and denoted by means of a small, bounded number of symbols or diacritics on the basic phoneme symbols.

As an experiment, say the word *like* to yourself. Feel the front of the tongue touching the alveolar ridge (cf. Figure 2.21) when realizing the initial phoneme */l/*. This is one allophone of */l/*, the so-called *light* or *clear /l/*. Now say *kill*. In this word, most English speakers will no longer feel the front part of the tongue touch the alveolar ridge. Rather, the */l/* is realized by stiffening the broad midsection of the tongue in the rear part of the mouth while the continuant airstream escapes laterally. This is another allophone of */l/*, conditioned by its syllable-final position, called the *dark /l/*. Predictable contextual effects on the realization of phones can be viewed as a nuisance for speech recognition, as will be discussed in Chapter 9. On the other hand, such variation, because it is systematic, could also serve as a cue to the syllable, word, and prosodic structure of speech.

Now experiment with the sound */p/* by holding a piece of tissue in front of your mouth while saying the word *pin* in a normal voice. Now repeat this experiment with *spin*. For most English speakers, the word *pin* produces a noticeable puff of air, called aspiration. But the same phoneme, */p/*, embedded in the consonant cluster */sp/* loses its aspiration (burst, see the lines bracketing the */p/* release in *pin* and *spin* in Figure 2.22), and because these two types of */p/* are in complementary distribution (completely determined by phonetic and syllabic context), the difference is considered allophonic.

Try to speak the word *bat* in a framing phrase *say bat again*. Now speak *say bad again*. Can you feel the length difference in the vowel */ae/*? A vowel before a voiced consonant, e.g., */d/*, seems typically longer than the same vowel before the unvoiced counterpart, in this case */t/*.

A sound phonemicized as */t/* or */d/*, that is, a stop made with the front part of the tongue, may be reduced to a quick tongue tap that has a different sound than either */t/* or */d/* in fuller contexts. This process is called flapping. It occurs when */t/* or */d/* closes a stressed vowel (coda position) followed by an unstressed vowel, as in: *bitter, batter, murder, quarter, humidity,* and can even occur across words as long as the preconditions are met, as in *you can say that again*. Sometimes the velar flap opens too soon (anticipation), giving a characteristically nasal quality to some pre-nasal vowels such as */ae/* in *ham* vs. *had*. We have a more detailed discussion on allophones in Chapter 9.

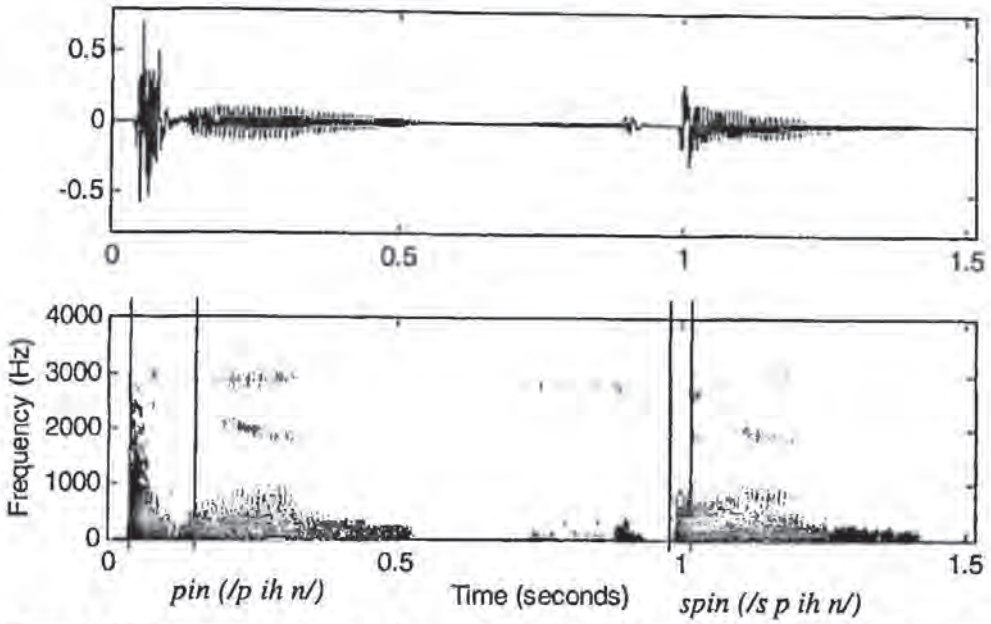


Figure 2.22 Spectrogram: bursts of *pin* and *spin*. The relative duration of a *p*-burst in different phonetic contexts is shown by the differing width of the area between the vertical lines.

2.2.3. Speech Rate and Coarticulation

In addition to allophones, there are other variations in speech for which no small set of established categories of variation can be established. These are *gradient*, existing along a scale for each relevant dimension, with speakers scattered widely. In general, it is harder to become consciously aware of coarticulation effects than of allophonic alternatives.

Individual speakers may vary their rates according to the content and setting of their speech, and there may be great inter-speaker differences as well. Some speakers may pause between every word, while others may speak hundreds of words per minute with barely a pause between sentences. At the faster rates, formant targets are less likely to be fully achieved. In addition, individual allophones may merge.

For example [20], consider the utterance *Did you hit it to Tom*? The pronunciation of this utterance is */d ih d y uw h ih t ih t t uw t aa m/*. However, a realistic, casual rendition of this sentence would appear as */d ih jh ax hh ih dx ih t ix t aa m/*, where */ix/* is a reduced schwa */ax/* that is short and often unvoiced, and */dx/* is a kind of shortened, indistinct stop, intermediate between */d/* and */t/*. The following five phonologic rules have operated on altering the pronunciation in the example:

- Palatalization of */d/* before */y/* in *did you*
- Reduction of unstressed */u/* to schwa in *you*

- Flapping of intervocalic /t/ in *hit it*
- Reduction of schwa and devoicing of /u/ in *to*
- Reduction of geminate (double consonant) /t/ in *it to*

There are also coarticulatory influences in the spectral appearance of speech sounds, which can only be understood at the level of spectral analysis. For example, in vowels, consonant neighbors can have a big effect on formant trajectories near the boundary. Consider the differences in *F1* and *F2* in the vowel */eh/* as realized in words with different initial consonants *bet*, *debt*, and *get*, corresponding to the three major places of articulation (labial, alveolar, and velar), illustrated in Figure 2.23. You can see the different relative spreads of *F1* and *F2* following the initial stop consonants.

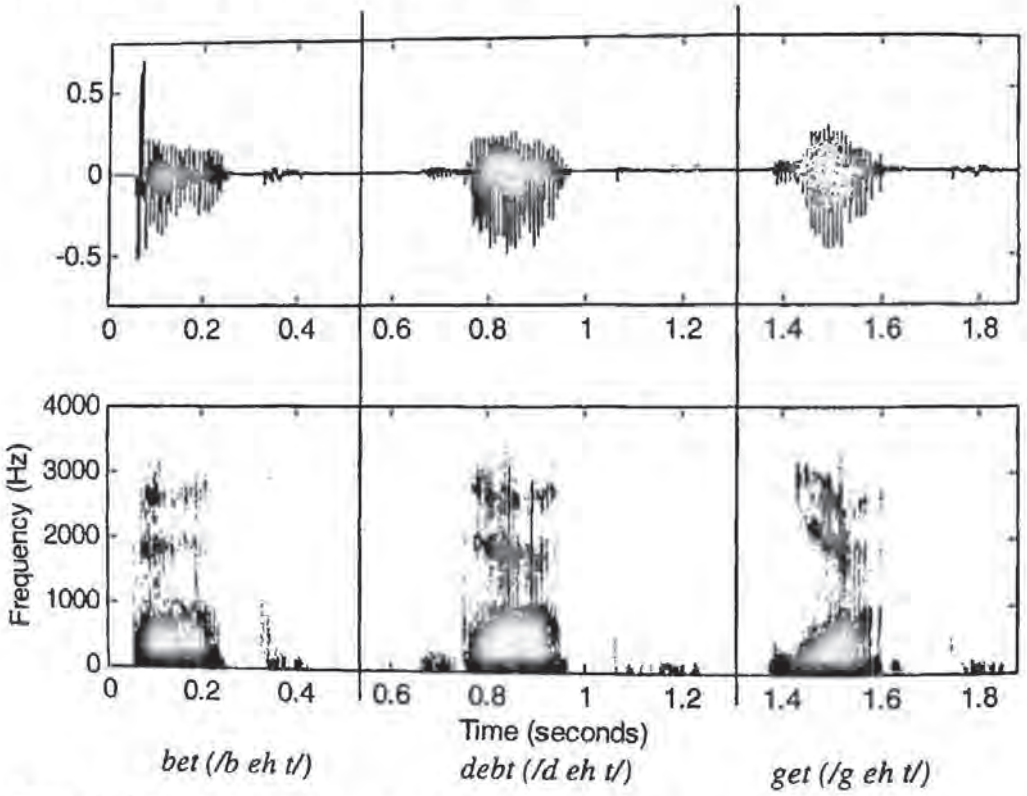


Figure 2.23 Spectrogram: *bet*, *debt*, and *get* (separated by vertical lines). Note different relative spreads of *F1* and *F2* following the initial stop consonants in each word.

Now let's see different consonants following the same vowel, *ebb*, *head*, and *egg*. In Figure 2.23, the coarticulatory effect is *perseverance*; i.e., in the early part of the vowel the articulators are still somewhat set from realization of the initial consonant. In the *ebb*, *head*, and *egg* examples shown in Figure 2.24, the coarticulatory effect is *anticipation*; i.e., in the latter part of the vowel the articulators are moving to prepare for the upcoming consonant articulation. You can see the increasing relative spread of *F1* and *F2* at the final vowel-consonant transition in each word.

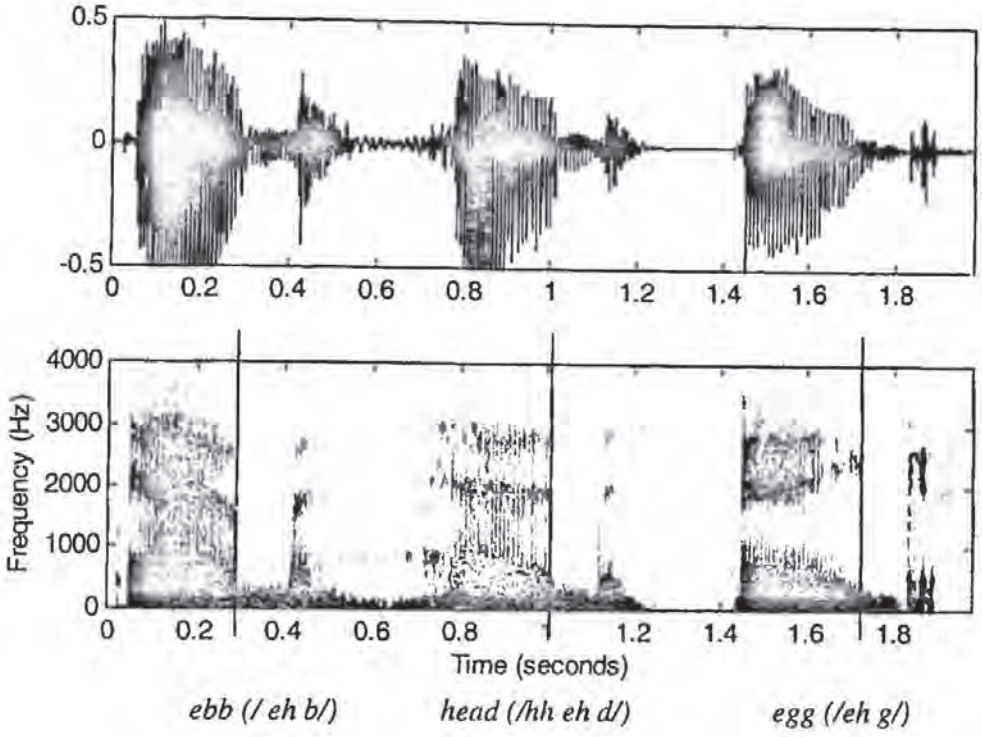


Figure 2.24 Spectrogram: *ebb*, *head*, and *egg*. Note the increasing relative spread of *F1* and *F2* at the final vowel-consonant transition in each word.

2.3. SYLLABLES AND WORDS

Phonemes are small building blocks. To contribute to language meaning, they must be organized into longer cohesive spans, and the units so formed must be combined in characteristic patterns to be meaningful, such as syllables and words in the English language.

2.3.1. Syllables

An intermediate unit, the *syllable*, is sometimes thought to interpose between the phones and the word level. The syllable is a slippery concept, with implications for both production and perception. Here we will treat it as a perceptual unit. Syllables are generally centered around vowels in English, giving two perceived syllables in a word like *tomcat*: */tOm-cAt/*. To completely parse a word into syllables requires making judgments of consonant affiliation (with the syllable peak vowels). The question of whether such judgments should be based on articulatory or perceptual criteria, and how they can be rigorously applied, remains unresolved.

Syllable centers can be thought of as *peaks* in sonority (high-amplitude, periodic sections of the speech waveform). These sonority peaks have affiliated *shoulders* of strictly non-increasing sonority. A scale of sonority can be used, ranking consonants along a continuum of stops, affricates, fricatives, and approximants. So, in a word like *verbal*, the syllabification would be *ver-bal*, or *verb-al*, but not *ve-rbal*, because putting the approximant /r/ before the stop /b/ in the second syllable would violate the non-decreasing sonority requirement heading into the syllable.

As long as the sonority conditions are met, the exact affiliation of a given consonant that could theoretically affiliate on either side can be ambiguous, unless determined by higher-order considerations of word structure, which may block affiliation. For example, in a word like *beekeeper*, an abstract boundary in the compound between the component words *bee* and *keeper* keeps us from accepting the syllable parse: *beek-eeper*, based on lexical interpretation. However, the same phonetic sequence in *beaker* could, depending on one's theory of syllabicity, permit affiliation of the *k: beak-er*. In general, the syllable is a unit that has intuitive plausibility but remains difficult to pin down precisely.

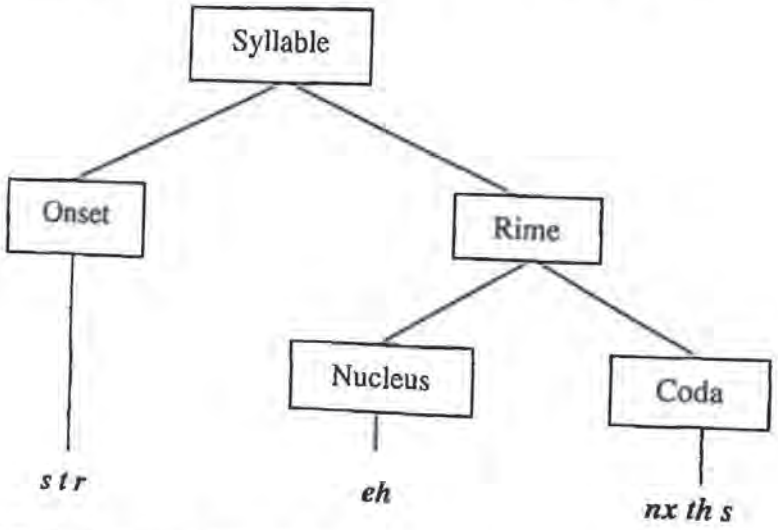


Figure 2.25 The word/syllable *strengths* (*/s t r eh nx th s/*) is the longest syllable of English.

Syllables are thought (by linguistic theorists) to have internal structure, and the terms used are worth knowing. Consider a big syllable such as *strengths /s t r eh nx th s/*. This consists of a vowel peak, called the *nucleus*, surrounded by the other sounds in characteristic positions. The *onset* consists of initial consonants if any, and the rime is the nucleus with trailing consonants (the part of the syllable that matters in determining poetic rhyme). The coda consists of consonants in the rime following the nucleus (in some treatments, the last consonant in a final cluster would belong to an *appendix*). This can be diagrammed as a syllable parse tree as shown in Figure 2.25. The syllable is sometimes thought to be the primary domain of coarticulation, that is, sounds within a syllable influence one another's realization more than the same sounds separated by a syllable boundary.

2.3.2. Words

The concept of words seems intuitively obvious to most speakers of Indo-European languages. It can be loosely defined as a lexical item, with an agreed-upon meaning in a given speech community, that has the freedom of syntactic combination allowed by its type (noun, verb, etc.).

In spoken language, there is a segmentation problem: words *run together* unless affected by a disfluency (unintended speech production problem) or by the deliberate placement of a pause (silence) for some structural or communicative reason. This is surprising to many people, because literacy has conditioned speakers/readers of Indo-European languages to expect a *blank space* between words on the printed page. But in speech, only a few true pauses (the aural equivalent of a blank space) may be present. So, what appears to the reading eye as *never give all the heart, for love* would appear to the ear, if we simply use letters to stand for their corresponding English speech sounds, as *nevergivealltheheart forlove* or, in phonemes, as *n eh v er g ih v ah l dh ax h aa r t \\ f ao r l ah v*. The \\ symbol marks a linguistically motivated pause, and the units so formed are sometimes called *intonation phrases*, as explained in Chapter 15.

Certain facts about word structure and combinatorial possibilities are evident to most native speakers and have been confirmed by decades of linguistic research. Some of these facts describe relations among words when considered in isolation, or concern groups of related words that seem intuitively similar along some dimension of form or meaning – these properties are *paradigmatic*. Paradigmatic properties of words include part-of-speech, inflectional and derivational morphology, and compound structure. Other properties of words concern their behavior and distribution when combined for communicative purposes in fully functioning utterances – these properties are *syntagmatic*.

2.3.2.1. Lexical Part-of-Speech

Lexical part-of-speech (POS) is a primitive form of linguistic theory that posits a restricted inventory of word-type categories, which capture generalizations of word forms and distributions. Assignment of a given POS specification to a word is a way of summarizing certain facts about its potential for syntagmatic combination. Additionally, paradigms of word formation processes are often similar within POS types and subtypes as well. The word properties upon which POS category assignments are based may include affixation behavior, very abstract semantic typologies, distributional patterns, compounding behavior, historical development, productivity and generalizabilty, and others.

A typical set of POS categories would include *noun*, *verb*, *adjective*, *adverb*, *interjection*, *conjunction*, *determiner*, *preposition*, and *pronoun*. Of these, we can observe that certain classes of words consist of infinitely large membership. This means new members can be added at any time. For example, the category of noun is constantly expanded to accommodate new inventions, such as Velcro or Spandex. New individuals are constantly being born, and their names are a type of noun called *proper noun*. The proliferation of words us-

ing the descriptive prefix *cyber* is another recent set of examples: *cyberscofflaw, cybersex,* and even *cyberia* illustrate the infinite creativity of humans in manipulating word structure to express new shades of meaning, frequently by analogy with, and using fragments of, existing vocabulary. Another example is the neologism *sheeple*, a noun combining the forms and meanings of *sheep* and *people* to refer to large masses of people who lack the capacity or willingness to take independent action. We can create new words whenever we like, but they had best fall within the predictable paradigmatic and syntagmatic patterns of use summarized by the existing POS generalizations, or there will be little hope of their adoption by any other speaker. These open POS categories are listed in Table 2.12. Nouns are inherently referential. They refer to persons, places, and things. Verbs are predicative; they indicate relations between entities and properties of entities, including participation in events. Adjectives typically describe and more completely specify noun reference, while adverbs describe, intensify, and more completely specify verbal relations. Open-class words are sometimes called *content* words, for their referential properties.

Table 2.12 Open POS categories.

Tag	Description	Function	Example
N	Noun	Names entity	*cat*
V	Verb	Names event or condition	*forget*
Adj	Adjective	Descriptive	*yellow*
Adv	Adverb	Manner of action	*quickly*
Interj	Interjection	Reaction	*oh!*

In contrast to the open-class categories, certain other categories of words only rarely and very slowly admit new members over the history of English development. These closed POS categories are shown in Table 2.13. The closed-category words are fairly stable over time. Conjunctions are used to join larger syntactically complete phrases. Determiners help to narrow noun reference possibilities. Prepositions denote common spatial and temporal relations of objects and actions to one another. Pronouns provide a convenient substitute for noun phrases that are fully understood from context. These words denote grammatical relations of other words to one another and fundamental properties of the world and how humans understand it. They can, of course, change slowly; for example, the Middle English pronoun *thee* is no longer in common use. The closed-class words are sometimes called *function* words.

Table 2.13 Closed POS categories.

Tag	Description	Function	Example
Conj	Conjunction	Coordinates phrases	*and*
Det	Determiner	Indicates definiteness	*the*
Prep	Preposition	Relations of time, space, direction	*from*
Pron	Pronoun	Simplified reference	*she*

The set of POS categories can be extended indefinitely. Examples can be drawn from the Penn Treebank project (http://www.cis.upenn.edu/ldc) as shown in Table 2.14, where you can find the proliferation of sub-categories, such as *Verb, base form* and *Verb, past tense*. These categories incorporate *morphological* attributes of words into the POS label system discussed in Section 2.3.2.2.

Table 2.14 Treebank POS categories – an expanded inventory.

String	Description	Example
CC	Coordinating conjunction	and
CD	Cardinal number	two
DT	Determiner	the
EX	Existential *there*	there (*There was an old lady*)
FW	Foreign word	omerta
IN	Preposition, subord. conjunction	over, but
JJ	Adjective	yellow
JJR	Adjective, comparative	better
JJS	Adjective, superlative	best
LS	List item marker	
MD	Modal	might
NN	Noun, singular or mass	rock, water
NNS	Noun, plural	rocks
NNP	Proper noun, singular	Joe
NNPS	Proper noun, plural	Red Guards
PDT	Predeterminer	all (*all the girls*)
POS	Possessive ending	's
PRP	Personal pronoun	I
PRP$	Possessive pronoun	mine
RB	Adverb	quickly
RBR	Adverb, comparative	higher (*shares closed higher.*)
RBS	Adverb, superlative	highest (*he jumped highest of all.*)
RP	Particle	up (*take up the cause*)
TO	*to*	to
UH	Interjection	hey!
VB	Verb, base form	choose
VBD	Verb, past tense	chose
VBG	Verb, gerund, or present participle	choosing
VBN	Verb, past participle	chosen
VBP	Verb, non-third person sing. present	jump
VBZ	Verb, third person singular present	jumps
WDT	Wh-determiner	which
WP	Wh-pronoun	who
WP$	Possessive wh-pronoun	whose
WRB	Wh-adverb	when (*When he came, it was late.*)

POS tagging is the process of assigning a part-of-speech or other lexical class marker to each word in a corpus. There are many algorithms to automatically tag input sentences into a set of tags. Rule-based methods [45], hidden Markov models (see Chapter 8) [23, 29, 46], and machine-learning methods [6] are used for this purpose.

2.3.2.2. Morphology

Morphology is about the subparts of words, i.e., the patterns of word formation including inflection, derivation, and the formation of compounds. English mainly uses prefixes and suffixes to express *inflection* and *derivational* morphology.

Inflectional morphology deals with variations in word form that reflect the contextual situation of a word in phrase or sentence syntax, and that rarely have direct effect on interpretation of the fundamental meaning expressed by the word. English inflectional morphology is relatively simple and includes person and number agreement and tense markings only. The variation in *cats* (vs. *cat*) is an example. The plural form is used to refer to an indefinite number of cats greater than one, depending on a particular situation. But the basic POS category (*noun*) and the basic meaning (*felis domesticus*) are not substantially affected. Words related to a common lemma via inflectional morphology are said to belong to a common paradigm, with a single POS category assignment. In English, common paradigm types include the verbal set of affixes (pieces of words): *-s, -ed, -ing*; the noun set: *-s*; and the adjectival *-er, -est*. Note that sometimes the base form may change spelling under affixation, complicating the job of automatic textual analysis methods. For historical reasons, certain paradigms may consist of highly idiosyncratic irregular variation as well, e.g., *go, going, went, gone* or *child, children*. Furthermore, some words may belong to defective paradigms, where only the singular (noun: *equipment*) or the plural (noun: *scissors*) is provided for.

In *derivational morphology*, a given root word may serve as the source for wholly new words, often with POS changes as illustrated in Table 2.15. For example, the terms *racial* and *racist*, though presumably based on a single root word *race*, have different POS possibilities (*adjective* vs. *noun-adjective*) and meanings. Derivational processes may induce pronunciation change or stress shift (e.g., *electric* vs. *electricity*). In English, typical derivational affixes (pieces of words) that are highly productive include prefixes and suffixes: *re-, pre-, -ial, -ism, -ish, -ity, -tion, -ness, -ment, -ious, -ify, -ize*, and others. In many cases, these can be added successively to create a complex layered form.

Table 2.15 Examples of stems and their related forms across POS categories.

Noun	Verb	Adjective	Adverb
criticism	*criticize*	*critical*	*critically*
fool	*fool*	*foolish*	*foolishly*
industry, industrialization	*industrialize*	*industrial,industrious*	*industriously*
employ, employee, employer	*employ*	*employable*	*employably*
certification	*certify*	*certifiable*	*certifiably*

Generally, word formation operates in layers, according to a kind of word syntax: *(deriv-prefix)* **root** (root)* (deriv-suffix)* (infl-suffix)*. This means that one or more *roots* can be compounded in the inner layer, with one or more optional *derivational prefixes*, followed by any number of optional *derivational suffixes*, capped off with no more than one *inflectional suffix*. There are, of course, limits on word formation, deriving both from semantics of the component words and simple lack of imagination. An example of a nearly maximal word in English might be *autocyberconceptualizations*, meaning (perhaps!) multiple instances of automatically creating computer-related concepts. This word lacks only compounding to be truly maximal. This word has a derivational prefix *auto-*, two root forms compounded (*cyber* and *concept*, though some may prefer to analyze *cyber-* as a prefix), three derivational suffixes (*-ual, -ize, -ation*), and is capped off with the plural inflectional suffix for nouns, *-s*.

2.3.2.3. Word Classes

POS classes are based on traditional grammatical and lexical analysis. With improved computational resources, it has become possible to examine words in context and assign words to groups according to their actual behavior in real text and speech from a statistical point of view. These kinds of classifications can be used in language modeling experiments for speech recognition, text analysis for text-to-speech synthesis, and other purposes.

One of the main advantages of word classification is its potential to derive more refined classes than traditional POS, while only rarely actually crossing traditional POS group boundaries. Such a system may group words automatically according to the similarity of usage with respect to their word neighbors. Consider classes automatically found by the classification algorithms of Brown *et al.* [7]:

{Friday Monday Thursday Wednesday Tuesday Saturday Sunday weekends}
{great big vast sudden mere sheer gigantic lifelong scant colossal}
{down backwards ashore sideways southward northward overboard aloft adrift}
{mother wife father son husband brother daughter sister boss uncle}
{John George James Bob Robert Paul William Jim David Mike}
{feet miles pounds degrees inches barrels tons acres meters bytes}

You can see that words are grouped together based on the semantic meaning, which is different from word classes created purely from syntactic point of view. Other types of classification are also possible, some of which can identify semantic relatedness across traditional POS categories. Some of the groups derived from this approach may include follows:

{problems problem solution solve analyzed solved solving}
{write writes writing written wrote pen}
{question questions asking answer answers answering}
{published publication author publish writer titled}

2.4. SYNTAX AND SEMANTICS

Syntax is the study of the patterns of formation of sentences and phrases from words and the rules for the formation of grammatical sentences. Semantics is another branch of linguistics dealing with the study of meaning, including the ways meaning is structured in language and changes in meaning and form over time.

2.4.1. Syntactic Constituents

Constituents represent the way a sentence can be divided into its grammatical subparts as constrained by common grammatical patterns (which implicitly incorporate normative judgments on acceptability). Syntactic constituents at least respect, and at best explain, the linear order of words in utterances and text. In this discussion, we will not strictly follow any of the many theories of syntax but will instead bring out a few basic ideas common to many approaches. We will not attempt anything like a complete presentation of the grammar of English but instead focus on a few simple phenomena.

Most work in syntactic theory has adopted machinery from traditional grammatical work on written language. Rather than analyze toy sentences, let's consider what kinds of superficial syntactic patterns are lurking in a random chunk of serious English text, excerpted from David Thoreau's essay *Civil Disobedience* [43]:

The authority of government, even such as I am willing to submit to – for I will cheerfully obey those who know and can do better than I, and in many things even those who neither know nor can do so well – is still an impure one: to be strictly just, it must have the sanction and consent of the governed. It can have no pure right over my person and property but what I concede to it. The progress from an absolute to a limited monarchy, from a limited monarchy to a democracy, is a progress toward a true respect for the individual.

2.4.1.1. Phrase Schemata

Words may be combined to form phrases that have internal structure and unity. We use generalized schemata to describe the phrase structure. The goal is to create a simple, uniform template that is independent of POS category.

Let's first consider nouns, a fundamental category referring to persons, places, and things in the world. The noun and its immediate modifiers form a constituent called the noun phrase (*NP*). To generalize this, we consider a word of arbitrary category, say category X (which could be a noun N or a verb V). The generalized rule for a phrase XP is $XP \Rightarrow$ *(modifiers) X-head (post-modifiers),* where X is the head, since it dominates the configuration and names the phrase. Elements preceding the head in its phrase are *premodifiers* and elements following the head are *postmodifiers*. XP, the culminating phrase node, is called a *maximal projection* of category X. We call the whole structure an *x-template*. Maximal projections, XP, are the primary currency of basic syntactic processes. The post-modifiers are usually maximal projections (another head, with its own post-modifiers forming an XP on its own) and are sometimes termed *complements*, because they are often required by the lexical properties of the head for a complete meaning to be expressed (e.g., when X is a preposition

or verb). Complements are typically noun phrases (*NP*), prepositional phrases (*PP*), verb phrases (*VP*), or sentence/clause (*S*), which make an essential contribution to the head's reference or meaning, and which the head requires for semantic completeness. Premodifiers are likely to be adverbs, adjectives, quantifiers, and determiners, i.e., words that help to specify the meaning of the head but may not be essential for completing the meaning. With minor variations, the *XP* template serves for most phrasal types, based on the POS of the head (*N, V, ADJ*, etc.).

For *NP*, we thus have *NP* $\Rightarrow$ *(det) (modifier)* ***head-noun*** *(post-modifier)*. This rule describes an *NP* (noun phrase – left side of arrow) in terms of its optional and required internal contents (right side of the arrow). *Det* is a word like *the* or *a* that helps to resolve the reference to a specific or an unknown instance of the noun. The *modifier* gives further information about the noun. The *head* of the phrase, and the only mandatory element, is the noun itself. *Post-modifiers* also give further information, usually in a more elaborate syntactic form than the simpler pre-modifiers, such as a relative clause or a prepositional phrase (covered below). The noun phrases of the passage above can be parsed as shown in Table 2.16. The head nouns may be personal pronouns (*I, it*), demonstrative and relative pronouns (*those*), coordinated nouns (*sanction and consent*), or common nouns (*individual*). The modifiers are mostly adjectives (*impure, pure*) or verbal forms functioning as adjectives (*limited*). The post-modifiers are interesting, in that, unlike the (pre-)modifiers, they are typically full phrases themselves, rather than isolated words. They include relative clauses (which are a kind of dependent sentence, e.g., *[those] who know and can do better than I*), as well as prepositional phrases (*of the governed*).

Table 2.16 *NPs* of the sample passage.

NP	Det	Mod	Head Noun	Post-Mod
1	the		**authority**	of government
2		even	**such**	as I am willing to submit to
3			**I**	
4			**those**	who know and can do better than I
5		many	**things**	
6		even	**those**	who neither know nor can do so well
7	an	impure	**one**	
8			**it**	
9	the		**sanction and consent**	of the governed
10	no	pure	**right**	over my person ... concede to it.
11	the		**progress**	from an absolute to a limited monarchy
12	an	absolute	**[monarchy]**	
13	a	limited	**monarchy**	
14	a		**democracy**	
15	a		**progress**	
16	a	true	**respect**	for the individual
17	the		**individual**	

Table 2.17 *PPs* of the sample passage.

Head Prep	Complement (Postmodifier)
of	Government
as	I am willing to submit to
than	I
in	many things
of	the governed
over	my person and property
to	it
from	an absolute [monarchy]
to	a limited monarchy
to	a democracy
toward	a true respect [for the individual]
for	the individual

Prepositions express spatial and temporal relations, among others. These are also said to project according to the *X*-template, but usually lack a pre-modifier. Some examples from the sample passage are listed in Table 2.17. The complements of *PP* are generally *NPs*, which may be simple head nouns like *government*. However, other complement types, such as the verb phrase in *after discussing it with Jo*, are also possible.

For verb phrases, the postmodifier (or complement) of a head verb would typically be one or more *NP* (noun phrase) maximal projections, which might, for example, function as a direct object in a *VP* like *pet the cat*. The complement may or may not be optional, depending on characteristics of the head. We can now make some language-specific generalizations about English. Some verbs, such as *give*, may take more than one kind of complement. So an appropriate template for a *VP* maximal projection in English would appear abstractly as *VP* $\Rightarrow$ *(modifier) verb (modifier) (Complement1, Complement2 ComplementN).* Complements are usually regarded as maximal projections, such as *NP*, *ADJP*, etc., and are enumerated in the template above, to cover possible multi-object verbs, such as *give*, which take both direct and indirect objects. Certain types of adverbs (*really*, *quickly*, *smoothly*, etc.) could be considered fillers for the *VP* modifier slots (before and after the head). In the sample passage, we find the following verb phrases as shown in Table 2.18.

VP presents some interesting issues. First, notice the multi-word verb *submit to*. Multi-word verbs such as *look after* and *put up with* are common. We also observe a number of auxiliary elements clustering before the verb in sentences of the sample passage: *am willing to submit to*, *will cheerfully obey*, and *can do better*. Rather than considering these as simple modifiers of the verbal head, they can be taken to have *scope* over the *VP* as a whole, which implies they are outside the *VP*. Since they are outside the *VP*, we can assume them to be heads in their own right, of phrases which require a *VP* as their complement. These elements mainly express tense (time or duration of verbal action) and modality (likelihood or probability of verbal action). In a full sentence, the *VP* has explicit or implicit inflection (projected from its verbal head) and indicates the person, number, and other context-dependent

features of the verb in relation to its arguments. In English, the person (first, second, third) and number (singular, plural) attributes, collectively called agreement features, of subject and verb must match. For simplicity, we will lump all these considerations together as inflectional elements, and posit yet another phrase type, the Inflectional Phrase (*IP*): *IP* ⇒ *premodifier head VP-complement.*

Table 2.18 *VP*s of the sample passage.

Pre-mod	Verb Head	Post-mod	Complement
	submit to		[the authority of government]
cheerfully	**obey**		those who know and can do better than I
	is	still	an impure one
	be		strictly just
	have		the sanction
	have		no pure right
	concede		to it
	is		a progress

The premodifier slot (sometimes called the *specifier* position in linguistic theory) of an *IP* is often filled by the subject of the sentence (typically a noun or *NP*). Since the *IP* unites the subject of a sentence with a *VP*, *IP* can also be considered simply as the sentence category, often written as *S* in speech grammars.

2.4.1.2. Clauses and Sentences

The *subject* of a sentence is what the sentence is mainly about. A *clause* is any phrase with both a subject and a *VP* (*predicate* in traditional grammars) that has potentially independent interpretation – thus, for us, a clause is an *IP*, a kind of sentence. A phrase is a constituent lacking either subject, predicate, or both. We have reviewed a number of phrase types above. There are also various types of clauses and sentences.

Even though clauses are sentences from an internal point of view (having subject and predicate), they often function as simpler phrases or words would, e.g., as modifiers (adjective and adverbs) or nouns and noun phrases. Clauses may appear as post-modifiers for nouns (so-called *relative clauses*), basically a kind of adjective clause, sharing their subjects with the containing sentence. Some clauses function as *NP*s in their own right. One common clause type substitutes a *wh-word* like *who* or *what* for a direct object of a verb in the embedded clause, to create a *questioned noun phrase* or indirect question: (*I don't know who Jo saw.*). In these clauses, it appears to syntacticians that the *questioned* object of the verb [*VP* saw **who**] has been extracted or moved to a new surface position (following the main clause verb *know*). This is sometimes shown in the phrase-structure diagram by co-indexing an empty *ghost* or trace constituent at the original position of the question pronoun with the question-*NP* appearing at the surface site:

I don't know $[_{NPobj}\,[_{IP}\,[_{NPi}$ *who] Jo saw* $[_{NPi}\,\_\,]]]$
$[_{NPsubj}[_{IP}$ *Whoever wins the game]] is our hero.*

There are various characteristic types of sentences. Some typical types include:

- Declarative: *I gave her a book.*
- Yes-no question: *Did you give her a book?*
- Wh-question: *What did you give her?*
- Alternatives question: *Did you give her a book, a scarf, or a knife?*
- Tag question: *You gave it to her, didn't you?*
- Passive: *She was given a book.*
- Cleft: *It must have been a book that she got.*
- Exclamative: *Hasn't this been a great birthday!*
- Imperative: *Give me the book.*

2.4.1.3. Parse Tree Representations

Sentences can be diagrammed in parse trees to indicate phrase-internal structure and linear precedence and immediate dominance among phrases. A typical phrase-structure tree for part of an embedded sentence is illustrated in Figure 2.26.

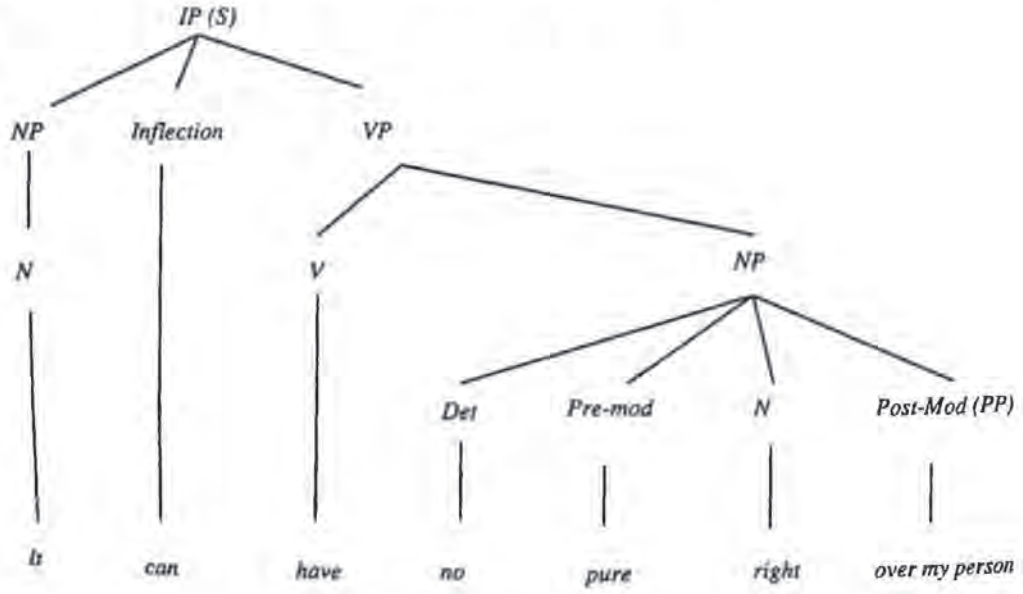


Figure 2.26 A simplified phrase-structure diagram.

For brevity, the same information illustrated in the tree can be represented as a bracketed string as follows:

$[_{IP}\,[_{NP}\,[_{N}\,It\,]_{N}]_{NP}\,[_{I}\,can\,]_{I}\,[_{VP}[_{V}\,have\,]_{V}\,[_{NP}\,no\ pure\ right\ [_{PP}\,over\ my\ person\,]_{PP}]_{NP}]_{VP}]_{IP}$

With such a bracketed representation, almost every type of syntactic constituent can be coordinated or joined with another of its type, and usually a new phrase node of the common type is added to subsume the constituents such as *NP: We have* $[_{NP}\,[_{NP}$ *tasty berries] and* $[_{NP}$ *tart juices]], IP/S:* $[_{IP}\,[_{IP}$ *Many have come] and* $[_{IP}$ *most have remained]], PP: We went* $[_{PP}\,[_{PP}$ *over the river] and* $[_{PP}$ *into the trees]],* and *VP: We want to* $[_{VP}\,[_{VP}$ *climb the mountains] and* $[_{VP}$ *sail the seas]].*

2.4.2. Semantic Roles

In traditional syntax, grammatical roles are used to describe the direction or control of action relative to the verb in a sentence. Examples include the ideas of *subject, object, indirect object*, etc. Semantic roles, sometimes called case relations, seem similar but dig deeper. They are used to make sense of the participants in an event, and they provide a vocabulary for us to answer the basic question *who did what to whom.* As developed by [13] and others, the theory of semantic roles posits a limited number of universal roles. Each basic meaning of each verb in our mental dictionary is tagged for the obligatory and optional semantic roles used to convey the particular meaning. A typical inventory of case roles is given below:

Agent	*cause or initiator of action, often intentional*
Patient/Theme	*undergoer of the action*
Instrument	*how action is accomplished*
Goal	*to whom action is directed*
Result	*result of action*
Location	*location of action*

These can be realized under various syntactic identities, and can be assigned to both required complement and optional adjuncts. A noun phrase in the Agentive role might be the surface subject of a sentence, or the object of the preposition *by* in a passive. For example, the verb *put* can be considered a process that has, in one of its senses, the case role specifications shown in Table 2.19.

Table 2.19 Analysis of a sentence with *put.*

Analysis	**Example**			
	Kim	*put*	*the book*	*on the table.*
Grammatical functions	*Subject (NP)*	*Predicate (VP)*	*Object (NP)*	*Adverbial (ADVP)*
Semantic roles	*Agent*	*Instrument*	*Theme*	*Location*

Now consider this passive-tense example, where the semantic roles align with different grammatical roles shown in Table 2.20. Words that look and sound identical can have different meaning or different *senses* as shown in Table 2.21. The sporting sense of *put* (as in the sport of shot-put) illustrates the meaning/sense-dependent nature of the role patterns, because in this sense the Locative case is no longer obligatory, as it is in the original sense illustrated in Table 2.19 and Table 2.20.

Table 2.20 Analysis of passive sentence with *put*.

Analysis	**Example**		
	The book	*was put*	*on the table.*
Grammatical functions	*Subject (NP)*	*Predicate (VP)*	*Adverbial (ADVP)*
Semantic roles	*Agent*	*Instrument*	*Location*

Table 2.21 Analysis of a different pattern of *put*.

Analysis	**Example**		
	Kim	*put*	*the shot.*
Grammatical functions	*Subject (NP)*	*Predicate (VP)*	*Object (NP)*
Semantic roles	*Agent*	*Instrument*	*Theme*

The lexical meaning of a verb can be further decomposed into primitive semantic relations such as CAUSE, CHANGE, and BE. The verb *open* might appear as *CAUSE(NP1,PHYSICAL-CHANGE(NP2,NOT-OPEN,OPEN))*. This says that for an agent (*NP1*) to *open* a theme (*NP2*) is to cause the patient to change from a not-opened state to an opened state. Such systems can be arbitrarily detailed and exhaustive, as the application requires.

2.4.3. Lexical Semantics

The specification of particular meaning templates for individual senses of particular words is called *lexical semantics*. When words combine, they may take on propositional meanings resulting from the composition of their meanings in isolation. We could imagine that a speaker starts with a proposition in mind (logical form as will be discussed in the next section), creating a need for particular words to express the idea (lexical semantics); the proposition is then linearized (syntactic form) and spoken (phonological/phonetic form). Lexical semantics is the level of meaning before words are composed into phrases and sentences, and it may heavily influence the possibilities for combination.

Words can be defined in a large number of ways including by relations to other words, in terms of decomposition semantic primitives, and in terms of non-linguistic cognitive constructs, such as perception, action, and emotion. There are hierarchical and non-hierarchical relations. The main hierarchical relations would be familiar to most object-oriented programmers. One is *is-a* taxonomies (a *crow* is-a *bird*), which have transitivity of properties

from type to subtype (inheritance). Another is *has-a* relations (a *car* has-a *windshield*), which are of several differing qualities, including process/subprocess (*teaching* has-a subprocess *giving exams*), and arbitrary or natural subdivisions of part-whole relations (*bread* has-a division into *slices*, *meter* has-a division into *centimeters*). Then there are non-branching hierarchies (no fancy name) that essentially form scales of degree, such as *frozen* $\Rightarrow$ *cold* $\Rightarrow$ *lukewarm* $\Rightarrow$ *hot* $\Rightarrow$ *burning*. Non-hierarchical relations include synonyms, such as *big/large*, and antonyms such as *good/bad*.

Words seem to have natural affinities and disaffinities in the semantic relations among the concepts they express. Because these affinities could potentially be exploited by future language understanding systems, researchers have used the generalizations above in an attempt to tease out a parsimonious and specific set of basic relations under which to group entire lexicons of words. A comprehensive listing of the families and subtypes of possible semantic relations has been presented in [10]. In Table 2.22, the leftmost column shows names for families of proposed relations, the middle column differentiates subtypes within each family, and the rightmost column provides examples of word pairs that participate in the proposed relation. Note that case roles have been modified for inclusion as a type of semantic relation within the lexicon.

We can see from Table 2.22 that a single word could participate in multiple relations of different kinds. For example, *knife* appears in the examples for *Similars: invited attribute* (i.e., a desired and expected property) as: *knife-sharp*, and also under *Case Relations*: *action-instrument*, which would label the relation of *knife* to the action *cut* in *He cut the bread with a knife*. This suggests that an entire lexicon could be viewed as a graph of semantic relations, with words or idioms as nodes and connecting edges between them representing semantic relations as listed above. There is a rich tradition of research in this vein.

The biggest practical problem of lexical semantics is the context-dependent resolution of senses of words – so-called polysemy. A classic example is *bank – bank of the stream* as opposed to *money in the bank*. While lexicographers try to identify distinct senses when they write dictionary entries, it has been generally difficult to rigorously quantify exactly what counts as a discrete sense of a word and to disambiguate the senses in practical contexts. Therefore, designers of practical speech understanding systems generally avoid the problem by limiting the domain of discourse. For example, in a financial application, generally only the sense of *bank* as a fiduciary institution is accessible, and others are assumed not to exist. It is sometimes difficult to make a principled argument as to how many distinct senses a word has, because at some level of depth and abstraction, what might appears as separate senses seem to be similar or related, as *face* could be *face of a clock* or *face of person*.

Senses are usually distinguished within a given part-of-speech (POS) category. Thus, when an occurrence of *bank* has been identified as a verb, the *shore* sense might be automatically eliminated, though depending on the sophistication of the system's lexicon and goals, there can be sense differences for many English verbs as well. Within a POS category, often the words that occur near a given ambiguous form in the utterance or discourse are clues to interpretation, where links can be established using semantic relations as described above. Mutual information measures as discussed in Chapter 3 can sometimes provide hints. In a context of dialog where other, less ambiguous financial terms come up

frequently, the sense of *bank* as fiduciary institution is more likely. Finally, when all else fails, often senses can be ranked in terms of their a priori likelihood of occurrence. It should always be borne in mind that language is not static; it can change form under a given analysis at any time. For example, the stable English form *spinster*, a somewhat pejorative term for an older, never-married female, has recently taken on a new morphologically complex form, with the new sense of a high political official, or media spokesperson, employed to provide bland disinformation (*spin*) on a given topic.

Table 2.22 Semantic relations.

Family	**Subtype**	**Example**
Contrasts	Contrary	*old-young*
	Contradictory	*alive-dead*
	Reverse	*buy-sell*
	Directional	*front-back*
	Incompatible	*happy-morbid*
	Asymmetric contrary	*hot-cool*
	Attribute similar	*rake-fork*
Similars	Synonymity	*car-auto*
	Dimensional similar	*smile-laugh*
	Necessary attribute	*bachelor-unmarried*
	Invited attribute	*knife-sharp*
	Action subordinate	*talk-lecture*
Class Inclusion	Perceptual subord.	*animal-horse*
	Functional subord.	*furniture-chair*
	State subord.	*disease-polio*
	Activity subord.	*game-chess*
	Geographic subord.	*country-Russia*
	Place	*Germany-Hamburg*
Case Relations	Agent-action	*artist-paint*
	Agent-instrument	*farmer-tractor*
	Agent-object	*baker-bread*
	Action-recipient	*sit-chair*
	Action-instrument	*cut-knife*
Part-Whole	Functional object	*engine-car*
	Collection	*forest-tree*
	Group	*choir-singer*
	Ingredient	*table-wood*
	Functional location	*kitchen-stove*
	Organization	*college-admissions*
	Measure	*mile-yard*

2.4.4. Logical Form

Because of all the lexical, syntactic, and semantic ambiguity in language, some of which requires external context for resolution, it is desirable to have a metalanguage in which to concretely and succinctly express all linguistically possible meanings of an utterance before discourse and world knowledge are applied to choose the most likely interpretation. The favored metalanguage for this purpose is called the predicate logic, used to represent the logical form, or context-independent meaning, of an utterance. The semantic component of many SLU architectures builds on a substrate of two-valued, first-order, logic. To distinguish *shades of meaning* beyond truth and falsity requires more powerful formalisms for knowledge representation.

In a typical first-order system, predicates correspond to events or conditions denoted by verbs (such as *Believe* or *Like*), states of identity (such as being a *Dog* or *Cat*), and properties of varying degrees of permanence (*Happy*). In this form of logical notation, predicates have open places, filled by arguments, as in a programming language subroutine definition. Since individuals may have identical names, subscripting can be used to preserve unique reference. In the simplest systems, predication ranges over individuals rather than higher-order entities such as properties and relations.

Predicates with filled argument slots map onto sets of individuals (constants) in the universe of discourse, in particular those individuals possessing the properties, or participating in the relation, named by the predicate. One-place predicates like *Soldier*, *Happy*, or *Sleeps* range over sets of individuals from the universe of discourse. Two-place predicates, like transitive verbs such as *loves*, range over a set consisting of ordered pairs of individual members (constants) of the universe of discourse. For example, we can consider the universe of discourse to be U = {*Romeo, Juliet, Paris, Rosaline, Tybalt*}, people as characters in a play. They do things with and to one another, such as loving and killing. Then we could imagine the relation *Loves* interpreted as the set of ordered pairs: {<*Romeo, Juliet*>, <*Juliet, Romeo*>, <*Tybalt, Tybalt*>, <*Paris, Juliet*>}, a subset of the Cartesian product of theoretically possible love matches $U \times U$. So, for any ordered pair x, y in U, *Loves*(x, y) is true if the ordered pair <x,y> is a member of the extension of the *Loves* predicate as defined, e.g., *Romeo loves Juliet, Juliet loves Romeo*, etc.. Typical formal properties of relations are sometimes specially marked by grammar, such as the reflexive relation *Loves*(*Tybalt, Tybalt*), which can rendered in natural language as *Tybalt loves himself*. Not every possibility is present; for instance in our example, the individual Rosaline does not happen to participate at all in this extensional definition of *Loves* over U, as her omission from the pairs list indicates. Notice that the subset of *Loves*(x, y) of ordered pairs involving both *Romeo* and *Juliet* is symmetric, also marked by grammar, as in *Romeo and Juliet love each other*. This general approach extends to predicates with any arbitrary number of arguments, such as intransitive verbs like *give*.

Just as in ordinary propositional logic, connectives such as negation, conjunction, disjunction, and entailment are admitted, and can be used with predicates to denote common natural language meanings:

Romeo isn't happy = ¬Happy(Romeo)
Romeo isn't happy, but Tybalt is (happy) = ¬Happy(Romeo) ∧ Happy(Tybalt)
Either Romeo or Tybalt is happy = Happy(Romeo) ∨ Happy(Tybalt)
If Romeo is happy, Juliet is happy = Happy(Romeo) → Happy(Juliet)

Formulae, such as those above, are also said to bear a binary truth value, true or false, with respect to a world of individuals and relations. The determination of the truth value is compositional, in the sense that the truth value of the whole depends on the truth value of the parts. This is a simplistic but formally tractable view of the relation between language and meaning.

Predicate logic can also be used to denote quantified noun phrases. Consider a simple case such as *Someone killed Tybalt*, predicated over our same U = {*Romeo, Juliet, Paris, Rosaline, Tybalt*}. We can now add an *existential* quantifier, ∃, standing for *there exists* or *there is at least one*. This quantifier will bind a variable over individuals in U, and will attach to a proposition to create a new, quantified proposition in logical form. The use of variables in propositions such as *killed(x, y)* creates open propositions. Binding the variables with a quantifier over them closes the proposition. The quantifier is prefixed to the original proposition: ∃*x Killed(x, Tybalt)*.

To establish a truth (semantic) value for the quantified proposition, we have to satisfy the disjunction of propositions in U: *Killed(Romeo, Tybalt)* ∨ *Killed(Juliet, Tybalt)* ∨ *Killed(Paris, Tybalt)* ∨ *Killed(Rosaline, Tybalt)* ∨ *Killed(Tybalt, Tybalt)*. The set of all such bindings of the variable x is the space that determines the truth or falsity of the proposition. In this case, the binding of x = *Romeo* is sufficient to assign a value true to the existential proposition.

2.5. Historical Perspective and Further Reading

Motivated to improve speech quality over the telephone, AT&T Bell Labs has contributed many influential discoveries in speech hearing, including the critical band and articulation index [2, 3]. The *Auditory Demonstration* CD prepared by Houtsma, Rossing, and Wagenaars [18] has a number of very interesting examples on psychoacoustics and its explanations. *Speech, Language, and Communication* [30] and *Speech Communication – Human and Machine* [32] are two good books that provide modern introductions to the structure of spoken language. Many speech perception experiments were conducted by exploring how phonetic information is distributed in the time or frequency domain. In addition to the formant structures for vowels, frequency importance function [12] has been developed to study how features related to phonetic categories are stored at various frequencies. In the time domain, it has been observed [16, 19, 42] that salient perceptual cues may not be evenly distributed over the speech segments and that certain perceptual critical points exist.

As intimate as speech and acoustic perception may be, there are also strong evidences that lexical and linguistic effects on speech perception are not always consistent with acoustic ones. For instance, it has long been observed that humans exhibit difficulties in distinguishing non-native phonemes. Human subjects also carry out categorical goodness

difference assimilation based on their mother tongue [34], and such perceptual mechanism can be observed as early as in six-month-old infants [22]. On the other hand, hearing-impaired listeners are able to effortlessly overcome their acoustical disabilities for speech perception [8]. Speech perception is not simply an auditory matter. McGurk and MacDonald (1976) [27, 28] dramatically demonstrated this when they created a videotape on which the auditory information (phonemes) did not match the visual speech information. The effect of this mismatch between the auditory signal and the visual signal was to create a third phoneme different from both the original auditory and visual speech signals. An example is dubbing the phoneme */ba/* to the visual speech movements */ga/*. This mismatch results in hearing the phoneme */da/*. Even when subjects know of the effect, they report the McGurk effect percept. The McGurk effect has been demonstrated for consonants, vowels, words, and sentences.

The earliest scientific work on phonology and grammars goes back to Panini, a Sanskrit grammarian of the fifth century B.C. (estimated), who created a comprehensive and scientific theory of phonetics, phonology, and morphology, based on data from Sanskrit (the classical literary language of the ancient Hindus). Panini created formal production rules and definitions to describe Sanskrit grammar, including phenomena such as construction of sentences, compound nouns, etc. Panini's formalisms function as ordered rules operating on underlying structures in a manner analogous to modern linguistic theory. Panini's phonological rules are equivalent in formal power to Backus-Nauer form (BNF). A general introduction to this pioneering scientist is Cardona [9].

