TR-H-240

Discriminative Feature Extraction Applied to Speech Recognition.

Alain BIEM

1998.3.16

ATR人間情報通信研究所
〒619-0288　京都府相楽郡精華町光台2-2　TEL: 0774-95-1011

ATR Human Information Processing Research Laboratories
2-2, Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-0288, Japan
Telephone: +81-774-95-1011
Fax : +81-774-95-1008

DISCRIMINATIVE FEATURE EXTRACTION APPLIED TO SPEECH RECOGNITION

By

Alain Ernest Biem

List of Symbols

$\boldsymbol{\Omega}_s$	Signal space (prior to feature extraction).
$\boldsymbol{\Omega}_x$	Feature space.
C	Category or class in the classification problem.
C_i	i-th category.
$\mathcal{C}$	Set of categories in the classification problem.
M	Total number of classes.
$\boldsymbol{s}$	Input to the feature extractor.
$\boldsymbol{x}$	Input to the classifier.