An excellent introduction to all aspects of phonetics is *A Course in Phonetics* [24]. A good treatment of the acoustic structure of English speech sounds and a through introduction and comparison of theories of speech perception is to be found in [33]. The basics of phonology as part of linguistic theory are treated in *Understanding Phonology* [17]. An interesting treatment of word structure (morphology) from a computational point of view can be found in *Morphology and Computation* [40]. A comprehensive yet readable treatment of English syntax and grammar can be found in *English Syntax* [4] and *A Comprehensive Grammar of the English Language* [36]. Syntactic theory has traditionally been the heart of linguistics, and has been an exciting and controversial area of research since the 1950s. Be aware that almost any work in this area will adopt and promote a particular viewpoint, often to the exclusion or minimization of others. A reasonable place to begin with syntactic theory is *Syntax: A Minimalist Introduction* [37]. An introductory textbook on syntactic and semantic theory that smoothly introduces computational issues is *Syntactic Theory: A Formal Introduction* [39]. For a philosophical and entertaining overview of various aspects of linguistic theory, see *Rhyme and Reason: An Introduction to Minimalist Syntax* [44]. A good and fairly concise treatment of basic semantics is *Introduction to Natural Language Semantics* [11]. Deeper issues are covered in greater detail and at a more advanced level in *The Handbook of Contemporary Semantic Theory* [25]. The intriguing area of lexical semantics (theory of word meanings) is comprehensively presented in *The Generative Lexicon* [35]. *Concise History of the Language Sciences* [21] is a good edited book if you are interested in the history of linguistics.

REFERENCES

[1] Aliprand, J., *et al.*, *The Unicode Standard, Version 2.0*, 1996, Addison Wesley.

[2] Allen, J.B., "How Do Humans Process and Recognize Speech?," *IEEE Trans. on Speech and Audio Processing*, 1994, **2**(4), pp. 567-577.

[3] Allen, J.B., "Harvey Fletcher 1884–1981" in *The ASA Edition of Speech and Hearing Communication* 1995, Woodbury, New York, pp. A1-A34, Acoustical Society of America.

[4] Baker, C.L., *English Syntax*, 1995, Cambridge, MA, MIT Press.

[5] Blauert, J., *Spatial Hearing*, 1983, MIT Press.

[6] Brill, E., "Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part-of-Speech Tagging," *Computational Linguistics*, 1995, **21**(4), pp. 543-566.

[7] Brown, P., *et al.*, "Class-Based N-gram Models of Natural Language," *Computational Linguistics*, 1992, **18**(4).

[8] Caplan, D. and J. Utman, "Selective Acoustic Phonetic Impairment and Lexical Access in an Aphasic Patient," *Journal of the Acoustical Society of America*, 1994, **95**(1), pp. 512-517.

[9] Cardona, G., *Panini: His Work and Its Traditions: Background and Introduction*, 1988, Motilal Banarsidass.

[10] Chaffin, R., and Herrmann, D., "The Nature of Semantic Relations: A Comparison of Two Approaches" in *Representing Knowledge in Semantic Networks*, M. Evens, ed., 1988, Cambridge, UK, Cambridge University Press.

[11] de Swart, H., *Introduction to Natural Language Semantics*, 1998, Stanford, CA, Center for the Study of Language and Information Publications.

[12] Duggirala, V., *et al.*, "Frequency Importance Function for a Feature Recognition Test Material," *Journal of the Acoustical Society of America*, 1988, **83**(9), pp. 2372-2382.

[13] Fillmore, C.J., "The Case for Case" in *Universals in Linguistic Theory*, E. Bach and R. Harms, eds. 1968, New York, NY, Holt, Rinehart and Winston.

[14] Fletcher, H., "Auditory Patterns," *Rev. Mod. Phys.*, 1940, **12**, pp. 47-65.

[15] Fry, D.B., *The Physics of Speech*, Cambridge Textbooks in Linguistics, 1979, Cambridge, U.K., Cambridge University Press.

[16] Furui, S., "On The Role of Spectral Transition for Speech Perception," *Journal of the Acoustical Society of America*, 1986, **80**(4), pp. 1016-1025.

[17] Gussenhoven, C., and Jacobs, H., *Understanding Phonology*, Understanding Language Series, 1998, Edward Arnold.

[18] Houtsma, A., T. Rossing, and W. Wagenaars, *Auditory Demonstrations*, 1987, Institute for Perception Research, Eindhovern, The Netherlands, Acoustic Society of America.

[19] Jenkins, J., W. Strange, and S. Miranda, "Vowel Identification in Mixed-Speaker Silent-Center Syllables," *Journal of the Acoustical Society of America*, 1994, **95**(2), pp. 1030-1041.

[20] Klatt, D., "Review of the ARPA Speech Understanding Project," *Journal of Acoustical Society of America*, 1977, **62**(6), pp. 1324-1366.

[21] Koerner, E. and E. Asher, eds. *Concise History of the Language Sciences*, 1995, Oxford, Elsevier Science.

[22] Kuhl, P., "Infant's Perception and Representation of Speech: Development of a New Theory," *Int. Conf. on Spoken Language Processing*, 1992, Alberta, Canada, pp. 449-452.

[23] Kupeic, J., "Robust Part-of-Speech Tagging Using a Hidden Markov Model," *Computer Speech and Language*, 1992, **6**, pp. 225-242.

[24] Ladefoged, P., *A Course in Phonetics*, 1993, Harcourt Brace Johanovich.

[25] Lappin, S., *The Handbook of Contemporary Semantic Theory*, Blackwell Handbooks in Linguistics, 1997, Oxford, UK, Blackwell Publishsers Inc.

[26] Lindsey, P. and D. Norman, *Human Information Processing*, 1972, New York and London, Academic Press.

[27] MacDonald, J. and H. McGurk, "Visual Influence on Speech Perception Process," *Perception and Psychophysics*, 1978, **24**(3), pp. 253-257.

[28] McGurk, H. and J. MacDonald, "Hearing Lips and Seeing Voices," *Nature*, 1976, **264**, pp. 746-748.

[29] Merialdo, B., "Tagging English Text with a Probabilistic Model," *Computational Linguistics*, 1994, **20**(2), pp. 155-172.

[30] Miller, J. and P. Eimas, *Speech, Language and Communication*, Handbook of Perception and Cognition, eds. E. Carterette and M. Friedman, 1995, Academic Press.

[31] Moore, B.C., *An Introduction to the Psychology of Hearing*, 1982, London, Academic Press.

[32] O'Shaughnessy, D., *Speech Communication – Human and Machine*, 1987, Addison-Wesley.

[33] Pickett, J.M., *The Acoustics of Speech Communication*, 1999, Needham Heights, MA, Allyn & Bacon.

[34] Polka, L., "Linguistic Influences in Adult Perception of Non-native Vowel Contrast," *Journal of the Acoustical Society of America*, 1995, **97**(2), pp. 1286-1296.

[35] Pustejovsky, J., *The Generative Lexicon*, 1998, Bradford Books.

[36] Quirk, R., Svartvik, J., Leech, G., *A Comprehensive Grammar of the English Language*, 1985, Addison-Wesley Pub. Co.

[37] Radford, A., *Syntax: A Minimalist Introduction*, 1997, Cambridge, U.K., Cambridge Univ. Press.

[38] Rossing, T.D., *The Science of Sound*, 1982, Reading, MA, Addison-Wesley.

[39] Sag, I., Wasow, T., *Syntactic Theory: A Formal Introduction*, 1999, Cambridge, UK, Cambridge University Press.

[40] Sproat, R., *Morphology and Computation*, ACL-MIT Press Series in Natural Language Processing, 1992, Cambridge, MA, MIT Press.

[41] Stevens, S.S. and J. Volkman, "The Relation of Pitch to Frequency," *Journal of Psychology*, 1940, **53**, pp. 329.

[42] Strange, W., J. Jenkins, and T. Johnson, "Dynamic Specification of Coarticulated Vowels," *Journal of the Acoustical Society of America*, 1983, 74(3), pp. 695-705.

[43] Thoreau, H.D., *Civil Disobedience, Solitude and Life Without Principle*, 1998, Prometheus Books.

[44] Uriagereka, J., *Rhyme and Reason: An Introduction to Minimalist Syntax*, 1998, Cambridge, MA, MIT Press.

[45] Voutilainen, A., "Morphological Disambiguation" in *Constraint Grammar: A Language-Independent System for Parsing Unrestricted Text* 1995, Berlin, Mouton de Gruyter.

[46] Weischedel, R., "BBN: Description of the PLUM System as Used for MUC-6," *The 6th Message Understanding Conferences (MUC-6)*, 1995, San Francisco, Morgan Kaufmann, pp. 55-70.

CHAPTER 3

Probability, Statistics, and Information Theory

Randomness and uncertainty play an important role in science and engineering. Most spoken language processing problems can be characterized in a probabilistic framework. Probability theory and statistics provide the mathematical language to describe and analyze such systems.

The criteria and methods used to estimate the unknown probabilities and probability densities form the basis for estimation theory. Estimation theory is critical to parameter learning in pattern recognition. In this chapter, three widely used estimation methods are discussed. They are *minimum mean squared error estimation* (MMSE), *maximum likelihood estimation* (MLE), and *maximum posterior probability estimation* (MAP).

Significance testing deals with the confidence of statistical inference, such as knowing whether the estimation of some parameter can be accepted with confidence. In pattern recognition, significance testing is important for determining whether the observed difference between two different classifiers is real. In our coverage of significance testing, we describe various methods that are used in pattern recognition, discussed in Chapter 4.

Information theory was originally developed for efficient and reliable communication systems. It has evolved into a mathematical theory concerned with the very essence of the communication process. It provides a framework for the study of fundamental issues, such as the efficiency of information representation and the limitations in reliable transmission of information over a communication channel. Many of these problems are fundamental to spoken language processing.

3.1. PROBABILITY THEORY

Probability theory deals with the averages of mass phenomena occurring sequentially or simultaneously. We often use probabilistic expressions in our day-to-day lives, such as when saying, *It is very likely that the Dow (Dow Jones Industrial index) will hit 12,000 points next month*, or, *The chance of scattered showers in Seattle this weekend is high.* Each of these expressions is based upon the concept of the probability, or the likelihood, that some specific event will occur.

Probability can be used to represent the degree of confidence in the outcome of some actions (observations), which are not definite. In probability theory, the term *sample space, S,* is used to refer to the collection (set) of all possible outcomes. An *event* refers to a subset of the sample space or a collection of outcomes. The *probability of event A,* denoted as $P(A)$, can be interpreted as the *relative frequency* with which event A would occur if the process were repeated a large number of times under similar conditions. Based on this interpretation, $P(A)$ can be computed simply by counting the total number, N_S, of all observations and the number of observations N_A whose outcome belongs to event A. That is,

$$P(A) = \frac{N_A}{N_S} \tag{3.1}$$

$P(A)$ is bounded between zero and one, i.e.,

$$0 \le P(A) \le 1 \text{ for all } A \tag{3.2}$$

The lower bound of probability $P(A)$ is zero when the event set A is an empty set. On the other hand, the upper bound of probability $P(A)$ is one when the event set A happens to be S.

If there are n events $A_1, A_2, \cdots, A_n$ in S such that $A_1, A_2, \cdots, A_n$ are disjoint and $\bigcup_{i=1}^{n} A_i = S$, events $A_1, A_2, \cdots, A_n$ are said to form a *partition* of S. The following obvious equation forms a fundamental axiom for probability theory.

$$P(A_1 \cup A_2 \cup \ldots A_n) = \sum_{i=1}^{n} P(A_i) = 1 \tag{3.3}$$

Based on the definition in Eq. (3.1), the *joint probability* of event A and event B occurring concurrently is denoted as $P(AB)$ and can be calculated as:

$$P(AB) = \frac{N_{AB}}{N_S} \tag{3.4}$$

3.1.1. Conditional Probability and Bayes' Rule

It is useful to study the way in which the probability of an event A changes after it has been learned that some other event B has occurred. This new probability denoted as $P(A \mid B)$ is called the *conditional probability* of event A given that event B has occurred. Since the set of those outcomes in B that also result in the occurrence of A is exactly the set AB as illustrated in Figure 3.1, it is natural to define the conditional probability as the proportion of the total probability $P(B)$ that is represented by the joint probability $P(AB)$. This leads to the following definition:

$$P(A \mid B) = \frac{P(AB)}{P(B)} = \frac{N_{AB}/N_S}{N_B/N_S} \tag{3.5}$$

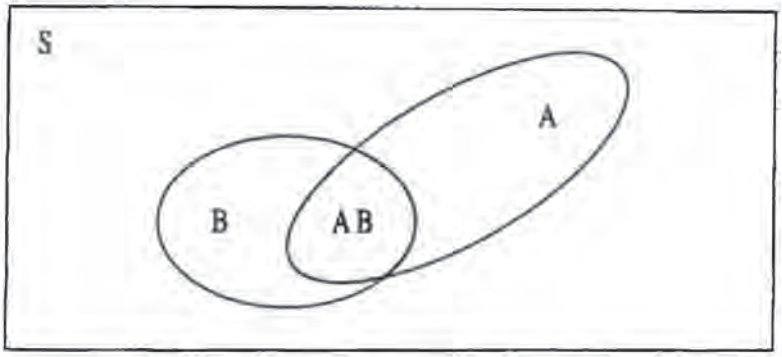


Figure 3.1 The intersection AB represents where the joint event A and B occurs concurrently.

Based on the definition of conditional probability, the following expressions can be easily derived.

$$P(AB) = P(A \mid B)P(B) = P(B \mid A)P(A) \tag{3.6}$$

Equation (3.6) is the simple version of the *chain rule*. The chain rule, which can specify a joint probability in terms of multiplication of several cascaded conditional probabilities, is often used to decompose a complicated joint probabilistic problem into a sequence of stepwise conditional probabilistic problems. Equation (3.6) can be converted to such a general chain:

$$P(A_1A_2 \cdots A_n) = P(A_n \mid A_1 \cdots A_{n-1}) \cdots P(A_2 \mid A_1)P(A_1) \tag{3.7}$$

When two events, A and B, are independent of each other, in the sense that the occurrence or of either of them has no relation to and no influence on the occurrence of the other, it is obvious that the conditional probability $P(B \mid A)$ equals to the unconditional probability

$P(B)$. It follows that the joint probability $P(AB)$ is simply the product of $P(A)$ and $P(B)$ if A and B, are independent.

If the n events $A_1, A_2, \cdots, A_n$ form a partition of S and B is any event in S as illustrated in Figure 3.2, the events $A_1B, A_2B, \cdots, A_nB$ form a partition of B. Thus, we can rewrite:

$$B = A_1B \cup A_2B \cup \cdots \cup A_nB \tag{3.8}$$

Since $A_1B, A_2B, \cdots, A_nB$ are disjoint,

$$P(B) = \sum_{k=1}^{n} P(A_kB) \tag{3.9}$$

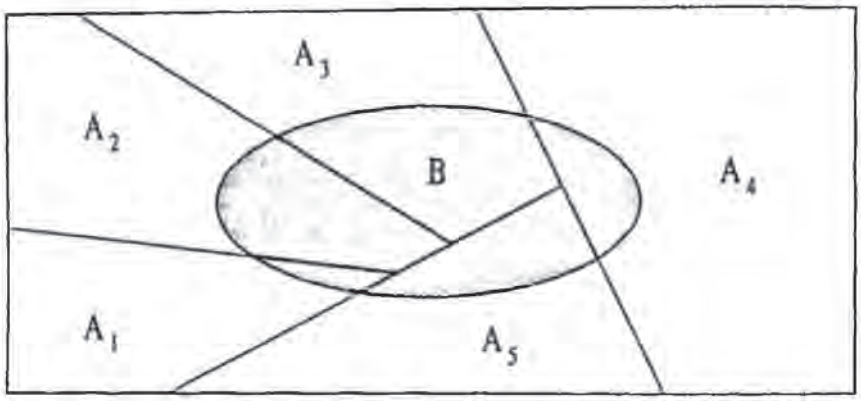


Figure 3.2 The intersections of B with partition events $A_1, A_2, \cdots, A_n$.

Equation (3.9) is called the *marginal probability* of event B, where the probability of event B is computed from the sum of joint probabilities.

According to the chain rule, Eq. (3.6), $P(A_iB) = P(A_i)P(B \mid A_i)$, it follows that

$$P(B) = \sum_{k=1}^{n} P(A_k)P(B \mid A_k) \tag{3.10}$$

Combining Eqs. (3.5) and (3.10), we get the well-known *Bayes' rule*:

$$P(A_i \mid B) = \frac{P(A_iB)}{P(B)} = \frac{P(B \mid A_i)P(A_i)}{\sum_{k=1}^{n} P(B \mid A_k)P(A_k)} \tag{3.11}$$

Bayes' rule is the basis for pattern recognition that is described in Chapter 4.

3.1.2. Random Variables

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

$$P(X=x)=P(s \mid X(s)=x) \tag{3.12}$$

A random variable X is a *discrete* random variable, or X has a *discrete distribution*, if X can take only a finite number n of different values $x_1,x_2,\cdots,x_n$, or at most, an infinite sequence of different values $x_1,x_2,\cdots$. If the random variable X is a discrete random variable, the *probability function* (pf) or *probability mass function* (pmf) of X is defined to be the function p such that for any real number x,

$$p_X(x)=P(X=x) \tag{3.13}$$

For the cases in which there is no confusion, we drop the subscription X for $p_X(x)$. The sum of probability mass over all values of the random variable is equal to unity.

$$\sum_{i=1}^{n} p(x_i)=\sum_{i=1}^{n} P(X=x_i)=1 \tag{3.14}$$

The marginal probability, chain rule and Bayes' rule can also be rewritten with respect to random variables:

$$p_X(x_i)=P(X=x_i)=\sum_{k=1}^{m} P(X=x_i,Y=y_k)=\sum_{k=1}^{m} P(X=x_i \mid Y=y_k)P(Y=y_k) \tag{3.15}$$

$$\begin{aligned} &P(X_1=x_1,\cdots,X_n=x_n)= \\ &P(X_n=x_n \mid X_1=x_1,\cdots,X_{n-1}=x_{n-1})\cdots P(X_2=x_2 \mid X_1=x_1)P(X_1=x_1) \end{aligned} \tag{3.16}$$

$$P(X=x_i \mid Y=y)=\frac{P(X=x_i,Y=y)}{P(Y=y)}=\frac{P(Y=y \mid X=x_i)P(X=x_i)}{\sum_{k=1}^{n} P(Y=y \mid X=x_k)P(X=x_k)} \tag{3.17}$$

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

$$P(X=x_i,Y=y_j)=P(X=x_i)P(Y=y_j)=p_X(x_i)p_Y(y_j) \ \forall \text{ all } i \text{ and } j \tag{3.18}$$

A random variable X is a *continuous* random variable, or X has a *continuous distribution*, if there exists a nonnegative function f, defined on the real line, such that for an interval A,

$$P(X \in A) = \int_A f_X(x)dx \tag{3.19}$$

The function f_X is called the *probability density function* (abbreviated pdf) of X. We drop the subscript X for f_X if there is no ambiguity. As illustrated in Figure 3.3, the area of shaded region is equal to the value of $P(a \le X \le b)$.

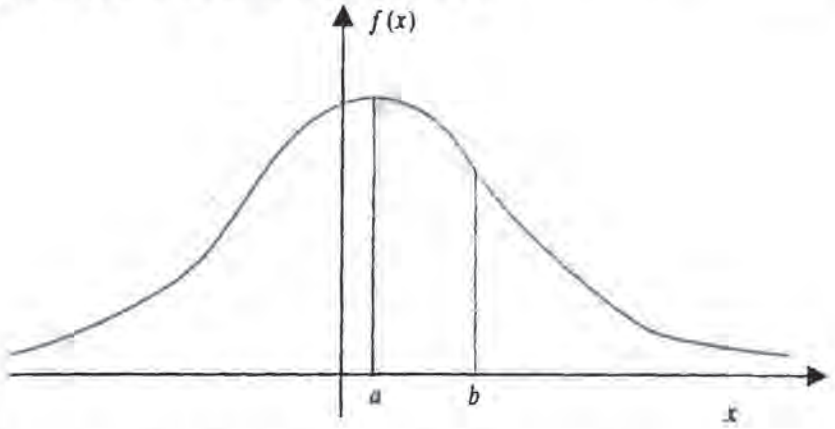


Figure 3.3 An example of pdf. The area of the shaded region is equal to the value of $P(a \le X \le b)$.

Every pdf must satisfy the following two requirements:

$$\begin{aligned} &f(x) \ge 0 \text{ for } -\infty \le x \le \infty \text{ and} \\ &\int_{-\infty}^{\infty} f(x)dx = 1 \end{aligned} \tag{3.20}$$

The marginal probability, chain rule, and Bayes' rule can also be rewritten with respect to continuous random variables:

$$f_X(x) = \int_{-\infty}^{\infty} f_{X,Y}(x,y)dy = \int_{-\infty}^{\infty} f_{X|Y}(x \mid y) f_Y(y)dy \tag{3.21}$$

$$f_{X_1,\cdots,X_n}(x_1,\cdots,x_n) = f_{X_n|X_1,\cdots,X_{n-1}}(x_n \mid x_1,\cdots,x_{n-1}) \cdots f_{X_2|X_1}(x_2 \mid x_1) f_{X_1}(x_1) \tag{3.22}$$

$$f_{X|Y}(x \mid y) = \frac{f_{X,Y}(x,y)}{f_Y(y)} = \frac{f_{Y|X}(y \mid x) f_X(x)}{\int_{-\infty}^{\infty} f_{Y|X}(y \mid x) f_X(x)dx} \tag{3.23}$$

The *distribution function* or *cumulative distribution function* F of a discrete or continuous random variable X is a function defined for every real number x as follows:

$$F(x) = P(X \leq x) \quad \text{for } -\infty \leq x \leq \infty \tag{3.24}$$

For continuous random variables, it follows that:

$$F(x) = \int_{-\infty}^{x} f_X(x)dx \tag{3.25}$$

$$f_X(x) = \frac{dF(x)}{dx} \tag{3.26}$$

3.1.3. Mean and Variance

Suppose that a discrete random variable X has a pf $f(x)$; the *expectation* or *mean* of X is defined as follows:

$$E(X) = \sum_{x} xf(x) \tag{3.27}$$

Similarly, if a continuous random variable X has a pdf f, the *expectation* or *mean* of X is defined as follows:

$$E(X) = \int_{-\infty}^{\infty} xf(x)dx \tag{3.28}$$

In physics, the mean is regarded as the center of mass of the probability distribution. The expectation can also be defined for any function of the random variable X. If X is a continuous random variable with pdf f, then the expectation of any function $g(X)$ can be defined as follows:

$$E\left[g(X)\right] = \int_{-\infty}^{\infty} g(x)f(x)dx \tag{3.29}$$

The expectation of a random variable is a linear operator. That is, it satisfies both additivity and homogeneity properties:

$$E(a_1X_1 + \cdots + a_nX_n + b) = a_1E(X_1) + \cdots + a_nE(X_n) + b \tag{3.30}$$

where $a_1, \cdots, a_n, b$ are constants.

Equation (3.30) is valid regardless of whether or not the random variables $X_1, \cdots, X_n$ are independent.

Suppose that X is a random variable with mean $\mu = E(X)$. The *variance* of X denoted as $Var(X)$ is defined as follows:

$$Var(X) = \sigma^2 = E\left[(X-\mu)^2\right] \tag{3.31}$$

where σ, the nonnegative square root of the variance is known as the *standard deviation* of random variable X. Therefore, the variance is also often denoted as σ^2.

The variance of a distribution provides a measure of the spread or dispersion of the distribution around its mean μ. A small value of the variance indicates that the probability distribution is tightly concentrated around μ, and a large value of the variance typically indicates the probability distribution has a wide spread around μ. Figure 3.4 illustrates three different Gaussian distributions[1] with the same mean, but different variances.

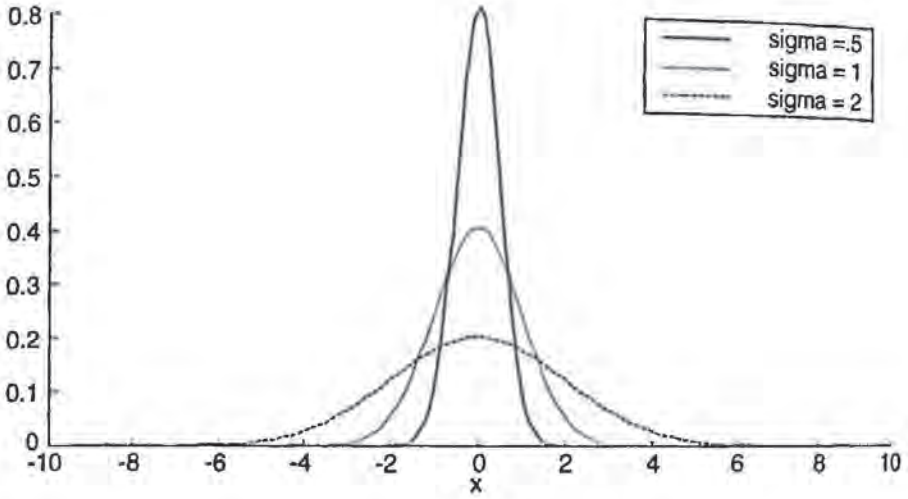


Figure 3.4 Three Gaussian distributions with same mean μ, but different variances, 0.5, 1.0, and 2.0, respectively. The distribution with a large value of the variance has a wide spread around the mean μ.

The variance of random variable X can be computed in the following way:

$$Var(X) = E(X^2) - \left[E(X)\right]^2 \tag{3.32}$$

In physics, the expectation $E(X^k)$ is called the k^{th} moment of X for any random variable X and any positive integer k. Therefore, the variance is simply the difference between the second moment and the square of the first moment.

The variance satisfies the following additivity property, if random variables X and Y are independent:

$$Var(X+Y) = Var(X) + Var(Y) \tag{3.33}$$

However, it does not satisfy the homogeneity property. Instead for constant a,

$$Var(aX) = a^2 Var(X) \tag{3.34}$$

[1] We describe Gaussian distributions in Section 3.1.7.

Since it is clear that $Var(b) = 0$ for any constant b, we have an equation similar to Eq. (3.30) if random variables $X_1, \cdots, X_n$ are independent.

$$Var(a_1X_1 + \cdots + a_nX_n + b) = a_1^2Var(X_1) + \cdots + a_n^2Var(X_n) \tag{3.35}$$

Conditional expectation can also be defined in a similar way. Suppose that X and Y are discrete random variables and let $f(y \mid x)$ denote the conditional pf of Y given $X = x$, then the conditional expectation $E(Y \mid X)$ is defined to be the function of X whose value $E(Y \mid x)$ when $X = x$ is

$$E_{Y|X}(Y \mid X = x) = \sum_y y f_{Y|X}(y \mid x) \tag{3.36}$$

For continuous random variables X and Y with $f_{Y|X}(y \mid x)$ as the conditional pdf. of Y given $X = x$, the conditional expectation $E(Y \mid X)$ is defined to be the function of X whose value $E(Y \mid x)$ when $X = x$ is

$$E_{Y|X}(Y \mid X = x) = \int_{-\infty}^{\infty} y f_{Y|X}(y \mid x) dy \tag{3.37}$$

Since $E(Y \mid X)$ is a function of random variable X, it itself is a random variable whose probability distribution can be derived from the distribution of X. It can be shown that

$$E_X\left[E_{Y|X}(Y \mid X)\right] = E_{X,Y}(Y) \tag{3.38}$$

More generally, suppose that X and Y have a continuous joint distribution and that $g(x, y)$ is any arbitrary function of X and Y. The conditional expectation $E[g(X,Y) \mid X]$ is defined to be the function of X whose value $E[g(X,Y) \mid x]$ when $X = x$ is

$$E_{Y|X}\left[g(X,Y) \mid X = x\right] = \int_{-\infty}^{\infty} g(x,y) f_{Y|X}(y \mid x) dy \tag{3.39}$$

Equation (3.38) can also be generalized into the following equation:

$$E_X\left\{E_{Y|X}\left[g(X,Y) \mid X\right]\right\} = E_{X,Y}\left[g(X,Y)\right] \tag{3.40}$$

Finally, it is worthwhile to introduce *median* and *mode*. The median of a distribution of X is defined to be a point m, such that $P(X \le m) \ge 1/2$ and $P(X \ge m) \ge 1/2$. Thus, the median m divides the total probability into two equal parts, i.e., the probability to the left of m and the probability to the right of m are exactly $1/2$.

Suppose a random variable X has either a discrete distribution with pf $p(x)$ or continuous pdf $f(x)$; a point ϖ is called the mode of the distribution if $p(x)$ or $f(x)$ attains the maximum value at the point ϖ. A distribution can have more than one mode.

3.1.3.1. The Law of Large Numbers

The concept of *sample mean* and *sample variance* is important in statistics because most statistical experiments involve sampling. Suppose that the random variables $X_1,\cdots,X_n$ form a random sample of size n from some distribution for which the mean is μ and the variance is σ^2. In other words, the random variables $X_1,\cdots,X_n$ are *independent identically distributed* (often abbreviated by iid) and each has mean μ and variance σ^2. Now if we denote $\bar{X}_n$ as the arithmetic average of the n observations in the sample, then

$$\bar{X}_n = \frac{1}{n}(X_1 + \cdots + X_n) \tag{3.41}$$

$\bar{X}_n$ is a random variable and is referred to as *sample mean*. The mean and variance of $\bar{X}_n$ can be easily derived based on the definition.

$$E(\bar{X}_n) = \mu \quad \text{and} \quad \text{Var}(\bar{X}_n) = \frac{\sigma^2}{n} \tag{3.42}$$

Equation (3.42) states that the mean of sample mean is equal to mean of the distribution, while the variance of sample mean is only $1/n$ times the variance of the distribution. In other words, the distribution of $\bar{X}_n$ will be more concentrated around the mean μ than was the original distribution. Thus, the sample mean is closer to μ than is the value of just a single observation X_i from the given distribution.

The *law of large numbers* is one of most important theorems in probability theory. Formally, it states that the sample mean $\bar{X}_n$ converges to the mean μ in probability, that is,

$$\lim_{n\to\infty} P\left(|\bar{X}_n - \mu| < \varepsilon\right) = 1 \quad \text{for any given number } \varepsilon > 0 \tag{3.43}$$

The law of large numbers basically implies that the sample mean is an excellent estimate of the unknown mean of the distribution when the sample size n is large.

3.1.4. Covariance and Correlation

Let X and Y be random variables having a specific joint distribution, and $E(X) = \mu_X$, $E(Y) = \mu_Y$, $Var(X) = \sigma_X^2$, and $Var(Y) = \sigma_Y^2$. The *covariance* of X and Y, denoted as $Cov(X,Y)$, is defined as follows:

$$Cov(X,Y) = E\left[(X-\mu_X)(Y-\mu_Y)\right] = Cov(Y,X) \tag{3.44}$$

In addition, the *correlation coefficient* of X and Y, denoted as ρ_{XY}, is defined as follows:

$$\rho_{XY} = \frac{Cov(X,Y)}{\sigma_X \sigma_Y} \tag{3.45}$$

It can be shown that $\rho(X,Y)$ should be bound within $[-1...1]$, that is,

$$-1 \le \rho(X,Y) \le 1 \tag{3.46}$$

X and Y are said to be *positively correlated* if $\rho_{XY} > 0$, *negatively correlated* if $\rho_{XY} < 0$, and *uncorrelated* if $\rho_{XY} = 0$. It can also be shown that $Cov(X,Y)$ and ρ_{XY} must have the same sign; that is, both are positive, negative, or zero at the same time. When $E(XY) = 0$, the two random variables are called *orthogonal.*

There are several theorems pertaining to the basic properties of covariance and correlation. We list here the most important ones:

Theorem 1 For any random variables X and Y

$$Cov(X,Y) = E(XY) - E(X)E(Y) \tag{3.47}$$

Theorem 2 If X and Y are independent random variables, then

$$Cov(X,Y) = \rho_{XY} = 0$$

Theorem 3 Suppose X is a random variable and Y is a linear function of X in the form of $Y = aX + b$ for some constant a and b, where $a \neq 0$. If $a > 0$, then $\rho_{XY} = 1$. If $a < 0$, then $\rho_{XY} = -1$. Sometimes, ρ_{XY} is referred to as the amount of linear dependency between random variables X and Y.

Theorem 4 For any random variables X and Y,

$$Var(X+Y) = Var(X) + Var(Y) + 2Cov(X,Y) \tag{3.48}$$

Theorem 5 If $X_1, \cdots, X_n$ are random variables, then

$$Var(\sum_{i=1}^{n} X_i) = \sum_{i=1}^{n} Var(X_i) + 2\sum_{i=1}^{n}\sum_{j=1}^{i-1} Cov(X_i, X_j) \tag{3.49}$$

3.1.5. Random Vectors and Multivariate Distributions

When a random variable is a vector rather than a scalar, it is called a random vector and we often use boldface variable like $\mathbf{X} = (X_1, \cdots, X_n)$ to indicate that it is a random vector. It is said that n random variables $X_1, \cdots, X_n$ have a *discrete joint distribution* if the random vector $\mathbf{X} = (X_1, \cdots, X_n)$ can have only a finite number or an infinite sequence of different

values $(x_1,\cdots,x_n)$ in R^n. The joint pf of $X_1,\cdots,X_n$ is defined to be the function f_X such that for any point $(x_1,\cdots,x_n)\in R^n$,

$$f_X(x_1,\cdots,x_n)=P(X_1=x_1,\cdots,X_n=x_n) \tag{3.50}$$

Similarly, it is said that n random variables $X_1,\cdots,X_n$ have a *continuous joint distribution* if there is a nonnegative function f defined on R^n such that for any subset $A\subset R^n$,

$$P[(X_1,\cdots,X_n)\in A]=\int_A\cdots\int f_X(x_1,\cdots,x_n)dx_1\cdots dx_n \tag{3.51}$$

The *joint distribution function* can also be defined similarly for n random variables $X_1,\cdots,X_n$ as follows:

$$F_X(x_1,\cdots,x_n)=P(X_1\le x_1,\cdots,X_n\le x_n) \tag{3.52}$$

The concept of mean and variance for a random vector can be generalized into *mean vector* and *covariance matrix*. Supposed that $\mathbf{X}$ is an n-dimensional random vector with components $X_1,\cdots,X_n$, under matrix representation, we have

$$\mathbf{X}=\begin{bmatrix}X_1\\ \vdots\\ X_n\end{bmatrix} \tag{3.53}$$

The *expectation (mean) vector* $E(\mathbf{X})$ of random vector $\mathbf{X}$ is an n-dimensional vector whose components are the expectations of the individual components of $\mathbf{X}$, that is,

$$E(\mathbf{X})=\begin{bmatrix}E(X_1)\\ \vdots\\ E(X_n)\end{bmatrix} \tag{3.54}$$

The *covariance matrix* $Cov(\mathbf{X})$ of random vector $\mathbf{X}$ is defined to be an $n\times n$ matrix such that the element in the i^{th} row and j^{th} column is $Cov(X_i,Y_j)$, that is,

$$Cov(\mathbf{X})=\begin{bmatrix}Cov(X_1,X_1) & \cdots & Cov(X_1,X_n)\\ \vdots & & \vdots\\ Cov(X_n,X_1) & \cdots & Cov(X_n,X_n)\end{bmatrix}=E\left[[X-E(X)][X-E(X)]^t\right] \tag{3.55}$$

It should be emphasized that the n diagonal elements of the covariance matrix $Cov(\mathbf{X})$ are actually the variances of $X_1,\cdots,X_n$. Furthermore, since the covariance is symmetric, i.e., $Cov(X_i,X_j)=Cov(X_j,X_i)$, the covariance matrix $Cov(\mathbf{X})$ must be a symmetric matrix.

There is an important theorem regarding the mean vector and covariance matrix for a linear transformation of random vector $\mathbf{X}$. Suppose $\mathbf{X}$ is an n-dimensional vector as specified by Eq. (3.53), with mean vector $E(\mathbf{X})$ and covariance matrix $Cov(\mathbf{X})$. Now, assume $\mathbf{Y}$ is a m-dimensional random vector which is a linear transform of random vector $\mathbf{X}$ by the

relation: $\mathbf{Y}=\mathbf{AX}+\mathbf{B}$, where $\mathbf{A}$ is a $m\times n$ transformation matrix whose elements are constants, and $\mathbf{B}$ is a m-dimensional constant vector. Then we have the following two equations:

$$E(\mathbf{Y})=\mathbf{A}E(\mathbf{X})+\mathbf{B} \tag{3.56}$$

$$Cov(\mathbf{Y})=\mathbf{A}Cov(\mathbf{X})\mathbf{A}' \tag{3.57}$$

3.1.6. Some Useful Distributions

In the following two sections, we will introduce several useful distributions that are widely used in applications of probability and statistics, particularly in spoken language systems.

3.1.6.1. Uniform Distributions

The simplest distribution is uniform distribution where the pf or pdf is a constant function. For uniform discrete random variable X, which only takes possible values from $\{x_i \mid 1\le i\le n\}$, the pf for X is

$$P(X=x_i)=\frac{1}{n} \quad 1\le i\le n \tag{3.58}$$

For uniform continuous random variable X, which only takes possible values in the real interval $[a,b]$, as shown in Figure 3.5, the pdf for X is

$$f(x)=\frac{1}{b-a} \quad a\le x\le b \tag{3.59}$$

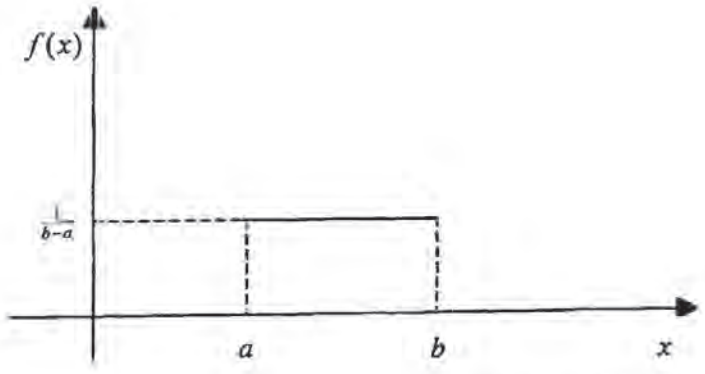


Figure 3.5 A uniform distribution for pdf in Eq. (3.59).

3.1.6.2. Binomial Distributions

The *binomial distribution* is used to describe binary-decision events. For example, suppose that a single coin toss will produce heads with probability p and produce tails with probability $1-p$. Now, if we toss the same coin n times and let X denote the number of heads observed, then the random variable X has the following *binomial* pf:

$$P(X=x)=f(x\mid n,p)=\binom{n}{x}p^x(1-p)^{n-x} \tag{3.60}$$

It can be shown that the mean and variance of a binomial distribution are:

$$E(X)=np \tag{3.61}$$

$$Var(X)=np(1-p) \tag{3.62}$$

Figure 3.6 illustrates three binomial distributions with p = 0.2, 0.3, and 0.4, and n = 10.

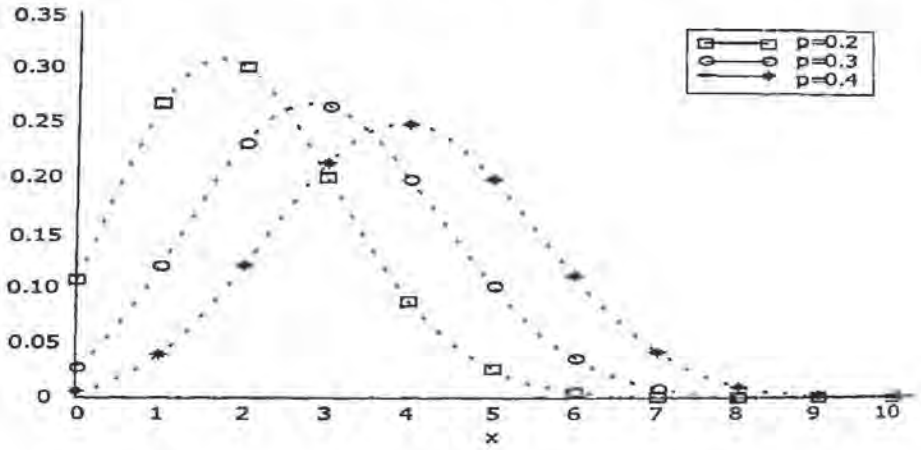


Figure 3.6 Three binomial distributions with p = 0.2, 0.3, and 0.4, and n = 10.

3.1.6.3. Geometric Distributions

The geometric distribution is related to the binomial distribution. As in the independent coin toss example, heads has a probability p and tails has a probability $1-p$. The geometric distribution is to model the time until tails appears. Let the random variable X

be the time (the number of tosses) until the first tail-up is shown. The pdf of X is in the following form:

$$P(X=x)=f(x\mid p)=p^{x-1}(1-p) \quad x=1,2,\ldots \text{ and } 0<p<1 \tag{3.63}$$

The mean and variance of a geometric distribution are given by:

$$E(X)=\frac{1}{1-p} \tag{3.64}$$

$$Var(X)=\frac{1}{(1-p)^2} \tag{3.65}$$

One example for the geometric distribution is the distribution of the state duration for a hidden Markov model, as described in Chapter 8. Figure 3.7 illustrates three geometric distributions with $p = 0.1$, 0.4, and 0.7.

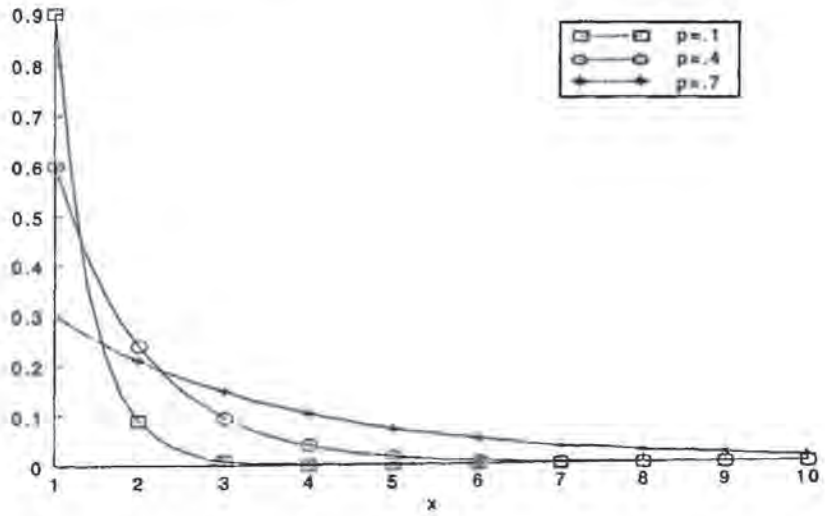


Figure 3.7 Three geometric distributions with different parameter p.

3.1.6.4. Multinomial Distributions

Suppose that a bag contains balls of k different colors, where the proportion of the balls of color i is p_i. Thus, $p_i > 0$ for $i=1,\ldots,k$ and $\sum_{i=1}^{k} p_i = 1$. Now suppose that n balls are randomly selected from the bag and there are enough balls ($> n$) of each color. Let X_i denote

the number of selected balls that are of color i. The random vector $\mathbf{X}=(X_1,\ldots,X_k)$ is said to have a *multinomial distribution* with parameters n and $\mathbf{p}=(p_1,\ldots,p_k)$. For a vector $\mathbf{x}=(x_1,\ldots,x_k)$, the pf of $\mathbf{X}$ has the following form:

$$P(\mathbf{X}=\mathbf{x})=f(\mathbf{x}\mid n,\mathbf{p})=\begin{cases}\dfrac{n!}{x_1!\ldots x_k!}p_1^{x_1}\ldots p_k^{x_k} & \text{where } x_i\geq 0\ \forall i=1\ \ldots\ k \\ & \text{and } x_1+\cdots+x_k=n \\ 0 & \text{otherwise}\end{cases} \tag{3.66}$$

It can be shown that the mean, variance and covariance of the multinomial distribution are:

$$E(X_i)=np_i \quad \text{and} \quad Var(X_i)=np_i(1-p_i) \quad \forall i=1,\ldots,k \tag{3.67}$$

$$Cov(X_i,X_j)=-np_ip_j \tag{3.68}$$

Figure 3.8 shows a multinomial distribution with $n=10$, $p_1=0.2$, and $p_2=0.3$. Since there are only two free parameters x_1 and x_2, the graph is illustrated only using x_1

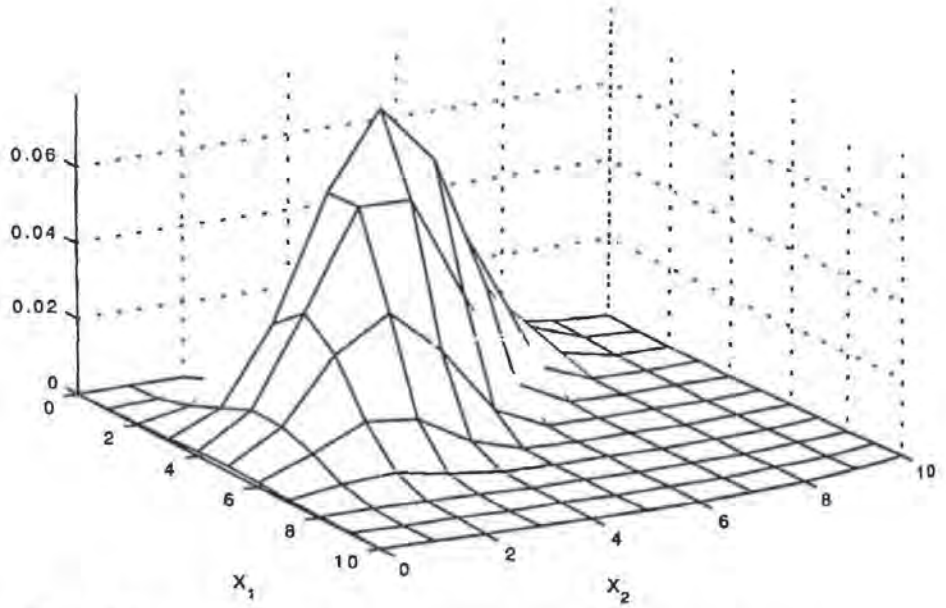


Figure 3.8 A multinomial distribution with n=10, $p_1=0.2$, and $p_2=0.3$.

and x_2 as axis. Multinomial distributions are typically used with the χ^2 test that is one of the most widely used goodness-of-fit hypotheses testing procedures described in Section 3.3.3.

3.1.6.5. Poisson Distributions

Another popular discrete distribution is the *Poisson distribution*. The random variable X has a Poisson distribution with mean λ ($\lambda > 0$) if the pf of X has the following form:

$$P(X = x) = f(x \mid \lambda) = \begin{cases} \dfrac{e^{-\lambda}\lambda^x}{x!} & \text{for } x = 0, 1, 2, \ldots \\ 0 & \text{otherwise} \end{cases} \tag{3.69}$$

The mean and variance of a Poisson distribution are the same and equal λ:

$$E(X) = Var(X) = \lambda \tag{3.70}$$

Figure 3.9 illustrates three Poisson distributions with $\lambda = 1$, 2, and 4. The Poisson distribution is typically used in queuing theory, where x is the total number of occurrences of some phenomenon during a fixed period of time or within a fixed region of space. Examples include the number of telephone calls received at a switchboard during a fixed period of time. In speech recognition, the Poisson distribution is used to model the duration for a phoneme.

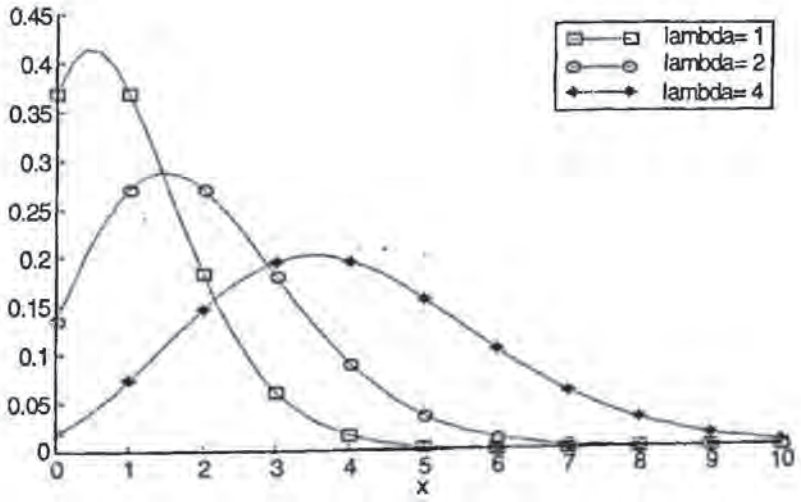


Figure 3.9 Three Poisson distributions with $\lambda = 1$, 2, and 4.

3.1.6.6. Gamma Distributions

A continuous random variable X is said to have a *gamma distribution* with parameters α and β ($\alpha > 0$ and $\beta > 0$) if X has a continuous pdf of the following form:

$$f(x \mid \alpha, \beta) = \begin{cases} \dfrac{\beta^{\alpha}}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x} & x > 0 \\ 0 & x \le 0 \end{cases} \tag{3.71}$$

where

$$\Gamma(\alpha) = \int_0^{\infty} x^{\alpha-1} e^{-x} dx \tag{3.72}$$

It can be shown that the function Γ is a factorial function when α is a positive integer.

$$\Gamma(n) = \begin{cases} (n-1)! & n = 2,3,... \\ 1 & n = 1 \end{cases} \tag{3.73}$$

The mean and variance of a gamma distribution are:

$$E(X) = \frac{\alpha}{\beta} \text{ and } Var(X) = \frac{\alpha}{\beta^2} \tag{3.74}$$

Figure 3.10 illustrates three gamma distributions with $\beta = 1.0$ and $\alpha = 2.0$, 3.0, and 4.0. There is an interesting theorem associated with gamma distributions. If the random variables $X_1, \ldots, X_k$ are independent and each random variable X_i has a gamma distribution with parameters α_i and β, then the sum $X_1 + \cdots + X_k$ also has a gamma distribution with parameters $\alpha_1 + \cdots + \alpha_k$ and β.

A special case of gamma distribution is called *exponential distribution*. A continuous random variable X is said to have an *exponential distribution* with parameters β ($\beta > 0$) if X has a continuous pdf of the following form:

$$f(x \mid \beta) = \begin{cases} \beta e^{-\beta x} & x > 0 \\ 0 & x \le 0 \end{cases} \tag{3.75}$$

It is clear that the exponential distribution is a gamma distribution with $\alpha = 1$. The mean and variance of the exponential distribution are:

$$E(X) = \frac{1}{\beta} \text{ and } Var(X) = \frac{1}{\beta^2} \tag{3.76}$$

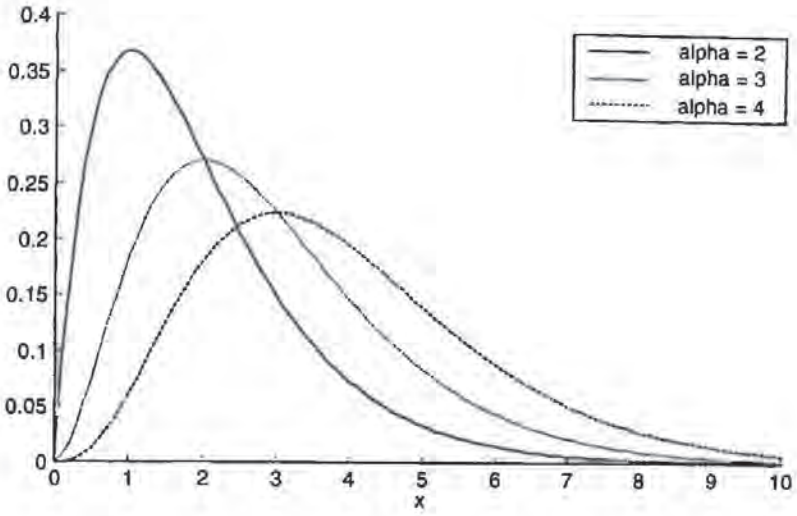


Figure 3.10 Three Gamma distributions with $\beta = 1.0$ and $\alpha = 2.0$, 3.0, and 4.0.

Figure 3.11 shows three exponential distributions with $\beta = 1.0$, 0.6, and 0.3. The exponential distribution is often used in queuing theory for the distributions of the duration of a service or the inter-arrival time of customers. It is also used to approximate the distribution of the life of a mechanical component.

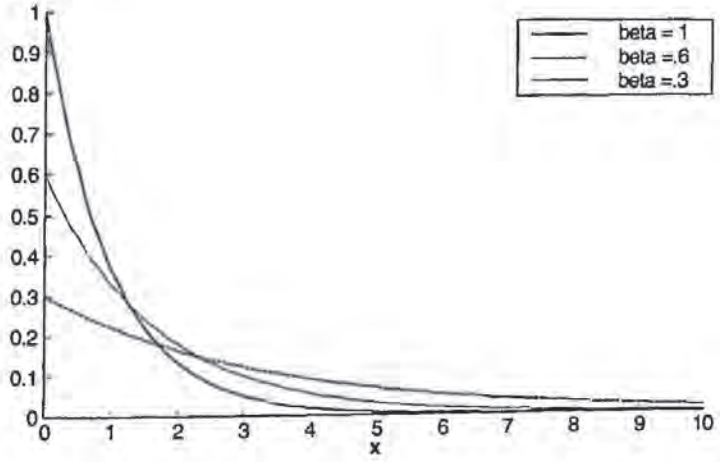


Figure 3.11 Three exponential distributions with $\beta = 1.0$, 0.6, and 0.3.

3.1.7. Gaussian Distributions

Gaussian distribution is by far the most important probability distribution mainly because many scientists have observed that the random variables studied in various physical experiments (including speech signals), often have distributions that are approximately Gaussian. The Gaussian distribution is also referred to as normal distribution. A continuous random variable X is said to have a *Gaussian distribution* with mean μ and variance σ^2 $(\sigma > 0)$ if X has a continuous pdf in the following form:

$$f(x \mid \mu,\sigma^2) = N(\mu,\sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right] \tag{3.77}$$

It can be shown that μ and σ^2 are indeed the mean and the variance for the Gaussian distribution. Some examples of Gaussians can be found in Figure 3.4.

The use of Gaussian distributions is justified by the *Central Limit Theorem*, which states that observable events considered to be a consequence of many unrelated causes with no single cause predominating over the others, tend to follow the Gaussian distribution [6].

It can be shown from Eq. (3.77) that the Gaussian $f(x \mid \mu,\sigma^2)$ is symmetric with respect to $x = \mu$. Therefore, μ is both the mean and the median of the distribution. Moreover, μ is also the mode of the distribution, i.e., the pdf $f(x \mid \mu,\sigma^2)$ attains its maximum at the mean point $x = \mu$.

Several Gaussian pdfs with the same mean μ, but different variances are illustrated in Figure 3.4. Readers can see that the curve has a *bell* shape. The Gaussian pdf with a small variance has a high peak and is very concentrated around the mean μ, whereas the Gaussian pdf with a large variance is relatively flat and is spread out more widely over the x-axis.

If the random variable X is a *Gaussian distribution* with mean μ and variance σ^2, then any linear function of X also has a Gaussian distribution. That is, if $Y = aX + b$, where a and b are constants and $a \neq 0$, Y has a Gaussian distribution with mean $a\mu + b$ and variance $a^2\sigma^2$. Similarly, the sum $X_1 + \cdots + X_k$ of random variables $X_1,\ldots,X_k$, where each random variable X_i has a Gaussian distribution, is also a Gaussian distribution.

3.1.7.1. Standard Gaussian Distributions

The Gaussian distribution with mean 0 and variance 1, denoted as $N(0,1)$, is called the *standard Gaussian distribution* or *unit Gaussian distribution*. Since the linear transformation of a Gaussian distribution is still a Gaussian distribution, the behavior of a Gaussian distribution can be solely described using a standard Gaussian distribution. If the random variable X is a Gaussian distribution with mean μ and variance σ^2, that is, $X \sim N(\mu,\sigma^2)$, it can be shown that

$$Z = \frac{X-\mu}{\sigma} \sim N(0,1) \tag{3.78}$$

Based on Eq. (3.78), the following property can be shown:

$$P(|X-\mu| \le k\sigma) = P(|Z| \le k) \tag{3.79}$$

Equation (3.79) demonstrates that every Gaussian distribution contains the same total amount of probability within any fixed number of standard deviations of its mean.

3.1.7.2. The Central Limit Theorem

If random variables $X_1,\ldots,X_n$ are i.i.d. according to a common distribution function with mean μ and variance σ^2, then as the random sample size n approaches ∞, the following random variable has a distribution converging to the standard Gaussian distribution:

$$Y_n = \frac{n(\bar{X}_n - \mu)}{\sqrt{n\sigma^2}} \sim N(0,1) \tag{3.80}$$

where $\bar{X}_n$ is the sample mean of random variables $X_1,\ldots,X_n$ as defined in Eq. (3.41).

Based on Eq. (3.80), the sample mean random variable $\bar{X}_n$ can be approximated by a Gaussian distribution with mean μ and variance σ^2/n.

The central limit theorem above is applied to i.i.d. random variables $X_1,\ldots,X_n$. A. Liapounov in 1901 derived another central limit theorem for independent but not necessarily identically distributed random variables $X_1,\ldots,X_n$. Suppose $X_1,\ldots,X_n$ are independent random variables and $E(|X_i-\mu_i|^3) < \infty$ for $1 \le i \le n$; the following random variable will converge to standard Gaussian distribution when $n \to \infty$.

$$Y_n = \left(\sum_{i=1}^{n} X_i - \sum_{i=1}^{n} \mu_i\right) \Big/ \left(\sum_{i=1}^{n} \sigma_i^2\right)^{1/2} \tag{3.81}$$

In other words, the sum of random variables $X_1,\ldots,X_n$ can be approximated by a Gaussian distribution with mean $\sum_{i=1}^{n} \mu_i$ and variance $\left(\sum_{i=1}^{n} \sigma_i^2\right)^{1/2}$.

Both central limit theorems essentially state that regardless of their original individual distributions, the sum of many independent random variables (effects) tends to be distributed like a Gaussian distribution as the number of random variables (effects) becomes large.

3.1.7.3. Multivariate Mixture Gaussian Distributions

When $\mathbf{X} = (X_1,\ldots,X_n)$ is an n-dimensional continuous random vector, the multivariate Gaussian pdf has the following form:

$$f(\mathbf{X}=\mathbf{x}\,|\,\boldsymbol{\mu},\boldsymbol{\Sigma}) = N(\mathbf{x};\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{n/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right] \tag{3.82}$$

where $\boldsymbol{\mu}$ is the n-dimensional mean vector, $\boldsymbol{\Sigma}$ is the $n \times n$ covariance matrix, and $|\boldsymbol{\Sigma}|$ is the determinant of the covariance matrix $\boldsymbol{\Sigma}$.

$$\boldsymbol{\mu} = E(\mathbf{x}) \tag{3.83}$$

$$\boldsymbol{\Sigma} = E\left[(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})'\right] \tag{3.84}$$

More specifically, the i-j^{th} element σ_{ij}^2 of covariance matrix $\boldsymbol{\Sigma}$ can be specified as follows:

$$\sigma_{ij}^2 = E\left[(x_i - \mu_i)(x_j - \mu_j)\right] \tag{3.85}$$

If $X_1, \ldots, X_n$ are independent random variables, the covariance matrix $\boldsymbol{\Sigma}$ is reduced to diagonal covariance where all the off-diagonal entries are zero. The distribution can be regarded as n independent scalar Gaussian distributions. The joint pdf is the product of all the individual scalar Gaussian pdfs. Figure 3.12 shows a two-dimensional multivariate Gaussian distribution with independent random variables x_1 and x_2 with the same variance. Figure 3.13 shows another two-dimensional multivariate Gaussian distribution with independent random variables x_1 and x_2 that have different variances.

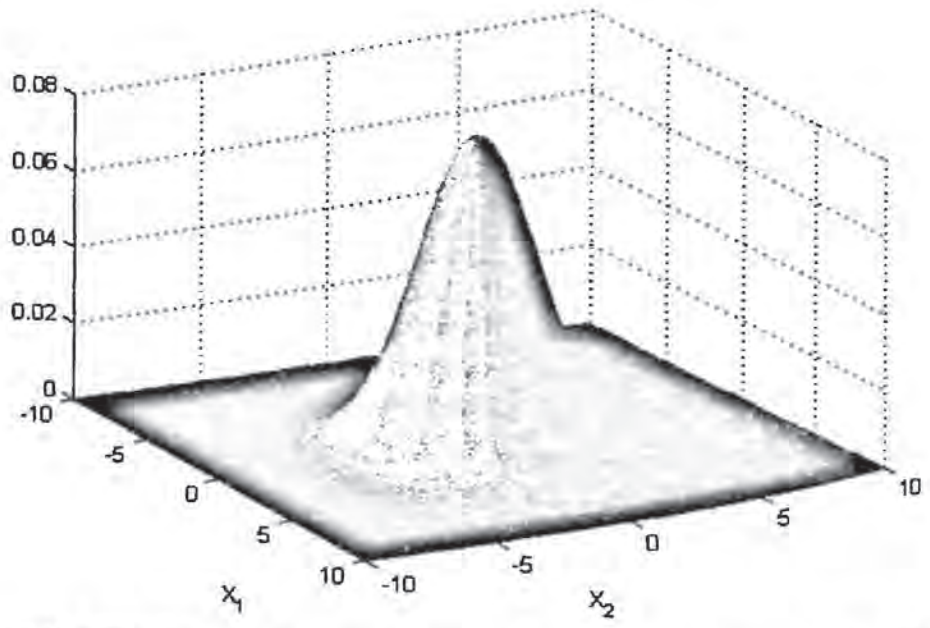


Figure 3.12 A two-dimensional multivariate Gaussian distribution with independent random variables x_1 and x_2 that have the same variance.

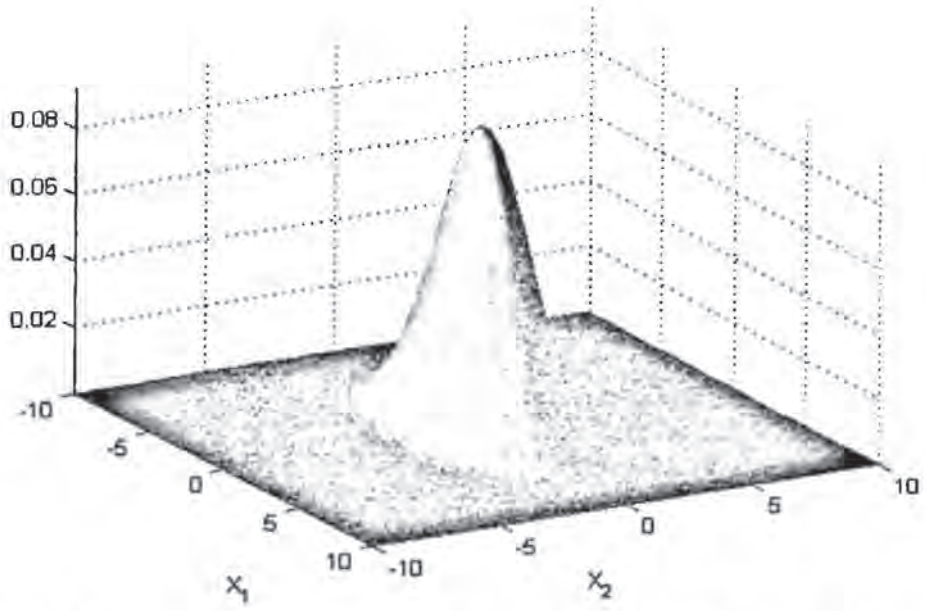


Figure 3.13 Another two-dimensional multivariate Gaussian distribution with independent random variable x_1 and x_2 which have different variances.

Although Gaussian distributions are unimodal,[2] more complex distributions with multiple local maxima can be approximated by Gaussian mixtures:

$$f(\mathbf{x}) = \sum_{k=1}^{K} c_k N_k(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{3.86}$$

where c_k, the mixture weight associated with kth Gaussian component, is subject to the following constraint:

$$c_k \geq 0 \text{ and } \sum_{k=1}^{K} c_k = 1$$

Gaussian mixtures with enough mixture components can approximate any distribution. Throughout this book, most continuous probability density functions are modeled with Gaussian mixtures.

3.1.7.4. χ^2 Distributions

The gamma distribution with parameters α and β is defined 1). For any given positive integer n, the gamma distribution for which $\alpha = n/2$ a is called the χ^2

[2] A unimodal distribution has a single maximum (bump) for the distribution. F ibution, the maximum occurs at the mean.

distribution with n degrees of freedom. It follows from Eq. (3.71) that the pdf for the χ^2 distribution is

$$f(x|n)=\begin{cases}\dfrac{1}{2^{n/2}\Gamma(n/2)}x^{(n/2)-1}e^{-x/2} & x>0\\ 0 & x\le 0\end{cases} \tag{3.87}$$

χ^2 distributions are important in statistics because they are closely related to random samples of Gaussian distribution. They are widely applied in many important problems of statistical inference and hypothesis testing. Specifically, if the random variables $X_1,\ldots,X_n$ are independent and identically distributed, and if each of these variables has a standard Gaussian distribution, then the sum of square $X_1^2+\ldots+X_n^2$ can be proved to have a χ^2 distribution with n degrees of freedom. Figure 3.14 illustrates three χ^2 distributions with $n=2$, 3, and 4.

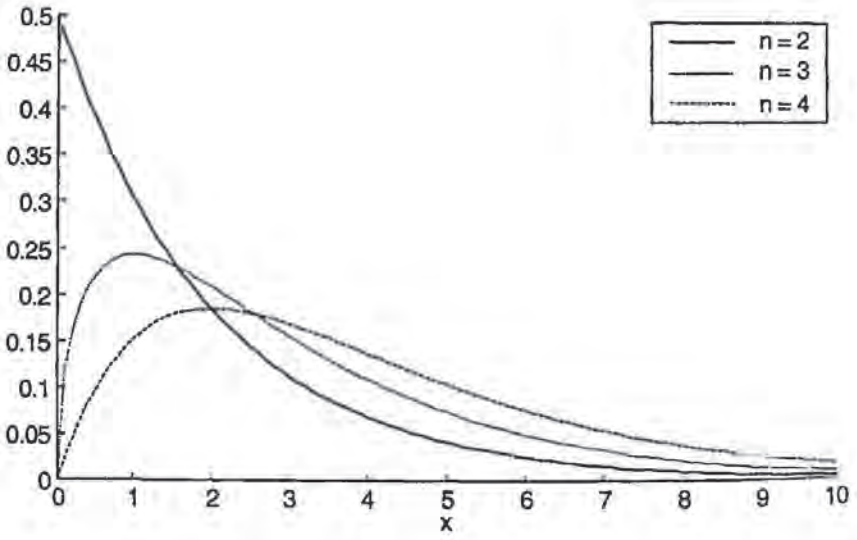


Figure 3.14 Three χ^2 distributions with $n=2$, 3, and 4.

The mean and variance for the χ^2 distribution are

$$E(X)=n \text{ and } Var(X)=2n \tag{3.88}$$

Following the additivity property of the gamma distribution, the χ^2 distribution also has the additivity property. That is, if the random variables $X_1,\ldots,X_n$ are independent and if X_i has a χ^2 distribution with k_i degrees of freedom, the sum $X_1+\ldots+X_n$ has a χ^2 distribution with $k_1+\ldots+k_n$ degrees of freedom.

3.1.7.5. Log-Normal Distribution

Let x be a Gaussian random variable with mean μ_x and standard deviation σ_x, then

$$y = e^x \tag{3.89}$$

follows the *lognormal* distribution

$$f(y \mid \mu_x, \sigma_x) = \frac{1}{y\sigma_x\sqrt{2\pi}} \exp\left\{-\frac{(\ln y - \mu_x)^2}{2\sigma_x^2}\right\} \tag{3.90}$$

shown in Figure 3.15, and whose mean is given by

$$\begin{aligned} \mu_y = E\{y\} = E\{e^x\} &= \int_{-\infty}^{\infty} \exp\{x\} \frac{1}{\sqrt{2\pi}\sigma_x} \exp\left\{-\frac{(x-\mu_x)^2}{2\sigma_x^2}\right\} dx \\ &= \int_{-\infty}^{\infty} \exp\{\mu_x + \sigma_x^2/2\} \frac{1}{\sqrt{2\pi}\sigma_x} \exp\left\{-\frac{(x-(\mu_x+\sigma_x^2))^2}{2\sigma_x^2}\right\} dx = \exp\{\mu_x + \sigma_x^2/2\} \end{aligned} \tag{3.91}$$

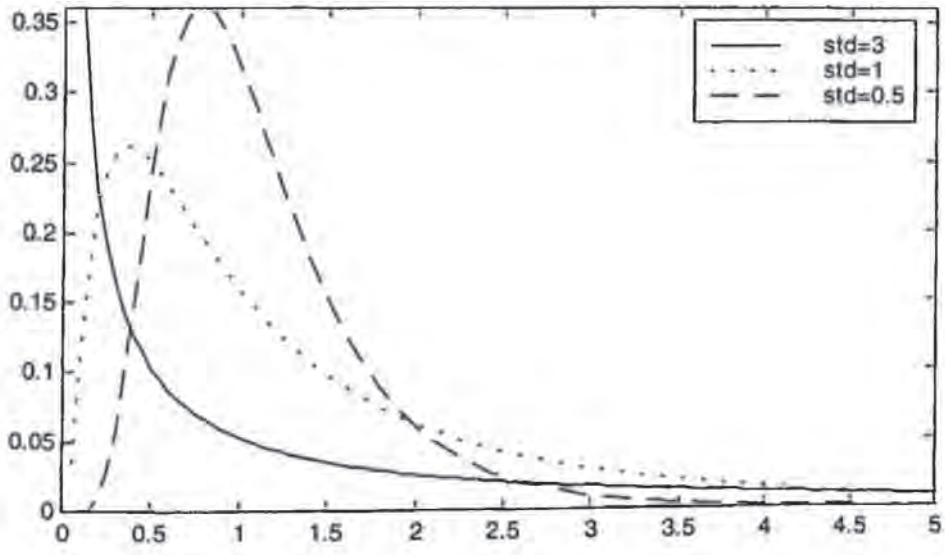


Figure 3.15 Lognormal distribution for $\mu_x = 0$ and $\sigma_x = 3$, 1, and 0.5, according to Eq. (3.90).

where we have rearranged the quadratic form of x and made use of the fact that the total probability mass of a Gaussian is 1. Similarly, the second order moment of y is given by

$$E\{y^2\} = \int_{-\infty}^{\infty} \exp\{2x\} \frac{1}{\sqrt{2\pi}\sigma_x} \exp\left\{-\frac{(x-\mu_x)^2}{2\sigma_x^2}\right\} dx$$
$$= \int_{-\infty}^{\infty} \exp\{2\mu_x + 2\sigma_x^2\} \frac{1}{\sqrt{2\pi}\sigma_x} \exp\left\{-\frac{(x-(\mu_x+2\sigma_x^2))^2}{2\sigma_x^2}\right\} dx = \exp\{2\mu_x + 2\sigma_x^2\} \tag{3.92}$$

and thus the variance of y is given by

$$\sigma_y^2 = E\{y^2\} - (E\{y\})^2 = \mu_y^2 \left(\exp\{\sigma_x^2\} - 1\right) \tag{3.93}$$

Similarly, if $\mathbf{x}$ is a Gaussian random vector with mean $\boldsymbol{\mu}_x$ and covariance matrix $\boldsymbol{\Sigma}_x$, then random vector $\mathbf{y} = e^{\mathbf{x}}$ is log-normal with mean and covariance matrix [8] given by

$$\boldsymbol{\mu}_y[i] = \exp\{\boldsymbol{\mu}_x[i] + \boldsymbol{\Sigma}_x[i,i]/2\}$$
$$\boldsymbol{\Sigma}_y[i,j] = \boldsymbol{\mu}_y[i]\boldsymbol{\mu}_y[j]\left(\exp\{\boldsymbol{\Sigma}_x[i,j]\} - 1\right) \tag{3.94}$$

using a similar derivation as in Eqs. (3.91) to (3.93).

3.2. ESTIMATION THEORY

Estimation theory and *significance testing* are two of the most important theories and methods of *statistical inference*. In this section, we describe estimation theory while significance testing is covered in the next section. A problem of statistical inference is one in which data generated in accordance with some unknown probability distribution must be analyzed, and some type of inference about the unknown distribution must be made. In a problem of statistical inference, any characteristic of the distribution generating the experimental data, such as the mean μ and variance σ^2 of a Gaussian distribution, is called a parameter of the distribution. The set Ω of all possible values of a *parameter* Φ or a group of parameters $\Phi_1, \Phi_2, \ldots, \Phi_n$ is called the *parameter space*. In this section we focus on how to estimate the parameter Φ from sample data.

Before we describe various estimation methods, we introduce the concept and nature of the estimation problems. Suppose that a set of random variables $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$ is iid according to a pdf $p(x|\Phi)$ where the value of the parameter Φ is unknown. Now, suppose also that the value of Φ must be estimated from the observed values in the sample. An *estimator* of the parameter Φ, based on the random variables $X_1, X_2, \ldots, X_n$, is a real-valued function $\theta(X_1, X_2, \ldots, X_n)$ that specifies the estimated value of Φ for each possible set of values of $X_1, X_2, \ldots, X_n$. That is, if the sample values of $X_1, X_2, \ldots, X_n$ turn out to be $x_1, x_2, \ldots, x_n$, then the estimated value of Φ will be $\theta(x_1, x_2, \ldots, x_n)$.

We need to distinguish between *estimator*, *estimate*, and *estimation*. An estimator $\theta(X_1, X_2, \ldots, X_n)$ is a function of the random variables, whose probability distribution can be derived from the joint distribution of $X_1, X_2, \ldots, X_n$. On the other hand, an estimate is a specific value $\theta(x_1, x_2, \ldots, x_n)$ of the estimator that is determined by using some specific

sample values $x_1, x_2, \ldots, x_n$. Estimation is usually used to indicate the process of obtaining such an estimator for the set of random variables or an estimate for the set of specific sample values. If we use the notation $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$ to represent the vector of random variables and $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$ to represent the vector of sample values, an estimator can be denoted as $\theta(\mathbf{X})$ and an estimate $\theta(\mathbf{x})$. Sometimes we abbreviate an estimator $\theta(\mathbf{X})$ by just the symbol θ.

In the following four sections we describe and compare three different estimators (estimation methods). They are *minimum mean square estimator*, *maximum likelihood estimator*, and *Bayes' estimator*. The first one is often used to estimate the random variable itself, while the latter two are used to estimate the parameters of the distribution of the random variables.

3.2.1. Minimum/Least Mean Squared Error Estimation

Minimum mean squared error (MMSE) estimation and *least squared error* (LSE) estimation are important methods for random variables since the goal (minimize the squared error) is an intuitive one. In general, two random variables X and Y are i.i.d. according to some pdf $f_{X,Y}(x,y)$. Suppose that we perform a series of experiments and observe the value of X. We want to find a transformation $\hat{Y} = g(X)$ such that we can predict the value of the random variable Y. The following quantity can measure the goodness of such a transformation:

$$E(Y-\hat{Y})^2 = E(Y-g(X))^2 \tag{3.95}$$

This quantity is called *mean squared error* (MSE) because it is the mean of the squared error of the predictor $g(X)$. The criterion of minimizing the mean squared error is a good one for picking the predictor $g(X)$. Of course, we usually specify the class of function G, from which $g(X)$ may be selected. In general, there is a parameter vector $\mathbf{\Phi}$ associated with the function $g(X)$, so the function can be expressed as $g(X,\mathbf{\Phi})$. The process to find the parameter vector $\hat{\mathbf{\Phi}}_{MMSE}$ that minimizes the mean of the squared error is called *minimum mean squared error estimation* and $\hat{\mathbf{\Phi}}_{MMSE}$ is called the *minimum mean squared error estimator*. That is,

$$\hat{\mathbf{\Phi}}_{MMSE} = \arg\min_{\mathbf{\Phi}} \left[E\left[(Y - g(X,\mathbf{\Phi}))^2 \right] \right] \tag{3.96}$$

Sometimes, the joint distribution of random variables X and Y is not known. Instead, samples of (x,y) pairs may be observable. In this case, the following criterion can be used instead,

$$\hat{\mathbf{\Phi}}_{LSE} = \arg\min_{\mathbf{\Phi}} \sum_{i=1}^{n} \left[y_i - g(x_i,\mathbf{\Phi}) \right]^2 \tag{3.97}$$

The argument of the minimization in Eq. (3.97) is called *sum-of-squared-error* (SSE) and the process of finding the parameter vector $\hat{\mathbf{\Phi}}_{LSE}$, which satisfies the criterion is called *least*

squared error estimation or *minimum squared error estimation*. LSE is a powerful mechanism for curve fitting, where the function $g(x,\Phi)$ describes the observation pairs (x_i, y_i). In general, there are more points (n) than the number of free parameters in function $g(x,\Phi)$, so the fitting is over-determined. Therefore, no exact solution exists, and LSE fitting becomes necessary.

It should be emphasized that MMSE and LSE are actually very similar and share similar properties. The quantity in Eq. (3.97) is actually n times the sample mean of the squared error. Based on the law of large numbers, when the joint probability $f_{X,Y}(x,y)$ is uniform or the number of samples approaches to infinity, MMSE and LSE are equivalent.

For the class of functions, we consider the following three cases:

- Constant functions, i.e.,

$$G_c = \{g(x) = c, c \in R\} \tag{3.98}$$

- Linear functions, i.e.,

$$G_l = \{g(x) = ax + b \quad a,b \in R\} \tag{3.99}$$

- Other non-linear functions G_{nl}

3.2.1.1. MMSE/LSE for Constant Functions

When $\hat{Y} = g(x) = c$, Eq. (3.95) becomes

$$E(Y - \hat{Y})^2 = E(Y - c)^2 \tag{3.100}$$

To find the MMSE estimate for c, we take the derivatives of both sides in Eq. (3.100) with respect to c and equate it to 0. The MMSE estimate c_{MMSE} is given as

$$c_{MMSE} = E(Y) \tag{3.101}$$

and the minimum mean squared error is exactly the variance of Y, $Var(Y)$.

For the LSE estimate of c, the quantity in Eq. (3.97) becomes

$$\min \sum_{i=1}^{n} [y_i - c]^2 \tag{3.102}$$

Similarly, the LSE estimate c_{LSE} can be obtained as follows:

$$c_{LSE} = \frac{1}{n} \sum_{i=1}^{n} y_i \tag{3.103}$$

The quantity in Eq. (3.103) is the sample mean.

3.2.1.2. MMSE and LSE for Linear Functions

When $\hat{Y} = g(x) = ax + b$, Eq. (3.95) becomes

$$e(a,b) = E(Y - \hat{Y})^2 = E(Y - ax - b)^2 \tag{3.104}$$

To find the MMSE estimate of a and b, we can first set

$$\frac{\partial e}{\partial a} = 0, \text{ and } \frac{\partial e}{\partial b} = 0 \tag{3.105}$$

and solve the two linear equations. Thus, we can obtain

$$a = \frac{\text{cov}(X,Y)}{Var(X)} = \rho_{XY}\frac{\sigma_Y}{\sigma_X} \tag{3.106}$$

$$b = E(Y) - \rho_{XY}\frac{\sigma_Y}{\sigma_X}E(X) \tag{3.107}$$

For LSE estimation, we assume that the sample $\mathbf{x}$ is a d-dimensional vector for generality. Assuming we have n sample-vectors $(\mathbf{x}_i, y_i) = (x_i^1, x_i^2, \cdots, x_i^d, y_i)$, $i = 1 \ldots n$, a linear function can be represented as

$$\hat{\mathbf{Y}} = \mathbf{XA} \text{ or } \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} 1 & x_1^1 & \cdots & x_1^d \\ 1 & x_2^1 & \cdots & x_2^d \\ \vdots & \vdots & & \vdots \\ 1 & x_n^1 & \cdots & x_n^d \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_d \end{pmatrix} \tag{3.108}$$

The sum of squared error can then be represented as

$$e(\mathbf{A}) = \| \hat{\mathbf{Y}} - \mathbf{Y} \|^2 = \sum_{i=1}^{n} \left(\mathbf{A}^t \mathbf{x}_i - y_i \right)^2 \tag{3.109}$$

A closed-form solution of the LSE estimate of $\mathbf{A}$ can be obtained by taking the gradient of $e(\mathbf{A})$,

$$\nabla e(\mathbf{A}) = \sum_{i=1}^{n} 2(\mathbf{A}^t \mathbf{x}_i - y_i)\mathbf{x}_i = 2\mathbf{X}^t(\mathbf{XA} - \mathbf{Y}) \tag{3.110}$$

and equating it to zero. This yields the following equation:

$$\mathbf{X}^t\mathbf{XA} = \mathbf{X}^t\mathbf{Y} \tag{3.111}$$

Thus the LSE estimate $\mathbf{A}_{LSE}$ will be of the following form:

$$\mathbf{A}_{LSE} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y} \tag{3.112}$$

$(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'$ in Eq. (3.112) is also refereed to as the *pseudo-inverse* of $\mathbf{X}$ and is sometimes denoted as $\mathbf{X}^{\perp}$.

When $\mathbf{X}'\mathbf{X}$ is singular or some boundary conditions cause the LSE estimation in Eq. (3.112) to be unattainable, some numeric methods can be used to find an approximate solution. Instead of minimizing the quantity in Eq. (3.109), one can minimize the following quantity:

$$e(\mathbf{A}) = \| \mathbf{X}\mathbf{A} - \mathbf{Y} \|^2 + \alpha \| \mathbf{A} \|^2 \tag{3.113}$$

Following a similar procedure, one can obtain the LSE estimate to minimize the quantity above in the following form.

$$\mathbf{A}^{*}_{LSE} = (\mathbf{X}'\mathbf{X} + \alpha\mathbf{I})^{-1}\mathbf{X}'\mathbf{Y} \tag{3.114}$$

The LSE solution in Eq. (3.112) can be used for polynomial functions too. In the problem of polynomial curve fitting using the least square criterion, we are aiming to find the coefficients $\mathbf{A} = (a_0, a_1, a_2, \cdots, a_d)'$ that minimize the following quantity:

$$\min_{a_0, a_1, a_2, \cdots, a_d} E(Y - \hat{Y})^2 \tag{3.115}$$

where $\hat{Y} = a_0 + a_1 x + a_2 x^2 + \cdots + a_d x^d$

To obtain the LSE estimate of coefficients $\mathbf{A} = (a_0, a_1, a_2, \cdots, a_d)'$, simply change the formation of matrix $\mathbf{X}$ in Eq. (3.108) to the following:

$$\mathbf{X} = \begin{pmatrix} 1 & x_1 & \cdots & x_1^d \\ 1 & x_2 & \cdots & x_2^d \\ \vdots & \vdots & & \vdots \\ 1 & x_n & \cdots & x_n^d \end{pmatrix} \tag{3.116}$$

Note that x_i^j in Eq. (3.108) means the j-th dimension of sample $\mathbf{x}_i$, while x_i^j in Eq. (3.116) means j-th order of value x_i. Therefore, the LSE estimate of polynomial coefficients $\mathbf{A}_{LSE} = (a_0, a_1, a_2, \cdots, a_d)'$ has the same form as Eq. (3.112).

3.2.1.3. MMSE/LSE for Nonlinear Functions

As the most general case, consider solving the following minimization problem:

$$\min_{g(\bullet) \in G_M} E\left[Y - g(X)\right]^2 \tag{3.117}$$

Since we need to deal with all possible nonlinear functions, taking a derivative does not work here. Instead, we use the property of conditional expectation to solve this minimization problem. By applying Eq. (3.38) to (3.117), we get

$$\begin{aligned} E_{X,Y}\left[Y-g(X)\right]^2 &= E_X\left\{E_{Y|X}\left[\left[Y-g(X)\right]^2 \mid X=x\right]\right\} \\ &= \int_{-\infty}^{\infty} E_{Y|X}\left[\left[Y-g(X)\right]^2 \mid X=x\right] f_X(x)dx \\ &= \int_{-\infty}^{\infty} E_{Y|X}\left[\left[Y-g(x)\right]^2 \mid X=x\right] f_X(x)dx \end{aligned} \tag{3.118}$$

Since the integrand is nonnegative in Eq. (3.118), the quantity in Eq. (3.117) will be minimized at the same time the following equation is minimized.

$$\min_{g(x)\in R} E_{Y|X}\left[\left[Y-g(x)\right]^2 \mid X=x\right] \tag{3.119}$$

Since $g(x)$ is a constant in the calculation of the conditional expectation above, the MMSE estimate can be obtained in the same way as the constant functions in Section 3.2.1.1. Thus, the MMSE estimate should take the following form:

$$\ddot{Y} = g_{MMSE}(X) = E_{Y|X}(Y \mid X) \tag{3.120}$$

If the value $X=x$ is observed and the value $E(Y \mid X=x)$ is used to predict Y, the mean squared error (MSE) is minimized and specified as follows:

$$E_{Y|X}\left[\left[Y-E_{Y|X}(Y \mid X=x)\right]^2 \mid X=x\right] = Var_{Y|X}(Y \mid X=x) \tag{3.121}$$

The overall MSE, averaged over all the possible values of X, is:

$$E_X\left[Y-E_{Y|X}(Y \mid X)\right]^2 = E_X\left\{E_{Y|X}\left[\left[Y-E_{Y|X}(Y \mid X)\right]^2 \mid X\right]\right\} = E_X\left[{}_{Y|X}Var(Y \mid X=x)\right] \tag{3.122}$$

It is important to distinguish between the overall MSE $E_X\left[Var_{Y|X}(Y \mid X)\right]$ and the MSE of the particular estimate when $X=x$, which is $Var_{Y|X}(Y \mid X=x)$. Before the value of X is observed, the expected MSE for the process of observing X and predicting Y is $E_X\left[Var_{Y|X}(Y \mid X)\right]$. On the other hand, after a particular value x of X has been observed and the prediction $E_{Y|X}(Y \mid X=x)$ has been made, the appropriate measure of MSE of the prediction is $Var_{Y|X}(Y \mid X=x)$.

In general, the form of the MMSE estimator for nonlinear functions depends on the form of the joint distribution of X and Y. There is no mathematical closed-form solution. To get the conditional expectation in Eq. (3.120), we have to perform the following integral:

$$\hat{Y}(x) = \int_{-\infty}^{\infty} y f_Y(y \mid X=x)dy \tag{3.123}$$

It is difficult to solve this integral calculation. First, different measures of x could determine different conditional pdf for the integral. Exact information about the pdf is often impossible to obtain. Second, there could be no analytic solution for the integral. Those difficulties reduce the interest of the MMSE estimation of nonlinear functions to theoretical aspects only. The same difficulties also exist for LSE estimation for nonlinear functions. Some certain classes of well-behaved nonlinear functions are typically assumed for LSE problems and numeric methods are used to obtain LSE estimate from sample data.

3.2.2. Maximum Likelihood Estimation

Maximum likelihood estimation (MLE) is the most widely used parametric estimation method, largely because of its efficiency. Suppose that a set of random samples $\mathbf{X}=\{X_1,X_2,\ldots,X_n\}$ is to be drawn independently according to a discrete or continuous distribution with the pf or the pdf $p(x|\Phi)$, where the parameter vector Φ belongs to some parameter space Ω. Given an observed vector $\mathbf{x}=(x_1,\cdots,x_n)$, the *likelihood* of the set of sample data vectors $\mathbf{x}$ with respect to Φ is defined as the joint pf or joint pdf $p_n(\mathbf{x}|\Phi)$; $p_n(\mathbf{x}|\Phi)$ is also referred to as the *likelihood function*.

MLE assumes the parameters of pdfs are fixed but unknown and aims to find the set of parameters that maximizes the likelihood of generating the observed data. For example, if the pdf $p_n(\mathbf{x}|\Phi)$ is assumed to be a Gaussian distribution $N(\boldsymbol{\mu},\boldsymbol{\Sigma})$, the components of Φ will then include exactly the components of mean-vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. Since $X_1,X_2,\ldots,X_n$ are independent random variables, the likelihood can be rewritten as follows:

$$p_n(\mathbf{x}|\Phi)=\prod_{k=1}^{n} p(x_k|\Phi) \tag{3.124}$$

The likelihood $p_n(\mathbf{x}|\Phi)$ can be viewed as the probability of generating the sample data set $\mathbf{x}$ based on parameter set Φ. The *maximum likelihood estimator* of Φ is denoted as Φ_{MLE} that maximizes the likelihood $p_n(\mathbf{x}|\Phi)$. That is,

$$\Phi_{MLE}=\underset{\Phi}{\operatorname{argmax}}\, p_n(\mathbf{x}|\Phi) \tag{3.125}$$

This estimation method is called the *maximum likelihood estimation* method and is often abbreviated as MLE. Since the logarithm function is a monotonically increasing function, the parameter set Φ_{MLE} that maximizes the log-likelihood should also maximize the likelihood. If $p_n(\mathbf{x}|\Phi)$ is differentiable function of Φ, Φ_{MLE} can be attained by taking the partial derivative with respect to Φ and setting it to zero. Specifically, let Φ be a k-component parameter vector $\Phi=(\Phi_1,\Phi_2,\ldots,\Phi_k)^t$ and ∇_Φ be the gradient operator:

$$\nabla_{\Phi} = \begin{bmatrix} \dfrac{\partial}{\partial \Phi_1} \\ \vdots \\ \dfrac{\partial}{\partial \Phi_k} \end{bmatrix} \tag{3.126}$$

The log-likelihood becomes:

$$l(\Phi) = \log p_n(\mathbf{x} \mid \Phi) = \sum_{k=1}^{n} \log p(x_k \mid \Phi) \tag{3.127}$$

and its partial derivative is:

$$\nabla_{\Phi}\, l(\Phi) = \sum_{k=1}^{n} \nabla_{\Phi} \log p(x_k \mid \Phi) \tag{3.128}$$

Thus, the maximum likelihood estimate of Φ can be obtained by solving the following set of k equations:

$$\nabla_{\Phi}\, l(\Phi) = 0 \tag{3.129}$$

Example 3.1

Let's take a look at the maximum likelihood estimator of a univariate Gaussian pdf, given as the following equation:

$$p(x \mid \Phi) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2} \right] \tag{3.130}$$

where μ and σ^2 are the mean and the variance respectively. The parameter vector Φ denotes (μ, σ^2). The log-likelihood is:

$$\begin{aligned} \log p_n(\mathbf{x} \mid \Phi) &= \sum_{k=1}^{n} \log p(x_k \mid \Phi) \\ &= \sum_{k=1}^{n} \log\left(\frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(x_k-\mu)^2}{2\sigma^2} \right] \right) \\ &= -\frac{n}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \sum_{k=1}^{n} (x_k - \mu)^2 \end{aligned} \tag{3.131}$$

and the partial derivative of the above expression is:

$$\frac{\partial}{\partial \mu}\log p_n(x|\Phi)=\sum_{k=1}^{n}\frac{1}{\sigma^2}(x_k-\mu)$$
$$\frac{\partial}{\partial \sigma^2}\log p_n(x|\Phi)=-\frac{n}{2\sigma^2}+\sum_{k=1}^{n}\frac{(x_k-\mu)^2}{2\sigma^4} \tag{3.132}$$

We set the two partial differential derivatives to zero,

$$\sum_{k=1}^{n}\frac{1}{\sigma^2}(x_k-\mu)=0$$
$$-\frac{n}{\sigma^2}+\sum_{k=1}^{n}\frac{(x_k-\mu)^2}{\sigma^4}=0 \tag{3.133}$$

The maximum likelihood estimates for μ and σ^2 are obtained by solving the above equations:

$$\mu_{MLE}=\frac{1}{n}\sum_{k=1}^{n}x_k=E(x)$$
$$\sigma^2_{MLE}=\frac{1}{n}\sum_{k=1}^{n}(x_k-\mu_{MLE})^2=E\left[(x-\mu_{MLE})^2\right] \tag{3.134}$$

Equation (3.134) indicates that the maximum likelihood estimation for mean and variance is just the sample mean and variance.

Example 3.2

For the multivariate Gaussian pdf $p(\mathbf{x})$

$$p(\mathbf{x}|\Phi)=\frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}}\exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})'\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right] \tag{3.135}$$

The maximum likelihood estimates of $\boldsymbol{\mu}$ and Σ can be obtained by a similar procedure.

$$\hat{\boldsymbol{\mu}}_{MLE}=\frac{1}{n}\sum_{k=1}^{n}\mathbf{x}_k$$
$$\hat{\Sigma}_{MLE}=\frac{1}{n}\sum_{k=1}^{n}(\mathbf{x}_k-\hat{\boldsymbol{\mu}}_{MLE})(\mathbf{x}_k-\hat{\boldsymbol{\mu}}_{MLE})'=E\left[(\mathbf{x}_k-\hat{\boldsymbol{\mu}}_{MLE})(\mathbf{x}_k-\hat{\boldsymbol{\mu}}_{MLE})'\right] \tag{3.136}$$

Once again, the maximum likelihood estimation for mean vector and covariance matrix is the sample mean vector and sample covariance matrix.

In some situations, maximum likelihood estimation of Φ may not exist, or the maximum likelihood estimator may not be uniquely defined, i.e., there may be more than one MLE of Φ for a specific set of sample values. Fortunately, according to Fisher's theorem, for most practical problems with a well-behaved family of distributions, the MLE exists and is uniquely defined [4, 25, 26].

In fact, the maximum likelihood estimator can be proven to be sound under certain conditions. As mentioned before, the estimator $\theta(\mathbf{X})$ is a function of the vector of random variables $\mathbf{X}$ that represent the sample data. $\theta(\mathbf{X})$ itself is also a random variable, with a distribution determined by joint distributions of $\mathbf{X}$. Let $\tilde{\Phi}$ be the parameter vector of true distribution $p(x|\Phi)$ from which the samples are drawn. If the following three conditions hold:

1. The sample $\mathbf{x}$ is a drawn from the assumed family of distribution,
2. The family of distributions is well behaved,
3. The sample $\mathbf{x}$ is large enough,

then maximum likelihood estimator, Φ_{MLE}, has a Gaussian distribution with a mean $\tilde{\Phi}$ and a variance of the form $1/nB_x^2$ [26], where n is the size of sample and B_x is the *Fisher information*, which is determined solely by $\tilde{\Phi}$ and $\mathbf{x}$. An estimator is said to be *consistent*, iff the estimate will converge to the true distribution when there is infinite number of training samples.

$$\lim_{n\to\infty} \Phi_{MLE} = \tilde{\Phi} \tag{3.137}$$

Φ_{MLE} is a consistent estimator based on the analysis above. In addition, it can be shown that no consistent estimator has a lower variance than Φ_{MLE}. In other words, no estimator provides a closer estimate of the true parameters than the maximum likelihood estimator.

3.2.3. Bayesian Estimation and MAP Estimation

Bayesian estimation has a different philosophy than maximum likelihood estimation. While MLE assumes that the parameter Φ[3] is fixed but unknown, Bayesian estimation assumes that the parameter Φ itself is a random variable with a prior distribution $p(\Phi)$. Suppose we observe a sequence of random samples $\mathbf{x}=\{x_1, x_2, \ldots, x_n\}$, which are i.i.d. with a pdf $p(x|\Phi)$. According to Bayes' rule, we have the posterior distribution of Φ as:

$$p(\Phi|\mathbf{x}) = \frac{p(\mathbf{x}|\Phi)p(\Phi)}{p(\mathbf{x})} \propto p(\mathbf{x}|\Phi)p(\Phi) \tag{3.138}$$

[3] For simplicity, we assume the parameter Φ is a scalar instead of a vector here. However, the extension to a parameter vector Φ can be derived according to a similar procedure.

In Eq. (3.138), we dropped the denominator $p(\mathbf{x})$ here because it is independent of the parameter Φ. The distribution in Eq. (3.138) is called the posterior distribution of Φ because it is the distribution of Φ after we observed the values of random variables $X_1, X_2, \ldots, X_n$.

3.2.3.1. Prior and Posterior Distributions

For mathematical tractability, conjugate priors are often used in Bayesian estimation. Suppose a random sample is taken of a known distribution with pdf $p(\mathbf{x}|\Phi)$. A conjugate prior for the random variable (or vector) is defined as the prior distribution for the parameters of the probability density function of the random variable (or vector), such that the class-conditional pdf $p(\mathbf{x}|\Phi)$, the posterior distribution $p(\Phi|\mathbf{x})$, and the prior distribution $p(\Phi)$ belong to the same distribution family. For example, it is well known that the conjugate prior for the mean of a Gaussian pdf is also a Gaussian pdf [4]. Now, let's derive such a posterior distribution $p(\Phi|\mathbf{x})$ from the widely used Gaussian conjugate prior.

Example 3.3

Suppose $X_1, X_2, \ldots, X_n$ are drawn from a Gaussian distribution for which the mean Φ is a random variable and the variance σ^2 is known. The likelihood function $p(\mathbf{x}|\Phi)$ can be written as:

$$p(\mathbf{x}|\Phi) = \frac{1}{(2\pi)^{n/2}\sigma^n}\exp\left[-\frac{1}{2}\sum_{i=1}^{n}\left(\frac{x_i-\Phi}{\sigma}\right)^2\right] \propto \exp\left[-\frac{1}{2}\sum_{i=1}^{n}\left(\frac{x_i-\Phi}{\sigma}\right)^2\right] \tag{3.139}$$

To further simply Eq. (3.139), we could use Eq. (3.140)

$$\sum_{i=1}^{n}(x_i-\Phi)^2 = n(\Phi-\bar{x}_n)^2 + \sum_{i=1}^{n}(x_i-\bar{x}_n)^2 \tag{3.140}$$

where $\bar{x}_n = \frac{1}{n}\sum_{i=1}^{n}x_i =$ the sample mean of $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$.

Let's rewrite $p(\mathbf{x}|\Phi)$ in Eq. (3.139) into Eq. (3.141):

$$p(\mathbf{x}|\Phi) \propto \exp\left[-\frac{n}{2\sigma^2}(\Phi-\bar{x}_n)^2\right]\exp\left[-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i-\bar{x}_n)^2\right] \tag{3.141}$$

Now suppose the prior distribution of Φ is also a Gaussian distribution with mean μ and variance ν^2, i.e., the prior distribution $p(\Phi)$ is given as follows:

$$p(\Phi) = \frac{1}{(2\pi)^{1/2}\nu}\exp\left[-\frac{1}{2}\left(\frac{\Phi-\mu}{\nu}\right)^2\right] \propto \exp\left[-\frac{1}{2}\left(\frac{\Phi-\mu}{\nu}\right)^2\right] \tag{3.142}$$

By combining Eqs. (3.141) and (3.142) while dropping the second term in Eq. (3.141) we could attain the posterior pdf $p(\Phi \mid \mathbf{x})$ in the following equation:

$$p(\Phi \mid \mathbf{x}) \propto \exp\left\{-\frac{1}{2}\left[\frac{n}{\sigma^2}(\Phi - \bar{x}_n)^2 + \frac{1}{\nu^2}(\Phi - \mu)^2\right]\right\} \tag{3.143}$$

Now if we define ρ and τ as follows:

$$\rho = \frac{\sigma^2\mu + n\nu^2\bar{x}_n}{\sigma^2 + n\nu^2} \tag{3.144}$$

$$\tau^2 = \frac{\sigma^2\nu^2}{\sigma^2 + n\nu^2} \tag{3.145}$$

We can rewrite Eq. (3.143) as:

$$p(\Phi \mid \mathbf{x}) \propto \exp\left\{-\frac{1}{2}\left[\frac{1}{\tau^2}(\Phi - \rho)^2 + \frac{n}{\sigma^2 + n\nu^2}(\bar{x}_n - \mu)^2\right]\right\} \tag{3.146}$$

Since the second term in Eq. (3.146) does not depend on Φ, it can be absorbed in the constant factor. Finally, we have the posterior pdf in the following form:

$$p(\Phi \mid \mathbf{x}) = \frac{1}{\sqrt{2\pi}\tau}\exp\left[\frac{-1}{2\tau^2}(\Phi - \rho)^2\right] \tag{3.147}$$

Equation (3.147) shows that the posterior pdf $p(\Phi \mid \mathbf{x})$ is a Gaussian distribution with mean ρ and variance τ^2 as defined in Eqs. (3.144) and (3.145). The Gaussian prior distribution defined in Eq. (3.142) is a conjugate prior.

3.2.3.2. General Bayesian Estimation

The foremost requirement of a good estimator θ is that it can yield an estimate of Φ ($\theta(\mathbf{X})$) which is close to the real value Φ. In other words, a good estimator is one for which it is highly probable that the error $\theta(\mathbf{X}) - \Phi$ is close to 0. In general, we can define a loss function[4] $R(\Phi, \bar{\Phi})$. It measures the loss or cost associated with the fact that the true value of the parameter is Φ while the estimate is $\bar{\Phi}$. When only the prior distribution

[4] Bayesian estimation and loss functions are based on Bayes' decision theory, described in Chapter 4.

$p(\Phi)$ is available and no sample data has been observed, if we choose one particular estimate $\bar{\Phi}$, the expected loss is:

$$E\left[R(\Phi,\bar{\Phi})\right]=\int R(\Phi,\bar{\Phi})p(\Phi)d\Phi \tag{3.148}$$

The fact that we could derive posterior distribution from the likelihood function and the prior distribution [as shown in the derivation of Eq. (3.147)] is very important here because it allows us to compute the expected posterior loss after sample vector $\mathbf{x}$ is observed. The expected posterior loss associated with estimate $\bar{\Phi}$ is:

$$E\left[R(\Phi,\bar{\Phi})\mid\mathbf{x}\right]=\int R(\Phi,\bar{\Phi})p(\Phi\mid\mathbf{x})d\Phi \tag{3.149}$$

The Bayesian estimator of Φ is defined as the estimator that attains minimum Bayes risk, that is, minimizes the expected posterior loss function (3.149). Formally, the Bayesian estimator is chosen according to:

$$\theta_{Bayes}(\mathbf{x})=\underset{\hat{\theta}}{\operatorname{argmin}}\ E\left[R(\Phi,\theta(\mathbf{x}))\mid\mathbf{x}\right] \tag{3.150}$$

The Bayesian estimator of Φ is the estimator θ_{Bayes} for which Eq. (3.150) is satisfied for every possible value of $\mathbf{x}$ of random vector $\mathbf{X}$. Therefore, the form of the Bayesian estimator θ_{Bayes} should depend only on the loss function and the prior distribution, but not the sample value $\mathbf{x}$.

One of the most common loss functions used in statistical estimation is the mean squared error function [20]. The mean squared error function for Bayesian estimation should have the following form:

$$R(\Phi,\theta(\mathbf{x}))=(\Phi-\theta(\mathbf{x}))^2 \tag{3.151}$$

In order to find the Bayesian estimator, we are seeking θ_{Bayes} to minimize the expected posterior loss function:

$$E\left[R(\Phi,\theta(\mathbf{x}))\mid\mathbf{x}\right]=E\left[(\Phi-\theta(\mathbf{x}))^2\mid\mathbf{x}\right]=E(\Phi^2\mid\mathbf{x})-2\theta(\mathbf{x})E(\Phi\mid\mathbf{x})-\theta(\mathbf{x})^2 \tag{3.152}$$

The minimum value of this function can be obtained by taking the partial derivative of Eq. (3.152) with respect to $\theta(\mathbf{x})$. Since the above equation is simply a quadratic function of $\theta(\mathbf{x})$, it can be shown that the minimum loss can be achieved when θ_{Bayes} is chosen based on the following equation:

$$\theta_{Bayes}(\mathbf{x})=E(\Phi\mid\mathbf{x}) \tag{3.153}$$

Equation (3.153) translates into the fact that the Bayesian estimate of the parameter Φ for mean squared error function is equal to the mean of the posterior distribution of Φ. In the following section, we discuss another popular loss function (MAP estimation) that also generates the same estimate for certain distribution functions.

3.2.3.3. MAP Estimation

One intuitive interpretation of Eq. (3.138) is that a prior pdf $p(\Phi)$ represents the relative likelihood before the values of $X_1, X_2, \ldots, X_n$ have been observed; while the posterior pdf $p(\Phi \mid \mathbf{x})$ represents the relative likelihood after the values of $X_1, X_2, \ldots, X_n$ have been observed. Therefore, choosing an estimate $\bar{\Phi}$ that maximizes the posterior probability is consistent without intuition. This estimator is in fact the *maximum posterior probability* (MAP) estimator and is the most popular Bayesian estimator.

The loss function associated with the MAP estimator is the so-called uniform loss function [20]:

$$R(\Phi, \theta(\mathbf{x})) = \begin{cases} 0, & \text{if } |\theta(\mathbf{x}) - \Phi| \le \Delta \\ 1, & \text{if } |\theta(\mathbf{x}) - \Phi| > \Delta \end{cases} \quad \text{where } \Delta > 0 \tag{3.154}$$

Now let's see how this uniform loss function results in MAP estimation. Based on the loss function defined above, the expected posterior loss function is:

$$\begin{aligned} E(R(\Phi, \theta(\mathbf{x})) \mid \mathbf{x}) &= P(|\theta(\mathbf{x}) - \Phi| > \Delta \mid \mathbf{x}) \\ &= 1 - P(|\theta(\mathbf{x}) - \Phi| \le \Delta \mid \mathbf{x}) = 1 - \int_{\theta(\mathbf{x})-\Delta}^{\theta(\mathbf{x})+\Delta} p(\Phi \mid \mathbf{x}) \end{aligned} \tag{3.155}$$

The quantity in Eq. (3.155) is minimized by maximizing the shaded area under $p(\Phi \mid \mathbf{x})$ over the interval $[\theta(\mathbf{x}) - \Delta, \theta(\mathbf{x}) + \Delta]$ in Figure 3.16. If $p(\Phi \mid \mathbf{x})$ is a smooth curve and Δ is small enough, the shaded area can be computed roughly as:

$$\int_{\theta(\mathbf{x})-\Delta}^{\theta(\mathbf{x})+\Delta} p(\Phi \mid \mathbf{x}) \cong 2\Delta p(\Phi \mid \mathbf{x})\Big|_{\Phi=\theta(\mathbf{x})} \tag{3.156}$$

Thus, the shaded area can be approximately maximized by choosing $\theta(\mathbf{x})$ to be the maximum point of $p(\Phi \mid \mathbf{x})$. This concludes our proof the using the error function in Eq. (3.154) indeed will generate MAP estimator.

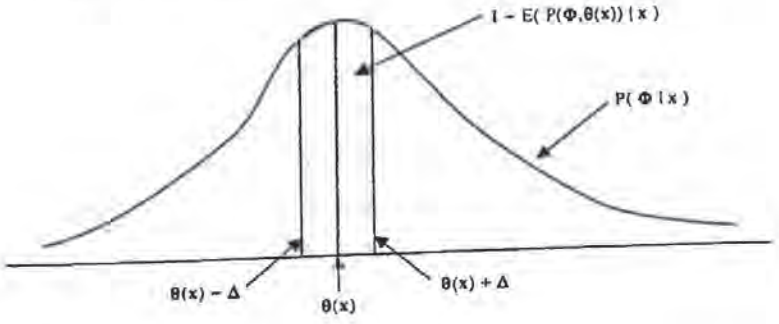


Figure 3.16 Illustration of the minimum expected posterior loss function for MAP estimation [20].

MAP estimation is to find the parameter estimate Φ_{MAP} or estimator $\theta_{MAP}(\mathbf{x})$ that maximizes the posterior probability,

$$\Phi_{MAP} = \theta_{MAP}(\mathbf{x}) = \underset{\Phi}{\operatorname{argmax}}\ p(\Phi \mid \mathbf{x}) = \underset{\Phi}{\operatorname{argmax}}\ p(\mathbf{x} \mid \Phi) p(\Phi) \tag{3.157}$$

Φ_{MAP} can also be specified in the logarithm form as follows:

$$\Phi_{MAP} = \underset{\Phi}{\operatorname{argmax}}\ \left[\log p(\mathbf{x} \mid \Phi) + \log p(\Phi)\right] \tag{3.158}$$

Φ_{MAP} can be attained by solving the following partial differential equation:

$$\frac{\partial \log p(\mathbf{x} \mid \Phi)}{\partial \Phi} + \frac{\partial \log p(\Phi)}{\partial \Phi} = 0 \tag{3.159}$$

Thus the MAP equation for finding Φ_{MAP} can be established.

$$\left.\frac{\partial \log p(\mathbf{x} \mid \Phi)}{\partial \Phi}\right|_{\Phi=\Phi_{MAP}} = \left.\frac{-\partial \log p(\Phi)}{\partial \Phi}\right|_{\Phi=\Phi_{MAP}} \tag{3.160}$$

There are interesting relationships between MAP estimation and MLE estimation. The prior distribution is viewed as the knowledge of the statistics of the parameters of interest before any sample data is observed. For the case of MLE, the parameter is assumed to be fixed but unknown. That is, there is no preference (knowledge) of what the values of parameters should be. The prior distribution $p(\Phi)$ can only be set to constant for the entire parameter space, and this type of prior information is often referred to as *non-informative prior* or *uniform prior*. By substituting $p(\Phi)$ with a uniform distribution in Eq. (3.157), MAP estimation is identical to MLE. In this case, the parameter estimation is solely determined by the observed data. A sufficient amount of training data is often a requirement for MLE. On the other hand, when the size of the training data is limited, the use of the prior density becomes valuable. If some prior knowledge of the distribution of the parameters can be obtained, MAP estimation provides a way of incorporating prior information in the parameter learning process.

Example 3.4

Now, let's formulate MAP estimation for Gaussian densities. As described in Section 3.2.3.1, the conjugate prior distribution for a Gaussian density is also a Gaussian distribution. Similarly, we assumed random variables $X_1, X_2, \ldots, X_n$ drawn from a Gaussian distribution for which the mean Φ is unknown and the variance σ^2 is known, while the conjugate prior distribution of Φ is a Gaussian distribution with mean μ and variance ν^2. It is shown in Section 3.2.3.1 that the posterior pdf can be formulated as in Eq. (3.147). The MAP estimation for Φ can be solved by taking the derivative of Eq. (3.147) with respect to Φ:

$$\Phi_{MAP} = \rho = \frac{\sigma^2 \mu + n\nu^2 \bar{x}_n}{\sigma^2 + n\nu^2} \tag{3.161}$$

where n is the total number of training samples and $\bar{x}_n$ the sample mean.

The MAP estimate of the mean Φ is a weighted average of the sample mean $\bar{x}_n$ and the prior mean. When n is zero (when there is no training data at all), the MAP estimate is simply the prior mean μ. On the other hand, when n is large ($n \to \infty$), the MAP estimate will converge to the maximum likelihood estimate. This phenomenon is consistent with our intuition and is often referred to as *asymptotic equivalence* or *asymptotic convergence*. Therefore, in practice, the difference between MAP estimation and MLE is often insignificant when a large amount of training data is available. When the prior variance ν^2 is very large (e.g., $\nu^2 >> \sigma^2 / n$), the MAP estimate will converge to the ML estimate because a very large ν^2 translates into a non-informative prior.

It is important to note that the requirement of learning prior distributions for MAP estimation is critical. In some cases, the prior distribution is very difficult to estimate and MLE is still an attractive estimation method. As mentioned before, the MAP estimation framework is particularly useful for dealing with sparse data, such as parameter adaptation. For example, in speaker adaptation, the speaker-independent (or multiple speakers) database can be used to first estimate the prior distribution [9]. The model parameters are adapted to a target speaker through a MAP framework by using limited speaker-specific training data as discussed in Chapter 9.

3.3. SIGNIFICANCE TESTING

Significance testing is one of the most important theories and methods of statistical *inference*. A problem of statistical inference, or, more simply, a statistics problem, is one in which data that have been generated in accordance with some unknown probability distribution must be analyzed, and some type of inference about the unknown distribution must be made. Hundreds of test procedures have developed in statistics for various kinds of hypotheses testing. We focus only on tests that are used in spoken language systems.

The selection of appropriate models for the data or systems is essential for spoken language systems. When the distribution of certain sample data is unknown, it is usually appropriate to make some assumptions about the distribution of the data with a distribution function whose properties are well known. For example, people often use Gaussian distributions to model the distribution of background noise in spoken language systems. One important issue is how good our assumptions are, and what the appropriate values of the parameters for the distributions are, even when we can use the methods in Section 3.2 to estimate parameters from sample data. Statistical tests are often applied to determine if the distribution with specific parameters is appropriate to model the sample data. In this section, we describe the most popular testing method for the goodness of distribution fitting – the χ^2 goodness-of-fit test.

Another important type of statistical tests is designed to evaluate the excellence of two different methods or algorithms for the same tasks when there is uncertainty regarding the results. To assure that the two systems are evaluated on the same or similar conditions, experimenters often carefully choose similar or even the exactly same data sets for testing. This is why we refer to this type of statistical test as a *paired observations* test. In both speech recognition and speech synthesis, the paired observations test is a very important tool for interpreting the comparison results.

3.3.1. Level of Significance

We now consider statistical problems involving a parameter ϕ whose value is unknown but must lie in a certain parameter space Ω. In statistical tests, we let H_0 denote the hypothesis that $\phi \in \Omega_0$ and let H_1 denote the hypothesis that $\phi \in \Omega_1$. The subsets Ω_0 and Ω_1 are disjoint and $\Omega_0 \cup \Omega_1 = \Omega$, so exactly one of the hypotheses H_0 and H_1 must be true. We must now decide whether to accept H_0 or H_1 by observing a random sample $X_1, \cdots, X_n$ drawn from a distribution involving the unknown parameter ϕ. A problem like this is called hypotheses testing. A procedure for deciding whether to accept H_0 or H_1 is called a *test procedure* or simply a *test*. The hypothesis H_0 is often referred to as the *null hypothesis* and the hypothesis H_1 as the *alternative hypothesis*. Since there are only two possible decisions, accepting H_0 is equivalent to rejecting H_1 and rejecting H_0 is equivalent to accepting H_1. Therefore, in testing hypotheses, we often use the terms *accepting or rejecting the null hypothesis* H_0 as the only decision choices.

Usually we are presented with a random sample $\mathbf{X} = (X_1, \cdots, X_n)$ to help us in making the test decision. Let S denote the sample space of n-dimensional random vector $\mathbf{X}$. The testing procedure is equivalent to partitioning the sample space S into two subsets. One subset specifies the values of $\mathbf{X}$ for which one will accept H_0 and the other subset specifies the values of $\mathbf{X}$ for which one will reject H_0. The second subset is called the *critical region* and is often denoted as C.

Since there is uncertainty associated with the test decision, for each value of $\phi \in \Omega$, we are interested in the probability $\rho(\phi)$ that the testing procedure rejects H_0. The function $\rho(\phi)$ is called the *power function* of the test and can be specified as follows:

$$\rho(\phi) = P(\mathbf{X} \in C \mid \phi) \tag{3.162}$$

For $\phi \in \Omega_0$, the decision to reject H_0 is incorrect. Therefore, if $\phi \in \Omega_0$, $\rho(\phi)$ is the probability that the statistician will make an incorrect decision (false rejection). In statistical tests, an upper bound α_0 $(0 < \alpha_0 < 1)$ is specified, and we only consider tests for which $\rho(\phi) \le \alpha_0$ for every value of $\phi \in \Omega_0$. The upper bound α_0 is called the *level of significance*. The smaller α_0 is, the less likely it is that the test procedure will reject H_0. Since α_0 specifies the upper bound for false rejection, once a hypothesis is rejected by the test procedure, we can be $(1-\alpha_0)$ confident the decision is correct. In most applications, α_0 is set to be 0.05 and the test is said to be carried out at the 0.05 level of significance or 0.95 level of confidence.

We define the size α of a given test as the maximum probability, among all the values of ϕ which satisfy the null hypothesis, of making an incorrect decision.

$$\alpha = \max_{\theta \in \Omega_0} \rho(\phi) \tag{3.163}$$

Once we obtain the value of α, the test procedure is straightforward. First, the statistician specifies a certain level of significance α_0 in a given problem of testing hypotheses, then he or she rejects the null hypothesis if the size α is such that $\alpha \leq \alpha_0$.

The size α of a given test is also called the *tail area* or the *p-value* corresponding to the observed value of data sample **X** because it corresponds to tail area of the distribution. The hypothesis will be rejected if the level of significance α_0 is such that $\alpha_0 > \alpha$ and should be accepted for any value of $\alpha_0 < \alpha$. Alternatively, we can say the observed value of **X** is *just significant* at the level of significance α without using the level of significance α_0. Therefore, if we had found that the observed value of one data sample **X** was just significant at the level of 0.0001, while the other observed value of data sample **Y** was just significant at the level of 0.001, then we can conclude the sample **X** provides much stronger evidence against H_0. In statistics, an observed value of one data sample **X** is generally said to be statistically significant if the corresponding tail area is smaller than the traditional value 0.05. For cases requiring more significance (confidence), 0.01 can be used.

A statistically significant observed data sample **X** that provides strong evidence against H_0 does not necessary provide strong evidence that the actual value of ϕ is significantly far away from parameter set Ω_0. This situation can arise, particularly when the size of random data sample is large, because a test with larger sample size will in general reject hypotheses with more confidence, unless the hypothesis is indeed the true one.

3.3.2. Normal Test (Z-Test)

Suppose we need to find whether a coin toss is fair or not. Let p be the probability of heads. The hypotheses to be tested are as follows:

$H_0 : p = 1/2$

$H_1 : p \neq 1/2$

We assume that a random sample size n is taken, and let random variable M denote the number of times we observe heads as the result. The random variable M has a binomial distribution $B(n, 1/2)$. Because of the shape of binomial distribution, M can lie on either side of the mean. This is why it is called a typical *two-tailed test*. The tail area or p-value for the observed value k can be computed as:

$$p = \begin{cases} 2P(k \leq M \leq n) & \text{for } k > n/2 \\ 2P(0 \leq M \leq k) & \text{for } k < n/2 \\ 1.0 & \text{for } k = n/2 \end{cases} \tag{3.164}$$

The p-value in Eq. (3.164) can be computed directly using the binomial distribution. The test procedure will reject H_0 when p is less than the significance level α_0.

In many situations, the p-value for the distribution of data sample $\mathbf{X}$ is difficult to obtain due to the complexity of the distribution. Fortunately, if some statistic Z of the data sample $\mathbf{X}$ has some well-known distribution, the test can then be done in the Z domain instead. If n is large enough ($n > 50$), a *normal test* (or *Z-test*) can be used to approximate a binomial probability. Under H_0, the mean and variance for M are $E(M) = n/2$ and $Var(M) = n/4$. The new random variable Z is defined as,

$$Z = \frac{|M - n/2| - 1/2}{\sqrt{n/4}} \tag{3.165}$$

which can be approximated as standard Gaussian distribution $N(0,1)$ under H_0. The p-value can now be computed as $p = 2P(Z \geq z)$ where z is the realized value of Z after M is observed. Thus, H_0 is rejected if $p < \alpha_0$, where α_0 is the level of significance.

3.3.3. χ^2 Goodness-of-Fit Test

The normal test (Z-test) can be extended to test the hypothesis that a given set of data came from a certain distribution with all parameters specified. First let's look at the case of discrete distribution fitting.

Suppose that a large population consists of items of k different types and let p_i be the probability that a random selected item belongs to type i. Now, let $q_1,\ldots,q_k$ be a set of specific numbers satisfying the probabilistic constraint ($q_i \geq 0$ for $i = 1,\ldots,k$ and $\sum_{i=1}^{k} q_i = 1$). Finally, suppose that the following hypotheses are to be tested:

$H_0 : p_i = q_i$ for $i = 1,\ldots,k$

$H_1 : p_i \neq q_i$ for at least one value of i

Assume that a random sample of size n is to be taken from the given population. For $i = 1,\ldots,k$, let N_i denote the number of observations in the random sample which are of type i. Here, $N_1,\ldots,N_k$ are nonnegative numbers and $\sum_{i=1}^{k} N_i = n$. Random variables $N_1,\ldots,N_k$ have a multinomial distribution. Since the p-value for the multinomial distribution is hard to obtain, instead we use another statistic about $N_1,\ldots,N_k$. When H_0 is true, the expected number of observations of type i is nq_i. In other words, the difference between the actual number of observations N_i and the expected number nq_i should be small when H_0 is true. It seems reasonable to base the test on the differences $N_i - nq_i$ and to reject H_0 when the differences are large. It can be proved [14] that the following random variable λ

$$\lambda = \sum_{i=1}^{k} \frac{(N_i - nq_i)^2}{nq_i} \tag{3.166}$$

converges to the χ^2 distribution with $k-1$ degrees of freedom as the sample size $n \to \infty$.

A χ^2 test of goodness-of-fit can be carried out in the following way. Once a level of significance α_0 is specified, we can use the following p-value function to find critical point c:[5]

$$P(\lambda > c) = 1 - F_{\chi^2}(x = c) \simeq \alpha_0 \tag{3.167}$$

where $F_{\chi^2}(x)$ is the distribution function for χ^2 distribution. The test procedure simply rejects H_0 when the realized value λ is such that $\lambda > c$. Empirical results show that the χ^2 distribution will be a good approximation to the actual distribution of λ as long as the value of each expectation nq_i is not too small (≥ 5). The approximation should still be satisfactory if $nq_i \geq 1.5$ for $i = 1,\ldots,k$.

For continuous distributions, a modified χ^2 goodness-of-fit test procedure can be applied. Suppose that we would like to hypothesize a null hypothesis H_0 in which continuous random sample data $X_1,\ldots,X_k$ are drawn from a certain continuous distribution with all parameters specified or estimated. Also, suppose the observed values of random sample $x_1,\ldots,x_k$ are bounded within interval Ω. First, we divide the range of the hypothesized distribution into m subintervals within interval Ω such that the expected number of values, say E_i, in each interval is at least 5. For $i = 1,\ldots,k$, we let N_i denote the number of observations in the i^{th} subintervals. As in Eq. (3.166), one can prove that the following random variable λ

$$\lambda = \sum_{i=1}^{m} \frac{(N_i - E_i)^2}{E_i} \tag{3.168}$$

converges to the χ^2 distribution with $m-k-1$ degrees of freedom as the sample size $n \to \infty$, where k is the number of parameters that must be estimated from the sample data in order to calculate the expected number of values, E_i. Once the χ^2 distribution is established, the same procedure can be used to find the critical c in Eq. (3.167) to make test decisions.

Example 3.5

Suppose we are given a random variable X of sample size 100 points and we want to determine whether we can reject the following hypothesis:

$$H_0 : X \sim N(0,1) \tag{3.169}$$

To perform χ^2 goodness-of-fit test, we first divide the range of X into 10 subintervals. The corresponding probability falling in each subinterval, the expected number of points falling in each subinterval and the actual number of points falling in each subintervals [10] are illustrated in Table 3.1.

[5] Since χ^2 pdf is a monotonic function, the test is a one-tail test. Thus, we only need to calculate one tail area.

Table 3.1 The probability falling in each subinterval of an $N(0,1)$, and 100 sample points, the expected number of points falling in each subinterval, and the actual number of points falling in each subinterval [10].

Subinterval I_i	$P(X \in I_i)$	$E_i = 100P(X \in I_i)$	N_i
[-∞, -1.6]	0.0548	5.48	2
[-1.6, -1.2]	0.0603	6.03	9
[-1.2, -0.8]	0.0968	9.68	6
[-0.8, -0.4]	0.1327	13.27	11
[-0.4, 0.0]	0.1554	15.54	19
[0.0, 0.4]	0.1554	15.54	25
[0.4, 0.8]	0.1327	13.27	17
[0.8, 1.2]	0.0968	9.68	2
[1.2, 1.6]	0.0603	6.03	6
[-1.6, ∞]	0.0548	5.48	3

The value for λ can then be calculated as follows:

$$\lambda = \sum_{i=1}^{m} \frac{(N_i - E_i)^2}{E_i} = 18.286$$

Since λ can be approximated as a χ^2 distribution with $m-k-1=10-0-1=9$ degrees of freedom, the critical point c at the 0.05 level of significance is calculated[6] to be 16.919 according to Eq. (3.167). Thus, we should reject the hypothesis H_0 because the calculated λ is greater than the critical point c.

The χ^2 goodness-of-fit test at the 0.05 significance level is in general used to determine when a hypothesized distribution is not an adequate distribution to use. To accept the diswtribution as a good fit, one needs to make sure the hypothesized distribution cannot be rejected at the 0.4 to 0.5 level-of-significance. The alternative is to use the χ^2 goodness-of-fit test for a number of potential distributions and select the one with smallest calculated χ^2.

When all the parameters are specified (instead of estimated), the Kolmogorov-Smirnov test [5] can also be used for the goodness-of-fit test. The Kolmogorov-Smirnov test in general is a more powerful test procedure when the sample size is relatively small.

3.3.4. Matched-Pairs Test

In this section, we discuss experiments in which two different methods (or systems) are to be compared to learn which one is better. To assure the two methods are evaluated under

[6] In general, we use a cumulative distribution function table to find the point with specific desired cumulative probability for complicated distributions, like the χ^2 distribution.

similar conditions, two closely resemble data samples or ideally the same data sample should be used to evaluate both methods. This type of hypotheses test is called *matched-paired* test [5].

3.3.4.1. The Sign Test

For $i = 1,\ldots,n$, let p_i denote the probability that method A is better than method B when testing on the i^{th} paired data sample. We shall assume that the probability p_i has the same value p for each of the n pairs. Suppose we wish to test the null hypothesis that method A is no better than method B. That is, the hypotheses to be tested have the following form:

$H_0 : p \leq 1/2$

$H_1 : p > 1/2$

Suppose that, for each pair of data samples, either one method or the other will appear to be better, and the two methods cannot tie. Under these assumptions, the n pairs represent n Bernoulli trials, for each of which there is probability p that method A yields better performance. Thus the number of pairs M in which method A yields better performance will have a binomial distribution $B(n, p)$. For the simple sign test where one needs to decide which method is better, p will be set to $1/2$. Hence a reasonable procedure is to reject H_0 if $M > c$, where c is a critical point. This procedure is called a signed test. The critical point can be found according to

$$P(M > c) = 1 - F_B(x = c) = \alpha_0 \tag{3.170}$$

where $F_B(x)$ is the distribution for binomial distribution. Thus, for observed values $M > c$, we will reject H_0.

3.3.4.2. Magnitude-Difference Test

The only information that the sign test utilizes from each pair of data samples, is the sign of the difference between two performances. To do a sign test, one does not need to obtain a numeric measurement of the magnitude of the difference between the two performances. However, if the measurement of magnitude of the difference for each pair is available, a test procedure based on the relative magnitudes of the differences can be used [11].

We assume now that the performance of each method can be measured for any data samples. For $i = 1,\ldots,n$, let A_i denote the performance of the method A on the i^{th} pair of data samples and B_i denote the performance of the method B on the i^{th} pair of data sample. Moreover, we shall let $D_i = A_i - B_i$. Since $D_1,\ldots,D_n$ are generated on n different pairs of data samples, they should be independent random variables. We also assume that $D_1,\ldots,D_n$ have the same distribution. Suppose now we are interested in testing the null hypothesis that method A and method B have on the average the same performance on the n pairs of data samples.

Let μ_D be the mean of D_i. The MLE estimate of μ_D is:

$$\mu_D = \sum_{i=1}^{n} \frac{D_i}{n} \tag{3.171}$$

The test hypotheses are:

$H_0 : \mu_D = 0$

$H_1 : \mu_D \neq 0$

The MLE estimate of the variance of D_i is

$$\sigma_D^2 = \frac{1}{n} \sum_{i=1}^{n} (D_i - \mu_D)^2 \tag{3.172}$$

We define a new random variable Z as follows:

$$Z = \frac{\mu_D}{\sigma_D / \sqrt{n}} \tag{3.173}$$

If n is large enough (> 50), Z is proved to have a standard Gaussian distribution $N(0,1)$. The normal test procedure described in Section 3.3.2 can be used to test H_0. This type of matched-paired tests usually depends on having enough pairs of data samples for the assumption that Z can be approximated with a Gaussian distribution. It also requires enough data samples to estimate the mean and variance of the D_i.

3.4. INFORMATION THEORY

Transmission of information is a general definition of what we call communication. Claude Shannon's classic paper of 1948 gave birth to a new field in information theory that has become the cornerstone for coding and digital communication. In the paper titled "A Mathematical Theory of Communication," he wrote [21]:

> *The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point.*

Information theory is a mathematical framework for approaching a large class of problems related to encoding, transmission, and decoding information in a systematic and disciplined way. Since speech is a form of communication, information theory has served as the underlying mathematical foundation for spoken language processing.

3.4.1. Entropy

Three interpretations can be used to describe the quantity of *information*: (1) the amount of uncertainty before seeing an event, (2) the amount of surprise when seeing an event, and (3)

the amount of information after seeing an event. Although these three interpretations seem slightly different, they are virtually the same under the framework of information theory.

According to information theory, the information derivable from outcome x_i depends on its probability. If the probability $P(x_i)$ is small, we can derive a large degree of information, because the outcome that it has occurred is very rare. On the other hand, if the probability is large, the information derived will be small, because the outcome is well expected. Thus, the amount of information is defined as follows:

$$I(x_i) = \log \frac{1}{P(x_i)} \tag{3.174}$$

The reason to use a logarithm can be interpreted as follows. The information for two independent events to occur (where the joint probability is the multiplication of both individual probabilities) can be simply carried out by the addition of the individual information of each event. When the logarithm base is 2, the unit of information is called a *bit*. This means that one bit of information is required to specify the outcome. In this probabilistic framework, the amount of information represents uncertainty. Suppose X is a discrete random variable taking value x_i (referred to as a symbol) from a finite or countable infinite sample space $S = \{x_1, x_2, \ldots, x_i, \ldots\}$ (referred to as an alphabet). The symbol x_i is produced from an information source with alphabet S, according to the probability distribution of the random variable X. One of the most important properties of an information source is the entropy $H(S)$ of the random variable X, defined as the average amount of information (expected information):

$$H(X) = E[I(X)] = \sum_S P(x_i) I(x_i) = \sum_S P(x_i) \log \frac{1}{P(x_i)} = E\left[-\log P(X)\right] \tag{3.175}$$

This entropy $H(X)$ is the amount of information required to specify what kind of symbol has occurred on average. It is also the average uncertainty for the symbol. Suppose that the sample space S has an alphabet size $\|S\| = N$. The entropy $H(X)$ attains the maximum value when the pf has a uniform distribution, i.e.:

$$P(x_i) = P(x_j) = \frac{1}{N} \quad \text{for all } i \text{ and } j \tag{3.176}$$

Equation (3.176) can be interpreted to mean that *uncertainty* reaches its maximum level when no outcome is more probable than any other. It can be proved that the entropy $H(X)$ is nonnegative and becomes zero only if the probability function is a deterministic one, i.e.,

$$H(X) \geq 0 \text{ with equality i.f.f. } P(x_i) = 1 \text{ for some } x_i \in S \tag{3.177}$$

There is another very interesting property for the entropy. If we replace the pf of generating symbol x_i in Eq. (3.175) with any other arbitrary pf, the new value is no smaller than the original entropy. That is,

$$H(X) \le E[-\log Q(X)] = -\sum_S P(x_i)\log Q(x_i) \tag{3.178}$$

Equation (3.178) has a very important meaning. It shows that we are more uncertain about the data if we misestimate the distribution governing the data source. The equality for Eq. (3.178) occurs if and only if $P(x_i) = Q(x_i)\ \ 1 \le i \le N$. Equation (3.178), often referred to as *Jensen's inequality*, is the basis for the proof of EM algorithm in Chapter 4. Similarly, Jensen's inequality can be extended to continuous pdf:

$$-\int f_x(x)\log f_x(x)dx \le -\int g_x(x)\log f_x(x)dx \tag{3.179}$$

with equality occurring if and only if $f_x(x) = g_x(x)\ \ \forall x$.

The proof of Eq. (3.178) follows from the fact $\log(x) \le x-1,\ \forall x$, so the following quantity must have an non-positive value.

$$\sum_S P(x_i)\log\frac{Q(x_i)}{P(x_i)} \le \sum_S P(x_i)\left[\frac{Q(x_i)}{P(x_i)} - 1\right] = 0 \tag{3.180}$$

Based on this property, the negation of the quantity in Eq. (3.180) can be used for the measurement of the distance of two probability distributions. Specifically, the *Kullback-Leibler* (KL) *distance* (*relative entropy*, *discrimination*, or *divergence*) is defined as:

$$KL(P \| Q) = E\left[\log\frac{P(X)}{Q(X)}\right] = \sum_S P(x_i)\log\frac{P(x_i)}{Q(x_i)} \tag{3.181}$$

As discussed in Chapter 11, the branching factor of a grammar or language is an important measure of degree of difficulty of a particular task in spoken language systems. This relates to the size of the word list from which a speech recognizer or a natural language processor needs to disambiguate in a given context. According to the entropy definition above, this branching factor estimate (or average choices for an alphabet) is defined as follows:

$$PP(X) = 2^{H(X)} \tag{3.182}$$

$PP(X)$ is called the *perplexity* of source X, since it describes how confusing the grammar (or language) is. The value of perplexity is equivalent to the size of an imaginary equivalent list, whose words are equally probable. The bigger the perplexity, the higher branching factor. To find out the perplexity of English, Shannon devised an ingenious way [22] to estimate the entropy and perplexity of English words and letters. His method is similar to a guessing game where a human subject guesses sequentially the words of a text hidden from him, using the relative frequencies of her/his guesses as the estimates of the probability distribution underlying the source of the text. Shannon's perplexity estimate of English comes out to be about 2.39 for English letters and 130 for English words. Chapter 11 has a detailed description on the use of perplexity for language modeling.

3.4.2. Conditional Entropy

Now let us consider transmission of symbols through an information channel. Suppose the input alphabet is $X=(x_1,x_2,\ldots,x_s)$, the output alphabet is $Y=(y_1,y_2,\ldots,y_t)$, and the information channel is defined by the channel matrix $M_{ij}=P(y_j\mid x_i)$, where $P(y_j\mid x_i)$ is the conditional probability of receiving output symbol y_j when input symbol x_i is sent. Figure 3.17 shows an example of an information channel.

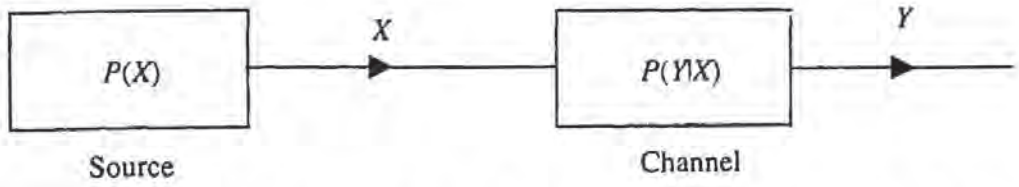


Figure 3.17 Example of information channel. The source is described by source pf $P(X)$ and the channel is characterized by the conditional pf $P(Y|X)$.

Before transmission, the average amount of information, or the uncertainty of the input alphabet X, is the prior entropy $H(X)$.

$$H(X)=\sum_X P(X=x_i)\log\frac{1}{P(X=x_i)} \tag{3.183}$$

where $P(x_i)$ is the prior probability. After transmission, suppose y_j is received; then the average amount of information, or the uncertainty of the input alphabet A, is reduced to the following *posterior* entropy:

$$H(X\mid Y=y_j)=-\sum_X P(X=x_i\mid Y=y_j)\log P(X=x_i\mid Y=y_j) \tag{3.184}$$

where the $P(x_i\mid y_j)$ are the posterior probabilities. Averaging the posterior entropy $H(X\mid y_j)$ over all output symbols y_j leads to the following equation:

$$\begin{aligned}H(X\mid Y)&=\sum_Y P(Y=y_j)H(X\mid Y=y_j)\\&=-\sum_Y P(Y=y_j)\sum_X P(X=x_i\mid Y=y_j)\log P(X=x_i\mid Y=y_j)\\&=-\sum_X\sum_Y P(X=x_i,Y=y_j)\log P(X=x_i\mid Y=y_j)\end{aligned} \tag{3.185}$$

This *conditional entropy*, defined in Eq. (3.185), is the average amount of information or the uncertainty of the input alphabet X given the outcome of the output event Y. Based on the definition of conditional entropy, we derive the following equation:

$$\begin{aligned} H(X,Y) &= -\sum_X \sum_Y P(X=x_i, Y=y_i) \log P(X=x_i, Y=y_i) \\ &= -\sum_X \sum_Y P(X=x_i, Y=y_i) \{\log P(X=x_i) + \log P(Y=y_i \mid X=x_i)\} \\ &= H(X) + H(Y \mid X) \end{aligned} \tag{3.186}$$

Equation (3.186) has an intuitive meaning – the uncertainty about two random variables equals the sum of uncertainty about the first variable and the conditional entropy for the second variable given the first variable is known. Equations (3.185) and (3.186) can be generalized to random vectors **X** and **Y** where each contains several random variables.

It can be proved that the chain rule [Eq. (3.16)] applies to entropy.

$$H(X_1, \cdots, X_n) = H(X_n \mid X_1, \cdots, X_{n-1}) + \cdots + H(X_2 \mid X_1) + H(X_1) \tag{3.187}$$

Finally, the following inequality can also be proved:

$$H(X \mid Y, Z) \le H(X \mid Y) \tag{3.188}$$

with equality i.f.f. X and Z being independent when conditioned on Y. Equation (3.188) basically confirms the intuitive belief that uncertainty decreases when more information is known.

3.4.3. The Source Coding Theorem

Information theory is the foundation for data compressing. In this section we describe *Shannon's source coding theorem*, also known as the *first coding theorem*. In source coding, we are interested in *lossless* compression, which means the compressed information (or symbols) can be recovered (decoded) perfectly. The entropy serves as the upper bound for a source lossless compression.

Consider an information source with alphabet $S = \{0,1,\ldots,N-1\}$. The goal of data compression is to *encode* the output symbols into a string of binary symbols. An interesting question arises: *What is the minimum number of bits required, on the average, to encode the output symbols of the information source?*

Let's assume we have a source that can emit four symbols {0,1,2,3} with equal probability $P(0) = P(1) = P(2) = P(3) = 1/4$. Its entropy is 2 bits as illustrated in Eq. (3.189):

$$H(S) = \sum_{i=0}^{3} P(i) \log_2 \frac{1}{P(i)} = 2 \tag{3.189}$$

It is obvious that 2 bits per symbol is good enough to encode this source. A possible binary code for this source is {00, 01, 10, 11}. It could happen, though some symbols are more likely than others, for example, $P(0) = 1/2$, $P(1) = 1/4$, $P(2) = 1/8$, $P(3) = 1/8$. In this case the entropy is only 1.75 bits. One obvious idea is to use fewer bits for lower values that are frequently used and more bits for larger values that are rarely used. To represent this

source we can use a variable-length code {0,10,110,111}, where no codeword is a prefix for the rest and thus a string of 0s and 1s can be uniquely broken into those symbols. The encoding scheme with such a property is called *uniquely decipherable* (or instantaneous) coding, because as soon as the decoder observes a sequence of codes, it can decisively determine the sequence of the original symbols. If we let $r(x)$ be the number of bits (length) used to encode symbol x, the average rate R of bits per symbol used for encoding the information source is:

$$R = \sum_x r(x)P(x) \tag{3.190}$$

In our case, R is 1.75 bits as shown in Eq. (3.191):

$$R = 0.5\times1 + 0.25\times2 + 0.125\times3 + 0.125\times3 = 1.75 \tag{3.191}$$

Such variable-length coding strategy is called *Huffman coding*. Huffman coding belongs to *entropy coding* because it matches the entropy of the source. In general, *Shannon's source coding* theorem says that a source cannot be coded with fewer bits than its entropy. We will skip the proof here. Interested readers can refer to [3, 15, 17] for the detailed proof. This theorem is consistent with our intuition because the entropy measure is exactly the information content of the information measured in bits. If the entropy increases, then uncertainty increases, resulting in a large amount of information. Therefore, it takes more bits to encode the symbols. In the case above, we are able to match this rate, but, in general, this is impossible, though we can get arbitrarily close to it. The Huffman code for this source offers a compression rate of 12.5% relative to the code designed for the uniform distribution.

Shannon's source coding theorem establishes not only the lower bound for lossless compression but also the upper bound. Let $\lceil x \rceil$ denote the smallest integer that greater or equal to x. As in the similar procedure above, we can make the code length assigned to source output x equal to

$$l(x) = \lceil -\log P(x) \rceil \tag{3.192}$$

The average length L satisfies the following inequality:

$$L = \sum_x l(x)P(x) < \sum_x [1 - \log P(x)]P(x) = 1 + H(X) \tag{3.193}$$

Equation (3.193) means that the average rate R only exceeds the value of entropy by less than one bit.

L can be made arbitrarily close to the entropy by block coding. Instead of encoding single output symbols of the information source, one can encode each block of length n. Let's assume the source is memoryless, so $X_1, X_2, \ldots, X_n$ are independent. According to Eq. (3.193), the average rate R for this block code satisfies:

$$L < 1 + H(X_1, X_2, \ldots, X_n) = 1 + nH(X) \tag{3.194}$$

This makes the average number of bits per output symbol, L/n, satisfy

$$\lim_{n\to\infty}\frac{1}{n}L \le H(X) \tag{3.195}$$

In general, Huffman coding arranges the symbols in order of decreasing probability, assigns the bit 0 to the symbol of highest probability and the bit 1 to what is left, and proceeds the same way for the second highest probability value (which now has a code 10) and iterate. This results in 2.25 bits for the uniform distribution case, which is higher than the 2 bits we obtain with equal-length codes.

Lempel-Ziv coding is a coding strategy that uses correlation to encode *strings* of symbols that occur frequently. Although it can be proved to converge to the entropy, its convergence rate is much slower [27]. Unlike Huffman coding, Lempel-Ziv coding is independent of the distribution of the source; i.e., it needs not be aware of the distribution of the source before encoding. This type of coding scheme is often referred to as *universal* encoding scheme.

3.4.4. Mutual Information and Channel Coding

Let's review the information channel illustrated in Figure 3.17. An intuitively plausible measure of the average amount of information provided by the random event Y about the random event X is the average difference between the number of bits it takes to specify the outcome of X when the outcome of Y is not known and the outcome of Y is known. *Mutual information* is defined as the difference in the entropy of X and the conditional entropy of X given Y:

$$\begin{aligned} I(X;Y) &= H(X) - H(X \mid Y) \\ &= \sum_X P(x_i)\log\frac{1}{P(x_i)} - \sum_X\sum_Y P(x_i,y_j)\log\frac{1}{P(x_i \mid y_j)} \\ &= \sum_X\sum_Y P(x_i,y_j)\log\frac{P(x_i \mid y_j)}{P(x_i)} = \sum_X\sum_Y P(x_i,y_j)\log\frac{P(x_i,y_j)}{P(x_i)P(y_i)} \\ &= E\left[\log\frac{P(X,Y)}{P(X)P(Y)}\right] \end{aligned} \tag{3.196}$$

$I(X;Y)$ is referred to as the mutual information between X and Y. $I(X;Y)$ is symmetrical; i.e., $I(X;Y) = I(Y;X)$. The quantity $P(x,y)/P(x)P(y)$ is often referred to as the *mutual information between symbol x and y*. $I(X;Y)$ is bounded:

$$0 \le I(X;Y) \le \min\left[H(X), H(Y)\right] \tag{3.197}$$

$I(X;Y)$ reaches the minimum value (zero) when the random variables X and Y are independent.

Mutual information represents the information obtained (or the reduction in uncertainty) through a channel by observing an output symbol. If the information channel is noiseless, the input symbol can be determined definitely by observing an output symbol. In this case, the conditional entropy $H(X|Y)$ equals zero and it is called a *noiseless channel*. We obtain the maximum mutual information $I(X; Y) = H(X)$. However, the information channel is generally *noisy* so that the conditional entropy $H(X|Y)$ is not zero. Therefore, maximizing the mutual information is equivalent to obtaining a low-noise information channel, which offers a closer relationship between input and output symbols.

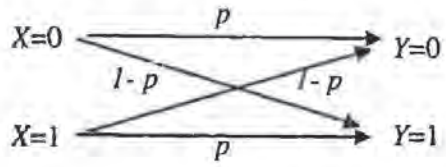


Figure 3.18 A binary channel with two symbols.

Let's assume that we have a binary channel, a channel with a binary input and output as shown in Figure 3.18. Associated with each output are a probability p that the output is correct, and a probability $(1-p)$ that it is not, so that the channel is *symmetric*.

If we observe a symbol $Y = 1$ at the output, we don't know for sure what symbol X was transmitted, though we know $P(X=1|Y=1)=p$ and $P(X=0|Y=1)=(1-p)$, so that we can measure our uncertainty about X by its conditional entropy:

$$H(X|Y=1) = -p\log p-(1-p)\log(1-p) \tag{3.198}$$

If we assume that our source X has a uniform distribution, $H(X|Y)=H(X|Y=1)$ as shown in Eq. (3.198) and $H(X) = 1$. The mutual information between X and Y is given by

$$I(X,Y) = H(X)-H(X|Y)=1+p\log p+(1-p)\log(1-p) \tag{3.199}$$

It measures the information that Y carries by about X. The channel capacity C is the maximum of the mutual information over all distributions of X. That is,

$$C=\max_{P(x)} I(X;Y) \tag{3.200}$$

The channel capacity C can be attained by varying the distribution of the information source until the mutual information is maximized for the channel. The channel capacity C can be regarded as a channel that can transmit at most C bits of information per unit of time. *Shannon's channel coding* theorem says that for a given channel there exists a code that permits error-free transmission across the channel, provided that $R \le C$, where R is the rate of the communication system, which is defined as the number of bits per unit of time being transmitted by the communication system. Shannon's channel coding theorem states the fact that *arbitrarily reliable communication is possible at any rate below channel capacity*.

Figure 3.19 illustrates a transmission channel with the source encoder and destination decoder. The source encoder will encode the source symbol sequence $\mathbf{x} = x_1, x_2, \ldots, x_n$ into

channel input sequence $y_1, y_2, \ldots, y_k$. The destination decoder takes the output sequence $z_1, z_2, \ldots, z_k$ from the channel and converts it into the estimates of the source output $\overline{\mathbf{x}} = \overline{x}_1, \overline{x}_2, \ldots, \overline{x}_n$. The goal of this transmission is to make the probability of correct decoding $P(\overline{\mathbf{x}} = \mathbf{x})$ asymptotically close to 1 while keeping the compression ratio $\Re = n/k$ as large as possible. *Shannon's source-channel coding* theorem (also referred to as *Shannon's second coding* theorem) says that it is possible to find an encoder-decoder pair of rate $\Re$ for a noisy information channel, provided that $\Re \times H(X) \leq C$.

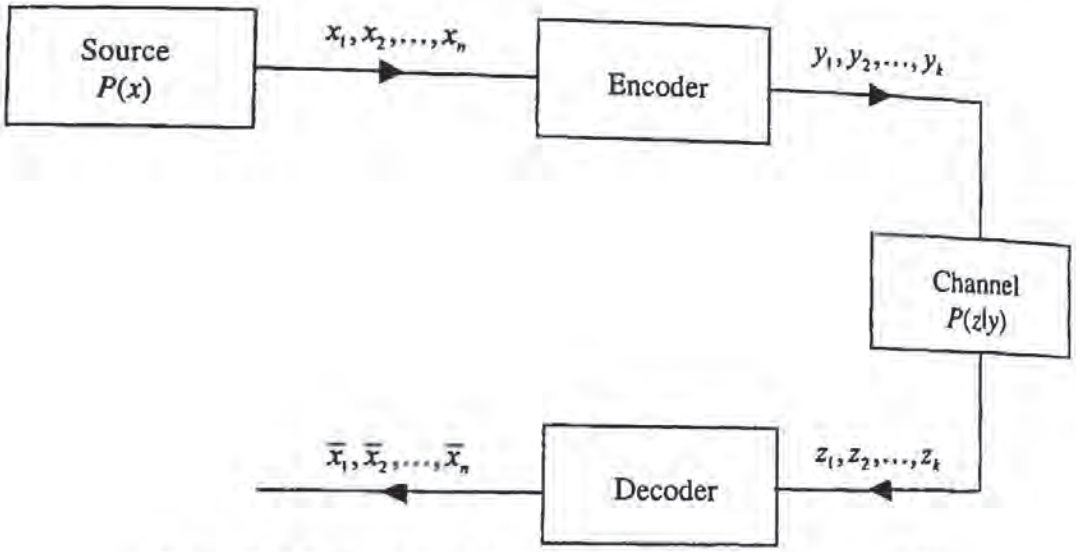


Figure 3.19 Transmission of information through a noisy channel [15].

Because of channel errors, speech coders need to provide error correction codes that will decrease the bit rate allocated to the speech. In practice, there is a tradeoff between the bit rate used for source coding and the bit rate for channel coding. In Chapter 7 we will describe speech coding in great detail.

3.5. HISTORICAL PERSPECTIVE AND FURTHER READING

The idea of uncertainty and probability can be traced all the way back to about 3500 B.C., when games of chance played with bone objects were developed in Egypt. Cubical dice with markings virtually identical to modern dice have been found in Egyptian tombs dating around 2000 B.C. Gambling with dice played an important part in the early development of probability theory. Modern mathematical theory of probability is believed to have been started by the French mathematicians Blaise Pascal (1623-1662) and Pierre Fermat (1601-1665) when they worked on certain gambling problems involving dice. English mathematician Thomas Bayes (1702-1761) was first to use probability inductively and established a mathematical basis for probability inference, leading to what is now known as Bayes' theorem. The theory of probability has developed steadily since then and has been widely ap-

plied in diverse fields of study. There are many good textbooks on probability theory. The book by DeGroot [6] is an excellent textbook for both probability and statistics which covers all the necessary elements for engineering majors. The authors also recommend [14], [19], or [24] for interested readers.

Estimation theory is a basic subject in statistics covered in textbooks. The books by DeGroot [6], Wilks [26] and Hoel [13] offer excellent discussions of estimation theory. They all include comprehensive treatments for maximum likelihood estimation and Bayesian estimation. Maximum likelihood estimation was introduced in 1912 by R. A. Fisher (1890-1962) and has been applied to various domains. It is arguably the most popular parameter estimation method due to its intuitive appeal and excellent performance with large training samples. The EM algorithm in Chapter 4 and the estimation of hidden Markov models in Chapter 8 are based on the principle of MLE. The use of prior distribution in Bayesian estimation is very controversial in statistics. Some statisticians adhere to the Bayesian philosophy of statistics by taking the Bayesian estimation view of the parameter Φ having a probability distribution. Others, however, believe that in many problems Φ is not a random variable but rather a fixed number whose value is unknown. Those statisticians believe that a prior distribution can be assigned to a parameter Φ only when there is extensive prior knowledge of the past; thus the non-informative priors are completely ruled out. Both groups of statisticians agree that whenever a meaningful prior distribution can be obtained, the theory of Bayesian estimation is applicable and useful. The books by DeGroot [6] and Poor [20] are excellent for learning the basics of Bayesian and MAP estimations. Bayesian and MAP adaptation are particularly powerful when the training samples are sparse. Therefore, they are often used for adaptation where the knowledge of prior distribution can help to adapt the model to a new but limited training set. The speaker adaptation work done by Brown et al. [2] first applied Bayesian estimation to speech recognition and [9] is another good paper on using MAP for hidden Markov models. References [4], [16] and [14] have extensive studies of different conjugate prior distributions for various standard distributions. Finally, [1] has an extensive reference for Bayesian estimation.

Significance testing is an essential tool for statisticians to interpret all the statistical experiments. Neyman and Pearson provided some of the most important pioneering work in hypotheses testing [18]. There are many different testing methods presented in most statistics books. The χ^2 test, invented in 1900 by Karl Pearson, is arguably the most widely used testing method. Again, the textbook by DeGroot [6] is an excellent source for the basics of testing and various testing methods. The authors recommend [7] as an interesting book that uses many real-world examples to explain statistical theories and methods, particularly the significance testing.

Information theory first appeared in Claude Shannon's historical paper: *A Mathematical Theory of Communication* [21]. In it, Shannon, analyzed communication as the transmission of a message from a source through a channel to a receiver. In order to solve the problem he created a new branch of applied mathematics – *information and coding theory*. IEEE published a collection of Shannon's papers [23] containing all of his published works, as well as many that have never been published. Those published include his classic papers on information theory and switching theory. Among the unpublished works are his once-

secret wartime reports, his Ph.D. thesis on population genetics, unpublished Bell Labs memoranda, and a paper on the theory of juggling. The textbook by McEliece [17] is excellent for learning all theoretical aspects of information and coding theory. However, it might be out of print now. Instead, the books by Hamming [12] and Cover [3] are two current great references for information and coding theory. Finally, F. Jelinek's *Statistical Methods for Speech Recognition* [15] approaches the speech recognition problem from an information-theoretic aspect. It is a useful book for people interested in both topics.

REFERENCES

[1] Bernardo, J.M. and A.F.M. Smith, *Bayesian Theory*, 1996, New York, John Wiley.

[2] Brown, P., C.-H. Lee, and J. Spohrer, "Bayesian Adaptation in Speech Recognition," *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1983, Boston, MA, pp. 761-764.

[3] Cover, T.M. and J.A. Thomas, *Elements of Information Theory*, 1991, New York, John Wiley and Sons.

[4] DeGroot, M.H., *Optimal Statistical Decisions*, 1970, New York, NY, McGraw-Hill.

[5] DeGroot, M.H., *Probability and Statistics*, Addison-Wesley Series in Behavioral Science: Quantitive Methods, F. Mosteller, ed., 1975, Reading, MA, Addison-Wesley Publishing Company.

[6] DeGroot, M.H., *Probability and Statistics*, 2nd ed, Addison-Wesley Series in Behavioral Science: Quantitive Methods, F. Mosteller, ed., 1986, Reading, MA, Addison-Wesley Publishing Company.

[7] Freedman, D., *et al.*, *Statistics*, 2nd ed, 1991, New York, W. W. Norton & Company, Inc.

[8] Gales, M.J., *Model Based Techniques for Noise Robust Speech Recognition*, PhD Thesis in Engineering Department 1995, Cambridge University.

[9] Gauvain, J.L. and C.H. Lee, "Maximum a Posteriori Estimation for Multivariate Gaussian Mixture Observations of Markov Chains," *IEEE Trans. on Speech and Audio Processing*, 1994, **2**(2), pp. 291-298.

[10] Gillett, G.E., *Introduction to Operations Research: A Computer-Oriented Algorithmic Approach*, McGraw-Hill Series in Industrial Engineering and Management Science, J. Riggs, ed., 1976, New York, McGraw-Hill.

[11] Gillick, L. and S.J. Cox, "Some Statistical Issues in the Comparison of Speech Recognition Algorithms," *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1989, Glasgow, Scotland, UK, IEEE, pp. 532-535.

[12] Hamming, R.W., *Coding and Information Theory*, 1986, Englewood Cliffs, NJ, Prentice-Hall.

[13] Hoel, P.G., *Introduction to Mathematical Statistics*, 5th ed., 1984, John Wiley & Sons.

[14] Jeffreys, H., *Theory of Probability*, 1961, Oxford University Press.

[15] Jelinek, F., *Statistical Methods for Speech Recognition*, Language, Speech, and Communication, 1998, Cambridge, MA, MIT Press.

[16] Lindley, D.V., "The Use of Prior Probability Distributions in Statistical Inference and Decision," *Fourth Berkeley Symposium on Mathematical Statistics and Probability*, 1961, Berkeley, CA, Univ. of California Press.

[17] McEliece, R., *The Theory of Information and Coding*, Encyclopedia of Mathematics and Its Applications, eds. R. Gian-Carlo. Vol. 3, 1977, Reading, MA, Addison-Wesley Publishing Company.

[18] Neyman, J. and E.S. Pearson, "On the Problem of the Most Efficient Tests of Statistical Hypotheses," *Philosophical Trans. of Royal Society*, 1928, **231**, pp. 289-337.

[19] Papoulis, A., *Probability, Random Variables, and Stochastic Processes*, 3rd ed., 1991, New York, McGraw-Hill.

[20] Poor, H.V., *An Introduction to Signal Detection and Estimation*, Springer Tests in Electrical Engineering, J.B. Thomas, ed., 1988, New York, Springer-Verlag.

[21] Shannon, C., "A Mathematical Theory of Communication," *Bell System Technical Journal*, 1948, **27**, pp. 379-423, 623-526.

[22] Shannon, C.E., "Prediction and Entropy of Printed English," *Bell System Technical Journal*, 1951, pp. 50-62.

[23] Shannon, C.E., *Claude Elwood Shannon: Collected Papers*, 1993, IEEE.

[24] Viniotis, Y., *Probability and Random Processes for Electrical Engineering*, Outline Series in Electronics & Electrical Engineering, Schaum, ed., 1998, New York, WCB McGraw-Hill.

[25] Wald, A., "Note of Consistency of Maximum Likelihood Estimate," *Ann. Mathematical Statistics*, 1949(20), pp. 595-601.

[26] Wilks, S.S., *Mathematical Statistics*, 1962, New York, John Wiley and Sons.

[27] Ziv, J. and A. Lempel, "A Universal Algorithm for Sequential Data Compression," *IEEE Trans. on Information Theory*, 1997, **IT-23**, pp. 337-343.

CHAPTER 4

Pattern Recognition

Spoken language processing relies heavily on pattern recognition, one of the most challenging problems for machines. In a broader sense, the ability to recognize patterns forms the core of our intelligence. If we can incorporate the ability to reliably recognize patterns in our work and life, we can make machines much easier to use. The process of human pattern recognition is not well understood.

Due to the inherent variability of spoken language patterns, we emphasize the use of statistical approaches in this book. The decision for pattern recognition is based on appropriate probabilistic models of the patterns. This chapter presents several mathematical fundamentals for statistical pattern recognition and classification. In particular, Bayes' decision theory and estimation techniques for parameters of classifiers are introduced. Bayes' decision theory, which plays a central role for statistical pattern recognition, introduces the concept of decision-making based on both *posterior* knowledge obtained from specific observation data, and *prior* knowledge of the categories. To build such a classifier or predictor, it is critical to estimate *prior* class probabilities and the class-conditional probabilities for a Bayes' classifier.

Supervised learning has class information for the data. Only the probabilistic structure needs to be learned. Maximum likelihood estimation (MLE) and maximum posterior probability estimation (MAP) that we discussed in Chapter 3 are two most powerful methods. Both MLE and MAP aim to maximize the likelihood function. The MLE criterion does not necessarily minimize the recognition error rate. Various discriminant estimation methods are introduced for that purpose. *Maximum mutual information estimation* (MMIE) is based on criteria to achieve maximum model separation (the model for the correct class is well separated from other competing models) instead of likelihood criteria. The MMIE criterion is one step closer but still is not directly related to minimizing the error rate. Other discriminant estimation methods, such as *minimum error-rate estimation*, use the ultimate goal of pattern recognition – minimizing the classification errors. *Neural networks* are one class of discriminant estimation methods.

The EM algorithm is an iterative algorithm for unsupervised learning in which class information is unavailable or only partially available. The EM algorithm forms the theoretical basis for training hidden Markov models (HMM) as described in Chapter 8. To better understand the relationship between MLE and EM algorithms, we first introduce *vector quantization* (VQ), a widely used source-coding technique in speech analysis. The well-known *k-means* clustering algorithm best illustrates the relationship between MLE and the EM algorithm. We close this chapter by introducing a powerful binary prediction and regression technique, classification and regression trees (CART). CART represents an important technique that combines rule-based expert knowledge and statistical learning.

4.1. BAYES' DECISION THEORY

Bayes' decision theory forms the basis of statistical pattern recognition. The theory is based on the assumption that the decision problem can be specified in probabilistic terms and that all of the relevant probability values are known. Bayes' decision theory can be viewed as a formalization of a common-sense procedure, i.e., the aim to achieve minimum-error-rate classification. This common-sense procedure can be best observed in the following real-world decision examples.

Consider the problem of making predictions for the stock market. We use the Dow Jones Industrial average index to formulate our example, where we have to decide tomorrow's Dow Jones Industrial average index in one of the three categories (events): *Up*, *Down*, or *Unchanged*. The available information is the probability function $P(\omega)$ of the three categories. The variable ω is a discrete random variable with the value $\omega = \omega_i$ $(i = 1,2,3)$. We call the probability $P(\omega_i)$ a *prior* probability, since it reflects prior knowledge of tomorrow's Dow Jones Industrial index. If we have to make a decision based only on the *prior* probability, the most plausible decision may be made by selecting the class ω_i with the highest prior probability $P(\omega_i)$. This decision is unreasonable, in that we always make the same decision even though we know that all three categories of Dow Jones Industrial index changes will possibly appear. If we are given further observable data, such as the federal-funds interest rate or the jobless rate, we can make a more informed decision. Let x be a con-

tinuous random variable whose value is the federal-fund interest rate, and $f_{x|\omega}(x|\omega)$ be a class-conditional pdf For simplicity, we denote the pdf $f_{x|\omega}(x|\omega)$ as $p(x|\omega_i)$, where $i = 1$, 2, 3 unless there is ambiguity. The class-conditional probability density function is often referred to as the *likelihood* function as well, since it measures how likely it is that the underlying parametric model of class ω_i will generate the data sample x. Since we know the prior probability $P(\omega_i)$ and class-conditional pdf $p(x|\omega_i)$, we can compute the conditional probability $P(\omega_i|x)$ using Bayes' rule:

$$P(\omega_i|x) = \frac{p(x|\omega_i)P(\omega_i)}{p(x)} \tag{4.1}$$

where $p(x) = \sum_{i=1}^{3} p(x|\omega_i)P(\omega_i)$.

The probability term in the left-hand side of Eq. (4.1) is called the *posterior* probability as it is the probability of class ω_i after observing the federal-funds interest rate x. An intuitive decision rule would be choosing the class ω_k with the greatest posterior probability. That is,

$$k = \arg\max_i P(\omega_i|x) \tag{4.2}$$

In general, the denominator $p(x)$ in Eq. (4.1) is unnecessary because it is a constant term for all classes. Therefore, Eq. (4.2) becomes

$$k = \arg\max_i P(\omega_i|x) = \arg\max_i p(x|\omega_i)P(\omega_i) \tag{4.3}$$

The rule in Eq. (4.3) is referred to as Bayes' decision rule. It shows how the observed data x changes the decision based on the *prior* probability $P(\omega_i)$ to one based on the posterior probability $P(\omega_i|x)$. Decision making based on the *posterior* probability is more reliable, because it employs prior knowledge together with the present observed data. As a matter of fact, when the prior knowledge is non-informative ($P(\omega_1) = P(\omega_2) = P(\omega_3) = 1/3$), the observed data fully control the decision. On the other hand, when observed data are ambiguous, then prior knowledge controls the decision. There are many kinds of decision rules based on posterior probability. Our interest is to find the decision rule that leads to minimum overall risk, or minimum error rate in decision.

4.1.1. Minimum-Error-Rate Decision Rules

Bayes' decision rule is designed to minimize the overall risk involved in making a decision. Bayes' decision based on posterior probability $P(\omega_i|x)$ instead of prior probability $P(\omega_i)$ is a natural choice. Given an observation x, if $P(\omega_k|x) \geq P(\omega_i|x)$ for all $i \neq k$, we can decide that the true class is ω_k. To justify this procedure, we show such a decision results in minimum decision error.

Let $\Omega=\{\omega_1,...,\omega_s\}$ be the finite set of s possible categories to be predicted and $\Delta=\{\delta_1,...,\delta_t\}$ be a finite set of t possible decisions. Let $l(\delta_i \mid \omega_j)$ be the loss function incurred for making decision δ_i when the true class is ω_j. Using the *prior* probability $P(\omega_i)$ and class-conditional pdf $p(x \mid \omega_i)$, the posterior probability $P(\omega_i \mid x)$ is computed by Bayes' rule as shown in Eq. (4.1). Since the posterior probability $P(\omega_j \mid x)$ is the probability that the true class is ω_j after observing the data x, the expected loss associated with making decision δ_i is:

$$R(\delta_i \mid x)=\sum_{j=1}^{s} l(\delta_i \mid \omega_j)P(\omega_j \mid x) \tag{4.4}$$

In decision-theoretic terminology, the above expression is called *conditional risks*. The overall risk R is the expected loss associated with a given decision rule. The decision rule is employed as a decision function $\delta(x)$ that maps the data x to one of the decisions $\Delta=\{\delta_1,...,\delta_t\}$. Since $R(\delta_i \mid x)$ is the conditional risk associated with decision δ_i, the overall risk is given by:

$$R=\int_{-\infty}^{\infty} R(\delta(x) \mid x)p(x)dx \tag{4.5}$$

If the decision function $\delta(x)$ is chosen so that the conditional risk $R(\delta(x) \mid x)$ is minimized for every x, the overall risk is minimized. This leads to the Bayes' decision rule: To minimize the overall risk, we compute the conditional risk shown in Eq. (4.4) for $i=1,...,t$ and select the decision δ_i for which the conditional risk $R(\delta_i \mid x)$ is minimum. The resulting minimum overall risk is known as *Bayes' risk* that has the best performance possible.

The loss function $l(\delta_i \mid \omega_j)$ in the Bayes' decision rule can be defined as:

$$l(\delta_i \mid \omega_j)=\begin{cases} 0 & i=j \\ & i,\ j=1,\ldots,s \\ 1 & i\neq j \end{cases} \tag{4.6}$$

This loss function assigns no loss to a correct decision where the true class is ω_i and the decision is δ_i, which implies that the true class must be ω_i. It assigns a unit loss to any error where $i\neq j$; i.e., all errors are equally costly. This type of loss function is known as a *symmetrical* or *zero-one* loss function. The risk corresponding to this loss function equals the classification error rate, as shown in the following equation.

$$\begin{aligned} R(\delta_i \mid x) &= \sum_{j=1}^{s} l(\delta_i \mid \omega_j)P(\omega_j \mid x)=\sum_{j\neq i} P(\omega_j \mid x) \\ &= \sum_{j=1}^{s} P(\omega_j \mid x)-P(\omega_i \mid x)=1-P(\omega_i \mid x) \end{aligned} \tag{4.7}$$

Here $P(\omega_i \mid x)$ is the conditional probability that decision δ_i is correct after observing the data x. Therefore, in order to minimize classification error rate, we have to choose the decision of class i that maximizes the posterior probability $P(\omega_i \mid x)$. Furthermore, since $p(x)$ is a constant, the decision is equivalent to picking the class i that maximizes $p(x \mid \omega_i)P(\omega_i)$. The Bayes' decision rule can be formulated as follows:

$$\delta(x) = \arg\max_i P(\omega_i \mid x) = \arg\max_i P(x \mid \omega_i)P(\omega_i) \tag{4.8}$$

This decision rule, which is based on the maximum of the posterior probability $P(\omega_i \mid x)$, is called the *minimum-error-rate decision rule*. It minimizes the classification error rate. Although our description is for random variable x, Bayes' decision rule is applicable to multivariate random vector $\mathbf{x}$ without loss of generality.

A pattern classifier can be regarded as a device for partitioning the feature space into decision regions. Without loss of generality, we consider a two-class case. Assume that the classifier divides the space $\Re$ into two regions, $\Re_1$ and $\Re_2$. To compute the likelihood of errors, we need to consider two cases. In the first case, x falls in $\Re_1$, but the true class is ω_2. In the other case, x falls in $\Re_2$, but the true class is ω_1. Since these two cases are mutually exclusive, we have

$$\begin{aligned} P(error) &= P(x \in \Re_1, \omega_2) + P(x \in \Re_2, \omega_1) \\ &= P(x \in \Re_1 \mid \omega_2)P(\omega_2) + P(x \in \Re_2 \mid \omega_1)P(\omega_1) \\ &= \int_{\Re_1} P(x \mid \omega_2)P(\omega_2)dx + \int_{\Re_2} P(x \mid \omega_1)P(\omega_1)dx \end{aligned} \tag{4.9}$$

Figure 4.1 illustrates the calculation of the classification error in Eq. (4.9). The two terms in the summation are merely the tail areas of the function $P(x \mid \omega_i)P(\omega_i)$. It is clear

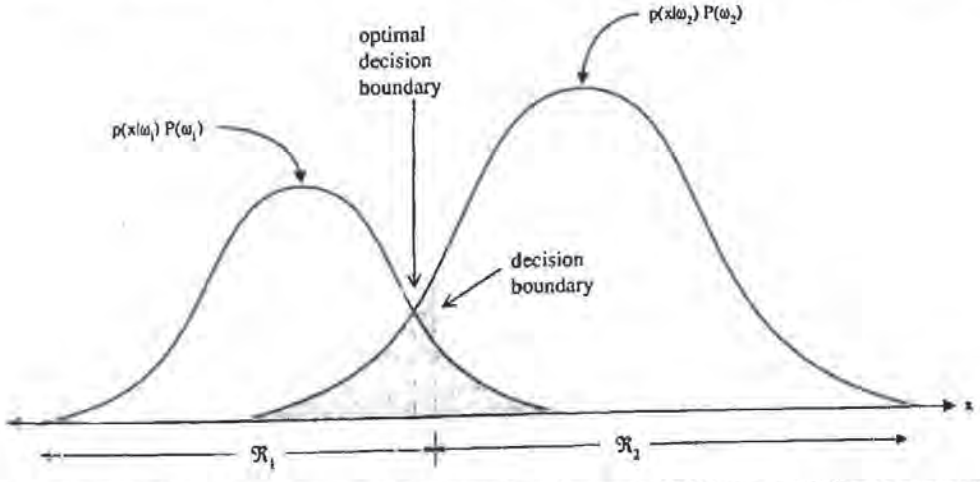


Figure 4.1 Calculation of the likelihood of classification error [22]. The shaded area represents the integral value in Eq. (4.9).

that this decision boundary is not optimal. If we move the decision boundary a little bit to the left, so that the decision is made to choose the class i based on the maximum value of $P(x|\omega_i)P(\omega_i)$, the tail integral area $P(error)$ becomes minimum, which is the Bayes' decision rule.

4.1.2. Discriminant Functions

The decision problem above can also be viewed as a pattern classification problem where unknown data $\mathbf{x}'$ are classified into known categories, such as the classification of sounds into phonemes using spectral data $\mathbf{x}$. A classifier is designed to classify data $\mathbf{x}$ into s categories by using s discriminant functions, $d_i(\mathbf{x})$, computing the similarities between the unknown data $\mathbf{x}$ and each class ω_i and assigning $\mathbf{x}$ to class ω_j if

$$d_j(\mathbf{x}) > d_i(\mathbf{x}) \quad \forall i \neq j \tag{4.10}$$

This representation of a classifier is illustrated in Figure 4.2.

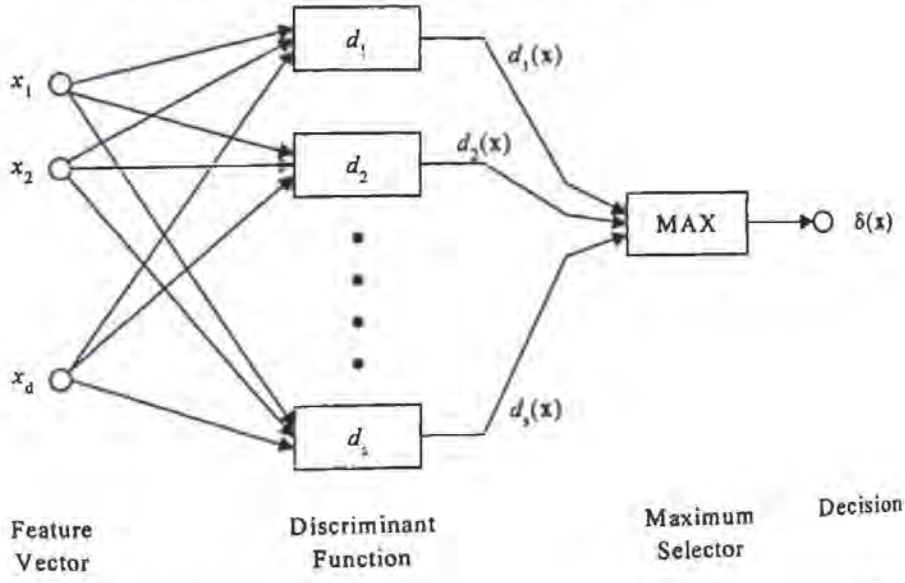


Figure 4.2 Block diagram of a classifier based on discriminant functions [22].

A Bayes' classifier can be represented in the same way. Based on the Bayes' classifier, unknown data $\mathbf{x}$ are classified on the basis of Bayes' decision rule, which minimizes the conditional risk $R(\delta_i|\mathbf{x})$. Since the classification decision of a pattern classifier is based on the maximum discriminant function shown in Eq. (4.10), we define our discriminant function as:

$$d_i(\mathbf{x}) = -R(\delta_i|\mathbf{x}) \tag{4.11}$$

[1] Assuming x is a d-dimensional vector.

As such, the maximum discriminant function corresponds to the minimum conditional risk. In the minimum-error-rate classifier, the decision rule is to maximize the posterior probability $P(\omega_i \mid \mathbf{x})$. Thus, the discriminant function can be written as follows:

$$d_i(\mathbf{x}) = P(\omega_i \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid \omega_i)P(\omega_i)}{p(\mathbf{x})} = \frac{p(\mathbf{x} \mid \omega_i)P(\omega_i)}{\sum_{j=1}^{s} p(\mathbf{x} \mid \omega_j)P(\omega_j)} \tag{4.12}$$

There is a very interesting relationship between Bayes' decision rule and the hypotheses testing method described in Chapter 3. For a two-class pattern recognition problem, the Bayes' decision rule in Eq. (4.2) can be written as follows:

$$p(\mathbf{x} \mid \omega_1)P(\omega_1) \underset{\omega_2}{\overset{\omega_1}{\gtrless}} p(\mathbf{x} \mid \omega_2)P(\omega_2) \tag{4.13}$$

Eq. (4.13) can be rewritten as:

$$\ell(x) = \frac{p(\mathbf{x} \mid \omega_1)}{p(\mathbf{x} \mid \omega_2)} \underset{\omega_2}{\overset{\omega_1}{\gtrless}} \frac{P(\omega_2)}{P(\omega_1)} \tag{4.14}$$

The term $\ell(x)$ is called *likelihood ratio* and is the basic quantity in hypothesis testing [73]. The term $P(\omega_2)/P(\omega_1)$ is called the *threshold value* of the likelihood ratio for the decision. Often it is convenient to use the log-likelihood ratio instead of the likelihood ratio for the decision rule. Namely, the following single discriminant function can be used instead of $d_1(x)$ and $d_2(x)$ for:

$$d(\mathbf{x}) = \log \ell(\mathbf{x}) = \log p(\mathbf{x} \mid \omega_1) - \log p(\mathbf{x} \mid \omega_2) \underset{\omega_2}{\overset{\omega_1}{\gtrless}} \log P(\omega_2) - \log P(\omega_1) \tag{4.15}$$

As the classifier assigns data $\mathbf{x}$ to class ω_i, the data space is divided into s regions, $\Re_1^d, \Re_2^d, \ldots, \Re_s^d$, called decision regions. The boundaries between decision regions are called decision boundaries and are represented as follows (if they are contiguous):

$$d_i(\mathbf{x}) = d_j(\mathbf{x}) \quad i \neq j \tag{4.16}$$

For points on the decision boundary, the classification can go either way. For a Bayes' classifier, the conditional risk associated with either decision is the same and how to break the tie does not matter. Figure 4.3 illustrates an example of decision boundaries and regions for a three-class classifier on a scalar data sample x.

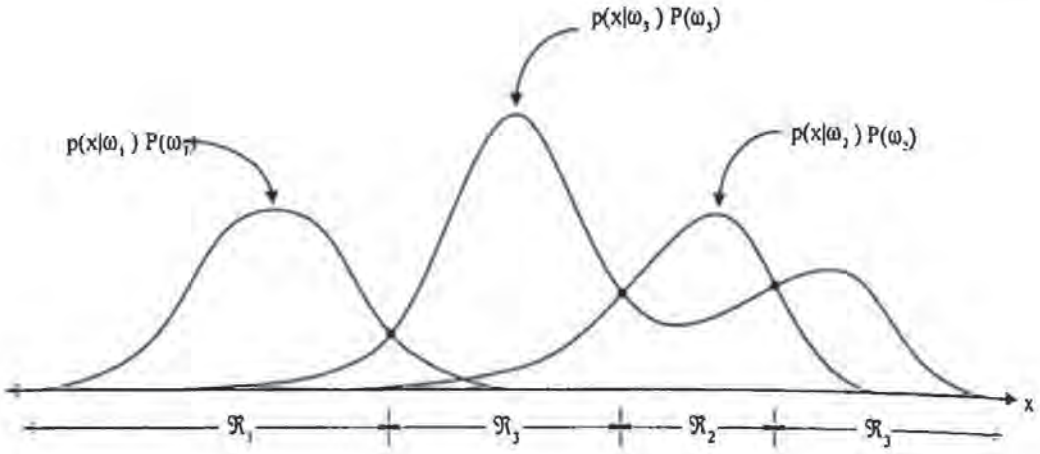


Figure 4.3 An example of decision boundaries and regions. For simplicity, we use scalar variable x instead of a multi-dimensional vector [22].

4.2. How To Construct Classifiers

In the Bayes' classifier, or the minimum-error-rate classifier, the prior probability $P(\omega_i)$ and class-conditional pdf $p(\mathbf{x} \mid \omega_i)$ are known. Unfortunately, in pattern recognition, we rarely have complete knowledge of class-conditional pdfs and/or prior probability. They often must be estimated or learned from the training data. In practice, the estimation of the prior probabilities is relatively easy. Estimation of the class-conditional pdf is more complicated. There is always concern to have sufficient training data relative to the tractability of the huge dimensionality of the sample data $\mathbf{x}$. In this chapter we focus on estimation methods for the class-conditional pdf.

The estimation of the class-conditional pdfs can be nonparametric or parametric. In nonparametric estimation, no model structure is assumed and the pdf is directly estimated from the training data. When large amounts of sample data are available, nonparametric learning can accurately reflect the underlying probabilistic structure of the training data. However, available sample data are normally limited in practice, and parametric learning can achieve better estimates if valid model assumptions are made. In parametric learning, some general knowledge about the problem space allows one to parameterize the class-conditional pdf, so the severity of sparse training data can be reduced significantly. Suppose the pdf $p(\mathbf{x} \mid \omega_i)$ is assumed to have a certain probabilistic structure, such as the Gaussian pdf In such cases, only the mean vector $\boldsymbol{\mu}_i$ (or mean μ_i) and covariance matrix Σ_i (or variance σ^2) need to be estimated.

When the observed data x only takes discrete values from a finite set of N values, the class-conditional pdf is often assumed nonparametric, so there will be $N-1$ free parameters in the probability function $p(\mathbf{x}|\omega_i)$.[2] When the observed data $\mathbf{x}$ takes continuous values, parametric approaches are usually necessary. In many systems, the continuous class-conditional pdf (likelihood) $p(\mathbf{x}|\omega_i)$ is assumed to be a Gaussian distribution or a mixture of Gaussian distributions.

In pattern recognition, the set of data samples, which is often collected to estimate the parameters of the recognizer (including the prior and class-conditional pdf), is referred to as the *training set*. In contrast to the training set, the *testing set* is referred to the independent set of data samples, which is used to evaluate the recognition performance of the recognizer.

For parameter estimation or learning, it is also important to distinguish between *supervised learning* and *unsupervised learning*. Let's denote the pair $(\mathbf{x},\omega)$ as a sample, where $\mathbf{x}$ is the observed data and ω is the class from which the data $\mathbf{x}$ comes. From the definition, it is clear that $(\mathbf{x},\omega)$ are jointly distributed random variables. In supervised learning, ω, information about the class of the sample data $\mathbf{x}$ is given. Such sample data are usually called labeled data or complete data, in contrast to incomplete data where the class information ω is missing for unsupervised learning. Techniques for parametric unsupervised learning are discussed in Section 4.4.

In Chapter 3 we introduced two most popular parameter estimation techniques – *maximum likelihood estimation* (MLE) and *maximum a posteriori probability estimation* (MAP). Both MLE and MAP are supervised learning methods since the class information is required. MLE is the most widely used because of its efficiency. The goal of MLE is to find the set of parameters that maximizes the probability of generating the training data actually observed. The class-conditional pdf is typically parameterized. Let $\mathbf{\Phi}_i$ denote the parameter vector for class i. We can represent the class-conditional pdf as a function of $\mathbf{\Phi}_i$ as $p(\mathbf{x}|\omega_i,\mathbf{\Phi}_i)$. As stated earlier, in supervised learning, the class name ω_i is given for each sample data in training set $\{\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_n\}$. We need to make an assumption[3] that samples in class ω_i give no information about the parameter vector $\mathbf{\Phi}_j$ of the other class ω_j. This assumption allows us to deal with each class independently, since the parameter vectors for different categories are functionally independent. The class-conditional pdf can be rewritten as $p(\mathbf{x}|\mathbf{\Phi})$, where $\mathbf{\Phi}=\{\mathbf{\Phi}_1,\mathbf{\Phi}_2,\ldots,\mathbf{\Phi}_n\}$. If a set of random samples $\{\mathbf{X}_1,\mathbf{X}_2,\ldots,\mathbf{X}_n\}$ is drawn independently according to a pdf $p(\mathbf{x}|\mathbf{\Phi})$, where the value of the parameter $\mathbf{\Phi}$ is unknown, the MLE method described in Chapter 3 can be directly applied to estimate $\mathbf{\Phi}$.

Similarly, MAP estimation can be applied to estimate $\mathbf{\Phi}$ if knowledge about a prior distribution is available. In general, MLE is used for estimating parameters from scratch without any prior knowledge, and MAP estimation is used for parameter adaptation where the behavior of a prior distribution is known and only a small amount of adaptation data is available. When the amount of adaptation data increases, MAP estimation converges to MLE.

---

[2] Since all the probabilities need to add up to one.

[3] This assumption is only true for non-discriminative estimation. Samples in class ω_i may affect parameter vector $\mathbf{\Phi}_i$ of the other classes in discriminative estimation methods as described in Section 4.3

4.2.1. Gaussian Classifiers

A *Gaussian classifier* is a Bayes' classifier where class-conditional probability density $p(\mathbf{x}\,|\,\omega_i)$ for each class ω_i is assumed to have a Gaussian distribution:[4]

$$p(\mathbf{x}\,|\,\omega_i) = \frac{1}{(2\pi)^{d/2}\,|\Sigma_i|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)'\,\Sigma_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)\right] \tag{4.17}$$

As discussed in Chapter 3, the parameter estimation techniques are well suited for the Gaussian family. The MLE of the Gaussian parameter is just its sample mean and variance (or co-variance matrix). A Gaussian classifier is equivalent to the one using a quadratic discriminant function. As noted in Eq. (4.12), the discriminant function for a Bayes' decision rule is the posterior probability $p(\omega_i\,|\,\mathbf{x})$ or $p(\mathbf{x}\,|\,\omega_i)P(\omega_i)$. Assuming $p(\mathbf{x}\,|\,\omega_i)$ is a multivariate Gaussian density as shown in Eq. (4.17), a discriminant function can be written as follows:

$$\begin{aligned} d_i(\mathbf{x}) &= \log p(\mathbf{x}\,|\,\omega_i)P(\omega_i) \\ &= -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)'\,\Sigma_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i) + \log P(\omega_i) - \frac{1}{2}\log|\Sigma| - \frac{d}{2}\log 2\pi \end{aligned} \tag{4.18}$$

If we have a uniform prior $P(\omega_i)$, it is clear that the above discriminant function $d_i(\mathbf{x})$ is a quadratic function. Once we have the s Gaussian discriminant functions, the decision process simply assigns data $\mathbf{x}$ to class ω_j if

$$j = \arg\max_i d_i(\mathbf{x}) \tag{4.19}$$

When all the Gaussian pdfs have the same covariance matrix ($\Sigma_i = \Sigma$ for $i = 1,2,\ldots,s$), the quadratic term $\mathbf{x}'\Sigma^{-1}\mathbf{x}$ is independent of the class and can be treated as a constant. Thus the following new discriminant function $d_i(\mathbf{x})$ can be used [22]:

$$d_i(\mathbf{x}) = \mathbf{a}_i'\mathbf{x} + \mathbf{c}_i \tag{4.20}$$

where $\mathbf{a}_i = \Sigma^{-1}\boldsymbol{\mu}_i$ and $\mathbf{c}_i = -\frac{1}{2}\boldsymbol{\mu}_i'\Sigma_i^{-1}\boldsymbol{\mu}_i + \log P(\omega_i)$. $d_i(\mathbf{x})$ in Eq. (4.20) is a linear discriminant function. For linear discriminant functions, the decision boundaries are hyperplanes. For the two-class case (ω_1 and ω_2), and assuming that data sample $\mathbf{x}$ is a real random vector, the decision boundary can be shown to be the following hyperplane:

[4] The Gaussian distribution may include a mixture of Gaussian pdfs.

$$\mathbf{A}^t(\mathbf{x}-\mathbf{b})=0 \tag{4.21}$$

where

$$\mathbf{A}=\Sigma^{-1}(\mu_1-\mu_2) \tag{4.22}$$

and

$$\mathbf{b}=\frac{1}{2}(\mu_1-\mu_2)-\frac{(\mu_1-\mu_2)\log[P(\omega_1/\omega_2]}{(\mu_1-\mu_2)'\Sigma^{-1}(\mu_1-\mu_2)} \tag{4.23}$$

Figure 4.4 shows a two dimensional decision boundary for a two-class Gaussian classifier with the same covariance matrix. Please note that the decision hyperplane is generally not orthogonal to the line between the means μ_1 and μ_2, although it does intersect that line at the point **b**, which is halfway between μ_1 and μ_2. The analysis above is based on the case of uniform priors ($p(\omega_1)=p(\omega_2)$). For nonuniform priors, the decision hyperplane moves away from the more likely mean.

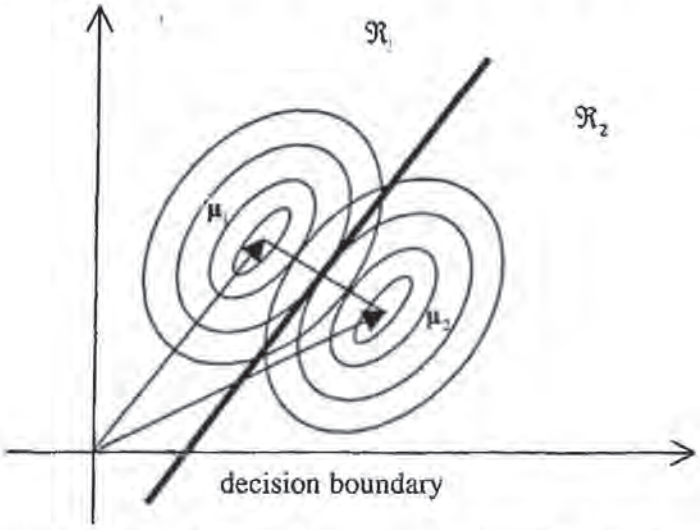


Figure 4.4 Decision boundary for a two-class Gaussian classifier. Gaussian distributions for the two categories have the same covariance matrix Σ. Each ellipse represents the region with the same likelihood probability [22].

Finally, if each dimension of random vector **x** is statistically independent and has the same variance σ^2, i.e., $\Sigma_1=\Sigma_2=\sigma^2\mathbf{I}$, Figure 4.4 becomes Figure 4.5. The ellipse in Figure 4.4 becomes a circle because the variance σ^2 is the same for all dimensions [22].

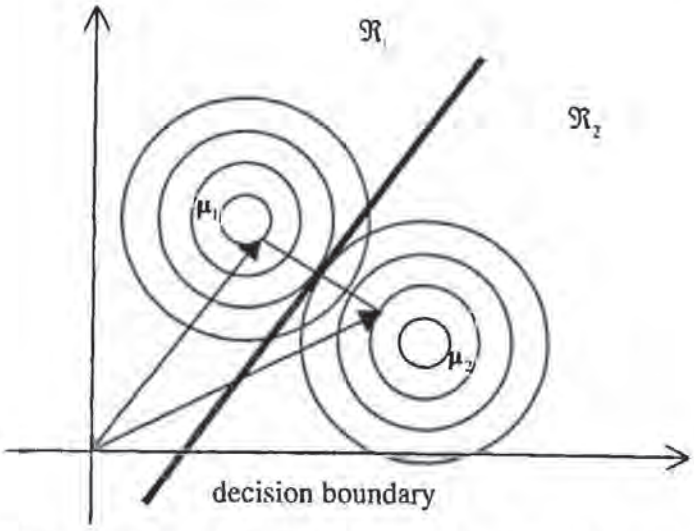


Figure 4.5 Decision boundary for a two-class Gaussian classifier. Gaussian distributions for the two categories have the same covariance matrix $\sigma^2\mathbf{I}$. Each circle represents the region with the same likelihood probability [22].

4.2.2. The Curse of Dimensionality

More features (higher dimensions for sample $\mathbf{x}$) and more parameters for the class-conditional pdf $p(\mathbf{x}\,|\,\Phi)$ may lead to lower classification error rate. If the features are statistically independent, there are theoretical arguments that support better classification performance with more features. Let us consider a simple two-class Gaussian classifier. Suppose the prior probabilities $p(\omega_i)$ are equal and the class-conditional Gaussian pdfs $p(\mathbf{x}\,|\,\boldsymbol{\mu}_i, \Sigma)$ share the same covariance matrix Σ. According to Eqs. (4.9) and (4.21), the Bayes' classification error rate is given by:

$$\begin{aligned} P(error) &= 2\int_{\Re_2} P(\mathbf{x}\,|\,\omega_1)P(\omega_1)d\mathbf{x} \\ &= 2\int_{\mathbf{A}'(\mathbf{x}-\mathbf{b})=0}^{\infty} \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left[-\tfrac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_1)'\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}_1)\right] d\mathbf{x} \\ &= \frac{1}{\sqrt{2\pi}}\int_{\frac{r}{2}}^{\infty} e^{-\frac{z^2}{2}}\, dz \end{aligned} \tag{4.24}$$

where $r = \sqrt{(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)'\Sigma^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)}$. When features are independent, the covariance matrix becomes a diagonal one. The following equation shows that each independent feature helps to reduce the error rate:[5]

[5] When the means of a feature for the two classes are exactly the same, adding such a feature does not reduce the Bayes' error. Nonetheless, according to Eq. (4.25), the Bayes' error cannot possibly be increased by incorporating an additional independent feature.

$$r = \sqrt{\sum_{i=1}^{d} \left(\frac{\mu_{1i} - \mu_{2i}}{\sigma_i} \right)^2} \tag{4.25}$$

where μ_{1i} and μ_{2i} are the i^{th} -dimension of mean vectors μ_1 and μ_2 respectively.

Unfortunately, in practice, the inclusion of additional features may lead to worse classification results. This paradox is called *the curse of dimensionality*. The fundamental issue, called *trainability*, refers to how well the parameters of the classifier are trained from the limited training samples. Trainability can be illustrated by a typical curve-fitting (or regression) problem. Figure 4.6 shows a set of eleven data points and several polynomial fitting curves with different orders. Both the first-order (linear) and second-order (quadratic) polynomials shown provide fairly good fittings for these data points. Although the tenth-order polynomial fits the data points perfectly, no one would expect such an under-determined solution to fit the new data well. In general, many more data samples would be necessary to get a good estimate of a tenth-order polynomial than of a second-order polynomial, because reliable interpolation or extrapolation can be attained only with an over-determined solution.



Figure 4.6 Fitting eleven data points with polynomial functions of different orders [22].

Figure 4.7 shows the error rates for two-phonemes (*/ae/* and */ih/*) classification where two phonemes are modeled by mixtures of Gaussian distributions. The parameters of mixtures of Gaussian are trained from a varied set of training samples via maximum likelihood estimation. The curve illustrates the classification error rate as a function of the number of training samples and the number of mixtures. For every curve associated with a finite number of samples, there are an optimal number of mixtures. This illustrates the importance of trainability: it is critical to assure there are enough samples for training an increased number of features or parameters. When the size of training data is fixed, increasing the number of features or parameters beyond a certain point is likely to be counterproductive.

When you have an insufficient amount of data to estimate the parameters, some simplification can be made to the structure of your models. In general, the estimation for higher-order statistics, like variances or co-variance matrices, requires more data than that for lower- order statistics, like mean vectors. Thus more attention often is paid to dealing with

the estimation of covariance matrices. Some frequently used heuristics for Gaussian distributions include the use of the same covariance matrix for all mixture components [77], diagonal covariance matrix, and shrinkage (also referred to as regularized discriminant analysis), where the covariance matrix is interpolated with the constant covariance matrix [23, 50].

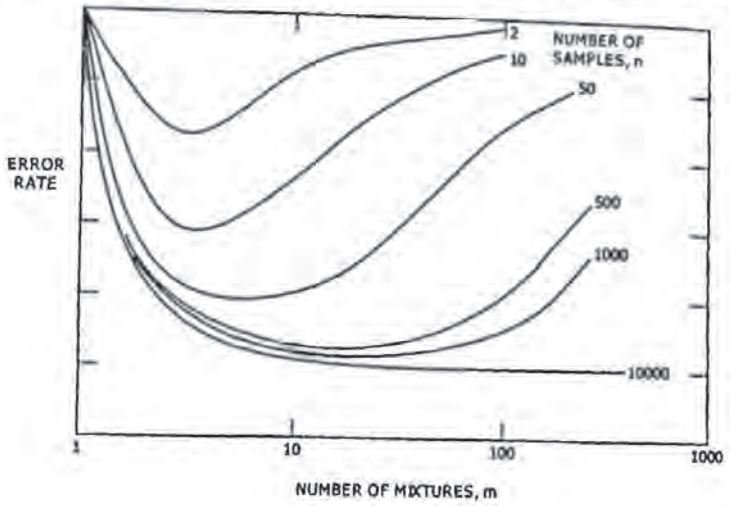


Figure 4.7 Two-phoneme (*/ae/* and */ih/*) classification results as a function of the number of Gaussian mixtures and the number of training samples.

4.2.3. Estimating the Error Rate

Estimating the error rate of a classifier is important. We want to see whether the classifier is good enough to be useful for our task. For example, telephone applications show that some minimum accuracy is required before users would switch from using the touch-tone to the speech recognizer. It is also critical to compare the performance of a classifier (algorithm) against an alternative. In this section we deal with how to estimate the true classification error rate.

One approach is to compute the theoretic error rate from the parametric model as shown in Eq. (4.24). However, there are several problems with this approach. First, such an approach almost always under-estimates, because the parameters estimated from the training samples might not be realistic unless the training samples are representative and sufficient. Second, all the assumptions about models and distributions might be severely wrong. Finally, it is very difficult to compute the exact error rate, as in the simple case illustrated in Eq. (4.24).

Instead, you can estimate the error rate empirically. In general, the recognition error rate on the training set should be viewed only as a lower bound, because the estimate can be made to minimize the error rate on the training data. Therefore, a better estimate of the rec-

ognition performance should be obtained on an independent test set. The question now is how representative is the error rate computed from an arbitrary independent test set. The common process of using some of the data samples for design and reserving the rest for test is called the *holdout* or *H* method.

Suppose the true but unknown classification error rate of the classifier is p, and one observes that k out of n independent randomly drawn test samples are misclassified. The random variable K should have a binomial distribution $B(n,p)$. The maximum likelihood estimation for p should be

$$\hat{p} = \frac{k}{n} \tag{4.26}$$

The statistical test for binomial distribution is discussed in Chapter 3. For a 0.05 significance level, we can compute the following equations to get the range (p_1, p_2):

$$2P(k \le m \le n) = 2\sum_{m=k}^{n} \binom{n}{m} (p_1)^m (1-p_1)^{n-m} = 0.05 \quad \text{when } k > np_1 \tag{4.27}$$

$$2P(0 \le m \le k) = 2\sum_{m=0}^{k} \binom{n}{m} (p_2)^m (1-p_2)^{n-m} = 0.05 \quad \text{when } k < np_2 \tag{4.28}$$

Equations (4.27) and (4.28) are cumbersome to solve, so the normal test described in Chapter 3 can be used instead. The null hypothesis H_0 is

$H_0 : p = \hat{p}$

We can use the normal test to find the two boundary points p_1 and p_2 at which we would not reject the null hypothesis H_0.

The range (p_1, p_2) is called the 0.95 confidence intervals because one can be 95% confident that the true error rate p falls in the range (p_1, p_2). Figure 4.8 illustrates 95% confidence intervals as a function of $\hat{p}$ and n. The curve certainly agrees with our intuition – the larger the number of test samples n, the more confidence we have in the MLE estimated error rate $\hat{p}$; otherwise, the $\hat{p}$ can be used only with caution.

Based on the description in the previous paragraph, the larger the test set is, the better it represents the recognition performance of possible data. On one hand, we need more training data to build a reliable and consistent estimate. On the other hand, we need a large independent test set to derive a good estimate of the true recognition performance. This creates a contradictory situation for dividing the available data set into training and independent test set. One way to effectively use the available database is *V-fold cross validation*. It first splits the entire database into V equal parts. Each part is used in turn as an independent test set while the remaining $(V - 1)$ parts are used for training. The error rate can then be better estimated by averaging the error rates evaluated on the V different testing sets. Thus, each part can contribute to both training and test sets during V-fold cross validation. This procedure, also called the *leave-one-out* or *U* method [53], is particularly attractive when the number of available samples are limited.

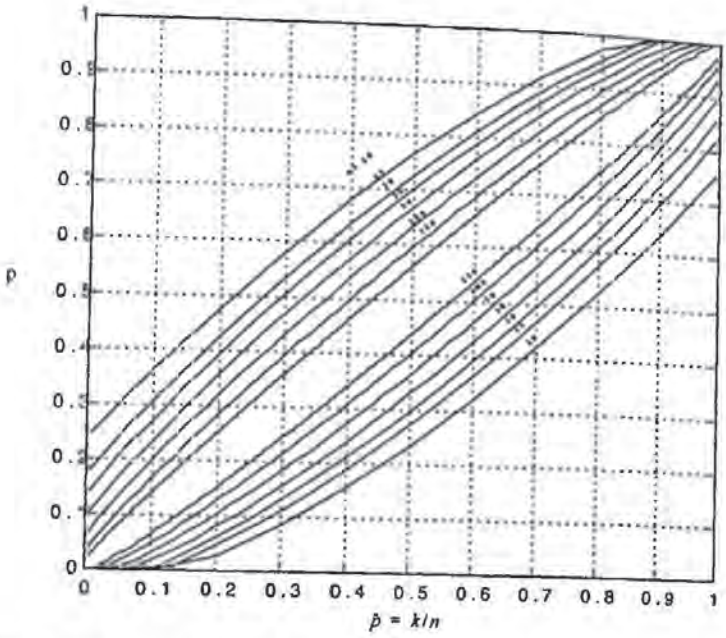


Figure 4.8 95% confidence intervals for classification error rate estimation with normal test.

4.2.4. Comparing Classifiers

Given so many design alternatives, it is critical to compare the performance of different classifiers so that the best classifier can be used for real-world applications. It is common for designers to test two classifiers on some test samples and decide if one is superior to the other. Relative efficacy can be claimed only if the difference in performance is statistically significant. In other words, we establish the null hypothesis H_0 that the two classifiers have the same error rates. Based on the observed error patterns, we decide whether we could reject H_0 at the 0.05 level of significance. The test for different classifiers falls into the category of *matched-pairs* tests described in Chapter 3. Classifiers are compared with the same test samples.

We present an effective matched-pairs test – McNemar's test [66] which is particularly suitable for comparing classification results. Suppose there are two classifiers: Q_1 and Q_2. The estimated classification error rates on the same test set for these two classifiers are p_1 and p_2 respectively. The null hypothesis H_0 is $p_1 = p_2$. The classification performance of the two classifiers can be summarized as in Table 4.1. We define q_{ij} as follows:

$$q_{00} = P(Q_1 \text{ and } Q_2 \text{ classify data sample correctly})$$

$$q_{01} = P(Q_1 \text{ classifies data sample correctly, but } Q_2 \text{ incorrectly})$$

q_{10} = $P(Q_2$ classifies data sample correctly, but Q_1 incorrectly)
q_{11} = $P(Q_1$ and Q_2 classify data sample incorrectly)

Table 4.1 Classification performance table for classifiers Q_1 and Q_2. N_{00} is the number of samples which Q_1 and Q_2 classify correctly, N_{01} is the number of samples which Q_1 classifies correctly, but Q_2 incorrectly, N_{10} is the number of samples which Q_2 classifies correctly, but Q_1 incorrectly, and N_{11} is the number of samples which Q_1 and Q_2 classify incorrectly [30].

		Q_2	
		Correct	Incorrect
Q_1	Correct	N_{00}	N_{01}
	Incorrect	N_{10}	N_{11}

The null hypothesis H_0 is equivalent to H_0^1 : $q_{01} = q_{10}$. If we define $q = q_{10}/(q_{01}+q_{10})$, H_0 is equivalent to H_0^2 : $q = ½$. H_0^2 represents the hypothesis that, given only one of the classifiers makes an error, it is equally likely to be either one. We can test H_0^2 based on the data samples on which only one of the classifiers made an error. Let $n = N_{01} + N_{10}$. The observed random variable N_{01} should have a binomial distribution $B(n, ½)$. Therefore, the normal test (z-test) described in Chapter 3 can be applied directly to test the null hypothesis H_0^2.

The above procedure is called the *McNemar's test* [66]. If we view the classification results as N (the total number of test samples) independent matched pairs, the sign test as described in Chapter 3 can be directly applied to test the null hypothesis that classifier Q_1 is not better than classifier Q_2, that is, the probability that classifier Q_1 performs better than classifier Q_2, p, is smaller than or equal to ½.

McNemar's test is applicable when the errors made by a classifier are independent among different test samples. Although this condition is true for most static pattern recognition problems, it is not the case for most speech recognition problems. In speech recognition, the errors are highly inter-dependent because of the use of higher-order language models (described in Chapter 11).

The solution is to divide the test data stream into segments in such a way that errors in one segment are statistically independent of errors in any other segment [30]. A natural candidate for such a segment is a sentence or a phrase after which the speaker pauses. Let N_1^i be the number of errors[6] made on the i^{th} segment by classifier Q_1 and N_2^i be the number of errors made on the i^{th} segment by classifier Q_2. Under this formulation, the magnitude-difference test described in Chapter 3 can be applied directly to test the null hypothesis that classifiers Q_1 and Q_2 have on the average the same error rate on the pairs of n independent segments.

[6] The errors for speech recognition include substitutions, insertions and deletions as discussed in Chapter 9.

4.3. DISCRIMINATIVE TRAINING

Both MLE and MAP criteria maximize the probability of the model associated with the corresponding data. Only data labeled as belonging to class ω_i are used to train the parameters. There is no guarantee that the observed data $\mathbf{x}$ from class ω_i actually have a higher likelihood $P(\mathbf{x}|\omega_i)$ than the likelihood $P(\mathbf{x}|\omega_j)$ associated with class j, given $j \neq i$. Models generated by MLE or MAP have a loose discriminant nature. Several estimation methods aim for maximum discrimination among models to achieve best pattern recognition performance.

4.3.1. Maximum Mutual Information Estimation

The pattern recognition problem can be formalized as an information channel, as illustrated in Figure 4.9. The source symbol ω is encoded into data $\mathbf{x}$ and transmitted through an information channel to the observer. The observer utilizes pattern recognition techniques to decode $\mathbf{x}$ into source symbol $\hat{\omega}$. Consistent with the goal of communication channels, the observer hopes the decoded symbol $\hat{\omega}$ is the same as the original source symbol ω. Maximum mutual information estimation tries to improve channel quality between input and output symbols.

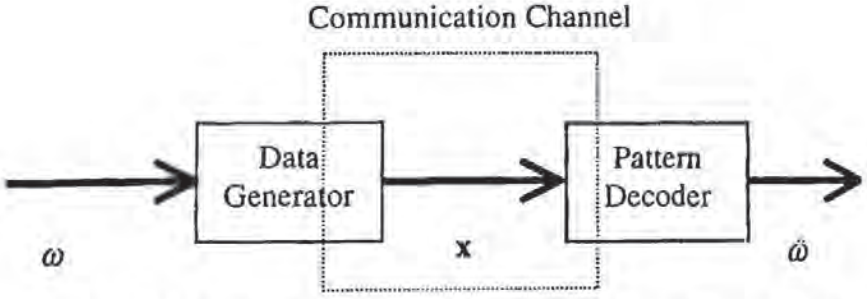


Figure 4.9 An information channel framework for pattern recognition.

As described in Section 4.1.1, the decision rule for the minimum-error-rate classifier selects the class ω_i with maximum posterior probability $P(\omega_i|\mathbf{x})$. It is a good criterion to maximize the posterior probability $P(\omega_i|\mathbf{x})$ for parameter estimation. Recalling Bayes' rule in Section 4.1, the posterior probability $p(\omega_i|\mathbf{x})$ (assuming $\mathbf{x}$ belongs to class ω_i) is:

$$P(\omega_i|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_i)P(\omega_i)}{p(\mathbf{x})} \tag{4.29}$$

and $p(\mathbf{x})$ can be expressed as follows:

$$p(\mathbf{x}) = \sum_k p(\mathbf{x}|\omega_k)p(\omega_k) \tag{4.30}$$

In the classification stage, $p(\mathbf{x})$ can be considered as a constant. However, during training, the value of $p(\mathbf{x})$ depends on the parameters of all models and is different for different $\mathbf{x}$. Equation (4.29) is referred to as *conditional likelihood*. A conditional maximum likelihood estimator (CMLE) θ_{CMLE} is defined as follows:

$$\theta_{CMLE}(\mathbf{x}) = \Phi_{MAP} = \underset{\Phi}{\operatorname{argmax}}\ p_{\Phi}(\omega_i \mid \mathbf{x}) \tag{4.31}$$

The summation in Eq. (4.30) extends over all possible classes that include the correct model and all the possible competing models. The parameter vector Φ in Eq. (4.31) includes not only the parameter Φ_i corresponding to class ω_i, but also those for all other classes.

Note that in Chapter 3, the mutual information between random variable $\mathbf{X}$ (observed data) and Ω (class label) is defined as:

$$I(\mathbf{X},\Omega) = E\left[\log\frac{p(\mathbf{X},\Omega)}{p(\mathbf{X})P(\Omega)}\right] = E\left[\log\frac{p(\mathbf{X}\mid\Omega)P(\Omega)}{p(\mathbf{X})P(\Omega)}\right] \tag{4.32}$$

Since we don't know the probability distribution for $p(\mathbf{X},\Omega)$, we assume our sample $(\mathbf{x},\omega_i)$ is representative and define the following *instantaneous mutual information*:

$$I(\mathbf{x},\omega_i) = \log\frac{p(\mathbf{x},\omega_i)}{p(\mathbf{x})P(\omega_i)} \tag{4.33}$$

If equal prior $p(\omega_i)$ is assumed for all classes, maximizing the conditional likelihood in Eq. (4.29) is equivalent to maximizing the mutual information defined in Eq.(4.33). CMLE becomes *maximum mutual information estimation* (MMIE). It is important to note that, in contrast to MLE, MMIE is concerned with distributions over all possible classes. Equation (4.30) can be rewritten as two terms, one corresponding to the correct one, and the other corresponding to the competing models:

$$p(\mathbf{x}) = p(\mathbf{x}\mid\omega_i)P(\omega_i) + \sum_{k\neq i} p(\mathbf{x}\mid\omega_k)P(\omega_k) \tag{4.34}$$

Based on the new expression of $p(\mathbf{x})$ shown in Eq. (4.34), the posterior probability $p(\omega_i \mid \mathbf{x})$ in Eq. (4.29) can be rewritten as:

$$P(\omega_i \mid \mathbf{x}) = \frac{p(\mathbf{x}\mid\omega_i)P(\omega_i)}{p(\mathbf{x}\mid\omega_i)P(\omega_i) + \sum_{k\neq i} p(\mathbf{x}\mid\omega_k)P(\omega_k)} \tag{4.35}$$

Now, maximization of the posterior probability $p(\omega_i \mid \mathbf{x})$ with respect to all models leads to a discriminant model.[7] It implies that the contribution of $p(\mathbf{x}\mid\omega_i)P(\omega_i)$ from the true model needs to be enforced, while the contribution of all the competing models, specified by

[7] General minimum-error-rate estimation is described in Section 4.3.2.

$\sum_{k \neq i} p(\mathbf{x} \mid \omega_k) P(\omega_k)$, needs to be minimized. Maximization of Eq. (4.35) can be further rewritten as:

$$P(\omega_i \mid \mathbf{x}) = \frac{1}{1 + \frac{\sum_{k \neq i} p(\mathbf{x} \mid \omega_i) p(\omega_i)}{p(\mathbf{x} \mid \omega_i) p(\omega_i)}} \tag{4.36}$$

Maximization is thus equivalent to maximization of the following term, which is clearly a discriminant criterion between model ω_i and the sum of all other competing models.

$$\frac{p(\mathbf{x} \mid \omega_i) p(\omega_i)}{\sum_{k \neq i} p(\mathbf{x} \mid \omega_k) p(\omega_k)} \tag{4.37}$$

Equation (4.37) also illustrates a fundamental difference between MLE and MMIE. In MLE, only the correct model needs to be updated during training. However, every MMIE model is updated even with one training sample. Furthermore, the greater the prior probability $p(\omega_k)$ for class ω_k, the more effect it has on the maximum mutual information estimator θ_{MMIE}. This makes sense, since the greater the prior probability $p(\omega_k)$, the greater the chance for the recognition system to mis-recognize ω_i as ω_k. MLE is a simplified version of MMIE by restricting the training of model using the data for the model only. This simplification allows the denominator term in Eq. (4.29) to contain the correct model so that it can be dropped as a constant term. Thus, maximization of the posterior probability $p(\omega_i \mid \mathbf{x})$ can be transformed into maximization of the likelihood $p(\mathbf{x} \mid \omega_i)$.

Although likelihood and posterior probability are transformable based on Bayes' rule, MLE and MMIE often generate different results. Discriminative criteria like MMIE attempt to achieve minimum error rate. It might actually produce lower likelihood for the underlying probability density $p(\mathbf{x} \mid \omega_k)$. However, if the assumption of the underlying distributions is correct and there are enough (or infinite) training data, the estimates should converge to the true underlying distributions. Therefore, Bayes' rule should be satisfied and MLE and MMIE should produce the same estimate.

Arthur Nadas [71] showed that if the prior distribution (language model) and the assumed likelihood distribution family are correct, both MLE and MMIE are consistent estimators, but MMIE has a greater variance. However, when some of those premises are not valid, it is desirable to use MMIE to find the estimate that maximizes the mutual information (instead of likelihood) between sample data and its class information. The difference between these two estimation techniques is that MMIE not only aims to increase the likelihood for the correct class, but also tries to decrease the likelihood for the incorrect classes. Thus, MMIE in general possesses more discriminating power among different categories. Although MMIE is theoretically appealing, computationally it is very expensive. Comparing with MLE, every data sample needs to train all the possible models instead of the corresponding model. It also lacks an efficient maximization algorithm. You need to use a gradient descent algorithm.

4.3.1.1. Gradient Descent

To maximize Eq. (4.37) over the entire parameter space $\Phi=\{\Phi_1,\Phi_2,\ldots,\Phi_S\}$ with S classes, we define the mutual information term in Eq. (4.37) to be a function of Φ. To fit into the traditional gradient descent framework, we take the inverse of Eq. (4.37) as our optimization function to minimize the following function:[8]

$$F(\Phi)=\frac{\sum_{k\neq i} p_{\lambda_i}(\mathbf{x}\mid\omega_k)p(\omega_k)}{p_{\lambda_i}(\mathbf{x}\mid\omega_i)p(\omega_i)} \tag{4.38}$$

The gradient descent algorithm starts with some initial estimate Φ^0 and computes the gradient vector $\nabla F(\Phi)$ (∇ is defined in Chapter 3). It obtains a new estimate Φ^1 by moving Φ^0 in the direction of the steepest descent, i.e., along the negative of the gradient. Once it obtains the new estimate, it can perform the same gradient descent procedure iteratively until $F(\Phi)$ converges to the local minimum. In summary, it obtains Φ^{t+1} from Φ^t by the following formula:

$$\Phi^{t+1}=\Phi^t-\varepsilon_t\nabla F(\Phi)\big|_{\Phi=\Phi^t} \tag{4.39}$$

where ε_t is the learning rate (or step size) for the gradient descent.

Why can gradient descent lead $F(\Phi)$ to a local minimum? Based on the definition of gradient vector, $F(\Phi)$ can be approximated by the first order expansion if the correction term $\Delta\Phi$ is small enough.

$$F(\Phi^{t+1})\approx F(\Phi^t)+\Delta\Phi*\nabla F(\Phi)\big|_{\Phi=\Phi^t} \tag{4.40}$$

$\Delta\Phi$ can be expressed as the following term based on Eq. (4.39)

$$\Delta\Phi=\Phi^{t+1}-\Phi^t=-\varepsilon_t\nabla F(\Phi)\big|_{\Phi=\Phi^t} \tag{4.41}$$

Thus, we can obtain the following equation:

$$\begin{aligned} F(\Phi^{t+1})-F(\Phi^t) &= -\varepsilon_t\left\langle \nabla F(\Phi)\big|_{\Phi=\Phi^t}, \nabla F(\Phi)\big|_{\Phi=\Phi^t}\right\rangle \\ &= -\varepsilon_t\left\|\nabla F(\Phi)\big|_{\Phi=\Phi^t}\right\|^2 < 0 \end{aligned} \tag{4.42}$$

where $\langle x,y\rangle$ represents the inner product of two vectors, and $\|x\|$ represents the Euclidean norm of the vector. Equation (4.42) means that the gradient descent finds a new estimate Φ^{t+1} that makes the value of the function $F(\Phi)$ decrease.

The gradient descent algorithm needs to go through an iterative hill-climbing procedure to converge to the local minimum (estimate). Gradient descent usually requires many iterations to converge. The algorithm usually stops when the change of the parameter $\Delta\Phi$ becomes small enough. That is,

[8] You can use the logarithm of the object function to make it easier to compute the derivative in gradient descent.

$$\left|\varepsilon_t \nabla F(\Phi)|_{\Phi=\Phi'}\right| < \lambda \tag{4.43}$$

where λ is a preset threshold.

Based on the derivation above, the learning rate coefficient ε_t must be small enough for gradient descent to converge. However, if ε_t is too small, convergence is needlessly slow. Thus, it is very important to choose an appropriate ε_t. It is proved [47] [48] that gradient converges almost surely if the learning rate coefficient ε_t satisfies the following condition:

$$\sum_{t=0}^{\infty} \varepsilon_t = \infty, \quad \sum_{t=0}^{\infty} \varepsilon_t^2 < \infty, \quad \text{and } \varepsilon_t > 0 \tag{4.44}$$

One popular choice of ε_t satisfying the above condition is

$$\varepsilon_t = \frac{1}{t+1} \tag{4.45}$$

Another way to find an appropriate ε_t is through the second-order expansion:

$$F(\Phi^{t+1}) \approx F(\Phi^t) + \Delta\Phi \nabla F(\Phi)|_{\Phi=\Phi'} + \frac{1}{2}(\Delta\Phi)^t \mathbf{D} \Delta\Phi \tag{4.46}$$

where $\mathbf{D}$ is the *Hessian* matrix [23] of the second-order gradient operator where the i-th row and j-th element $D_{i,j}$ are given by the following partial derivative:

$$D_{i,j} = \frac{\partial^2 F(\Phi)}{\partial \Phi_i \partial \Phi_j} \tag{4.47}$$

By substituting $\Delta\Phi$ from Eq. (4.41) into Eq. (4.46), we can obtain

$$F(\Phi^{t+1}) \approx F(\Phi^t) - \varepsilon_t \|\nabla F\|^2 + \frac{1}{2}\varepsilon_t^2 \nabla F^t \mathbf{D} \nabla F \tag{4.48}$$

From this, it follows that ε_t can be chosen as follows to minimize $F(\Phi)$ [23]:

$$\varepsilon_t = \frac{\|\nabla F\|^2}{\nabla F^t \mathbf{D} \nabla F} \tag{4.49}$$

Sometimes it is desirable to impose a different learning rate for the correct model vs. competing models. Therefore re-estimation Eq. (4.39) can be generalized to the following form [19, 48]:

$$\Phi^{t+1} = \Phi^t - \varepsilon_t U_t \nabla F(\Phi)|_{\Phi=\Phi'} \tag{4.50}$$

where U_t is the learning bias matrix which is a positive definite matrix. One particular choice of U_t is $\mathbf{D}^{-1}$, where $\mathbf{D}$ is the Hessian matrix defined in Eq. (4.47). When the learning

rate is set to be 1.0, Eq. (4.50) becomes Newton's algorithm, where the gradient descent is chosen to minimize the second-order expansion. Equation (4.50) becomes:

$$\mathbf{\Phi}^{t+1} = \mathbf{\Phi}^{t} - \mathbf{D}^{-1}\nabla F(\mathbf{\Phi})|_{\mathbf{\Phi}=\mathbf{\Phi}^{t}} \tag{4.51}$$

When probabilistic parameters are iteratively re-estimated, probabilistic constraints must be satisfied in each iteration as probability measure, such as:

1. For discrete distributions, all the values of the probability function ought to be nonnegative. Moreover the sum of all discrete probability values needs to be one, i.e., $\sum_i a_i = 1$
2. For continuous distributions (assuming Gaussian density family), the variance needs to be nonnegative. For Gaussian mixtures, the diagonal covariance entries need to be nonnegative and the sum of mixture weights needs to be one, i.e., $\sum_i c_i = 1$

In general, gradient descent is an unconstrained minimization (or maximization) process that needs to be modified to accommodate constrained minimization (or maximization) problems. The tricks to use are parameter transformations that implicitly maintain these constraints during gradient descent. The original parameters are updated through the inverse transform from the transformed parameter space to the original parameter space. The transformation is done in such a way that constraints on the original parameter are always maintained. Some of these transformations are listed as follows [48]:

1. For probabilities which need to be nonnegative and sum to one, like discrete probability function and mixture weight, the following transformation can be performed:

$$a_i = \frac{\exp(\tilde{a}_i)}{\sum_k \exp(\tilde{a}_k)} \tag{4.52}$$

2. For mean μ and variance (or diagonal covariance entries) σ^2, the following transformation can be used.

$$\mu = \tilde{\mu}\sigma \tag{4.53}$$

$$\sigma = \exp(\tilde{\sigma}) \tag{4.54}$$

After the transformations, we can now compute the gradient with respect to the transformed parameters $(\tilde{a}_i, \tilde{\mu}, \tilde{\sigma})$ using the chain rule. Once the new estimate for the transformed parameters is obtained through gradient descent, one can easily transform them back to the original parameter domain.

4.3.2. Minimum-Error-Rate Estimation

Parameter estimation techniques described so far aim to maximize either the likelihood (class-conditional probability) (MLE and MAP) or the posterior probability (MMIE) in Bayes' equation, Eq. (4.1). Although the criteria used in those estimation methods all have their own merit and under some conditions should lead to satisfactory results, the ultimate parameter estimation criterion for pattern recognition should be to minimize the recognition error rate (or the Bayes' risk) directly. *Minimum-error-rate estimation* is also called *minimum-classification-error* (MCE) training, or discriminative training. Similar to MMIE, the algorithm generally tests the classifier using re-estimated models in the training procedure, and subsequently improves the correct models and suppresses mis-recognized or near-miss models.[9] Neural networks are in this class. Although minimum-error-rate estimation cannot be easily applied, it is still attractive that the criterion is identical to the goal of pattern recognition.

We have used the posterior probability $p(\omega_i \mid \mathbf{x})$ in Bayes' rule as the discriminant function. In fact, just about any discriminant function can be used for minimum-error-rate estimation. For example, as described in Section 4.2.1, a Bayes' Gaussian classifier is equivalent to a quadratic discriminant function. The goal now is to find the estimation of parameters for a discriminant function family $\{d_i(\mathbf{x})\}$ to achieve the minimum error rate. One such error measure is defined in Eq. (4.5). The difficulty associated with the discriminative training approach lies in the fact that the error function needs to be consistent with the true error rate measure and also suitable for optimization.[10] Unfortunately, the error function defined in Section 4.1.1 [Eq. (4.5)] is based on a finite set, which is a piecewise constant function of the parameter vector Φ. It is not suitable for optimization.

To find an alternative smooth error function for MCE, let us assume that the discriminant function family contains s discriminant functions $d_i(\mathbf{x}, \Phi)$, $i = 1, 2, \ldots, s$. Φ denotes the entire parameter set for s discriminant functions. We also assume that all the discriminant functions are nonnegative. We define the following error (misclassification) measure:

$$e_i(\mathbf{x}) = -d_i(\mathbf{x}, \Phi) + \left[\frac{1}{s-1} \sum_{j \neq i} d_j(\mathbf{x}, \Phi)^{\eta} \right]^{1/\eta} \tag{4.55}$$

where η is a positive number. The intuition behind the above measure is the attempt to enumerate the decision rule. For a ω_i class input $\mathbf{x}$, $e_i(\mathbf{x}) > 0$ implies recognition error, while $e_i(\mathbf{x}) \leq 0$ implies correct recognition. The number η can be thought to be a coefficient to select competing classes in Eq. (4.55). When $\eta = 1$, the competing class term is the average of all the competing discriminant function scores. When $\eta \to \infty$, the competing class term becomes $\max_{j \neq i} d_j(\mathbf{x}, \Phi)$ representing the discriminant function score for the top

[9] A near-miss model occurs when the incorrect model has higher likelihood than the correct model.
[10] In general, a function is optimizable if it is a smooth function and has a derivative.

competing class. By varying the value of η, one can take all the competing classes into account based on their individual significance.

To transform $e_i(\mathbf{x})$ into a normalized smooth function, we can use the sigmoid function to embed $e_i(\mathbf{x})$ in a smooth zero-one function. The loss function can be defined as follows:

$$l_i(\mathbf{x};\Phi) = sigmoid(e_i(\mathbf{x})) \tag{4.56}$$

where $sigmoid(x) = \frac{1}{1+e^{-x}}$ (4.57)

When $e_i(\mathbf{x})$ is a big negative number, which indicates correct recognition, the loss function $l_i(\mathbf{x};\Phi)$ has a value close to zero, which implies no loss incurred. On the other hand, when $e_i(\mathbf{x})$ is a positive number, it leads to a value between zero and one that indicates the likelihood of an error. Thus $l_i(\mathbf{x};\Phi)$ essentially represents a soft recognition error count.

For any data sample $\mathbf{x}$, the recognizer's loss function can be defined as:

$$l(\mathbf{x},\Phi) = \sum_{i=1}^{s} l_i(\mathbf{x},\Phi)\delta(\omega = \omega_i) \tag{4.58}$$

where $\delta(\bullet)$ is a Boolean function which will return 1 if the argument is true and 0 if the argument is false. Since $\mathbf{x}$ is a random vector, the expected loss according to Eq. (4.58) can be defined as:

$$L(\Phi) = E_{\mathbf{X}}(l(\mathbf{x},\Phi)) = \sum_{i=1}^{s} \int_{\omega=\omega_i} l(\mathbf{x},\Phi)p(\mathbf{x})d\mathbf{x} \tag{4.59}$$

Since $\max_{\Phi}\left[\int f(\mathbf{x},\Phi)d\mathbf{x}\right] = \int\left[\max_{\Phi} f(\mathbf{x},\Phi)\right]d\mathbf{x}$, Φ can be estimated by gradient descent over $l(\mathbf{x},\Phi)$ instead of expected loss $L(\Phi)$. That is, minimum classification error training of parameter Φ can be estimated by first choosing an initial estimate Φ_0 and the following iterative estimation equation:

$$\Phi^{t+1} = \Phi^t - \varepsilon_t \nabla l(\mathbf{x},\Phi)|_{\Phi=\Phi^t} \tag{4.60}$$

You can follow the gradient descent procedure described in Section 4.3.1.1 to achieve the MCE estimate of Φ.

Both MMIE and MCE are much more computationally intensive than MLE, owing to the inefficiency of gradient descent algorithms. Therefore, discriminant estimation methods, like MMIE and MCE, are usually used for tasks containing few classes or data samples. A

more pragmatic approach is *corrective training* [6], which is based on a very simple error-correcting procedure. First, a labeled training set is used to train the parameters for each corresponding class by standard MLE. For each training sample, a list of confusable classes is created by running the recognizer and kept as its *near-miss* list. Then, the parameters of the correct class are moved in the direction of the data sample, while the parameters of the "near-miss" class are moved in the opposite direction of the data samples. After all training samples have been processed; the parameters of all classes are updated. This procedure is repeated until the parameters for all classes converge. Although there is no theoretical proof that such a process converges, some experimental results show that it outperforms both MLE and MMIE methods [4].

We have described various estimators: minimum mean square estimator, maximum likelihood estimator, maximum posterior estimator, maximum mutual information estimator, and minimum error estimator. Although based on different training criteria, they are all powerful estimators for various pattern recognition problems. Every estimator has its strengths and weaknesses. It is almost impossible always to favor one over the others. Instead, you should study their characteristics and assumptions and select the most suitable ones for the domains you are working on.

In the following section we discuss *neural networks*. Both neural networks and MCE estimations follow a very similar discriminant training framework.

4.3.3. Neural Networks

In the area of pattern recognition, the advent of new learning procedures and the availability of high-speed parallel supercomputers have given rise to a renewed interest in neural networks.[11] Neural networks are particularly interesting for speech recognition, which requires massive constraint satisfaction, i.e., the parallel evaluation of many clues and facts and their interpretation in the light of numerous interrelated constraints. The computational flexibility of the human brain comes from its large number of neurons in a mesh of axons and dendrites. The communication between neurons is via the synapse and afferent fibers. There are many billions of neural connections in the human brain. At a simple level it can be considered that nerve impulses are comparable to the phonemes of speech. or to letters, in that they do not themselves convey meaning but indicate different intensities [95, 101] that are interpreted as meaningful units by *the language of the brain.* Neural networks attempt to achieve real-time response and humanlike performance using many simple processing elements operating in parallel as in biological nervous systems. Models of neural networks use a particular topology for the interactions and interrelations of the connections of the *neural units.* In this section we describe the basics of neural networks, including the multi-layer perceptrons and the back-propagation algorithm for training neural networks.

[11] A neural network is sometimes called an artificial neural network (ANN), a neural net, or a connectionist model.

4.3.3.1. Single-Layer Perceptrons

Figure 4.10 shows a basic *single-layer perceptron*. Assuming there are N inputs, labeled as $x_1, x_2, \ldots, x_N$, we can form a linear function with weights $w_{0j}, w_{1j}, w_{2j}, \ldots, w_{Nj}$ to give the output y_j, defined as

$$y_j = w_{0j} + \sum_{i=1}^{N} w_{ij} x_i = \mathbf{w}_j \mathbf{x}^t = g_j(\mathbf{x}) \tag{4.61}$$

where $\mathbf{w}_j = (w_{0j}, w_{1j}, w_{2j}, \ldots, w_{Nj})$ and $\mathbf{x} = (1, x_1, x_2, \ldots, x_N)$.

For pattern recognition purposes, we associate each class ω_j out of s classes $(\omega_1, \omega_2, \ldots, \omega_s)$ with such a linear discriminant function $g_j(\mathbf{x})$. By collecting all the discriminant functions, we will have the following matrix representation:

$$\mathbf{y} = \mathbf{g}(\mathbf{x}) = \mathbf{W}^t \mathbf{x} \tag{4.62}$$

where $\mathbf{g}(\mathbf{x}) = (g_1(\mathbf{x}), g_2(\mathbf{x}), \ldots, g_s(\mathbf{x}))^t$; $\mathbf{W} = (\mathbf{w}_1^t, \mathbf{w}_2^t, \ldots, \mathbf{w}_s^t)^t$ and $\mathbf{y} = (y_1, y_2, \ldots, y_s)^t$. The pattern recognition decision can then be based on these discriminant functions as in Bayes' decision theory. That is,

$$\mathbf{x} \in \omega_k \quad \text{iff} \quad k = \arg\max_i g_i(\mathbf{x}) \tag{4.63}$$

The *perceptron training algorithm* [68], guaranteed to converge for linearly separable classes, is often used for training the weight matrix $\mathbf{W}$. The algorithm basically divides the sample space $\Re^N$ into regions of corresponding classes. The decision boundary is characterized by hyper-planes of the following form:

$$g_i(\mathbf{x}) - g_j(\mathbf{x}) = 0 \quad \forall i \neq j \tag{4.64}$$

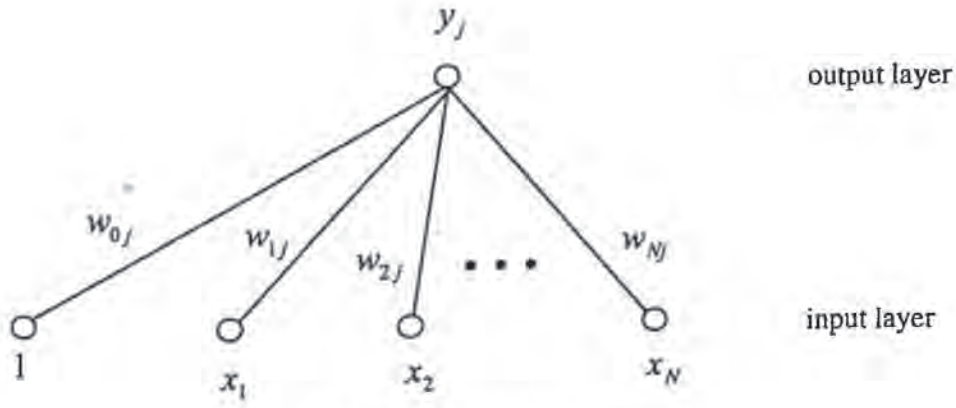


Figure 4.10 A single-layer perceptron.

Unfortunately, for data samples that are not linearly separable, the perceptron algorithm does not converge. However, if we can relax the definition of classification errors in this case, we can still use a powerful algorithm to train the weight matrix $\mathbf{W}$. This approach is the *least square error* (LSE) algorithm described in Chapter 3, which aims at minimizing *sum-of-squared-error* (SSE) criterion, instead of minimizing the classification errors. The sum-of-squared-error is defined as:

$$\text{SSE} = \sum_i \sum_{\mathbf{x}\in\omega_i} \| \mathbf{g}(\mathbf{x}) - \delta_i \|^2 \tag{4.65}$$

where δ_i is an M-dimensional *index vector* with all zero components except that the i^{th} one is 1.0, since the desired output for $\mathbf{g}(\mathbf{x})$ is typically equal to 1.0 if $\mathbf{x}\in\omega_i$ and 0 if $\mathbf{x}\notin\omega_i$.

The use of LSE leads to discriminant functions that have real outputs approximating the values 1 or 0. Suppose there are M input vectors $\mathbf{X} = (\mathbf{x}_1', \mathbf{x}_2', \ldots, \mathbf{x}_M')$ in the training set. Similar to the LSE for linear functions described in Chapter 3 (cf. Section 3.2.1.2), the LSE estimate of weight matrix $\mathbf{W}$ will have the following closed form:

$$\mathbf{W} = ((\mathbf{X}\mathbf{X}'))^{-1}\mathbf{L}\Sigma \tag{4.66}$$

where $\mathbf{L}$ is a $(N+1)\times s$ matrix where the k-th column is the mean vector $\boldsymbol{\mu}_k = (1, \mu_{k1}, \mu_{k2}, \ldots, \mu_{kN})'$ of all the vectors classified into class ω_k, and Σ is an $s\times s$ diagonal matrix with diagonal entry $c_{k,k}$ representing the number of vectors classified into ω_k. LSE estimation using linear discriminant functions is equivalent to estimating Bayes' Gaussian densities where all the densities are assumed to share the same covariance matrix [98], as described in Section 4.2.1.

Although the use of LSE algorithm solves the convergence problems, it loses the power of nonlinear logical decision (i.e., minimizing the classification error rate), since it is only approximating the simple logical decision between alternatives. An alternative approach is to use a smooth and differential sigmoid function as the threshold function:

$$\begin{aligned}\mathbf{y} &= sigmoid(\mathbf{g}(\mathbf{x})) = sigmoid((g_1(\mathbf{x}), g_2(\mathbf{x}), \ldots, g_s(\mathbf{x}))') \\ &= (sigmoid(g_1(\mathbf{x})), sigmoid(g_2(\mathbf{x})), \ldots, sigmoid(g_s(\mathbf{x})))'\end{aligned} \tag{4.67}$$

where $sigmoid(x)$ is the sigmoid function defined in Eq. (4.57). With the sigmoid function, the following new sum-of-squared-error term closely tracks the classification error:

$$\text{NSSE} = \sum_i \sum_{\mathbf{x}\in\omega_i} \| sigmoid(\mathbf{g}(\mathbf{x})) - \delta_i \|^2 \tag{4.68}$$

where δ_i is the same index vector defined above. Since there is no analytic way of minimizing a nonlinear function, the use of the sigmoid threshold function requires an iterative gradient descent algorithm, back-propagation, which will be described in the next section.

4.3.3.2. Multi-Layer Perceptron

One of the technical developments sparking the recent resurgence of interest in neural networks has been the popularization of *multi-layer perceptrons* (MLP) [37, 90]. Figure 4.11 shows a multi-layer perceptron. In contrast to a single-layer perceptron, it has two hidden layers. The hidden layers can be viewed as feature extractors. Each layer has the same computation models as the single-layer perceptron; i.e., the value of each node is computed as a linear weighted sum of the input nodes and passed to a sigmoid type of threshold function.

$$\begin{aligned} \mathbf{h}_1 &= sigmoid(\mathbf{g}_{h1}(\mathbf{x})) = sigmoid(\mathbf{W}_{h1}^t \mathbf{x}) \\ \mathbf{h}_2 &= sigmoid(\mathbf{g}_{h2}(\mathbf{h}_1)) = sigmoid(\mathbf{W}_{h2}^t \mathbf{h}_1) \\ \mathbf{y} &= sigmoid(\mathbf{g}_y(\mathbf{h}_2)) = sigmoid(\mathbf{W}_y^t \mathbf{h}_2) \end{aligned} \tag{4.69}$$

where $sigmoid(x)$ is the sigmoid function defined in Eq. (4.57).

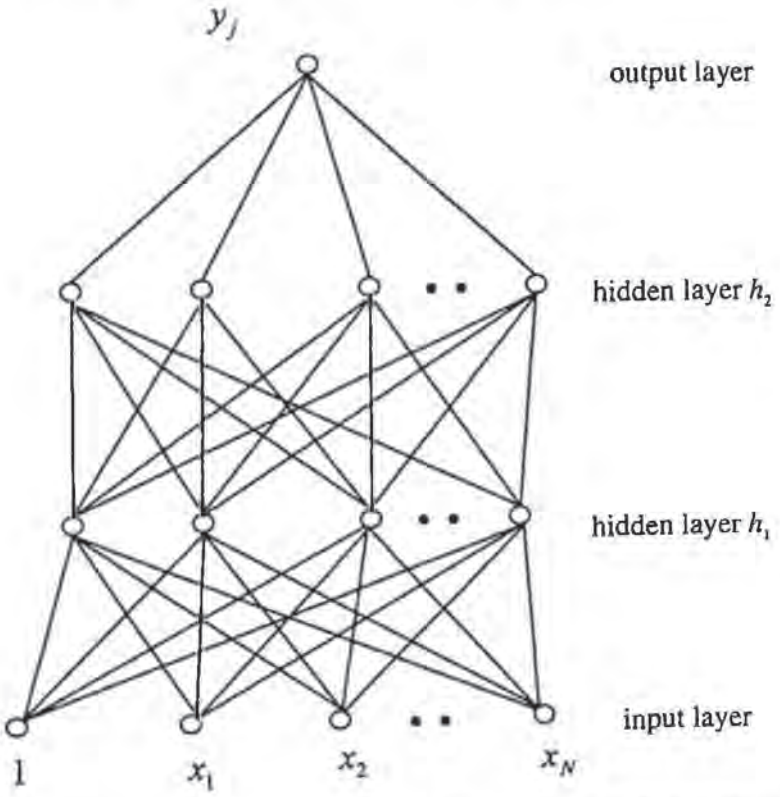


Figure 4.11 A multi-layer perceptron with four total layers. The middle two layers are hidden.

According to Eq. (4.69), we can propagate the computation from input layer to output layer and denote the output layer as a nonlinear function of the input layer.

$$\mathbf{Y} = MLP(\mathbf{x}) \tag{4.70}$$

Let's denote $O(\mathbf{x})$ as the desired output for input vector $\mathbf{x}$. For pattern classification, $O(\mathbf{x})$ will be an s-dimensional vector with the desired output pattern set to one and the remaining patterns set to zero. As we mentioned before, there is no analytic way to minimize the mean square error $E = \sum \| MLP(\mathbf{x}) - O(\mathbf{x}) \|^2$. Instead, an iterative gradient descent algorithm called back propagation [89, 90] needs to be used to reduce error. Without loss of generality, we assume there is only one input vector $\mathbf{x} = (1, x_1, x_2, \ldots, x_N)$ with desired output $\mathbf{o} = (o_1, o_2, \ldots, o_s)$. All the layers in the MLP are numbered 0, 1, 2,... upward from the input layer. The back propagation algorithm can then be described as in Algorithm 4.1.

In computing the partial derivative $\frac{\partial E}{\partial w_{ij}^k(t)}$, you need to use the chain rule. w_{ij}^K is the weight connecting the output layer and the last hidden layer; the partial derivative is:

$$\begin{aligned}
\frac{\partial E}{\partial w_{ij}^K} &= \frac{\partial (\sum_{i=1}^{s} (y_i - o_i)^2)}{\partial w_{ij}^K} \\
&= \frac{\partial (\sum_{i=1}^{s} (y_i - o_i)^2)}{\partial y_j} \times \frac{\partial y_j}{\partial (w_{0j}^K + \sum_{i=1}^{N} w_{ij}^K v_i^{K-1})} \times \frac{\partial (w_{0j}^K + \sum_{i=1}^{N} w_{ij}^K v_i^{K-1})}{\partial w_{ij}^K} \\
&= 2(y_i - o_i) y_j (y_j - 1) v_i^{K-1}
\end{aligned} \tag{4.71}$$

For layers $k = K-1, K-2, \cdots$, one can apply chain rules similarly for gradient $\frac{\partial E}{\partial w_{ij}^k(t)}$.

The back propagation algorithm is a generalization of the minimum mean squared error (MMSE) algorithm. It uses a gradient search to minimize the difference between the desired outputs and the actual net outputs, where the optimized criterion is directly related to pattern classification. With initial parameters for the weights, the training procedure is then repeated to update the weights until the cost function is reduced to an acceptable value or remains unchanged. In the algorithm described above, we assume a single training example. In real-world application, these weights are estimated from a large number of training observations in a manner similar to hidden Markov modeling. The weight updates in Step 3 are accumulated over all the training data. The actual gradient is then estimated for the complete set of training data before the beginning of the next iteration. Note that the estimation criterion for neural networks is directly related to classification rather than maximum likelihood.

ALGORITHM 4.1: *THE BACK PROPAGATION ALGORITHM*

Step 1: Initialization: Set $t = 0$ and choose initial weight matrices $\mathbf{W}$ for each layer. Let's denote $w_{ij}^k(t)$ as the weighting coefficients connecting i^{th} input node in layer $k-1$ and j^{th} output node in layer k at time t.

Step 2: Forward Propagation: Compute the values in each node from input layer to output layer in a propagating fashion, for $k = 1$ to K

$$v_j^k = sigmoid(w_{0j}(t) + \sum_{i=1}^{N} w_{ij}^k(t) v_i^{k-1}) \quad \forall j \tag{4.72}$$

where $sigmoid(x) = \frac{1}{1+e^{-x}}$ and v_j^k is denoted as the j^{th} node in the k^{th} layer

Step 3: Back Propagation: Update the weights matrix for each layer from output layer to input layer according to:

$$\overline{w}_{ij}^k(t+1) = w_{ij}^k(t) - \alpha \frac{\partial E}{\partial w_{ij}^k(t)} \tag{4.73}$$

where $E = \sum_{i=1}^{s} \| y_i - o_i \|^2$ and $(y_1, y_2, \ldots y_s)$ is the computed output vector in Step 2.

α is referred to as the learning rate and has to be small enough to guarantee convergence. One popular choice is $1/(t+1)$.

Step 4: Iteration: Let $t = t+1$. Repeat Steps 2 and 3 until some convergence condition is met.

4.4. UNSUPERVISED ESTIMATION METHODS

As described in Section 4.2, in unsupervised learning, information about class ω of the data sample $\mathbf{x}$ is unavailable. Data observed are *incomplete* since the class data ω is missing. One might wonder why we are interested in such an unpromising problem, and whether or not it is possible to learn anything from incomplete data. Interestingly enough, the formal solution to this problem is almost identical to the solution for the supervised learning case – MLE. We discuss vector quantization (VQ), which uses principles similar to the EM algorithm. It is important in its own right in spoken language systems.

4.4.1. Vector Quantization

As described in Chapter 3, source coding refers to techniques that convert the signal source into a sequence of bits that are transmitted over a communication channel and then used to

reproduce the original signal at a different location or time. In speech communication, the reproduced sound usually allows some acceptable level of distortion to achieve low bit rate. The goal of source coding is to reduce the number of bits necessary to transmit or store data, subject to a distortion or fidelity criterion, or equivalently, to achieve the minimum possible distortion for a prescribed bit rate. *Vector quantization* (VQ) is one of the most efficient source-coding techniques.

Quantization is the process of approximating continuous amplitude signals by discrete symbols. The quantization of a single signal value or parameter is referred to as scalar quantization. In contrast, joint quantization of multiple signal values or parameters is referred to as vector quantization. Conventional pattern recognition techniques have been used effectively to solve the quantization or data compression problem with successful application to speech coding, image coding, and speech recognition [36, 85]. In both speech recognition and synthesis systems, vector quantization serves an important role in many aspects of the systems, ranging from discrete acoustic prototypes of speech signals for the discrete HMM, to robust signal processing and data compression.

A vector quantizer is described by a codebook, which is a set of fixed *prototype vectors* or reproduction vectors. Each of these prototype vectors is also referred to as a codeword. To perform the quantization process, the input vector is matched against each codeword in the codebook using some *distortion measure*. The input vector is then replaced by the index of the codeword with the smallest distortion. Therefore, a description of the vector quantization process includes:

1. the distortion measure;
2. the generation of each codeword in the codebook.

4.4.1.1. Distortion Measures

Since vectors are replaced by the index of the codeword with smallest distortion, the transmitted data can be recovered only by replacing the code index sequence with the corresponding codeword sequence. This inevitably causes distortion between the original data and the transmitted data. How to minimize the distortion is thus the central goal of vector quantization. This section describes a couple of the most common distortion measures.

Assume that $\mathbf{x} = (x_1, x_2, \ldots, x_d)^t \in R^d$ is a d-dimensional vector whose components $\{x_k, 1 \le k \le d\}$ are real-valued, continuous-amplitude random variables. After vector quantization, the vector $\mathbf{x}$ is mapped (quantized) to another discrete-amplitude d-dimensional vector $\mathbf{z}$.

$$\mathbf{z} = q(\mathbf{x}) \tag{4.74}$$

In Eq. (4.74) $q()$ is the quantization operator. Typically, $\mathbf{z}$ is a vector from a finite set $\mathbf{Z} = \{\mathbf{z}_j, 1 \le j \le M\}$, where $\mathbf{z}_j$ is also a d-dimensional vector. The set $\mathbf{Z}$ is referred to as the codebook, M is the size of the codebook, and $\mathbf{z}_j$ is j^{th} *codeword*. The size M of the codebook is also called the number of partitions (or levels) in the codebook. To design a codebook, the

d-dimensional space of the original random vector $\mathbf{x}$ can be partitioned into M regions or cells $\{C_i, 1 \le i \le M\}$, and each cell C_i is associated with a codeword vector $\mathbf{z}_i$. VQ then maps (quantizes) the input vector $\mathbf{x}$ to codeword $\mathbf{z}_i$ if $\mathbf{x}$ lies in C_i. That is,

$$q(\mathbf{x}) = \mathbf{z}_i \text{ if } \mathbf{x} \in C_i \tag{4.75}$$

An example of partitioning of a two-dimensional space ($d = 2$) for the purpose of vector quantization is shown in Figure 4.12. The shaded region enclosed by the dashed lines is the cell C_i. Any input vector $\mathbf{x}$ that lies in the cell C_i is quantized as $\mathbf{z}_i$. The shapes of the various cells can be different. The positions of the codewords within each cell are shown by dots in Figure 4.12. The codeword $\mathbf{z}_i$ is also referred to as the *centroid* of the cell C_i because it can be viewed as the central point of the cell C_i.

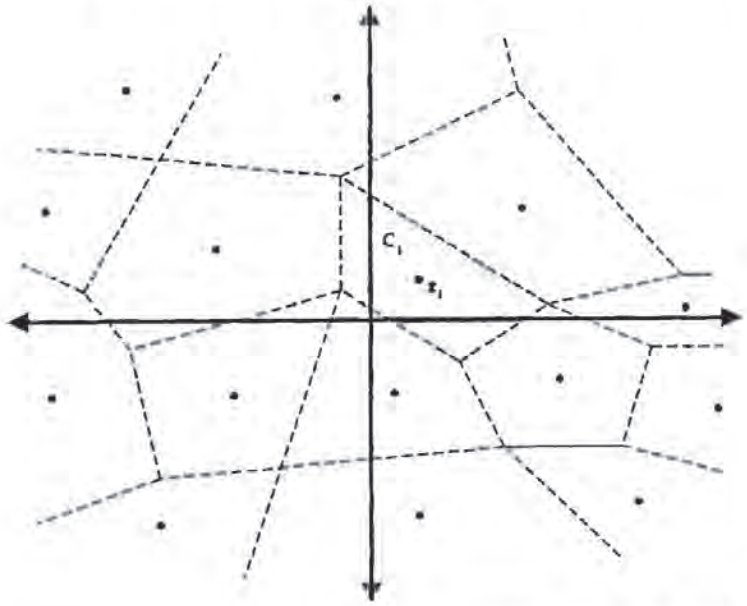


Figure 4.12 Partitioning of a two-dimensional space into 16 cells.

When $\mathbf{x}$ is quantized as $\mathbf{z}$, a quantization error results. A distortion measure $d(\mathbf{x}, \mathbf{z})$ can be defined between $\mathbf{x}$ and $\mathbf{z}$ to measure the quantization quality. Using this distortion measure, Eq. (4.75) can be reformulated as follows:

$$q(\mathbf{x}) = \mathbf{z}_i \text{ if and only if } i = \underset{k}{\operatorname{argmin}}\, d(\mathbf{x}, \mathbf{z}_k) \tag{4.76}$$

The distortion measure between $\mathbf{x}$ and $\mathbf{z}$ is also known as a distance measure in the speech context. The measure must be tractable in order to be computed and analyzed, and also must be subjectively relevant so that differences in distortion values can be used to indicate differences in original and transmitted signals. The most commonly used measure is the Euclidean distortion measure, which assumes that the distortions contributed by quantiz-

ing the different parameters are equal. Therefore, the distortion measure $d(\mathbf{x}, \mathbf{z})$ can be defined as follows:

$$d(\mathbf{x},\mathbf{z}) = (\mathbf{x}-\mathbf{z})'(\mathbf{x}-\mathbf{z}) = \sum_{i=1}^{d}(x_i - z_i)^2 \tag{4.77}$$

The distortion defined in Eq. (4.77) is also known as sum of squared error. In general, unequal weights can be introduced to weight certain contributions to the distortion more than others. One choice for weights that is popular in many practical applications is to use the inverse of the covariance matrix of $\mathbf{z}$.

$$d(\mathbf{x},\mathbf{z}) = (\mathbf{x}-\mathbf{z})'\Sigma^{-1}(\mathbf{x}-\mathbf{z}) \tag{4.78}$$

This distortion measure, known as the *Mahalanobis* distance, is actually the exponential term in a Gaussian density function.

Another way to weight the contributions to the distortion measure is to use *perceptually*-based distortion measures. Such distortion measures take advantage of subjective judgments of perceptual difference caused by two different signals. A perceptually-based distortion measure has the property that signal changes that make the sounds being perceived different should be associated with large distances. Similarly signal changes that keep the sound perceived the same should be associated with small distances. A number of perceptually based distortion measures have been used in speech coding [3, 75, 76].

4.4.1.2. The *K*-Means Algorithm

To design an M-level codebook, it is necessary to partition d-dimensional space into M cells and associate a quantized vector with each cell. Based on the source-coding principle, the criterion for optimization of the vector quantizer is to minimize overall average distortion over all M-levels of the VQ. The overall average distortion can be defined by

$$\begin{aligned} D &= E[d(\mathbf{x},\mathbf{z})] = \sum_{i=1}^{M} p(\mathbf{x}\in C_i)E[d(\mathbf{x},\mathbf{z}_i)|\,\mathbf{x}\in C_i] \\ &= \sum_{i=1}^{M} p(\mathbf{x}\in C_i)\int_{\mathbf{x}\in C_i} d(\mathbf{x},\mathbf{z}_i)p(\mathbf{x}|\,\mathbf{x}\in C_i)d\mathbf{x} = \sum_{i=1}^{M} D_i \end{aligned} \tag{4.79}$$

where the integral is taken over all components of vector $\mathbf{x}$; $p(\mathbf{x}\in C_i)$ denotes the prior probability of codeword $\mathbf{z}_i$; $p(\mathbf{x}|\,\mathbf{x}\in C_i)$ denotes the multidimensional probability density function of $\mathbf{x}$ in cell C_i; and D_i is the average distortion in cell C_i. No analytic solution exists to guarantee global minimization of the average distortion measure for a given set of speech data. However, an iterative algorithm, which guarantees a local minimum, exists and works well in practice.

We say a quantizer is optimal if the overall average distortion is minimized over all M-levels of the quantizer. There are two necessary conditions for optimality. The first is that the optimal quantizer is realized by using a nearest-neighbor selection rule as specified by Eq. (4.76). Note that the average distortion for each cell C_i

$$E\left[d(\mathbf{x},\mathbf{z}_i)\,|\,\mathbf{x}\in C_i\right] \tag{4.80}$$

can be minimized when $\mathbf{z}_i$ is selected such that $d(\mathbf{x},\mathbf{z}_i)$ is minimized for $\mathbf{x}$. This means that the quantizer must choose the codeword that results in the minimum distortion with respect to $\mathbf{x}$. The second condition for optimality is that each codeword $\mathbf{z}_i$ is chosen to minimize the average distortion in cell C_i. That is, $\mathbf{z}_i$ is the vector that minimizes

$$D_i = p(\mathbf{z}_i)E\left[d(\mathbf{x},\mathbf{z})|\mathbf{x}\in C_i\right] \tag{4.81}$$

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

$$\mathbf{z}_i = \text{cent}(C_i) \tag{4.82}$$

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

$$D_i = \frac{1}{T}\sum_{\mathbf{x}\in C_i} d(\mathbf{x},\mathbf{z}_i) \tag{4.83}$$

The second condition for optimality can then be rewritten as follows:

$$\hat{\mathbf{z}}_i = \arg\min_{\mathbf{z}_i} D_i(\mathbf{z}_i) = \arg\min_{\mathbf{z}_i}\frac{1}{T}\sum_{\mathbf{x}\in C_i} d(\mathbf{x},\mathbf{z}_i) \tag{4.84}$$

When the sum of squared error in Eq. (4.77) is used for the distortion measure, the attempt to find such $\hat{\mathbf{z}}_i$ to minimize the sum of squared error is equivalent to least squared error estimation, which was described in Chapter 3. Minimization of D_i in Eq. (4.84) with respect to $\mathbf{z}_i$ is given by setting the derivative of D_i to zero:

$$\begin{aligned}\nabla_{\mathbf{z}_i} D_i &= \nabla_{\mathbf{z}_i}\frac{1}{T}\sum_{\mathbf{x}\in C_i}(\mathbf{x}-\mathbf{z}_i)'(\mathbf{x}-\mathbf{z}_i)\\ &= \frac{1}{T}\sum_{\mathbf{x}\in C_i}\nabla_{\mathbf{z}_i}(\mathbf{x}-\mathbf{z}_i)'(\mathbf{x}-\mathbf{z}_i)\\ &= \frac{-2}{T}\sum_{\mathbf{x}\in C_i}(\mathbf{x}-\mathbf{z}_i) = 0\end{aligned} \tag{4.85}$$

By solving Eq. (4.85), we obtain the least square error estimate of centroid $\hat{\mathbf{z}}_i$ simply as the sample mean of all the training vectors $\mathbf{x}$, quantized to cell C_i:

$$\hat{\mathbf{z}}_i = \frac{1}{K_i} \sum_{\mathbf{x} \in C_i} \mathbf{x} \tag{4.86}$$

If the Mahalanobis distance measure [Eq. (4.78)] is used, minimization of D_i in Eq. (4.84) can be done similarly:

$$\begin{aligned} \nabla_{\mathbf{z}_i} D_i &= \nabla_{\mathbf{z}_i} \frac{1}{T} \sum_{\mathbf{x} \in C_i} (\mathbf{x} - \mathbf{z}_i)' \Sigma^{-1} (\mathbf{x} - \mathbf{z}_i) \\ &= \frac{1}{T} \sum_{\mathbf{x} \in C_i} \nabla_{\mathbf{z}_i} (\mathbf{x} - \mathbf{z}_i)' \Sigma^{-1} (\mathbf{x} - \mathbf{z}_i) \\ &= \frac{-2}{T} \sum_{\mathbf{x} \in C_i} \Sigma^{-1} (\mathbf{x} - \mathbf{z}_i) = 0 \end{aligned} \tag{4.87}$$

and centroid $\hat{\mathbf{z}}_i$ is obtained from

$$\hat{\mathbf{z}}_i = \frac{1}{K_i} \sum_{\mathbf{x} \in C_i} \mathbf{x} \tag{4.88}$$

One can see that $\hat{\mathbf{z}}_i$ is again the sample mean of all the training vectors $\mathbf{x}$, quantized to cell C_i. Although Eq. (4.88) is obtained based on the Mahalanobis distance measure, it also works with a large class of Euclidean-like distortion measures [61]. Since the Mahalanobis distance measure is actually the exponential term in a Gaussian density, minimization of the distance criterion can be easily translated into maximization of the logarithm of the Gaussian likelihood. Therefore, similar to the relationship between least square error estimation for the linear discrimination function and the Gaussian classifier described in Section 4.3.3.1, the distance minimization process (least square error estimation) above is in fact a *maximum likelihood estimation*.

According to these two conditions for VQ optimality, one can iteratively apply the nearest-neighbor selection rule and Eq. (4.88) to get the new centroid $\hat{\mathbf{z}}_i$ for each cell in order to minimize the average distortion measure. This procedure is known as the *k-means* algorithm or the *generalized Lloyd* algorithm [29, 34, 56]. In the *k*-means algorithm, the basic idea is to partition the set of training vectors into M clusters C_i $(1 \le i \le M)$ in such a way that the two necessary conditions for optimality described above are satisfied. The *k*-means algorithm can be described as in Algorithm 4.2.

ALGORITHM 4.2: *THE K-MEANS ALGORITHM*

Step 1: Initialization: Choose some adequate method to derive initial VQ codewords ($\mathbf{z}_i$, $1 \le i \le M$) in the codebook.

Step 2: Nearest-neighbor Classification: Classify each training vector $\{\mathbf{x}_k\}$ into one of the cells C_i by choosing the closest codeword $\mathbf{z}_i$ ($\mathbf{x} \in C_i$, i.f.f. $d(\mathbf{x},\mathbf{z}_i) \le d(\mathbf{x},\mathbf{z}_j)$ for all $j \ne i$). This classification is also called minimum-distance classifier.

Step 3: Codebook Updating: Update the codeword of every cell by computing the centroid of the training vectors in each cell according to Eq. (4.84) ($\hat{\mathbf{z}}_i = cent(C_i)$, $1 \le i \le M$).

Step 4: Iteration: Repeat steps 2 and 3 until the ratio of the new overall distortion D at the current iteration relative to the overall distortion at the previous iteration is above a preset threshold.

In the process of minimizing the average distortion measure, the k-means procedure actually breaks the minimization process into two steps. Assuming that the centroid $\mathbf{z}_i$ (or mean) for each cell C_i has been found, then the minimization process is found simply by partitioning all the training vectors into their corresponding cells according to the distortion measure. After all of the new partitions are obtained, the minimization process involves finding the new centroid within each cell to minimize its corresponding within-cell average distortion D_i based on Eq. (4.84). By iterating over these two steps, a new overall distortion D smaller than that of the previous step can be obtained.

Theoretically, the k-means algorithm can converge only to a local optimum [56]. Furthermore, any such solution is, in general, not unique [33]. Initialization is often critical to the quality of the eventual converged codebook. Global optimality may be approximated by repeating the k-means algorithm for several sets of codebook initialization values, and then one can choose the codebook that produces the minimum overall distortion. In the next subsection we will describe methods for finding a decent initial codebook.

4.4.1.3. The LBG Algorithm

Since the initial codebook is critical to the ultimate quality of the final codebook, it has been shown that it is advantageous to design an M-vector codebook in stages. This extended k-means algorithm is known as the LBG algorithm proposed by Linde, Buzo, and Gray [56]. The LBG algorithm first computes a 1-vector codebook, then uses a splitting algorithm on the codewords to obtain the initial 2-vector codebook, and continues the splitting process until the desired M-vector codebook is obtained. The procedure is formally implemented by Algorithm 4.3.

ALGORITHM 4.3: *THE LBG ALGORITHM*

Step 1: Initialization: Set M (number of partitions or cells) =1. Find the centroid of all the training data according to Eq. (4.84).
Step 2: Splitting: Split M into $2M$ partitions by splitting each current codeword by finding two points that are far apart in each partition using a heuristic method, and use these two points as the new centroids for the new $2M$ codebook. Now set $M = 2M$.
Step 3: K-means Stage: Now use the k-means iterative algorithm described in the previous section to reach the best set of centroids for the new codebook.
Step 4: Termination: If M equals the VQ codebook size required, STOP; otherwise go to Step 2.

4.4.2. The EM Algorithm

We introduce the EM algorithm that is important to hidden Markov models and other learning techniques. It discovers model parameters by maximizing the log-likelihood of incomplete data and by iteratively maximizing the expectation of log-likelihood from complete data. The EM algorithm is a generalization of the VQ algorithm described above.

The EM algorithm can also be viewed as a generalization of the MLE method, when the data observed is incomplete. Without loss of generality, we use scalar random variables here to describe the EM algorithm. Suppose we observe training data y. In order to determine the parameter vector Φ that maximizes $P(Y=y|\Phi)$, we would need to know some hidden data x (that is unobserved). For example, x may be a hidden number that refers to component densities of observable data y, or x may be the underlying hidden state sequence in *hidden Markov models* (as discussed in Chapter 8). Without knowing this hidden data x, we could not easily use the maximum likelihood estimation to estimate $\hat{\Phi}$, which maximizes $P(Y=y|\Phi)$. Instead, we assume a parameter vector Φ and estimate the probability that each x occurred in the generation of y. This way we can pretend that we had in fact observed a complete data pair (x, y), with frequency proportional to the probability $P(X=x,Y=y|\Phi)$, to compute a new $\bar{\Phi}$, the maximum likelihood estimate of Φ. We can then set the parameter vector Φ to be this new $\bar{\Phi}$ and repeat the process to iteratively improve our estimate.

The issue now is whether or not the process (EM algorithm) described above converges. Without loss of generality, we assume that both random variables X (unobserved) and Y (observed) are discrete random variables. According to Bayes' rule,

$$P(X=x,Y=y|\bar{\Phi}) = P(X=x|Y=y,\bar{\Phi})P(Y=y|\bar{\Phi}) \tag{4.89}$$

Our goal is to maximize the log-likelihood of the observable, real data y generated by parameter vector $\bar{\Phi}$. Based on Eq. (4.89), the log-likelihood can be expressed as follows:

$$\log P(Y=y|\bar{\Phi}) = \log P(X=x,Y=y|\bar{\Phi}) - \log P(X=x|Y=y,\bar{\Phi}) \tag{4.90}$$

Now, we take the conditional expectation of $\log P(Y=y|\bar{\Phi})$ over X computed with parameter vector Φ:

$$\begin{aligned} E_{\Phi}[\log P(Y=y|\bar{\Phi})]_{X|Y=y} &= \sum_{x}\left(P(X=x|Y=y,\Phi)\log P(Y=y|\bar{\Phi})\right) \\ &= \log P(Y=y|\bar{\Phi}) \end{aligned} \tag{4.91}$$

where we denote $E_{\Phi}[f]_{X|Y=y}$ as the expectation of function f over X computed with parameter vector Φ. Then using Eq. (4.90) and (4.91), the following expression is obtained:

$$\begin{aligned} \log P(Y=y|\bar{\Phi}) &= E_{\Phi}[\log P(X,Y=y|\bar{\Phi})]_{X|Y=y} - E_{\Phi}[\log P(X|Y=y,\bar{\Phi})]_{X|Y=y} \\ &= Q(\Phi,\bar{\Phi}) - H(\Phi,\bar{\Phi}) \end{aligned} \tag{4.92}$$

where

$$\begin{aligned} Q(\Phi,\bar{\Phi}) &= E_{\Phi}[\log P(X,Y=y|\bar{\Phi})]_{X|Y=y} \\ &= \sum_{x}\left(P(X=x|Y=y,\Phi)\log P(X=x,Y=y|\bar{\Phi})\right) \end{aligned} \tag{4.93}$$

and

$$\begin{aligned} H(\Phi,\bar{\Phi}) &= E_{\Phi}[\log P(X|Y=y,\bar{\Phi})]_{X|Y=y} \\ &= \sum_{x}\left(P(X=x|Y=y,\bar{\Phi})\log P(X=x|Y=y,\bar{\Phi})\right) \end{aligned} \tag{4.94}$$

The convergence of the EM algorithm lies in the fact that if we choose $\bar{\Phi}$ so that

$$Q(\Phi,\bar{\Phi}) \geq Q(\Phi,\Phi) \tag{4.95}$$

then

$$\log P(Y=y|\bar{\Phi}) \geq \log P(Y=y|\Phi) \tag{4.96}$$

since it follows from Jensen's inequality that $H(\Phi,\bar{\Phi}) \leq H(\Phi,\Phi)$ [21]. The function $Q(\Phi,\bar{\Phi})$ is known as the Q-function or auxiliary function. This fact implies that we can maximize the Q-function, which is the expectation of log-likelihood from complete data pair (x, y), to update parameter vector from Φ to $\bar{\Phi}$, so that the incomplete log-likelihood $L(x,\Phi)$ increases monotonically. Eventually, the likelihood will converge to a local maximum if we iterate the process.

The name of the EM algorithm comes from E for expectation and M for maximization. The implementation of the EM algorithm includes the E (expectation) step, which calculates the auxiliary Q-function $Q(\Phi,\bar{\Phi})$ and the M (maximization) step, which maximizes $Q(\Phi,\bar{\Phi})$ over $\bar{\Phi}$ to obtain $\hat{\Phi}$. The general EM algorithm can be described in the following way.

ALGORITHM 4.4: *THE EM ALGORITHM*

Step 1: Initialization: Choose an initial estimate Φ.
Step 2: E-Step: Compute auxiliary Q-function $Q(\Phi, \bar{\Phi})$ (which is also the expectation of log-likelihood from complete data) based on Φ.
Step 3: M-Step: Compute $\hat{\Phi} = \arg\max_{\Phi} Q(\Phi, \bar{\Phi})$ to maximize the auxiliary Q-function.
Step 4: Iteration: Set $\Phi = \hat{\Phi}$, repeat from Step 2 until convergence.

The M-step of the EM algorithm is actually a maximum likelihood estimation of complete data (assuming we know the unobserved data x based on observed data y and initial parameter vector Φ). The EM algorithm is usually used in applications where no analytic solution exists for maximization of log-likelihood of incomplete data. Instead, the Q-function is iteratively maximized to obtain the estimation of parameter vector.

4.4.3. Multivariate Gaussian Mixture Density Estimation

The vector quantization process described in Section 4.4.1 partitions the data space into separate regions based on some distance measure regardless of the probability distributions of the original data. This process may introduce errors in partitions that could potentially destroy the original structure of data. An alternative way for modeling a VQ codebook is to use a family of Gaussian probability density functions, such that each cell will be represented by a (Gaussian) probability density function as shown in Figure 4.13. These probability density functions can then overlap, rather than partition, in order to represent the entire data space. The objective for a mixture Gaussian VQ is to maximize the likelihood of the observed data (represented by the product of the Gaussian mixture scores) instead of minimizing the overall distortion. The centroid of each cell (the mean vectors of each Gaussian pdf) obtained via such a representation may be quite different from that obtained using the traditional k-means algorithm, since the distribution properties of the data are taken into account.

There should be an obvious analogy between the EM algorithm and the k-means algorithm described in the Section 4.4.1.2. In the k-means algorithm, the class information for the observed data samples is hidden and unobserved, so an EM-like algorithm instead of maximum likelihood estimate needs to be used. Therefore, instead of a single process of maximum likelihood estimation, the k-means algorithm first uses the old codebook to find the nearest neighbor for each data sample followed by maximum likelihood estimation of the new codebook and iterates the process until the distortion stabilizes. The steps 2 and 3 in the k-means algorithm are actually the E and M steps in the EM algorithm respectively.

Mixture density estimation [41] is a typical example of EM estimation. In the mixtures of Gaussian density, the probability density for observable data $\mathbf{y}$ is the weighted sum of each Gaussian component:

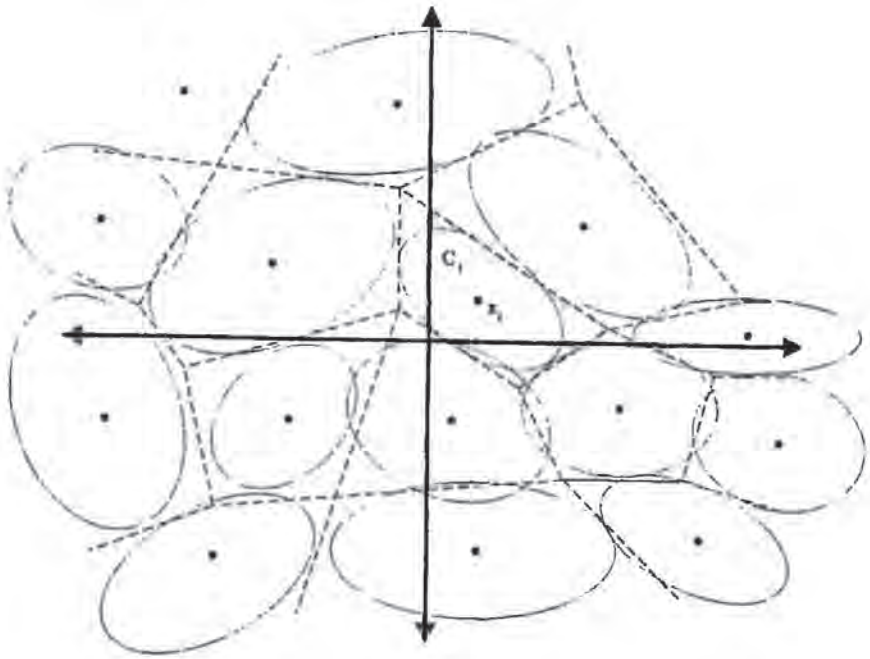


Figure 4.13 Partitioning of a two-dimensional space with 16 Gaussian density functions.

$$p(\mathbf{y} \mid \Phi) = \sum_{k=1}^{K} c_k p_k(\mathbf{y} \mid \Phi_k) = \sum_{k=1}^{K} c_k N_k(\mathbf{y} \mid \mu_k, \Sigma_k) \tag{4.97}$$

where $0 \le c_k \le 1$, for $1 \le k \le K$ and $\sum_{k=1}^{K} c_k = 1$.

Unlike the case of a single Gaussian estimation, we also need to estimate the mixture weight c_k. In order to do so, we can assume that observable data $\mathbf{y}$ come from one of the component densities $p_X(\mathbf{y} \mid \Phi_X)$, where X is a random variable taking value from $\{1,2,\ldots K\}$ to indicate the Gaussian component. It is clear that x is unobserved and used to specify the pdf component Φ_X. Assuming that the probability density function for complete data $(x,\mathbf{y})$ is given by the joint probability:

$$p(\mathbf{y}, x \mid \Phi) = P(X = x) p_x(\mathbf{y} \mid \Phi_x) = P(X = x) N_x(\mathbf{y} \mid \mu_x, \Sigma_x) \tag{4.98}$$

$P(X = x)$ can be regarded as the probability of the unobserved data x used to specify the component density $p_x(\mathbf{y} \mid \Phi_x)$ from which the observed data $\mathbf{y}$ is drawn. If we assume the number of components is K and Φ is the vector of all probability parameters $(P(X), \Phi_1, \Phi_2, \ldots, \Phi_K)$, the probability density function of incomplete (observed) data $\mathbf{y}$ can be specified as the following marginal probability:

$$p(\mathbf{y} \mid \Phi) = \sum_{x} p(\mathbf{y}, x \mid \Phi) = \sum_{x} P(X = x) p_x(\mathbf{y} \mid \Phi_x) \tag{4.99}$$

By comparing Eq. (4.97) and (4.99), we can see that the mixture weight is represented as the probability function $P(X = x)$. That is,

$$c_k = P(X = k) \tag{4.100}$$

According to the EM algorithm, the maximization of the logarithm of the likelihood function $\log p(\mathbf{y} \mid \Phi)$ can be performed by iteratively maximizing the conditional expectation of the logarithm of Eq. (4.98), i.e., $\log p(\mathbf{y}, x \mid \Phi)$. Suppose we have observed N independent samples: $\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N\}$ with hidden unobserved data $\{x_1, x_2, \ldots, x_N\}$; the Q-function can then be written as follows:

$$\begin{aligned} Q(\Phi, \bar{\Phi}) &= \sum_{l=1}^{N} Q_l(\Phi, \bar{\Phi}) = \sum_{l=1}^{N} \sum_{x_l} P(x_l \mid \mathbf{y}_l, \Phi) \log p(\mathbf{y}_l, x_l \mid \bar{\Phi}) \\ &= \sum_{l=1}^{N} \sum_{x_l} \frac{p(\mathbf{y}_l, x_l \mid \Phi)}{p(\mathbf{y}_l \mid \Phi)} \log p(\mathbf{y}_l, x_l \mid \bar{\Phi}) \end{aligned} \tag{4.101}$$

By replacing items in Eq. (4.101) with Eqs. (4.98) and (4.100), the following equation can be obtained:

$$Q(\Phi, \bar{\Phi}) = \sum_{k=1}^{K} \gamma_k \log \bar{c}_k + \sum_{k=1}^{K} Q_\lambda(\Phi, \bar{\Phi}_k) \tag{4.102}$$

where

$$\gamma_k^l = \frac{c_k p_k(\mathbf{y}_l \mid \Phi_k)}{P(\mathbf{y}_l \mid \Phi)} \tag{4.103}$$

$$\gamma_k = \sum_{l=1}^{N} \gamma_k^l = \sum_{l=1}^{N} \frac{c_k p_k(\mathbf{y}_l \mid \Phi_k)}{P(\mathbf{y}_l \mid \Phi)} \tag{4.104}$$

$$Q_\lambda(\Phi, \bar{\Phi}_k) = \sum_{l=1}^{N} \gamma_k^l \log p_k(\mathbf{y}_l \mid \bar{\Phi}_k) = \sum_{l=1}^{N} \frac{c_k p_k(\mathbf{y}_l \mid \Phi_k)}{P(\mathbf{y}_l \mid \Phi)} \log p_k(\mathbf{y}_l \mid \bar{\Phi}_k) \tag{4.105}$$

Now we can perform a maximum likelihood estimation on the complete data (x, y) during the M-step. By taking the derivative with respect to each parameter and setting it to zero, we obtain the following EM re-estimate of c_k, μ_k, and Σ_k:

$$\hat{c}_k = \frac{\gamma_k}{\sum_{k=1}^{K} \gamma_k} = \frac{\gamma_k}{N} \tag{4.106}$$

$$\hat{\boldsymbol{\mu}}_k = \frac{\sum_{i=1}^{N} \gamma_k^i \mathbf{y}_i}{\sum_{i=1}^{N} \gamma_k^i} = \frac{\sum_{i=1}^{N} \frac{c_k p_k(\mathbf{y}_i \mid \boldsymbol{\Phi}_k)\mathbf{y}_i}{P(\mathbf{y}_i \mid \boldsymbol{\Phi})}}{\sum_{i=1}^{N} \frac{c_k p_k(\mathbf{y}_i \mid \boldsymbol{\Phi}_k)}{P(\mathbf{y}_i \mid \boldsymbol{\Phi})}} \tag{4.107}$$

$$\hat{\boldsymbol{\Sigma}}_k = \frac{\sum_{i=1}^{N} \gamma_k^i (\mathbf{y}_i - \boldsymbol{\mu}_k)(\mathbf{y}_i - \boldsymbol{\mu}_k)^t}{\sum_{i=1}^{N} \gamma_k^i} = \frac{\sum_{i=1}^{N} \frac{c_k p_k(\mathbf{y}_i \mid \boldsymbol{\Phi}_k)(\mathbf{y}_i - \boldsymbol{\mu}_k)(\mathbf{y}_i - \boldsymbol{\mu}_k)^t}{P(\mathbf{y}_i \mid \boldsymbol{\Phi})}}{\sum_{i=1}^{N} \frac{c_k p_k(\mathbf{y}_i \mid \boldsymbol{\Phi}_k)}{P(\mathbf{y}_i \mid \boldsymbol{\Phi})}} \tag{4.108}$$

The quantity γ_k^i defined in Eq. (4.103) can be interpreted as the posterior probability that the observed data $\mathbf{y}_i$ belong to Gaussian component k ($N_k(\mathbf{y} \mid \boldsymbol{\mu}_k, \Sigma_k)$). This information as to whether the observed data $\mathbf{y}_i$ should belong to Gaussian component k is hidden and can only be observed through the hidden variable $x(c_k)$. The EM algorithm described above is used to uncover how likely the observed data $\mathbf{y}_i$ are expected to be in each Gaussian component. The re-estimation formulas are consistent with our intuition. These MLE formulas calculate the weighted contribution of each data sample according to the mixture posterior probability γ_k^i.

In fact, VQ is an approximate version of EM algorithms. A traditional VQ with the Mahalanobis distance measure is equivalent to a mixture Gaussian VQ with the following conditions:

$$c_k = 1/K \tag{4.109}$$

$$\gamma_k^i = \begin{cases} 1, & \mathbf{y}_i \in C_k \\ 0, & \text{otherwise} \end{cases} \tag{4.110}$$

The difference between VQ and the EM algorithm is that VQ performs a hard assignment of the data sample $\mathbf{y}_i$ to clusters (cells) while the EM algorithm performs a soft assignment of the data sample $\mathbf{y}_i$ to clusters. As discussed in Chapter 8, this difference carries over to the case of the Viterbi algorithm vs. the Baum-Welch algorithm in hidden Markov models.

4.5. CLASSIFICATION AND REGRESSION TREES

Classification and regression trees (CART) [15, 82] have been used in a variety of pattern recognition applications. Binary decision trees, with splitting questions attached to each

node, provide an easy representation that interprets and predicts the structure of a set of data. The application of binary decision trees is much like playing the *number-guessing* game, where the examinee tries to deduce the chosen number by asking a series of binary number-comparing questions.

Consider a simple binary decision tree for height classification. Every person's data in the study may consist of several measurements, including race, gender, weight, age, occupation, and so on. The goal of the study is to develop a classification method to assign a person one of the following five height classes: *tall* (T), *medium-tall* (t), *medium* (M), *medium-short*(s) and *short* (S). Figure 4.14 shows an example of such a binary tree structure. With this binary decision tree, one can easily predict the height class for any new person (with all the measured data, but no height information) by traversing the binary trees. Traversing the binary tree is done through answering a series of yes/no questions in the traversed nodes with the measured data. When the answer is *no*, the right branch is traversed next; otherwise, the left branch will be traversed instead. When the path ends at a leaf node, you can use its attached label as the height class for the new person. If you have the average height for each leaf node (computed by averaging the heights from those people who fall in the same leaf node during training), you can actually use the average height in the leaf node to predict the height for the new person.

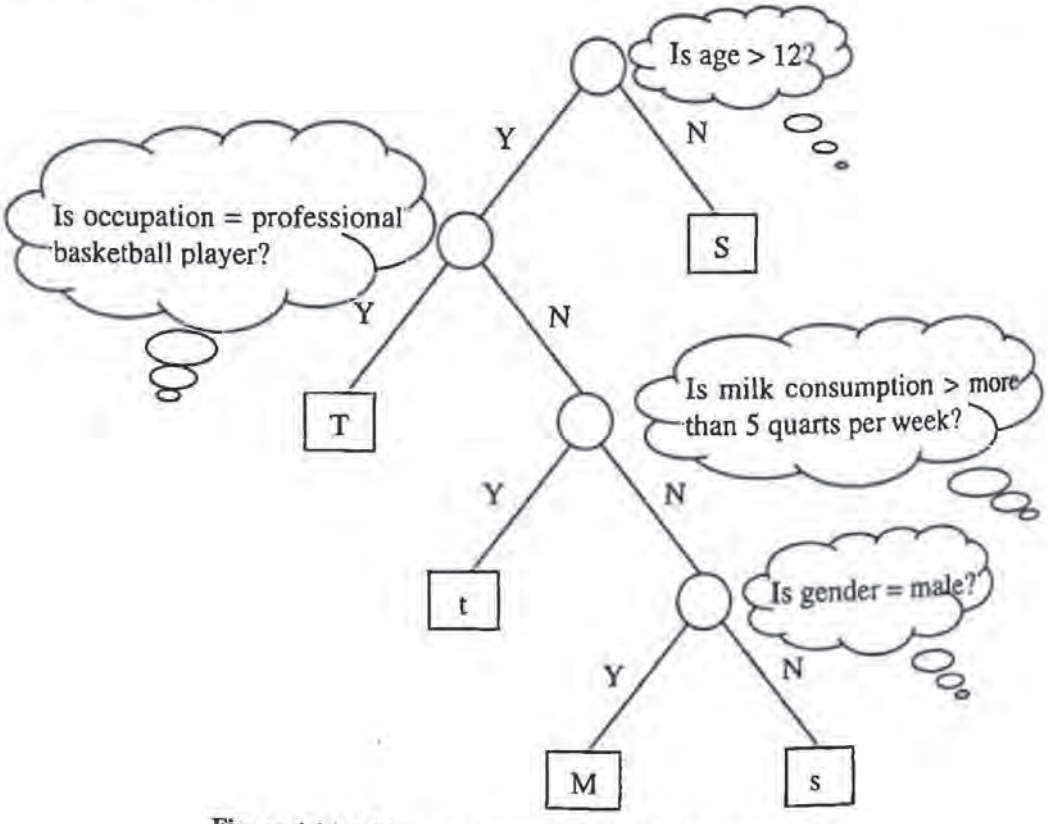


Figure 4.14 A binary tree structure for height classification.

This classification process is similar to a rule-based system where the classification is carried out by a sequence of decision rules. The choice and order of rules applied in a rule-based system is typically designed subjectively by hand through an introspective analysis based on the impressions and intuitions of a limited number of data samples. CART, on the other hand, provides an automatic and data-driven framework to construct the decision process based on objective criteria. Most statistical pattern recognition techniques are designed for data samples having a standard structure with homogeneous variables. CART is designed instead to handle data samples with high dimensionality, mixed data types, and nonstandard data structure. It has the following advantages over other pattern recognition techniques:

- CART can be applied to any data structure through appropriate formulation of the set of potential questions.
- The binary tree structure allows for compact storage, efficient classification, and easily understood interpretation of the predictive structure of the data.
- It often provides, without additional effort, not only classification and recognition, but also an estimate of the misclassification rate for each class.
- It not only handles missing data, but also is very robust to outliers and mislabeled data samples.

To construct a CART from the training samples with their classes (let's denote the set as $\mathfrak{I}$), we first need to find a set of questions regarding the measured variables; e.g., "*Is age* > *12*?", "*Is occupation = professional basketball player*?", "*Is gender = male*?" and so on. Once the question set is determined, CART uses a greedy algorithm to generate the decision trees. All training samples $\mathfrak{I}$ are placed in the root of the initial tree. The *best question* is then chosen from the question set to split the root into two nodes. Of course, we need a measurement of how well each question splits the data samples to pick the best question. The algorithm recursively splits the most promising node with the best question until the right-sized tree is obtained. We describe next how to construct the question set, how to measure each split, how to grow the tree, and how to choose the right-sized tree.

4.5.1. Choice of Question Set

Assume that the training data has the following format:

$$\mathbf{x} = (x_1, x_2, \ldots x_d) \tag{4.111}$$

where each variable x_i is a discrete or continuous data type. We can construct a *standard set* of questions Q as follows:

1. Each question is about the value of only a single variable. Questions of this type are called *simple* or *singleton* questions.

2. If x_i is a discrete variable from the set $\{c_1, c_2, \ldots, c_K\}$, Q includes all questions of the following form:

$$\{\text{Is } x_i \in S?\} \quad \text{where } S \text{ is any subset of } \{c_1, c_2, \ldots, c_K\} \tag{4.112}$$

3. If x_i is a continuous variable, Q includes all questions of the following form:

$$\{\text{Is } x_i \le c?\} \text{ for } c \in (-\infty, \infty) \tag{4.113}$$

The question subset generated from discrete variables (in condition 2 above) is clearly a finite set ($2^{K-1}-1$). On the other hand, the question subset generated from continuous variables (in condition 3 above) seems to be an infinite set based on the definition. Fortunately, since the training data samples are finite, there are only finite number of distinct splits for the training data. For a continuous variable x_i, the data points in $\mathcal{S}$ contain at most M distinct values $v_1, v_2, \ldots, v_M$. There are only at most M different splits generated by the set of questions in the form:

$$\{\text{Is } x_i \le c_n\} \quad n = 1, 2, \ldots, M \tag{4.114}$$

where $c_n = \frac{v_{n-1} + v_n}{2}$ and $v_0 = 0$. Therefore, questions related to a continuous variable also form a finite subset. The fact that Q is a finite set allows the enumerating of all possible questions in each node during tree growing.

The construction of a question set is similar to that of rules in a rule-based system. Instead of using the all-possible question set Q, some people use knowledge selectively to pick a subset of Q, which is sensitive to pattern classification. For example, the vowel subset and consonant subset are a natural choice for these sensitive questions for phoneme classification. However, the beauty of CART is the ability to use all possible questions related to the measured variables, because CART has a statistical data-driven framework to determine the decision process (as described in subsequent sections). Instead of setting some constraints on the questions (splits), most CART systems use all the possible questions for Q.

4.5.2. Splitting Criteria

A question in CART framework represents a split (partition) of data samples. All the leaf nodes (L in total) represent L disjoint subsets $A_1, A_2, \ldots, A_L$. Now we have the entire potential question set Q, the task is how to find the best question for a node split. The selection of the best question is equivalent to finding the best split for the data samples of the node.

Since each node t in the tree contains some training samples, we can compute the corresponding class probability density function $P(\omega \mid t)$. The classification process for the node can then be interpreted as a random process based on $P(\omega \mid t)$. Since our goal is classi-

fication, the objective of a decision tree is to reduce the uncertainty of the event being decided upon. We want the leaf nodes to be as pure as possible in terms of the class distribution. Let Y be the random variable of classification decision for data sample $\mathbf{X}$. We could define the weighted entropy for any node t as follows:

$$\bar{H}_t(Y) = H_t(Y)P(t) \tag{4.115}$$

$$H_t(Y) = -\sum_i P(\omega_i \mid t) \log P(\omega_i \mid t) \tag{4.116}$$

where $P(\omega_i \mid t)$ is the percentage of data samples for class i in node t; and $P(t)$ is the prior probability of visiting node t (equivalent to the ratio of number of data samples in node t and the total number of training data samples). With this weighted entropy definition, the splitting criterion is equivalent to finding the question which gives the greatest entropy reduction, where the entropy reduction for a question q to split a node t into nodes l and r can be defined as:

$$\Delta\bar{H}_t(q) = \bar{H}_t(Y) - (\bar{H}_l(Y) + \bar{H}_r(Y)) = \bar{H}_t(Y) - \bar{H}_t(Y \mid q) \tag{4.117}$$

The reduction in entropy is also the mutual information between Y and question q. The task becomes that of evaluating the entropy reduction $\Delta\bar{H}_q$ for each potential question (split), and picking the question with the greatest entropy reduction, that is,

$$q^* = \underset{q}{\operatorname{argmax}}(\Delta\bar{H}_t(q)) \tag{4.118}$$

If we define the entropy for a tree, T, as the sum of weighted entropies for all the terminal nodes, we have:

$$\bar{H}(T) = \sum_{t \text{ is terminal}} \bar{H}_t(Y) \tag{4.119}$$

It can be shown that the tree-growing (splitting) process repeatedly reduces the entropy of the tree. The resulting tree thus has a better classification power. For continuous pdf, likelihood gain is often used instead, since there is no straightforward entropy measurement [43]. Suppose one specific split divides the data into two groups, $\mathbf{X}_1$ and $\mathbf{X}_2$, which can then be used to train two Gaussian distributions $N_1(\mu_1, \Sigma_1)$ and $N_2(\mu_2, \Sigma_2)$. The log-likelihoods for generating these two data groups are:

$$L_1(\mathbf{X}_1 \mid N_1) = \log \prod_{\mathbf{x}_1} N(\mathbf{x}_1, \mu_1, \Sigma_1) = -\left(d \log 2\pi + \log|\Sigma_1| + d\right) a/2 \tag{4.120}$$

$$L_2(\mathbf{X}_2 \mid N_2) = \log \prod_{\mathbf{x}_2} N(\mathbf{x}_2, \mu_2, \Sigma_2) = -\left(d \log 2\pi + \log|\Sigma_2| + d\right) b/2 \tag{4.121}$$

where d is the dimensionality of the data; and a and b are the sample counts for the data groups $\mathbf{X}_1$ and $\mathbf{X}_2$ respectively. Now if the data $\mathbf{X}_1$ and $\mathbf{X}_2$ are merged into one group and modeled by one Gaussian $N(\mu,\Sigma)$, according to MLE, we have

$$\mu = \frac{a}{a+b}\mu_1 + \frac{b}{a+b}\mu_2 \tag{4.122}$$

$$\Sigma = \frac{a}{a+b}\left[\Sigma_1 + (\mu_1 - \mu)(\mu_1 - \mu)^t\right] + \frac{b}{a+b}\left[\Sigma_2 + (\mu_2 - \mu)(\mu_2 - \mu)^t\right] \tag{4.123}$$

Thus, the likelihood gain of splitting the data $\mathbf{X}$ into two groups $\mathbf{X}_1$ and $\mathbf{X}_2$ is:

$$\begin{aligned}\Delta\bar{L}_t(q) &= L_1(\mathbf{X}_1 \mid N) + L_2(\mathbf{X}_2 \mid N) - L_x(\mathbf{X} \mid N) \\ &= (a+b)\log|\Sigma| - a\log|\Sigma_1| - b\log|\Sigma_2|\end{aligned} \tag{4.124}$$

For regression purposes, the most popular splitting criterion is the mean squared error measure, which is consistent with the common *least squared* regression methods. For instance, suppose we need to investigate the real height as a regression function of the measured variables in the height study. Instead of finding height classification, we could simply use the average height in each node to predict the height for any data sample. Suppose Y is the actual height for training data $\mathbf{X}$, then overall regression (prediction) error for a node t can be defined as:

$$E(t) = \sum_{\mathbf{X}\in t} |Y - d(\mathbf{X})|^2 \tag{4.125}$$

where $d(\mathbf{X})$ is the regression (predictive) value of Y.

Now, instead of finding the question with greatest entropy reduction, we want to find the question with largest squared error reduction. That is, we want to pick the question q^* that maximizes:

$$\Delta E_t(q) = E(t) - (E(l) + E(r)) \tag{4.126}$$

where l and r are the leaves of node t. We define the expected square error $V(t)$ for a node t as the overall regression error divided by the total number of data samples in the node.

$$V(t) = E\left(\sum_{\mathbf{X}\in t} |Y - d(\mathbf{X})|^2\right) = \frac{1}{N(t)}\sum_{\mathbf{X}\in t} |Y - d(\mathbf{X})|^2 \tag{4.127}$$

Note that $V(t)$ is actually the variance estimate of the height, if $d(\mathbf{X})$ is made to be the average height of data samples in the node. With $V(t)$, we define the weighted squared error $\bar{V}(t)$ for a node t as follows.

$$\bar{V}(t) = V(t)P(t) = \left(\frac{1}{N(t)}\sum_{\mathbf{X}\in t} |Y - d(\mathbf{X})|^2\right)P(t) \tag{4.128}$$

Finally, the splitting criterion can be rewritten as:

$$\Delta\bar{V}_t(q) = \bar{V}(t) - (\bar{V}(l) + \bar{V}(r)) \tag{4.129}$$

Based on Eqs. (4.117) and (4.129), one can see the analogy between entropy and variance in the splitting criteria for CART. The use of entropy or variance as splitting criteria is under the assumption of uniform misclassification costs and uniform prior distributions. When nonuniform misclassification costs and prior distributions are used, some other splitting might be used for splitting criteria. Noteworthy ones are *Gini index of diversity* and *twoing rule*. Those interested in alternative splitting criteria can refer to [11, 15].

For a wide range of splitting criteria, the properties of the resulting CARTs are empirically insensitive to these choices. Instead, the criterion used to get the right-sized tree is much more important. We discuss this issue in Section 4.5.6.

4.5.3. Growing the Tree

Given the question set Q and splitting criteria $\Delta\bar{H}_t(q)$, the tree-growing algorithm starts from the initial root-only tree. At each node of tree, the algorithm searches through the variables one by one, from x_1 to x_N. For each variable, it uses the splitting criteria to find the best question (split). Then it can pick the best question out of the N best single-variable questions. The procedure can continue splitting each node until either of the following conditions is met for a node:

1. No more splits are possible; that is, all the data samples in the node belong to the same class;
2. The greatest entropy reduction of best question (split) falls below a pre-set threshold β, i.e.:

$$\max_{q \in Q} \Delta\bar{H}_t(q) < \beta \tag{4.130}$$

3. The number of data samples falling in the leaf node t is below some threshold α. This is to assure that there are enough training tokens for each leaf node if one needs to estimate some parameters associated with the node.

When a node cannot be further split, it is declared a terminal node. When all active (non-split) nodes are terminal, the tree-growing algorithm stops.

The algorithm is greedy because the question selected for any given node is the one that seems to be the best, without regard to subsequent splits and nodes. Thus, the algorithm constructs a tree that is locally optimal, but not necessarily globally optimal (but hopefully globally *good enough*). This tree-growing algorithm has been successfully applied in many applications [5, 39, 60]. A dynamic programming algorithm for determining global optimality is described in [78]; however, it is suitable only in restricted applications with relatively few variables.

4.5.4. Missing Values and Conflict Resolution

Sometimes, the available data sample $\mathbf{x}=(x_1,x_2,\ldots x_d)$ has some value x_j missing. This missing-value case can be handled by the use of *surrogate questions* (*splits*). The idea is intuitive. We define a similarity measurement between any two questions (splits) q and $\tilde{q}$ of a node t. If the best question of node t is the question q on the variable x_i, we can find the question $\tilde{q}$ that is most similar to q on a variable other than x_i. $\tilde{q}$ is our best surrogate question. Similarly, we find the second-best surrogate question, third-best and so on. The surrogate questions are considered as the backup questions in the case of missing x_i values in the data samples. The surrogate question is used in descending order to continue tree traversing for those data samples. The surrogate question gives CART unique ability to handle the case of missing data. The similarity measurement is basically a measurement reflecting the similarity of the class probability density function [15].

When choosing the best question for splitting a node, several questions on the same variable x_i may achieve the same entropy reduction and generate the same partition. As in rule-based problem solving systems, a *conflict resolution* procedure [99] is needed to decide which question to use. For example, discrete questions q_1 and q_2 have the following format:

$$q_1 \ : \ \{\text{Is } x_i \in S_1 \text{?}\} \tag{4.131}$$

$$q_2 \ : \ \{\text{Is } x_i \in S_2 \text{?}\} \tag{4.132}$$

Suppose S_1 is a subset of S_2, and one particular node contains only data samples whose x_i value contains only values in S_1, but no other. Now question q_1 or q_2 performs the same splitting pattern and therefore achieves exactly the same amount of entropy reduction. In this case, we call q_1 a sub-question of question q_2, because q_1 is a more specific version.

A *specificity ordering* conflict resolution strategy is used to favor the discrete question with fewer elements because it is more specific to the current node. In other words, if the elements of a question are a subset of the elements of another question with the same entropy reduction, the question with the subset of elements is preferred. Preferring more specific questions will prevent decision trees from over-generalizing. The specificity ordering conflict resolution can be implemented easily by presorting the set of discrete questions by the number of elements they contain in descending order, before applying them to decision trees. A similar specificity ordering conflict resolution can also be implemented for continuous-variable questions.

4.5.5. Complex Questions

One problem with allowing only simple questions is that the data may be over-fragmented, resulting in similar leaves in different locations of the tree. For example, when the best ques-

tion (rule) to split a node is actually a composite question of the form "*Is* $x_i \in S_1$?" or "*Is* $x_i \in S_2$?", a system allowing only simple questions will generate two separate questions to split the data into three clusters rather than two as shown in Figure 4.15. Also data for which the answer is *yes* are inevitably fragmented across two shaded nodes. This is inefficient and ineffective since these two very similar data clusters may now both contain insufficient training examples, which could potentially handicap future tree growing. Splitting data unnecessarily across different nodes leads to unnecessary computation, redundant clusters, reduced trainability, and less accurate entropy reduction.

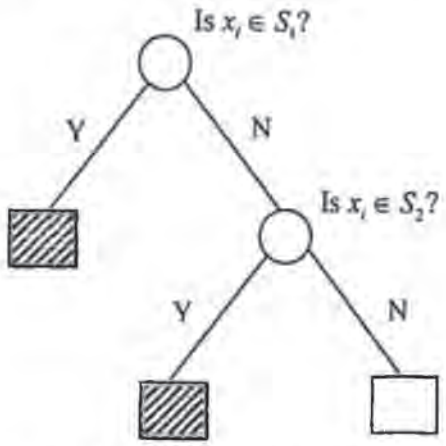


Figure 4.15 An over-split tree for the question "*Is* $x_i \in S_1$?" or "*Is* $x_i \in S_2$?"

We deal with this problem by using a composite-question construction [38, 40]. It involves conjunctive and disjunctive combinations of all questions (and their negations). A composite question is formed by first growing a tree with simple questions only and then clustering the leaves into two sets. Figure 4.16 shows the formation of one composite question. After merging, the structure is still a binary question. To construct the composite question, multiple OR operators are used to describe the composite condition leading to either one of the final clusters, and AND operators are used to describe the relation within a particular route. Finally, a Boolean reduction algorithm is used to simplify the Boolean expression of the composite question.

To speed up the process of constructing composite questions, we constrain the number of leaves or the depth of the binary tree through heuristics. The most frequently used heuristics is the limitation of the depth when searching a composite question. Since composite questions are essentially binary questions, we use the same greedy tree-growing algorithm to find the best composite question for each node and keep growing the tree until the stop criterion is met. The use of composite questions not only enables flexible clustering, but also improves entropy reduction. Growing the sub-tree a little deeper before constructing the composite question may achieve longer-range optimum, which is preferable to the local optimum achieved in the original greedy algorithm that used simple questions only.

The construction of composite questions can also be applied to continuous variables to obtained complex rectangular partitions. However, some other techniques are used to obtain

general partitions generated by hyperplanes not perpendicular to the coordinate axes. Questions typically have a linear combination of continuous variables in the following form [15]:

$$\{\text{Is } \sum_l a_l x_l \le c?\} \tag{4.133}$$

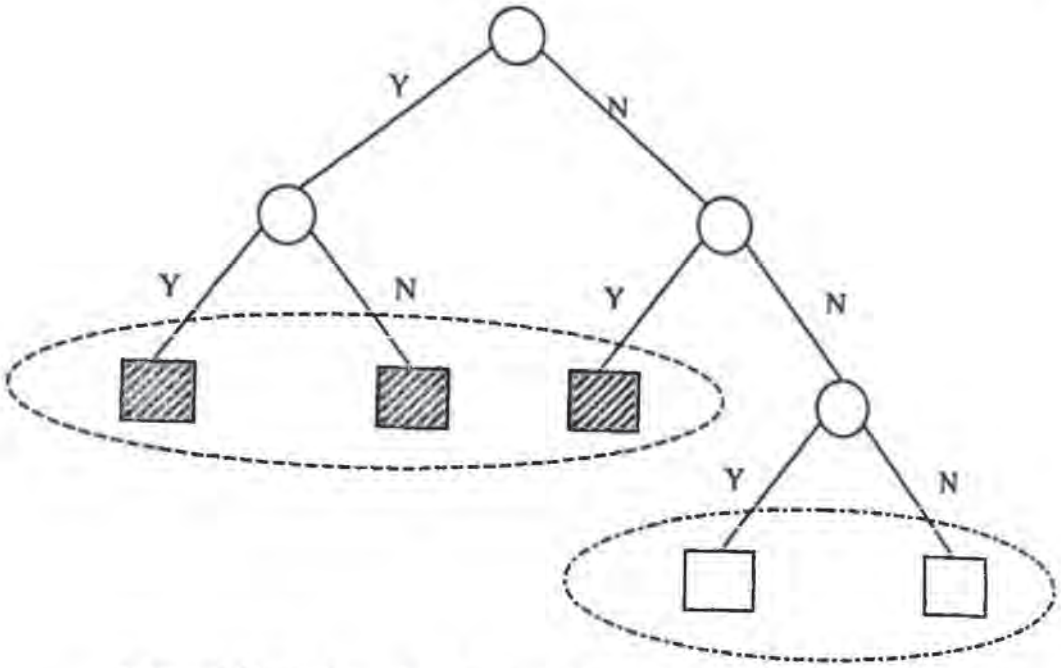


Figure 4.16 The formation of a composite question from simple questions.

4.5.6. The Right-Sized Tree

One of the most critical problems for CART is that the tree may be strictly tailored to the training data and has no generalization capability. When you split a leaf node in the tree to get entropy reduction until each leaf node contains data from one single class, that tree possesses a zero percent classification error on the training set. This is an over-optimistic estimate of the test-set misclassification rate. Independent test sample estimation or cross-validation is often used to prevent decision trees from over-modeling idiosyncrasies of the training data. To get a right-sized tree, you can minimize the misclassification rate for future independent test data.

Before we describe the solution for finding the right sized tree, let's define a couple of useful terms. Naturally we will use the plurality rule $\delta(t)$ to choose the class for a node t:

$$\delta(t) = \underset{i}{\text{argmax}}\, P(\omega_i \mid t) \tag{4.134}$$

Similar to the notation used in Bayes' decision theory, we can define the misclassification rate $R(t)$ for a node t as:

$$R(t) = r(t)P(t) \tag{4.135}$$

where $r(t) = 1 - \max_i P(\omega_i \mid t)$ and $P(t)$ is the frequency (probability) of the data falling in node t. The overall misclassification rate for the whole tree T is defined as:

$$R(T) = \sum_{t \in \tilde{T}} R(t) \tag{4.136}$$

where $\tilde{T}$ represents the set of terminal nodes. If a nonuniform misclassification cost $c(i \mid j)$, the cost of misclassifying class j data as class i data, is used, $r(t)$ is redefined as:

$$r(t) = \min_i \sum_j c(i \mid j)P(j \mid t) \tag{4.137}$$

As we mentioned, $R(T)$ can be made arbitrarily small (eventually reduced to zero) for the training data if we keep growing the tree. The key now is how we choose the tree that can minimize $R^*(T)$, which is denoted as the misclassification rate of independent test data. Almost no tree initially grown can perform well on independent test data. In fact, using more complicated stopping rules to limit the tree growing seldom works, and it is either stopped too soon at some terminal nodes, or continued too far in other parts of the tree. Instead of inventing some clever stopping criteria to stop the tree growing at the right size, we let the tree over-grow (based on rules in Section 4.5.3). We use a pruning strategy to gradually cut back the tree until the minimum $R^*(T)$ is achieved. In the next section we describe an algorithm to prune an over-grown tree, *minimum cost-complexity pruning*.

4.5.6.1. Minimum Cost-Complexity Pruning

To prune a tree, we need to find a subtree (or branch) that makes the least impact in terms of a cost measure, whether it is pruned or not. This candidate to be pruned is called the *weakest subtree*. To define such a weakest subtree, we first need to define the cost measure.

DEFINITION 1: For any sub-tree T of $T_{\max}$ ($T \prec T_{\max}$), let $|\tilde{T}|$ denote the number of terminal nodes in tree T.

DEFINITION 2: Let $\alpha \geq 0$ be a real number called the *complexity parameter*. The cost-complexity measure can be defined as:

$$R_\alpha(T) = R(T) + \alpha |\tilde{T}| \tag{4.138}$$

DEFINITION 3: For each α, define the minimal cost-complexity subtree $T(\alpha) \prec T_{\max}$ that minimizes $R_\alpha(T)$, that is,

$$T(\alpha) = \arg\min_{T \prec T_{\max}} R_\alpha(T) \tag{4.139}$$

Based on Definitions 2 and 3, if α is small, the penalty for having a large tree is small and $T(\alpha)$ will be large. In fact, $T(0) = T_{max}$ because T_{max} has a zero misclassification rate, so it will minimize $R_o(T)$. On the other hand, when α increases, $T(\alpha)$ becomes smaller and smaller. For a sufficient large α, $T(\alpha)$ may collapse into a tree with only the root. The increase of α produces a sequence of pruned trees and it is the basis of the pruning process. The pruning algorithm rests on two theorems. The first is given as follows.

THEOREM 1: For every value of α, there exists a unique minimal cost-complexity subtree $T(\alpha)$ as defined in Definition 3.[12]

To progressively prune the tree, we need to find the weakest subtree (node). The idea of a weakest subtree is the following: *if we collapse the weakest subtree into a single terminal node, the cost-complexity measure would increase least.* For any node t in the tree T, let $\{t\}$ denote the subtree containing only the node t, and T_t denote the branch starting at node t. Then we have

$$R_\alpha(T_t) = R(T_t) + \alpha |\tilde{T}_t| \tag{4.140}$$

$$R_\alpha(\{t\}) = R(t) + \alpha \tag{4.141}$$

When α is small, T_t has a smaller cost-complexity than the single-node tree $\{t\}$. However, when α increases to a point where the cost-complexity measures for T_t and $\{t\}$ are the same, it makes sense to collapse T_t into a single terminal node $\{t\}$. Therefore, we decide the critical value for α by solving the following inequality:

$$R_\alpha(T_t) \le R_\alpha(\{t\}) \tag{4.142}$$

We obtain:

$$\alpha \le \frac{R(t) - R(T_t)}{|\tilde{T}_t| - 1} \tag{4.143}$$

Based on Eq. (4.143), we define a measurement $\eta(t)$ for each node t in tree T:

$$\eta(t) = \begin{cases} \dfrac{R(t) - R(T_t)}{|\tilde{T}_t| - 1}, & t \notin \tilde{T} \\ +\infty, & t \in \tilde{T} \end{cases} \tag{4.144}$$

Based on measurement $\eta(t)$, we then define the weakest subtree T_{t_1} as the tree branch starting at the node t_1 such that

[12] You can find the proof to this in [15].

$$t_1 = \arg\min_{t \in T} \eta(t) \tag{4.145}$$

$$\alpha_1 = \eta(t_1) \tag{4.146}$$

As α increases, the node t_1 is the first node such that $R_\alpha(\{t\})$ becomes equal to $R_\alpha(T_t)$. At this point, it would make sense to prune subtree T_{t_1} (collapse T_{t_1} into a single-node subtree $\{t_1\}$), and α_1 is the value of α where the pruning occurs.

Now the tree T after pruning is referred to as T_1, i.e.,

$$T_1 = T - T_{t_1} \tag{4.147}$$

We then use the same process to find the weakest subtree T_{t_2} in T_1 and the new pruning point α_2. After pruning away T_{t_2} from T_1 to form the new pruned tree T_2, we repeat the same process to find the next weakest subtree and pruning point. If we continue the process, we get a sequence of decreasing pruned trees:

$$T \succ T_1 \succ T_2 \succ T_2 \cdots \succ \{r\} \tag{4.148}$$

where $\{r\}$ is the single-node tree containing the root of tree T with corresponding pruning points:

$$\alpha_0 < \alpha_1 < \alpha_2 < \alpha_3 < \cdots \tag{4.149}$$

where $\alpha_0 = 0$.

With the process above, the following theorem (which is basic for the minimum cost-complexity pruning) can be proved.

THEOREM 2: Let T_0 be the original tree T.

$$\text{For } k \geq 0,\ \alpha_k \leq \alpha < \alpha_{k+1},\ T(\alpha) = T(\alpha_k) = T_k \tag{4.150}$$

4.5.6.2. Independent Test Sample Estimation

The minimum cost-complexity pruning algorithm can progressively prune the over-grown tree to form a decreasing sequence of subtrees $T \succ T_1 \succ T_2 \succ T_2 \cdots \succ \{r\}$, where $T_k = T(\alpha_k)$, $\alpha_0 = 0$ and $T_0 = T$. The task now is simply to choose one of those subtrees as the optimal-sized tree. Our goal is to find the optimal-sized tree that minimizes the misclassification for

independent test set $R^*(T)$. When the training set $\Im$ is abundant, we can afford to set aside an independent test set $\Re$ from the training set. Usually $\Re$ is selected as one third of the training set $\Im$. We use the remaining two thirds of the training set $\Im - \Re$ (still abundant) to train the initial tree T and apply the minimum cost-complexity pruning algorithm to attain the decreasing sequence of subtrees $T \succ T_1 \succ T_2 \succ T_2 \cdots \succ \{r\}$. Next, the test set $\Re$ is run through the sequence of subtrees to get corresponding estimates of test-set misclassification $R^*(T), R^*(T_1), R^*(T_2), \cdots, R^*(\{r\})$. The optimal-sized tree T_{k^*} is then picked as the one with minimum test-set misclassification measure, i.e.:

$$k^* = \arg\min_k R^*(T_k) \tag{4.151}$$

The *independent test sample estimation* approach has the drawback that it reduces the effective training sample size. This is why it is used only when there is abundant training data. Under most circumstances where training data is limited, *cross-validation* is often used.

4.5.6.3. Cross-Validation

CART can be pruned via *v-fold cross-validation*. It follows the same principle of cross validation described in Section 4.2.3. First it randomly divides the training set $\Im$ into v disjoint subsets $\Im_1, \Im_2, \cdots, \Im_v$, each containing roughly the same data samples. It then defines the i^{th} training set

$$\Im^i = \Im - \Im_i \quad i = 1, 2, \ldots, v \tag{4.152}$$

so that $\Im^i$ contains the fraction $(v-1)/v$ of the original training set. v is usually chosen to be large, like 10.

In v-fold cross-validation, v auxiliary trees are grown together with the main tree T grown on $\Im$. The i^{th} tree is grown on the i^{th} training set $\Im^i$. By applying minimum cost-complexity pruning, for any given value of the cost-complexity parameter α, we can obtain the corresponding minimum cost-complexity subtrees $T(\alpha)$ and $T^i(\alpha)$, $i = 1, 2, \ldots, v$. According to Theorem 2 in Section 4.5.6.1, those minimum cost-complexity subtrees will form $v+1$ sequences of subtrees:

$$T \succ T_1 \succ T_2 \succ T_2 \cdots \succ \{r\} \text{ and} \tag{4.153}$$

$$T^i \succ T_1^i \succ T_2^i \succ T_3^i \cdots \succ \{r^i\} \quad i = 1, 2, \ldots, v \tag{4.154}$$

ALGORITHM 4.5: *THE CART ALGORITHM*

Step 1: Question Set: Create a standard set of questions Q that consists of all possible questions about the measured variables.
Step 2: Splitting Criterion: Pick a splitting criterion that can evaluate all the possible questions in any node. Usually it is either entropy-like measurement for classification trees or mean square errors for regression trees.
Step 3: Initialization: Create a tree with one (root) node, consisting of all training samples.
Step 4: Split Candidates: Find the best composite question for each terminal node:
a. Generate a tree with several simple-question splits as described in Section 4.5.3.
b. Cluster leaf nodes into two classes according to the same splitting criterion.
c. Based on the clustering done in (b), construct a corresponding composite question.
Step 5: Split: Out of all the split candidates in Step 4, split the one with best criterion.
Step 6: Stop Criterion: If all the leaf nodes containing data samples from the same class or all the potential splits generating improvement fall below a pre-set threshold β, go to Step 7; otherwise go to Step 4.
Step 7: Use *independent test sample estimate* or *cross-validation estimate* to prune the original tree into the optimal size.

The basic assumption of cross-validation is that the procedure is *stable* if v is large. That is, $T(\alpha)$ should have the same classification accuracy as $T^{i}(\alpha)$. Although we cannot directly estimate the test-set misclassification for the main tree $R^{*}(T(\alpha))$, we could approximate it via the test-set misclassification measure $R^{*}(T^{i}(\alpha))$, since each data sample in $\mathfrak{I}$ occurs in one and only one test set $\mathfrak{I}_i$. The v-fold cross-validation estimate $R^{CV}(T(\alpha))$ can be computed as:

$$R^{CV}(T(\alpha)) = \frac{1}{v}\sum_{i=1}^{v} R^{*}(T^{i}(\alpha)) \tag{4.155}$$

Similar to Eq. (4.151), once $R^{CV}(T(\alpha))$ is computed, the optimal v-fold cross-validation tree $T_{k^{CV}}$ can be found through

$$k^{CV} = \arg\min_{k} R^{CV}(T_k) \tag{4.156}$$

Cross-validation is computationally expensive in comparison with independent test sample estimation, though it makes more effective use of all training data and reveals useful information regarding the stability of the tree structure. Since the auxiliary trees are grown on a smaller training set (a fraction $v-1/v$ of the original training data), they tend to have a higher misclassification rate. Therefore, the cross-validation estimates $R^{CV}(T)$ tend to be an over-estimation of the misclassification rate. The algorithm for generating a CART tree is illustrated in Algorithm 4.5.

4.6. HISTORICAL PERSPECTIVE AND FURTHER READING

Pattern recognition is a multidisciplinary field that comprises a broad body of loosely related knowledge and techniques. Historically, there are two major approaches to pattern recognition – the statistical and the syntactical approaches. Although this chapter is focused on the statistical approach, syntactical pattern recognition techniques, which aim to address the limitations of the statistical approach in handling contextual or structural information, can be complementary to statistical approaches for spoken language processing, such as parsing. Syntactic pattern recognition is based on the analogy that complex patterns can be decomposed recursively into simpler subpatterns, much as a sentence can be decomposed into words and letters. Fu [24] provides an excellent book on syntactic pattern recognition.

The foundation of statistical pattern recognition is Bayesian theory, which can be traced back to the 18th century [9, 54] and its invention by the British mathematician Thomas Bayes (1702-1761). Chow [20] was the first to use Bayesian decision theory for pattern recognition. Statistical pattern recognition has been used successfully in a wide range of applications, from optical/handwritten recognition [13, 96], to speech recognition [7, 86] and to medical/machinery diagnosis [1, 27]. The books by Duda et al. [22] and Fukunaga [25] are two classic treatments of statistical pattern recognition. Duda et al. have a second edition of the classic pattern recognition book [23] that includes many up-to-date topics.

MLE and MAP are two most frequently used estimation methods for pattern recognition because of their simplicity and efficiency. In Chapters 8 and 9, they play an essential role in model parameter estimation. Estimating the recognition performance and comparing different recognition systems are important subjects in pattern recognition. The importance of a large number of test samples was reported in [49]. McNemar's test is dated back to the 1940s [66]. The modification of the test for continuous speech recognition systems presented in this chapter is based on an interesting paper [30] that contains a general discussion on using hypothesis-testing methods for continuous speech recognition.

Gradient descent is fundamental for most discriminant estimation methods, including MMIE, MCE, and neural networks. The history of gradient descent can be traced back to Newton's method for root finding [72, 81]. Both the book by Duda et al. [23] and the paper by Juang et al. [48] provide a good description of gradient descent. MMIE was first proposed in [16, 71] for the speech recognition problem. According to these two works, MMIE is more robust than MLE to incorrect model assumptions. MCE was first formulated by Juang et al. [48] and successfully applied to small-vocabulary speech recognition [47].

The modern era of neural networks was brought to the scientific community by McCulloch and Pitts. In the pioneering paper [64], McCulloch and Pitts laid out the mathematical treatment of the behavior of networks of simple neurons. The most important result they showed is that a network would compute any computable function. John von Neumann was influenced by this paper to use switch-delay elements derived from the McCulloch-Pitts neuron in the construction of the EDVAC (Electronic Discrete Variable Automatic Computer) that was developed based on ENIAC (Electronic Numerical Integrator and Computer) [2, 35]. The ENIAC was the famous first general-purpose electronic computer built at the

Moore School of Electrical Engineering at the University of Pennsylvania from 1943 to 1946 [31]. The two-layer perceptron work [87] by Rosenblatt, was the first to provide rigorous proofs about perceptron convergence. A 1969 book by Minsky and Papert [68] reveals that there are fundamental limits to what single-layer perceptrons can compute. It was not until the 1980s that the discovery of multi-layer perceptrons (with hidden layers and nonlinear threshold functions) and back-propagation [88] reawakened interest in neural networks. The two-volume PDP book [90, 91], *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, edited by Rummelhart and McClelland, brought the back-propagation learning method to the attention of the widest audience. Since then, various applications of neural networks in diverse domains have been developed, including speech recognition [14, 58], speech production and perception [93, 94], optical and handwriting character recognition [55, 92], visual recognition [26], game playing [97], and natural language processing [63]. There are several good textbooks for neural networks. In particular, the book by Haykin [35] provides a very comprehensive coverage of all foundations of neural networks. Bishop [12] provides a thoughtful treatment of neural networks from the perspective of pattern recognition. Short, concise tutorial papers on neural networks can be found in [44, 57].

Vector quantization originated from speech coding [17, 32, 45, 61]. The k-means algorithm was introduced by Lloyd [59]. Over the years, there have been many variations of VQ, including fuzzy VQ [10], learning VQ (LVQ) [51], and supervised VQ [18, 42]. The first published investigation toward the EM-like algorithm for incomplete data learning can be attributed to Pearson [79]. The modern EM algorithm was formalized by Dempster, Laird, and Rubin [21]. McLachlan and Krishnan [65] provide a thorough overview and history of the EM algorithm. The convergence of the EM algorithm is an interesting research topic and Wu [100] has an extensive description of the rate of convergence. The EM algorithm is the basis for all unsupervised learning that includes hidden variables. The famous HMM training algorithm, as described in Chapter 8, is based on the EM algorithm.

CART uses a very intuitive and natural principle of sequential questions and answers, which can be traced back to 1960s [70]. The popularity of CART is attributed to the book by Breiman *et al.* [15]. Quinlan proposed some interesting variants of CART, like ID3 [82] and C4.5 [84]. CART has recently been one of the most popular techniques in machine learning. Mitchell includes a good overview chapter on the latest CART techniques in his machine-learning book [69]. In addition to the strategies of node splitting and pruning mentioned in this chapter, [62] used a very interesting approach for splitting and pruning criteria based on a statistical significance testing of the data's distributions. Moreover, [28] proposed an iterative expansion pruning algorithm which is believed to perform as well as cross-validation pruning and yet is computationally cheaper [52]. CART has been successfully used in a variety of spoken language applications such as letter-to-sound conversion [46, 60], allophone model clustering [8, 38, 39], language models [5], automatic rule generation [83], duration modeling of phonemes [74, 80], and supervised vector quantization [67].

REFERENCES

[1] Albert, A. and E.K. Harris, *Multivariate Interpretation of Clinical Laboratory Data*, 1987, New York, Marcel Dekker.

[2] Aspray, W. and A. Burks, "Papers of John von Neumann on Computing and Computer Theory" in *Charles Babbage Institute Reprint Series for the History of Computing* 1986, Cambridge, MA, MIT Press.

[3] Atal, B.S. and M.R. Schroeder, "Predictive Coding of Speech Signals and Subjective Error Criteria," *IEEE Trans. on Acoustics, Speech and Signal Processing*, 1979, **ASSP-27**(3), pp. 247-254.

[4] Bahl, L.R., *et al.*, "A New Algorithm for the Estimation of Hidden Markov Model Parameters," *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1988, New York, NY, pp. 493-496.

[5] Bahl, L.R., *et al.*, "A Tree-Based Statistical Language Model for Natural Language Speech Recognition," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 1989, **37**(7), pp. 1001-1008.

[6] Bahl, L.R., *et al.*, "Estimating Hidden Markov Model Parameters so as to Maximize Speech Recognition Accuracy," *IEEE Trans. on Speech and Audio Processing*, 1993, **1**(1), pp. 77-83.

[7] Bahl, L.R., F. Jelinek, and R.L. Mercer, "A Maximum Likelihood Approach to Continuous Speech Recognition," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 1983, **5**(2), pp. 179-190.

[8] Bahl, L.R., *et al.*, "Decision Trees for Phonological Rules in Continuous Speech" in *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing* 1991, Toronto, Canada, pp. 185-188.

[9] Bayes, T., "An Essay Towards Solving a Problem in the Doctrine of Chances," *Philosophical Tansactions of the Royal Society*, 1763, **53**, pp. 370-418.

[10] Bezdek, J., *Pattern Recognition with Fuzzy Objective Function Algorithms*, 1981, New York, NY, Plenum Press.

[11] Bhargava, T.N. and V.R.R. Uppuluri, "Sampling Distribution of Gini's Index of Diversity," *Applied Mathematics and Computation*, 1977, **3**, pp. 1-24.

[12] Bishop, C.M., *Neural Networks for Pattern Recognition*, 1995, Oxford, UK, Oxford University Press.

[13] Blesser, B., *et al.*, "A Theoretical Approach for Character Recognition Based on Phenomenological Attributes," *Int. Journal of Man-Machine Studies*, 1974, 6(6), pp. 701-714.

[14] Bourlard, H. and N. Morgan, *Connectionist Speech Recognition – A Hybrid Approach*, 1994, Boston, MA, Kluwer Academic Publishers.

[15] Breiman, L., *et al.*, *Classification and Regression Trees*, 1984, Pacific Grove, CA, Wadsworth.

[16] Brown, P.F., *The Acoustic-Modeling Problem in Automatic Speech Recognition*, PhD Thesis in Computer Science Department 1987, Carnegie Mellon University, Pittsburgh, PA.

[17] Buzo, A., *et al.*, "Speech Coding Based upon Vector Quantization," *IEEE Trans. on Acoustics, Speech and Signal Processing*, 1980, **28**(5), pp. 562-574.

[18] Cerf, P.L., W. Ma, and D.V. Compernolle, "Multilayer Perceptrons as Labelers for Hidden Markov Models," *IEEE Trans. on Speech and Audio Processing*, 1994, **2**(1), pp. 185-193.

[19] Chang, P.C. and B.H. Juang, "Discriminative Training of Dynamic Programming Based Speech Recognizers," *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1992, San Fancisco.

[20] Chow, C.K., "An Optimum Character Recognition System Using Decision Functions," *IRE Trans.*, 1957, pp. 247-254.

[21] Dempster, A.P., N.M. Laird, and D.B. Rubin, "Maximum-Likelihood from Incomplete Data via the EM Algorithm," *Journal of Royal Statistical Society ser. B*, 1977, **39**, pp. 1-38.

[22] Duda, R.O. and P.E. Hart, *Pattern Classification and Scene Analysis*, 1973, New York, N.Y., John Wiley and Sons.

[23] Duda, R.O., D.G. Stork, and P.E. Hart, *Pattern Classification and Scene Analysis: Pattern Classification*, 2nd ed, 1999, John Wiley & Sons.

[24] Fu, K.S., *Syntactic Pattern Recognition and Applications*, 1982, Englewood Cliffs, NJ, Prentice Hall.

[25] Fukunaga, K., *Introduction to Statistical Pattern Recognition*, 2nd ed., 1990, Orlando, FL, Academic Press.

[26] Fukushima, K., S. Miykake, and I. Takayuki, "Neocognition: A Neural Network Model for a Mechanism of Visual Pattern Recognition," *IEEE Trans. on Systems, Man and Cybernetics*, 1983, **SMC-13**(5), pp. 826-834.

[27] Gastwirth, J.L., "The Statistical Precision of Medical Screening Procedures: Application to Polygraph and AIDS Antibodies Test Data (with Discussion)," *Statistics Science*, 1987, **2**, pp. 213-238.

[28] Gelfand, S., C. Ravishankar, and E. Delp, "An Iterative Growing and Pruning Algorithm for Classification Tree Design," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 1991, **13**(6), pp. 163-174.

[29] Gersho, A., "On the Structure of Vector Quantization," *IEEE Trans. on Information Theory*, 1982, **IT-28**, pp. 256-261.

[30] Gillick, L. and S.J. Cox, "Some Statistical Issues in the Comparison of Speech Recognition Algorithms," *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1989, Glasgow, Scotland, UK, IEEE, pp. 532-535.

[31] Goldstine, H., *The Computer from Pascal to von Neumann*, 2nd ed, 1993, Princeton, NJ, Princeton University Press.

[32] Gray, R.M., "Vector Quantization," *IEEE ASSP Magazine*, 1984, **1**(April), pp. 4-29.

[33] Gray, R.M. and E.D. Karnin, "Multiple Local Optima in Vector Quantizers," *IEEE Trans. on Information Theory*, 1982, **IT-28**, pp. 256-261.

[34] Hartigan, J.A., *Clustering Algorithm*, 1975, New York, NY, J. Wiley.

[35] Haykin, S., *Neural Networks: A Comprehensive Foundation*, 2nd ed, 1999, Upper Saddle River, NJ, Prentice-Hall.

[36] Hedelin, P., P. Knagenhjelm, and M. Skoglund, "Vector Quantization for Speech Transmission" in *Speech Coding and Synthesis*, W.B. Kleijn and K.K. Paliwal, eds., 1995, Amsterdam, pp. 311-396, Elsevier.

[37] Hinton, G.E., "Connectionist Learning Procedures," *Artificial Intelligence*, 1989, **40**, pp. 185--234.

[38] Hon, H.W., *Vocabulary-Independent Speech Recognition: The VOCIND System*, Ph.D Thesis in Department of Computer Science 1992, Carnegie Mellon University, Pittsburgh, PA.

[39] Hon, H.W. and K.F. Lee, "On Vocabulary-Independent Speech Modeling," *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1990, Albuquerque, NM, pp. 725-728.

[40] Hon, H.W. and K.F. Lee, "Vocabulary-Independent Subword Modeling and Adaptation," *IEEE Workshop on Speech Recognition*, 1991, Harriman, NY, Arden House.

[41] Huang, X.D., Y. Ariki, and M.A. Jack, *Hidden Markov Models for Speech Recognition*, 1990, Edinburgh, U.K., Edinburgh University Press.

[42] Hunt, M.J., *et al.*, "An Investigation of PLP and IMELDA Acoustic Representations and of Their Potential for Combination" in *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing* 1991, Toronto, Canada, pp. 881-884.

[43] Hwang, M.Y. and X.D. Huang, "Dynamically Configurable Acoustic Modelings for Speech Recognition," *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1998, Seattle, WA.

[44] Jain, A., J. Mao, and K.M. Mohiuddin, "Artificial Neural Networks: A Tutorial," *Computer*, 1996, **29**(3), pp. 31-44.

[45] Jayant, N.S. and P. Noll, *Digital Coding of Waveforms*, 1984, Englewood Cliffs, NJ, Prentice-Hall.

[46] Jiang, L., H.W. Hon, and X.D. Huang, "Improvements on a Trainable Letter-to-Sound Converter," *Eurospeech*, 1997, Rhodes, Greece.

[47] Juang, B.H., W. Chou, and C.H. Lee, "Statistical and Discriminative Methods for Speech Recognition" in *Automatic Speech and Speaker Recognition – Advanced Topics*, C.H. Lee, F.K. Soong, and K.K. Paliwal, eds., 1996, Boston, MA, pp. 109-132, Kluwer Academic Publishers.

[48] Juang, B.H. and S. Katagiri, "Discriminative Learning for Minimum Error Classification," *IEEE Trans. on Acoustics, Speech and Signal Processing*, 1992, **SP-40**(12), pp. 3043-3054.

[49] Kanal, L.N. and B. Chandrasekaran, "On Dimensionality and Sample Size in Statistical Pattern Classification," *Proc. of NEC*, 1968, **24**, pp. 2-7.

[50] Kanal, L.N. and N.C. Randall, "Recognition System Design by Statistical Analysis," *ACM Proc. of 19th National Conf.*, 1964, pp. D2.5-1-D2.5-10.

[51] Kohonen, T., *Learning Vector Quantization for Pattern Recognition*, 1986, Helsinki University of Technology, Finland.

[52] Kuhn, R. and R.D. Mori, "The Application of Semantic Classification Trees to Natural Language Understanding," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 1995(7), pp. 449-460.

[53] Lachenbruch, P.A. and M.R. Mickey, "Estimation of Error Rates in Discriminant Analysis," *Technometrics*, 1968, **10**, pp. 1-11.

[54] Laplace, P.S., *Theorie Analytique des Probabilities*, 1812, Paris, Courcier.

[55] Lee, D.S., S.N. Srihari, and R. Gaborski, "Bayesian and Neural Network Pattern Recognition: A Theoretical Connection and Empirical Results with Handwriting Characters" in *Artificial Neural Network and Statistical Pattern Recognition: Old and New Connections*, I.K. Sethi and A.K. Jain, eds., 1991, Amsterdam, North-Holland.

[56] Linde, Y., A. Buzo, and R.M. Gray, "An Algorithm for Vector Quantizer Design," *IEEE Trans. on Communication*, 1980, **COM-28**(1), pp. 84-95.

[57] Lippmann, R.P., "An Introduction to Computing with Neural Nets," *IEEE ASSP Magazine*, 1987, pp. 4-22.

[58] Lippmann, R.P., "Review of Neural Nets for Speech Recognition," *Neural Computation*, 1989, **1**, pp. 1-38.

[59] Lloyd, S.P., "Least Squares Quantization in PCM," *IEEE Trans. on Information Theory*, 1982, **IT-2**, pp. 129-137.

[60] Lucassen, J.M. and R.L. Mercer, "An Information-Theoretic Approach to the Automatic Determination of Phonemic Baseforms," *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1984, San Diego, California, pp. 42.5.1-42.5.4.

[61] Makhoul, J., S. Roucos, and H. Gish, "Vector Quantization in Speech Coding," *Proc. of the IEEE*, 1985, **73**(11), pp. 1551-1588.

[62] Martin, J.K., "An Exact Probability Metric for Decision Tree Splitting and Stopping," *Articial Intelligence and Statistics*, 1995, **5**, pp. 379-385.

[63] McClelland, J., "The Programmable Blackboard Model" in *Parallel Distributed Processing – Explorations in the Microstructure of Cognition, Volume II: Psychological and Biological Models* 1986, Cambridge, MA, MIT Press.

[64] McCulloch, W.S. and W. Pitts, "A Logical Calculus of Ideas Immanent in Nervous Activity," *Bulletin of Mathematical Biophysics*, 1943.

[65] McLachlan, G. and T. Krishnan, *The EM Algorithm and Extensions*, 1996, New York, NY, Wiley Interscience.

[66] McNemar, E.L., "Note on the Sampling Error of the Difference Between Correlated Proportions or Percentages," *Psychometrika*, 1947, **12**, pp. 153-157.

[67] Meisel, W.S., *et al.*, "The SSI Large-Vocabulary Speaker-Independent Continuous Speech Recognition System," 1991, pp. 337-340.

[68] Minsky, M. and S. Papert, *Perceptrons*, 1969, Cambridge, MA, MIT Press.

[69] Mitchell, T., *Machine Learning*, McGraw-Hill Series in Computer Science, 1997, McGraw-Hill.

[70] Morgan, J.N. and J.A. Sonquist, "Problems in the Analysis of Survey Data and a Proposal," *Journal of American Statistics Association*, 1962, **58**, pp. 415-434.

[71] Nadas, A., "A Decision-Theoretic Formulation of a Training Problem in Speech Recognition and a Comparison of Training by Unconditional Versus Conditional Maximum Likelihood," *IEEE Trans. on Acoustics, Speech and Signal Processing*, 1983, **4**, pp. 814-817.

[72] Newton, I., *Philosophiae Naturalis Principlea Mathematics*, 1687, London, Royal Society Press.

[73] Neyman, J. and E.S. Pearson, "On the Problem of the Most Efficient Tests of Statistical Hypotheses," *Philosophical Trans. of Royal Society*, 1928, **231**, pp. 289-337.

[74] Ostendorf, M. and N. Velleux, "A Hierarchical Stochastic Model for Automatic Prediction of Prosodic Boundary Location," *Computational Linguistics*, 1995, **20**(1), pp. 27-54.

[75] Paliwal, K. and W.B. Kleijn, "Quantization of LPC Parameters" in *Speech Coding and Synthesis*, W.B. Kleijn and K.K. Paliwal, eds., 1995, Amsterdam, pp. 433-466, Elsevier.

[76] Paul, D.B., "An 800 bps Adaptive Vector Quantization in Speech Coding," *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1983, pp. 73-76.

[77] Paul, D.B., "The Lincoln Tied-Mixture HMM Continuous Speech Recognizer" in *Morgan Kaufmann Publishers* 1990, San Mateo, CA, pp. 332-336.

[78] Payne, H.J. and W.S. Meisel, "An Algorithm for Constructing Optimal Binary Decision Trees," *IEEE Trans. on Computers*, 1977, **C-26**(September), pp. 905-916.

[79] Pearson, K., "Contributions to The Mathematical Theorem of Evolution," *Philosophical Trans. of Royal Society*, 1894, **158A**, pp. 71-110.

[80] Pitrelli, J. and V. Zue, "A Hierarchical Model for Phoneme Duration in American English" in *Proc. of Eurospeech*, 1989.

[81] Press, W.H., *et al.*, *Numerical Recipes in C*, 1988, Cambridge, UK, Cambridge University Press.

[82] Quinlan, J.R., "Introduction of Decision Trees" in *Machine Learning: An Artifical Intelligence Approach*, R. Michalski, J. Carbonell, and T. Mitchell, eds., 1986, Boston, M.A., pp. 1-86, Kluwer Academic Publishers.

[83] Quinlan, J.R., "Generating Production Rules from Decision Trees," *Int. Joint Conf. on Artificial Intelligence*, 1987, pp. 304-307.

[84] Quinlan, J.R., *C4.5: Programs for Machine Learning*, 1993, San Francisco, Morgan Kaufmann.

[85] Rabiner, L. and B.H. Juang, "Speech Recognition System Design and Implementation Issues" in *Fundamentals of Speech Recognition*, L. Rabiner and B.H. Juang, eds., 1993, Englewood Cliffs, NJ, Prentice Hall, pp. 242-340.

[86] Reddy, D.R., "Speech Recognition by Machine: A Review," *IEEE Proc.*, 1976, **64**(4), pp. 502-531.

[87] Rosenblatt, F., "The Perceptron – A Probabilistic Model for Information Storage and Organization in the Brain," *Psychological Review*, 1958. **65**, pp. 386-408.

[88] Rummelhart, D.E., G.E. Hinton, and R.J. Williams, "Learning Internal Representations by Error Propagation" in *Parallel Distributed Processing*, D.E. Rumelhart and J.L. McClelland, eds., 1986, Cambridge, MA, MIT Press, pp. 318-362.

[89] Rummelhart, D.E., G.E. Hinton, and R.J. Williams, "Learning Representations by Back-Propagating Errors," *Nature*, 1986, **323**, pp. 533-536.

[90] Rummelhart, D.E. and J.L. McClelland, *Parallel Distributed Processing – Explorations in the Microstructure of Cognition, Volume I: Foundations*, 1986, Cambridge, MA, MIT Press.

[91] Rummelhart, D.E. and J.L. McClelland, *Parallel Distributed Processing – Explorations in the Microstructure of Cognition, Volume II: Psychological and Biological Models*, 1986, Cambridge, MA, MIT Press.

[92] Schalkoff, R.J., *Digital Image Processing and Computer Vision*, 1989, New York, NY, John Wiley & Sons.

[93] Sejnowski, T.J. and C.R. Rosenberg, "Parallel Networks that Learn to Pronounce English Text," *Complex Systems*, 1987, **1**, pp. 145-168.

[94] Sejnowski, T.J., *et al.*, "Combining Visual and Acoustic Speech Signals with a Neural Network Improve Intelligibility" in *Advances in Neural Information Processing Systems* 1990, San Mateo, CA, Morgan Kaufmann, pp. 232-239.

[95] Sparkes, J.J., "Pattern Recognition and a Model of the Brain," *Int. Journal of Man-Machine Studies*, 1969, **1**(3), pp. 263-278.

[96] Tappert, C., C.Y. Suen, and T. Wakahara, "The State of the Art in On-Line Handwriting Recognition," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 1990, **12**(8), pp. 787-808.

[97] Tesauro, G. and T.J. Sejnowski, "A Neural Network That Learns to Play Backgammon," *Neural Information Processing Systems, American Institute of Physics*, 1988, pp. 794-803.

[98] White, H., "Learning in Artificial Neural Networks: A Statistical Perspective," *Neural Computation*, 1989, **1**(4), pp. 425-464.

[99] Winston, P.H., *Artificial Intelligence*, 1984, Reading, MA, Addison-Wesley.

[100] Wu, C.F.J., "On the Convergence Properties of the EM Algorithm," *The Annals of Statistics*, 1983, **11**(1), pp. 95-103.

[101] Young, J.Z., *Programmes of the Brain*, 1975, Oxford, England, Oxford University Press.

PART II

SPEECH PROCESSING

CHAPTER 5

Digital Signal Processing

One of the most popular ways of characterizing speech is in terms of a *signal* or acoustic waveform. Shown in Figure 5.1 is a representation of the speech signal that ensures that the information content can be easily extracted by human listeners or computers. This is why digital signal processing plays a fundamental role for spoken language processing. We describe here the fundamentals of digital signal processing: digital signals and systems, frequency-domain transforms for both continuous and discrete frequencies, digital filters, the relationship between analog and digital signals, filterbanks, and stochastic processes. In this chapter we set the mathematical foundations of frequency analysis that allow us to develop specific techniques for speech signals in Chapter 6.

The main theme of this chapter is the development of frequency-domain methods computed through the Fourier transform. When we boost the bass knob in our amplifier we are increasing the gain at low frequencies, and when we boost the treble knob we are increasing the gain at high frequencies. Representation of speech signals in the frequency domain is especially useful because the frequency structure of a phoneme is generally unique.

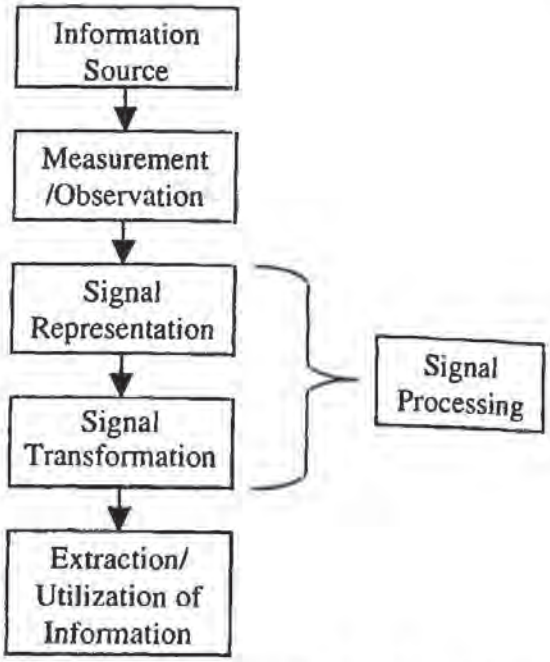


Figure 5.1 Signal processing is both a representation and a transformation that allows a useful information extraction from a source. The representation and transformation are based on a model of the signal, often parametric, that is convenient for subsequent processing.

5.1. DIGITAL SIGNALS AND SYSTEMS

To process speech signals, it is convenient to represent them mathematically as functions of a continuous variable t, which represents time. Let us define an *analog signal* $x_a(t)$ as a function varying continuously in time. If we sample the signal x with a sampling period T (i.e., $t = nT$), we can define a discrete-time signal as $x[n] = x_a(nT)$, also known as *digital signal.*[1] In this book we use parentheses to describe an analog signal and brackets for digital signals. Furthermore we can define the sampling frequency F_s as $F_s = 1/T$, the inverse of the sampling period T. For example, for a sampling rate $F_s = 8\,\text{kHz}$, its corresponding sampling period is 125 microseconds. In Section 5.5 it is shown that, under some circumstances, the analog signal $x_a(t)$ can be recovered exactly from the digital signal $x[n]$. Figure 5.2 shows an analog signal and its corresponding digital signal. In subsequent figures, for convenience, we will sometimes plot digital signals as continuous functions.

The term *Digital Signal Processing* (DSP) refers to methods for manipulating the sequence of numbers $x[n]$ in a digital computer. The acronym DSP is also used to refer to a *Digital Signal Processor*, i.e., a microprocessor specialized to perform DSP operations.

[1] Actually the term digital signal is defined as a discrete-time signal whose values are represented by integers within a range, whereas a general discrete-time signal would be represented by real numbers. Since the term digital signal is much more commonly used, we will use that term, except when the distinction between them is necessary.

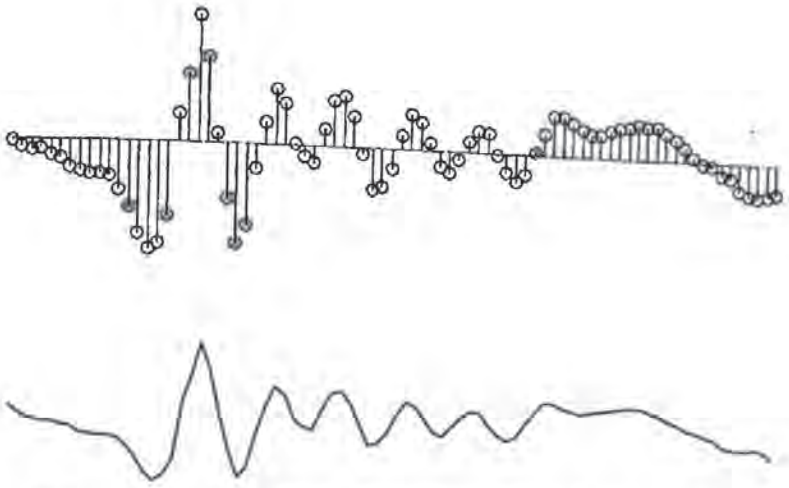


Figure 5.2 Analog signal and its corresponding digital signal.

We start with sinusoidal signals and show they are the fundamental signals for linear systems. We then introduce the concept of convolution and linear time-invariant systems. Other digital signals and nonlinear systems are also introduced.

5.1.1. Sinusoidal Signals

One of the most important signals is the sine wave or *sinusoid*

$$x_0[n] = A_0 \cos(\omega_0 n + \phi_0) \tag{5.1}$$

where A_0 is the sinusoid's amplitude, ω_0 the angular frequency, and ϕ_0 the phase. The angle in the trigonometric functions is expressed in radians, so that the angular frequency ω_0 is related to the normalized linear frequency f_0 by the relation $\omega_0 = 2\pi f_0$, and $0 \le f_0 \le 1$. This signal is *periodic*[2] with period $T_0 = 1/f_0$. In Figure 5.3 we can see an example of a sinusoid with frequency $f_0 = 0.04$, or a period of $T_0 = 25$ samples.

Sinusoids are important because speech signals can be decomposed as sums of sinusoids. When we boost the bass knob in our amplifier we are increasing the gain for sinusoids of low frequencies, and when we boost the treble knob we are increasing the gain for sinusoids of high frequencies.

[2] A signal $x[n]$ is periodic with period N if and only if $x[n]=x[n+N]$, which requires $\omega_0 = 2\pi / N$. This means that the digital signal in Eq. (5.1) is not periodic for all values of ω_0, even though its continuous signal counterpart $x(t) = A_0 \cos(\omega_0 t + \phi_0)$ is periodic for all values of ω_0 (see Section 5.5).

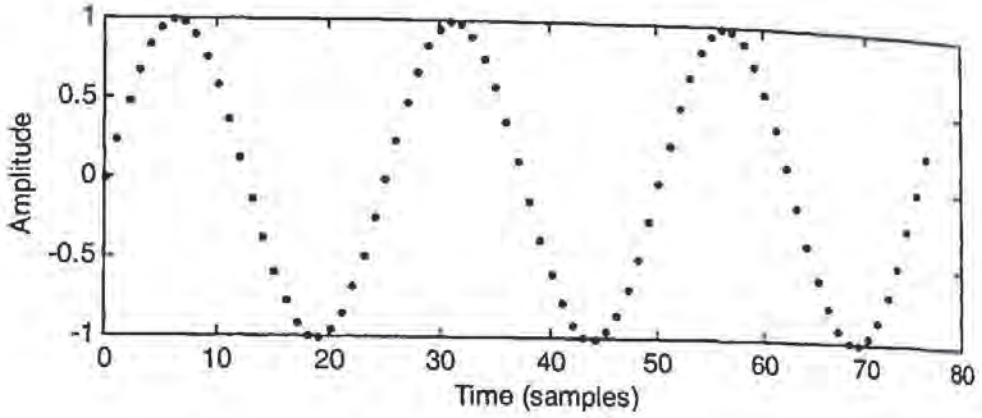


Figure 5.3 A digital sinusoid with a period of 25 samples.

What is the sum of two sinusoids $x_0[n]$ and $x_1[n]$ of the same frequency ω_0 but different amplitudes A_0 and A_1, and phases ϕ_0 and ϕ_1? The answer is another sinusoid of the same frequency but a different amplitude A and phase ϕ. While this can be computed through trigonometric identities, it is somewhat tedious and not very intuitive. For this reason we introduce another representation based on complex numbers, which proves to be very useful when we study digital filters.

A complex number x can be expressed as $z = x+jy$, where $j=\sqrt{-1}$, x is the real part and y is the imaginary part, with both x and y being real numbers. Using Euler's relation, given a real number ϕ, we have

$$e^{j\phi} = \cos\phi + j\sin\phi \tag{5.2}$$

so that the complex number z can also be expressed in polar form as $z = Ae^{j\phi}$, where A is the amplitude and ϕ is the phase. Both representations can be seen in Figure 5.4, where the real part is shown in the abscissa (x-axis) and the imaginary part in the ordinate (y-axis).

Using complex numbers, the sinusoid in Eq. (5.1) can be expressed as the real part of the corresponding complex exponential

$$x_0[n] = A_0\cos(\omega_0 n+\phi_0) = \mathrm{Re}\{A_0 e^{j(\omega_0 n+\phi_0)}\} \tag{5.3}$$

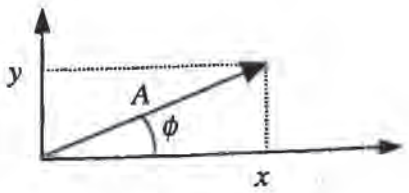


Figure 5.4 Complex number representation in Cartesian form $z=x+jy$ and polar form $z=Ae^{j\phi}$. Thus $x=A\cos\phi$ and $y=A\sin\phi$.

and thus the sum of two complex exponential signals equals

$$A_0 e^{j(\omega_0 n+\phi_0)} + A_1 e^{j(\omega_0 n+\phi_1)} = e^{j\omega_0 n}\left(A_0 e^{j\phi_0} + A_1 e^{j\phi_1}\right) = e^{j\omega_0 n} A e^{j\phi} = A e^{j(\omega_0 n+\phi)} \tag{5.4}$$

Taking the real part in both sides results in

$$A_0 \cos(\omega_0 n+\phi_0) + A_1 \cos(\omega_0 n+\phi_1) = A\cos(\omega_0 n+\phi) \tag{5.5}$$

or in other words, the sum of two sinusoids of the same frequency is another sinusoid of the same frequency.

To compute A and ϕ, dividing Eq. (5.4) by $e^{j\omega_0 n}$ leads to a relationship between the amplitude A and phase ϕ:

$$A_0 e^{j\phi_0} + A_1 e^{j\phi_1} = A e^{j\phi} \tag{5.6}$$

Equating real and imaginary parts in Eq. (5.6) and dividing them we obtain:

$$\tan\phi = \frac{A_0 \sin\phi_0 + A_1 \sin\phi_1}{A_0 \cos\phi_0 + A_1 \cos\phi_1} \tag{5.7}$$

and adding the squared of real and imaginary parts and using trigonometric identities[3]

$$A^2 = A_0^2 + A_1^2 + 2A_0 A_1 \cos(\phi_0 - \phi_1) \tag{5.8}$$

This complex representation of Figure 5.5 lets us analyze and visualize the amplitudes and phases of sinusoids of the same frequency as vectors. The sum of N sinusoids of the same frequency is another sinusoid of the same frequency that can be obtained by adding the real and imaginary parts of all complex vectors. In Section 5.2.1 we show that the output of a linear time-invariant system to a sinusoid is another sinusoid of the same frequency.

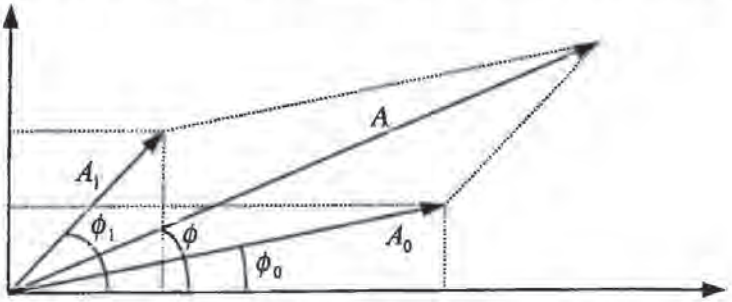


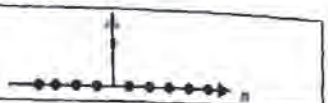
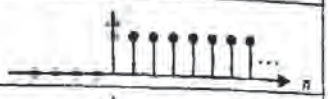
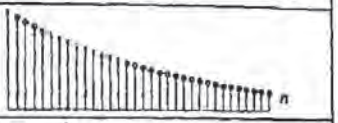
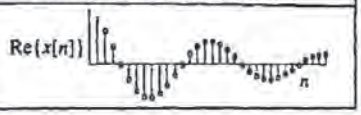
Figure 5.5 Geometric representation of the sum of two sinusoids of the same frequency. It follows the complex number representation in Cartesian form of Figure 5.4.

[3] $\sin^2\phi + \cos^2\phi = 1$ and $\cos(a-b) = \cos a \cos b + \sin a \sin b$.

5.1.2. Other Digital Signals

In the field of digital signal processing there are other signals that repeatedly arise and that are shown in Table 5.1.

Table 5.1 Some useful digital signals: the Kronecker delta, unit step, rectangular signal, real exponential ($a<1$) and real part of a complex exponential ($r<1$).

Kronecker delta, or unit impulse	$\delta[n]=\begin{cases}1 & n=0\\ 0 & otherwise\end{cases}$	
Unit step	$u[n]=\begin{cases}1 & n\geq 0\\ 0 & n<0\end{cases}$	
Rectangular signal	$\mathrm{rect}_N[n]=\begin{cases}1 & 0\leq n<N\\ 0 & otherwise\end{cases}$	
Real exponential	$x[n]=a^n u[n]$	
Complex exponential	$x[n]=a^n u[n]=r^n e^{jn\omega_0}u[n]$ $=r^n(\cos n\omega_0+j\sin n\omega_0)u[n]$	

If $r=1$ and $\omega_0\neq 0$ we have a complex sinusoid as shown in Section 5.1.1. If $\omega_0=0$ we have a real exponential signal, and if $r<1$ and $\omega_0\neq 0$ we have an exponentially decaying oscillatory sequence, also known as a damped sinusoid.

5.1.3. Digital Systems

A digital system is a system that, given an input signal $x[n]$, generates an output signal $y[n]$:

$$y[n]=T\{x[n]\} \tag{5.9}$$

whose input/output relationship can be seen in Figure 5.6.

In general, a digital system T is defined to be linear *iff* (if and only if)

$$T\{a_1x_1[n]+a_2x_2[n]\}=a_1T\{x_1[n]\}+a_2T\{x_2[n]\} \tag{5.10}$$

for any values of a_1, a_2 and any signals $x_1[n]$ and $x_2[n]$.

Here, we study systems according to whether or not they are linear and/or time invariant.

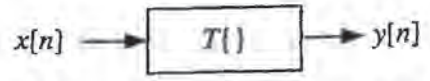


Figure 5.6 Block diagram of a digital system whose input is digital signal $x[n]$, and whose output is digital signal $y[n]$.

5.1.3.1. Linear Time-Invariant Systems

A system is *time-invariant* if given Eq. (5.9), then

$$y[n-n_0]=T\{x[n-n_0]\} \tag{5.11}$$

Linear digital systems of a special type, the so-called *linear time-invariant* (LTI),[4] are described by

$$y[n]=\sum_{k=-\infty}^{\infty} x[k]h[n-k]=x[n]*h[n] \tag{5.12}$$

where $*$ is defined as the *convolution* operator. It is left to the reader to show that the linear system in Eq. (5.12) indeed satisfies Eq. (5.11).

LTI systems are completely characterized by the signal $h[n]$, which is known as the system's *impulse response* because it is the output of the system when the input is an impulse $x[n]=\delta[n]$. Most of the systems described in this book are LTI systems.

Table 5.2 Properties of the convolution operator.

Commutative	$x[n]*h[n]=h[n]*x[n]$
Associative	$x[n]*(h_1[n]*h_2[n])=(x[n]*h_1[n])*h_2[n]=x[n]*h_1[n]*h_2[n]$
Distributive	$x[n]*(h_1[n]+h_2[n])=x[n]*h_1[n]+x[n]*h_2[n]$

The convolution operator is commutative, associative and distributive as shown in Table 5.2 and Figure 5.7.

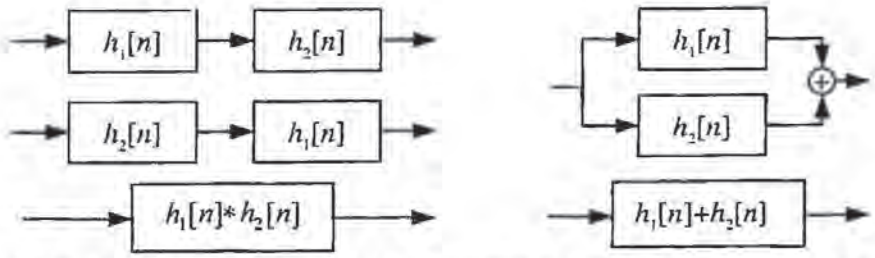


Figure 5.7 The block diagrams on the left, representing the commutative property, are equivalent. The block diagrams on the right, representing the distributive property, are also equivalent.

[4] Actually the term linear time-invariant (LTI) systems is typically reserved for continuous or analog systems, and linear shift-invariant system is used for discrete-time signals, but we will use LTI for discrete-time signals too since it is widely used in this context.

5.1.3.2. Linear Time-Varying Systems

An interesting type of digital systems is that whose output is a linear combination of the input signal at different times:

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]g[n,n-k] \tag{5.13}$$

The digital system in Eq. (5.13) is linear, since it satisfies Eq. (5.10). The Linear Time-Invariant systems of Section 5.1.3.1 are a special case of Eq. (5.13) when $g[n,n-k] = h[n-k]$. The systems in Eq. (5.13) are called *linear time-varying* (LTV) systems, because the weighting coefficients can vary with time.

A useful example of such system is the so-called *amplitude modulator*

$$y[n] = x[n]\cos\omega_0 n \tag{5.14}$$

used in AM transmissions. As we show in Chapter 6, speech signals are the output of LTV systems. Since these systems are difficult to analyze, we often approximate them with linear time-invariant systems.

5.1.3.3. Nonlinear Systems

Many *nonlinear* systems do not satisfy Eq. (5.10). Table 5.3 includes a list of typical nonlinear systems used in speech processing. All these nonlinear systems are memoryless, because the output at time n depends only on the input at time n, except for the *median smoother* of order $(2N + 1)$ whose output depends also on the previous and the following N samples.

5.2. CONTINUOUS-FREQUENCY TRANSFORMS

A very useful transform for LTI systems is the Fourier transform, because it uses complex exponentials as its basis functions, and its generalization: the z-transform. In this section we cover both transforms, which are continuous functions of frequency, and their properties.

5.2.1. The Fourier Transform

It is instructive to see what the output of a LTI system with impulse response $h[n]$ is when the input is a complex exponential. Substituting $x[n] = e^{j\omega_0 n}$ in Eq. (5.12) and using the commutative property of the convolution we obtain

$$y[n] = \sum_{k=-\infty}^{\infty} h[k]e^{j\omega_0(n-k)} = e^{j\omega_0 n}\sum_{k=-\infty}^{\infty} h[k]e^{-j\omega_0 k} = e^{j\omega_0 n}H(e^{j\omega_0}) \tag{5.15}$$

Table 5.3 Examples of nonlinear systems for speech processing. All of them are memoryless except for the median smoother.

Nonlinear System	Equation
Median Smoother of order $(2N+1)$	$y[n] = \text{median}\{x[n-N],\cdots,x[n],\cdots,x[n+N]\}$
Full-Wave Rectifier	$y[n] = \lvert x[n]\rvert$
Half-Wave Rectifier	$y[n] = \begin{cases} x[n] & x[n] \geq 0 \\ 0 & x[n] < 0 \end{cases}$
Frequency Modulator	$y[n] = A\cos\left(\omega_0 + \Delta\omega x[n]\right)n$
Hard-Limiter	$y[n] = \begin{cases} A & x[n] \geq A \\ x[n] & \lvert x[n]\rvert < A \\ -A & x[n] \leq -A \end{cases}$
Uniform Quantizer (L-bit) with $2N = 2^L$ intervals of width Δ	$y[n] = \begin{cases} (N-1/2)\Delta & x[n] \geq (N-1)\Delta \\ (m+1/2)\Delta & m\Delta \leq x[n] < (m+1)\Delta \quad 0 \leq m < N-1 \\ (-m+1/2)\Delta & -m\Delta \leq x[n] < -(m-1)\Delta \quad 0 < m < N-1 \\ (-N+1/2)\Delta & x[n] < -(N-1)\Delta \end{cases}$

which is another complex exponential of the same frequency and amplitude multiplied by the complex quantity $H(e^{j\omega_0})$ given by

$$H(e^{j\omega}) = \sum_{n=-\infty}^{\infty} h[n]e^{-j\omega n} \tag{5.16}$$

Since the output of a LTI system to a complex exponential is another complex exponential, it is said that complex exponentials are *eigensignals* of LTI systems, with the complex quantity $H(e^{j\omega_0})$ being their *eigenvalue*.

The quantity $H(e^{j\omega})$ is defined as the *discrete-time Fourier transform* of $h[n]$. It is clear from Eq. (5.16) that $H(e^{j\omega})$ is a periodic function of ω with period 2π, and therefore we need to keep only one period to fully describe it, typically $-\pi < \omega < \pi$ (Figure 5.8).

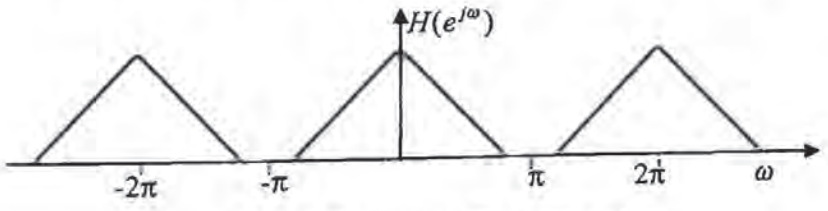


Figure 5.8 $H(e^{j\omega})$ is a periodic function of ω.

$H(e^{j\omega})$ is a complex function of ω which can be expressed in terms of the real and imaginary parts:

$$H(e^{j\omega}) = H_r(e^{j\omega}) + jH_i(e^{j\omega}) \tag{5.17}$$

or in terms of the magnitude and phase as

$$H(e^{j\omega}) = \left|H(e^{j\omega})\right| e^{j\arg\{H(e^{j\omega})\}} \tag{5.18}$$

Thus if the input to the LTI system is a sinusoid as in Eq. (5.1), the output will be

$$y_0[n] = A_0 \left|H(e^{j\omega_0})\right| \cos\left(\omega_0 n + \phi_0 + \arg\{H(e^{j\omega_0})\}\right) \tag{5.19}$$

according to Eq. (5.15). Therefore if $\left|H(e^{j\omega_0})\right| > 1$, the LTI system will amplify that frequency, and likewise it will attenuate, or *filter* it, if $\left|H(e^{j\omega_0})\right| < 1$. That is one reason why these systems are also called filters. The Fourier transform $H(e^{j\omega})$ of a filter $h[n]$ is called the system's *frequency response* or *transfer function.*

The angular frequency ω is related to the normalized linear frequency f by the simple relation $\omega = 2\pi f$. We show in Section 5.5 that linear frequency f_l and normalized frequency f are related by $f_l = fF_s$, where F_s is the sampling frequency.

The inverse *discrete-time Fourier transform* is defined as

$$h[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi} H(e^{j\omega}) e^{j\omega n} d\omega \tag{5.20}$$

The Fourier transform is invertible, and Eq. (5.16) and (5.20) are transform pairs:

$$h[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi} H(e^{j\omega}) e^{j\omega n} d\omega = \frac{1}{2\pi}\int_{-\pi}^{\pi}\left(\sum_{m=-\infty}^{\infty} h[m] e^{-j\omega m}\right) e^{j\omega n} d\omega \tag{5.21}$$

$$= \sum_{m=-\infty}^{\infty} h[m] \frac{1}{2\pi}\int_{-\pi}^{\pi} e^{j\omega(n-m)} d\omega = \sum_{m=-\infty}^{\infty} h[m]\delta[n-m] = h[n]$$

since

$$\frac{1}{2\pi}\int_{-\pi}^{\pi} e^{j\omega(n-m)} d\omega = \delta[n-m] \tag{5.22}$$

A sufficient condition for the existence of the Fourier transform is

$$\sum_{n=-\infty}^{\infty} \left|h[n]\right| < \infty \tag{5.23}$$

Although we have computed the Fourier transform of the impulse response of a filter $h[n]$, Eq. (5.16) and (5.20) can be applied to any signal $x[n]$.

5.2.2. Z-Transform

The *z-transform* is a generalization of the Fourier transform. The z-transform of a digital signal $h[n]$ is defined as

$$H(z) = \sum_{n=-\infty}^{\infty} h[n] z^{-n} \tag{5.24}$$

where z is a complex variable. Indeed, the Fourier transform of $h[n]$ equals its z-transform evaluated at $z = e^{j\omega}$. While the Fourier and z-transforms are often used interchangeably, we normally use the Fourier transform to plot the filter's frequency response, and the z-transform to analyze more general filter characteristics, given its polynomial functional form. We can also use the z-transform for unstable filters, which do not have Fourier transforms.

Since Eq. (5.24) is an infinite sum, it is not guaranteed to exist. A sufficient condition for convergence is:

$$\sum_{n=-\infty}^{\infty} |h[n]| |z|^{-n} < \infty \tag{5.25}$$

which is true only for a *region of convergence* (ROC) in the complex z-plane $R_1 < |z| < R_2$ as indicated in Figure 5.9.

For a signal $h[n]$ to have a Fourier transform, its z-transform $H(z)$ has to include the unit circle, $|z| = 1$, in its convergence region. Therefore, a sufficient condition for the existence of the Fourier transform is given in Eq. (5.23) by applying Eq. (5.25) to the unit circle.

An LTI system is defined to be *causal* if its impulse response is a causal signal, i.e. $h[n] = 0$ for $n < 0$. Similarly, a LTI system is *anti-causal* if $h[n] = 0$ for $n > 0$. While all physical systems are causal, noncausal systems are still useful since causal systems could be decomposed into causal and anti-causal systems.

A system is defined to be *stable* if for every bounded input it produces a bounded output. A necessary and sufficient condition for an LTI system to be stable is

$$\sum_{n=-\infty}^{\infty} |h[n]| < \infty \tag{5.26}$$

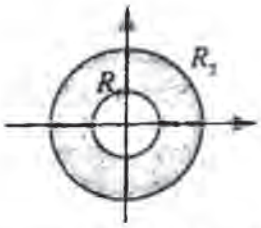


Figure 5.9 Region of convergence of the z-transform in the complex plane.

which means, according to Eq. (5.23), that $h[n]$ has a Fourier transform, and therefore that its z-transform includes the unit circle in its region of convergence.

Just as in the case of Fourier transforms, we can use the z-transform for any signal, not just for a filter's impulse response.

The *inverse z-transform* is defined as

$$h[n] = \frac{1}{2\pi j}\oint H(z)z^{n-1}dz \tag{5.27}$$

where the integral is performed along a closed contour that is within the *region of convergence*. Eqs. (5.24) and (5.27) plus knowledge of the region of convergence form a transform pair: *i.e.* one can be exactly determined if the other is known. If the integral is performed along the unit circle (i.e., doing the substitution $z = e^{j\omega}$) we obtain Eq. (5.20), the inverse Fourier transform.

5.2.3. Z-Transforms of Elementary Functions

In this section we compute the z-transforms of the signals defined in Table 5.1. The z-transforms of such signals are summarized in Table 5.4. In particular we compute the z-transforms of left-sided and right-sided complex exponentials, which are essential to compute the inverse z-transform of rational polynomials. As we see in Chapter 6, speech signals are often modeled as having z-transforms that are rational polynomials.

Table 5.4 Z-transforms of some useful signals together with their region of convergence.

Signal	Z-Transform	Region of Convergence
$h_1[n] = \delta[n-N]$	$H_1(z) = z^{-N}$	$z \neq 0$
$h_2[n] = u[n]-u[n-N]$	$H_2(z) = \frac{1-z^{-N}}{1-z^{-1}}$	$z \neq 0$
$h_3[n] = a^n u[n]$	$H_3(z) = \frac{1}{1-az^{-1}}$	$\lvert a\rvert < \lvert z\rvert$
$h_4[n] = -a^n u[-n-1]$	$H_4(z) = \frac{1}{1-az^{-1}}$	$\lvert z\rvert < \lvert a\rvert$

5.2.3.1. Right-Sided Complex Exponentials

A right-sided complex exponential sequence

$$h_3[n] = a^n u[n] \tag{5.28}$$

has a z-transform given by

$$H_3(z)=\sum_{n=0}^{\infty}a^n z^{-n}=\lim_{N\to\infty}\frac{1-(az^{-1})^N}{1-az^{-1}}=\frac{1}{1-az^{-1}}\quad\text{for}\quad |a|<|z| \tag{5.29}$$

by using the sum of the terms of a geometric sequence and making $N\to\infty$. This region of convergence ($|a|<|z|$) is typical of causal signals (those that are zero for $n<0$).

When a z-transform is expressed as the ratio of two polynomials, the roots of the numerator are called *zeros*, and the roots of the denominator are called *poles*. Zeros are the values of z for which the z-transform equals 0, and poles are the values of z for which the z-transform equals infinity.

$H_3(z)$ has a pole at $z=a$, because its value goes to infinity at $z=a$. According to Eq. (5.26), $h_3[n]$ is a stable signal if and only if $|a|<1$, or in other words, if its pole is inside the unit circle. In general, a causal and stable system has all its poles inside the unit circle. As a corollary, a system which has poles outside the unit circle is either noncausal or unstable or both. This is a very important fact, which we exploit throughout the book.

5.2.3.2. Left-Sided Complex Exponentials

A left-sided complex exponential sequence

$$h_4[n]=-a^n u[-n-1] \tag{5.30}$$

has a z-transform given by

$$\begin{aligned}H_4(z)&=-\sum_{n=-\infty}^{-1}a^n z^{-n}=-\sum_{n=1}^{\infty}a^{-n}z^n=1-\sum_{n=0}^{\infty}a^{-n}z^n \\ &=1-\frac{1}{1-a^{-1}z}=\frac{-a^{-1}z}{1-a^{-1}z}=\frac{1}{1-az^{-1}}\end{aligned}\qquad\text{for } |z|<|a| \tag{5.31}$$

This region of convergence ($|z|<|a|$) is typical of noncausal signals (those that are nonzero for $n<0$). Observe that $H_3(z)$ and $H_4(z)$ are functionally identical and only differ in the region of convergence. In general, the region of convergence of a signal that is nonzero for $-\infty<n<\infty$ is $R_1<|z|<R_2$.

5.2.3.3. Inverse Z-Transform of Rational Functions

Integrals in the complex plane such as Eq. (5.27) are not easy to do, but fortunately they are not necessary for the special case of $H(z)$ being a rational polynomial transform. In this case, partial fraction expansion can be used to decompose the signal into a linear combination of signals like $h_1[n]$, $h_3[n]$ and $h_4[n]$ as in Table 5.4.

For example,

$$H_5(z)=\frac{2+8z^{-1}}{2-5z^{-1}-3z^{-2}} \tag{5.32}$$

has as roots of its denominator $z=3,-1/2$. Therefore it can be decomposed as

$$H_5(z)=\frac{A}{1-3z^{-1}}+\frac{B}{1+(1/2)z^{-1}}=\frac{(2A+2B)+(A-6B)z^{-1}}{2-5z^{-1}-3z^{-2}} \tag{5.33}$$

so that A and B are the solution of the following set of linear equations:

$$\begin{aligned} 2A+2B&=2\\ A-6B&=8 \end{aligned} \tag{5.34}$$

whose solution is $A=2$ and $B=-1$, and thus Eq. (5.33) is expressed as

$$H_5(z)=2\left(\frac{1}{1-3z^{-1}}\right)-\left(\frac{1}{1+(1/2)z^{-1}}\right) \tag{5.35}$$

However, we cannot compute the inverse z-transform unless we know the region of convergence. If, for example, we are told that the region of convergence includes the unit circle (necessary for the system to be stable), then the inverse transform of

$$H_4(z)=\frac{1}{1-3z^{-1}} \tag{5.36}$$

must have a region of convergence of $|z|<3$ according to Table 5.4, and thus be a left-sided complex exponential:

$$h_4[n]=-3^n u[-n-1] \tag{5.37}$$

and the transform of

$$H_3(z)=\frac{1}{1+(1/2)z^{-1}} \tag{5.38}$$

must have a region of convergence of $1/2<|z|$ according to Table 5.4, and thus be a right-sided complex exponential:

$$h_3[n]=(-1/2)^n u[n] \tag{5.39}$$

so that

$$h_5[n]=-2\cdot 3^n u[-n-1]-(-1/2)^n u[n] \tag{5.40}$$

While we only showed an example here, the method used generalizes to rational transfer functions with more poles and zeros.

5.2.4. Properties of the Z- and Fourier Transforms

In this section we include a number of properties that are used throughout the book and that can be derived from the definition of Fourier and z-transforms (see Table 5.5). Of special interest are the convolution property and Parseval's theorem, which are described below.

5.2.4.1. The Convolution Property

The z-transform of $y[n]$, convolution of $x[n]$ and $h[n]$, can be expressed as a function of their z-transforms:

$$\begin{aligned} Y(z) &= \sum_{n=-\infty}^{\infty} y[n]z^{-n} = \sum_{n=-\infty}^{\infty}\left(\sum_{k=-\infty}^{\infty} x[k]h[n-k]\right)z^{-n} \\ &= \sum_{k=-\infty}^{\infty} x[k]\left(\sum_{n=-\infty}^{\infty} h[n-k]z^{-n}\right) = \sum_{k=-\infty}^{\infty} x[k]\left(\sum_{n=-\infty}^{\infty} h[n]z^{-(n+k)}\right) \\ &= \sum_{k=-\infty}^{\infty} x[k]z^{-k}H(z) = X(z)H(z) \end{aligned} \tag{5.41}$$

which is the fundamental property of LTI systems: "The z-transform of the convolution of two signals is the product of their z-transforms." This is also known as the convolution property. The ROC of $Y(z)$ is now the intersection of the ROCs of $X(z)$ and $H(z)$ and cannot be empty for $Y(z)$ to exist.

Likewise, we can obtain a similar expression for the Fourier transforms:

$$Y(e^{j\omega}) = X(e^{j\omega})H(e^{j\omega}) \tag{5.42}$$

A dual version of the convolution property can be proven for the product of digital signals:

$$x[n]y[n] \leftrightarrow \frac{1}{2\pi} X(e^{j\omega}) * Y(e^{j\omega}) \tag{5.43}$$

whose transform is the continuous convolution of the transforms with a scale factor. The convolution of functions of continuous variables is defined as

$$y(t) = x(t) * h(t) = \int_{-\infty}^{\infty} x(\tau)h(t-\tau)d\tau \tag{5.44}$$

Note how this differs from the discrete convolution of Eq. (5.12).

5.2.4.2. Power Spectrum and Parseval's Theorem

Let's define the *autocorrelation* of signal $x[n]$ as

$$R_{xx}[n] = \sum_{m=-\infty}^{\infty} x[m+n]x^*[m] = \sum_{l=-\infty}^{\infty} x[l]x^*[-(n-l)] = x[n] * x^*[-n] \tag{5.45}$$

where the superscript asterisk (*) means complex conjugate[5] and should not be confused with the convolution operator.

Using the fundamental property of LTI systems in Eq. (5.42) and the symmetry properties in Table 5.5, we can express its Fourier transform $S_{xx}(\omega)$ as

$$S_{xx}(\omega) = X(\omega)X^*(\omega) = |X(\omega)|^2 \tag{5.46}$$

which is the *power spectrum*. The Fourier transform of the autocorrelation is the power spectrum:

$$R_{xx}[n] \leftrightarrow S_{xx}(\omega) \tag{5.47}$$

or alternatively

$$R_{xx}[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi} S_{xx}(\omega)e^{j\omega n}d\omega \tag{5.48}$$

If we set $n = 0$ in Eq. (5.48) and use Eq. (5.45) and (5.46), we obtain

$$\sum_{n=-\infty}^{\infty} |x[n]|^2 = \frac{1}{2\pi}\int_{-\pi}^{\pi} |X(\omega)|^2 d\omega \tag{5.49}$$

which is called *Parseval's theorem* and says that we can compute the signal's energy in the time domain or in the frequency domain.

5.3. DISCRETE-FREQUENCY TRANSFORMS

Here we describe transforms, including the DFT, DCT and FFT, that take our discrete-time signal into a discrete frequency representation. Discrete-frequency transforms are the natural transform for periodic signals, though we show in Section 5.7 and Chapter 6 how they are also useful for aperiodic signals such as speech.

[5] If $z = x + jy = Ae^{j\phi}$, its complex conjugate is defined as $z^* = x - jy = Ae^{-j\phi}$.

Table 5.5 Properties of the Fourier and z-transforms.

Property	Signal	Fourier Transform	z-Transform
Linearity	$ax_1[n]+bx_2[n]$	$aX_1(e^{j\omega})+bX_2(e^{j\omega})$	$aX_1(z)+bX_2(z)$
Symmetry	$x[-n]$	$X(e^{-j\omega})$	$X(z^{-1})$
	$x^*[n]$	$X^*(e^{-j\omega})$	$X^*(z^*)$
	$x^*[-n]$	$X^*(e^{j\omega})$	$X^*(1/z^*)$
	$x[n]$ real	$X(e^{j\omega})$ is *Hermitian* $X(e^{-j\omega})=X^*(e^{j\omega})$ $\lvert X(e^{j\omega})\rvert$ is even[6] $\mathrm{Re}\{X(e^{j\omega})\}$ is even $\arg\{X(e^{j\omega})\}$ is odd[7] $\mathrm{Im}\{X(e^{j\omega})\}$ is odd	$X(z^*)=X^*(z)$
	$\mathrm{Even}\{x[n]\}$	$\mathrm{Re}\{X(e^{j\omega})\}$	
	$\mathrm{Odd}\{x[n]\}$	$j\,\mathrm{Im}\{X(e^{j\omega})\}$	
Time-shifting	$x[n-n_0]$	$X(e^{j\omega})e^{-j\omega n_0}$	$X(z)z^{-n_0}$
Modulation	$x[n]e^{j\omega_0 n}$	$X(e^{j(\omega-\omega_0)})$	$X(e^{-j\omega_0}z)$
	$x[n]z_0^n$		$X(z/z_0)$
Convolution	$x[n]*h[n]$	$X(e^{j\omega})H(e^{j\omega})$	$X(z)H(z)$
	$x[n]y[n]$	$\frac{1}{2\pi}X(e^{j\omega})*Y(e^{j\omega})$	
Parseval's Theorem	$R_{xx}[n]=\sum_{m=-\infty}^{\infty}x[m+n]x^*[m]$	$S_{xx}(\omega)=\lvert X(\omega)\rvert^2$	$X(z)X^*(1/z^*)$

A discrete transform of a signal $x[n]$ is another signal defined as

$$X[k]=\mathrm{T}\{x[n]\} \tag{5.50}$$

Linear transforms are special transforms that decompose the input signal $x[n]$ into a linear combination of other signals:

$$x[n]=\sum_{k=-\infty}^{\infty}X[k]\varphi_k[n] \tag{5.51}$$

[6] A function $f(x)$ is called even if and only if $f(x)=f(-x)$.

[7] A function $f(x)$ is called odd if and only if $f(x)=-f(-x)$.

where $\varphi_k[n]$ is a set of *orthonormal* functions

$$< \varphi_k[n], \varphi_l[n] >= \delta[k-l] \tag{5.52}$$

with the inner product defined as

$$< \varphi_k[n], \varphi_l[n] >= \sum_{n=-\infty}^{\infty} \varphi_k[n] \varphi_l^*[n] \tag{5.53}$$

With this definition, the coefficients $X[k]$ are the projection of $x[n]$ onto $\varphi_k[n]$:

$$X[k] =< x[n], \varphi_k[n] > \tag{5.54}$$

as illustrated in Figure 5.10.

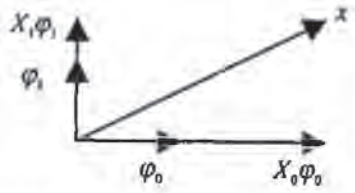


Figure 5.10 Orthonormal expansion of a signal $x[n]$ in a two-dimensional space.

5.3.1. The Discrete Fourier Transform (DFT)

If a $x_N[n]$ signal is periodic with period N then

$$x_N[n] = x_N[n+N] \tag{5.55}$$

and the signal is uniquely represented by N consecutive samples. Unfortunately, since Eq. (5.23) is not met, we cannot guarantee the existence of its Fourier transform. The *Discrete Fourier Transform* (DFT) of a periodic signal $x_N[n]$ is defined as

$$X_N[k] = \sum_{n=0}^{N-1} x_N[n] e^{-j2\pi nk/N} \qquad 0 \le k < N \tag{5.56}$$

$$x_N[n] = \frac{1}{N} \sum_{k=0}^{N-1} X_N[k] e^{j2\pi nk/N} \qquad 0 \le n < N \tag{5.57}$$

which are transform pairs. Equation (5.57) is also referred as a *Fourier series* expansion.

In Figure 5.11 we see the approximation of a periodic square signal with period $N = 100$ as a sum of 19 *harmonic* sinusoids, i.e., we used only the first 19 $X_N[k]$ coefficients in Eq. (5.57).

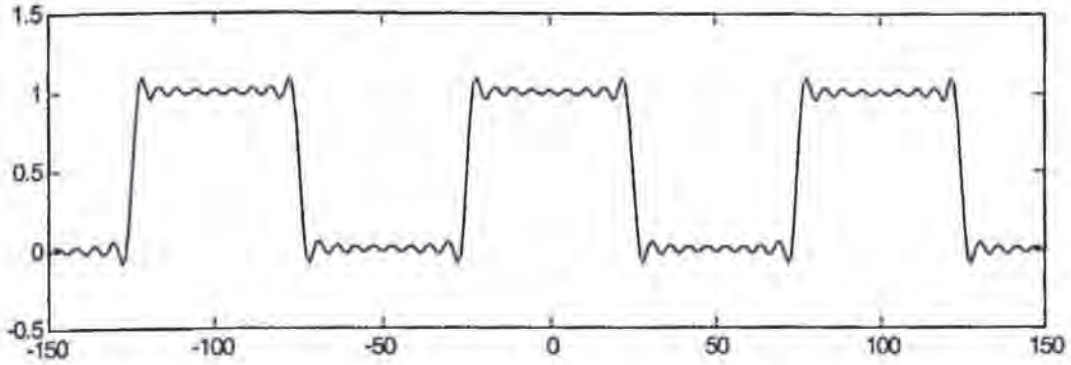


Figure 5.11 Decomposition of a periodic square signal with period 100 samples as a sum of 19 harmonic sinusoids with frequencies $\omega_k = 2\pi k/100$.

$$\tilde{x}_N[n] = \frac{1}{N}\sum_{k=-18}^{18} X_N[k]e^{j2\pi nk/N} = \frac{X_N[0]}{N} + \frac{2}{N}\sum_{k=1}^{18} X_N[k]\cos(2\pi nk/N) \tag{5.58}$$

Had we used 100 harmonic sinusoids, the periodic signal would have been reproduced *exactly*. Nonetheless, retaining a smaller number of sinusoids can provide a decent approximation for a periodic signal.

5.3.2. Fourier Transforms of Periodic Signals

Using the DFT, we now discuss how to compute the Fourier transforms of a complex exponential, an impulse train, and a general periodic signal, since they are signals often used in DSP. We also present a relationship between the continuous-frequency Fourier transform and the discrete Fourier transform.

5.3.2.1. The Complex Exponential

One of the simplest periodic functions is the complex exponential $x[n] = e^{j\omega_0 n}$. Since it has infinite energy, we cannot compute its Fourier transform in its strict sense. Since such signals are so useful, we devise an alternate formulation.

First, let us define the function

$$d_\Delta(\omega) = \begin{cases} 1/\Delta & 0 \le \omega < \Delta \\ 0 & otherwise \end{cases} \tag{5.59}$$

which has the following property

$$\int_{-\infty}^{\infty} d_\Delta(\omega)\,d\omega = 1 \tag{5.60}$$

for all values of $\Delta > 0$.

It is useful to define the continuous delta function $\delta(\omega)$, also known as the *Dirac* delta, as

$$\delta(\omega) = \lim_{\Delta \to 0} d_\Delta(\omega) \tag{5.61}$$

which is a *singular function* and can be seen in Figure 5.12. The Dirac delta is a function of a continuous variable and should not be confused with the Kronecker delta, which is a function of a discrete variable.

Using Eqs. (5.59) and (5.61) we can then see that

$$\int_{-\infty}^{\infty} X(\omega)\delta(\omega)d\omega = \lim_{\Delta \to 0} \int_{-\infty}^{\infty} X(\omega) d_\Delta(\omega) d\omega = X(0) \tag{5.62}$$

and similarly

$$\int_{-\infty}^{\infty} X(\omega)\delta(\omega - \omega_0)d\omega = X(\omega_0) \tag{5.63}$$

so that

$$X(\omega)\delta(\omega - \omega_0) = X(\omega_0)\delta(\omega - \omega_0) \tag{5.64}$$

because the integrals on both sides are identical.

Using Eq. (5.63), we see that the convolution of $X(\omega)$ and $\delta(\omega - \omega_0)$ is

$$X(\omega) * \delta(\omega - \omega_0) = \int_{-\infty}^{\infty} X(u)\delta(\omega - \omega_0 - u)du = X(\omega - \omega_0) \tag{5.65}$$

For the case of a complex exponential, inserting $X(\omega) = e^{j\omega n}$ into Eq. (5.63) results in

$$\int_{-\infty}^{\infty} \delta(\omega - \omega_0)e^{j\omega n} d\omega = e^{j\omega_0 n} \tag{5.66}$$

By comparing Eq. (5.66) with (5.20) we can then obtain

$$e^{j\omega_0 n} \leftrightarrow 2\pi\delta(\omega - \omega_0) \tag{5.67}$$

so that the Fourier transform of a complex exponential is an impulse concentrated at frequency ω_0.



Figure 5.12 Representation of the $\delta(\omega)$ function and its approximation $d_\Delta(\omega)$.

5.3.2.2. The Impulse Train

Since the impulse train

$$p_N[n] = \sum_{k=-\infty}^{\infty} \delta[n-kN] \tag{5.68}$$

is periodic with period N, it can be expanded in Fourier series according to (5.56) as

$$P_N[k] = 1 \tag{5.69}$$

so that using the inverse Fourier series Eq. (5.57), $p_N[n]$ can alternatively be expressed as

$$p_N[n] = \frac{1}{N}\sum_{k=0}^{N-1} e^{j2\pi kn/N} \tag{5.70}$$

which is an alternate expression to Eq. (5.68) as a sum of complex exponentials. Taking the Fourier transform of Eq. (5.70) and using Eq. (5.67) we obtain

$$P_N(e^{j\omega}) = \frac{2\pi}{N}\sum_{k=0}^{N-1} \delta(\omega - 2\pi k/N) \tag{5.71}$$

which is another impulse train in the frequency domain (See Figure 5.13). The impulse train in the time domain is given in terms of the Kronecker delta, and the impulse train in the frequency domain is given in terms of the Dirac delta.



Figure 5.13 An impulse train signal and its Fourier transform, which is also an impulse train.

5.3.2.3. General Periodic Signals

We now compute the Fourier transform of a general periodic signal using the results of Section 5.3.2.2 and show that, in addition to being periodic, the transform is also discrete. Given a periodic signal $x_N[n]$ with period N, we define another signal $x[n]$:

$$x[n] = \begin{cases} x_N[n] & 0 \le n < N \\ 0 & otherwise \end{cases} \tag{5.72}$$

so that

$$x_N[n] = \sum_{k=-\infty}^{\infty} x[n-kN] = x[n] * \sum_{k=-\infty}^{\infty} \delta[n-kN] = x[n] * p_N[n] \tag{5.73}$$

which is the convolution of $x[n]$ with an impulse train $p_N[n]$ as in Eq. (5.68). Since $x[n]$ is of finite length, it has a Fourier transform $X(e^{j\omega})$. Using the convolution property $X_N(e^{j\omega}) = X(e^{j\omega})P_N(e^{j\omega})$, where $P_N(e^{j\omega})$ is the Fourier transform of $p_N[n]$ as given by Eq. (5.71), we obtain another impulse train:

$$X_N(e^{j\omega}) = \frac{2\pi}{N}\sum_{k=-\infty}^{\infty} X(e^{j2\pi k/N})\delta(\omega - 2\pi k/N) \tag{5.74}$$

Therefore the Fourier transform $X_N(e^{j\omega})$ of a periodic signal $x_N[n]$ can be expressed in terms of samples $\omega_k = 2\pi k/N$, spaced $2\pi/N$ apart, of the Fourier transform $X(e^{j\omega})$ of $x[n]$, one period of the signal $x_N[n]$. The relationships between $x[n]$, $x_N[n]$, $X(e^{j\omega})$ and $X_N(e^{j\omega})$ are shown in Figure 5.14.

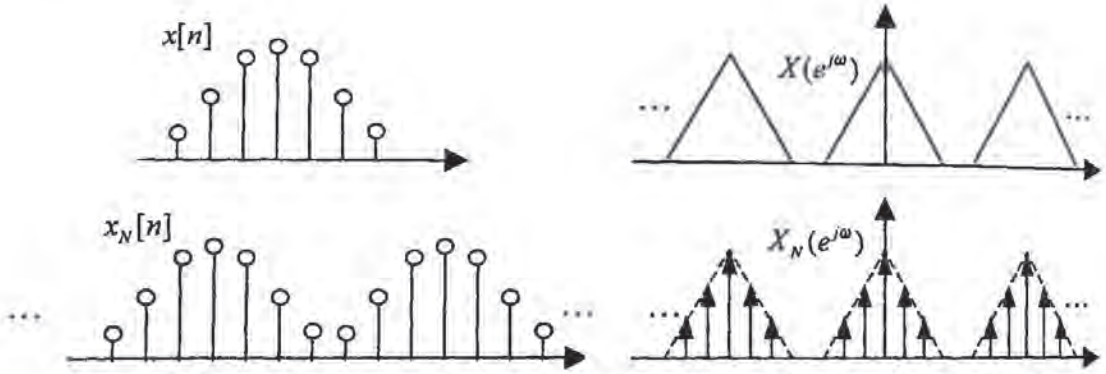


Figure 5.14 Relationships between finite and periodic signals and their Fourier transforms. On one hand, $x[n]$ is a length N discrete signal whose transform $X(e^{j\omega})$ is continuous and periodic with period 2π. On the other hand, $x_N[n]$ is a periodic signal with period N whose transform $X_N(e^{j\omega})$ is discrete and periodic.

5.3.3. The Fast Fourier Transform (FFT)

There is a family of fast algorithms to compute the DFT, which are called Fast Fourier Transforms (FFT). Direct computation of the DFT from Eq. (5.56) requires N^2 operations, assuming that the trigonometric functions have been pre-computed. The FFT algorithm only requires on the order of $N\log_2 N$ operations, so it is widely used for speech processing.

5.3.3.1. Radix-2 FFT

Let's express the discrete Fourier transform of $x[n]$

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-j2\pi nk/N} = \sum_{n=0}^{N-1} x[n]W_N^{nk} \qquad 0 \le k < N \tag{5.75}$$

where we have defined for convenience

$$W_N = e^{-j2\pi/N} \tag{5.76}$$

Equation (5.75) requires N^2 complex multiplies and adds. Now, let's suppose N is even, and let $f[n] = x[2n]$ represent the even-indexed samples of $x[n]$, and $g[n] = x[2n+1]$ the odd-indexed samples. We can express Eq. (5.75) as

$$X[k] = \sum_{n=0}^{N/2-1} f[n] W_{N/2}^{nk} + W_N^k \sum_{n=0}^{N/2-1} g[n] W_{N/2}^{nk} = F[k] + W_N^k G[k] \tag{5.77}$$

where $F[k]$ and $G[k]$ are the $N/2$ point DFTs of $f[n]$ and $g[n]$, respectively. Since both $F[k]$ and $G[k]$ are defined for $0 \le k < N/2$, we need to also evaluate them for $N/2 \le k < N$, which is straightforward, since

$$F[k + N/2] = F[k] \tag{5.78}$$

$$G[k + N/2] = G[k] \tag{5.79}$$

If $N/2$ is also even, then both $f[n]$ and $g[n]$ can be decomposed into sequences of even and odd indexed samples and therefore its DFT can be computed using the same process. Furthermore, if N is an integer power of 2, this process can be iterated and it can be shown that the number of multiplies and adds is $N \log_2 N$, which is a significant saving from N^2. This is the *decimation-in-time* algorithm and can be seen in Figure 5.15. A dual algorithm called *decimation-in-frequency* can be derived by decomposing the signal into its first $N/2$ and its last $N/2$ samples.

5.3.3.2. Other FFT Algorithms

Although the radix-2 FFT is the best known algorithm, there are other variants that are faster and are more often used in practice. Among those are the radix-4, radix-8, split-radix and prime-factor algorithm.

The same process used in the derivation of the radix-2 decimation-in-time algorithm applies if we decompose the sequences into four sequences: $f_1[n] = x[4n]$, $f_2[n] = x[4n+1]$, $f_3[n] = x[4n+2]$, and $f_4[n] = x[4n+3]$. This is the radix-4 algorithm, which can be applied when N is a power of 4, and is generally faster than an equivalent radix-2 algorithm.

Similarly there are radix-8 and radix-16 algorithms for N being powers of 8 and 16 respectively, which use fewer multiplies and adds. But because of possible additional control logic, it is not obvious that they will be faster, and every algorithm needs to be optimized for a given processor.

There are values of N, such as $N = 128$, for which we cannot use radix-4, radix-8 nor radix-16, so we have to use the less efficient radix-2. A combination of radix-2 and radix-4, called *split-radix* [5], has been shown to have fewer multiplies than both radix-2 and radix-4, and can be applied to N being a power of 2.

Finally, another possible decomposition is $N = p_1 p_2 \cdots p_L$ with p_l being prime numbers. This leads to the *prime-factor algorithm* [2]. While this family of algorithms offers a similar number of operations as the algorithms above, it offers more flexibility in the choice of N.

5.3.3.3. FFT Subroutines

Typically, FFT subroutines are computed *in-place* to save memory and have the form `fft (float *xr, float *xi, int n)` where `xr` and `xi` are the real and imaginary parts respectively of the input sequence, before calling the subroutine, and the real and imaginary parts of the output transform, after returning from it. C code that implements a decimation-in-time radix-2 FFT of Figure 5.15 is shown in Figure 5.16.

The first part of the subroutine in Figure 5.16 is doing the so-called *butterflies*, which use the trigonometric factors, also called *twiddle factors*. Normally, those twiddle factors are pre-computed and stored in a table. The second part of the subroutine deals with the fact that the output samples are not linearly ordered (see Figure 5.15); in fact, the indexing has the bits reversed, which is why we need to do *bit reversal*, also called *descrambling*.

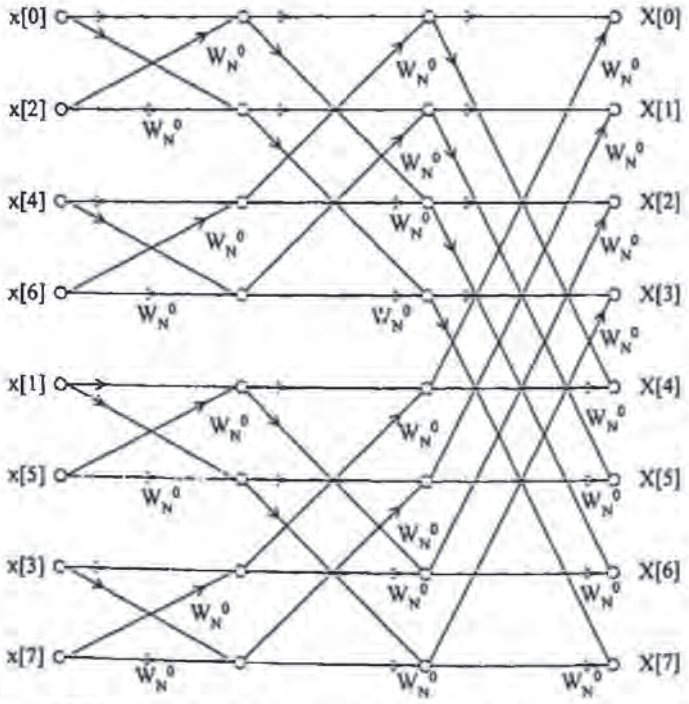


Figure 5.15 Decimation in time radix-2 algorithm for an 8-point FFT.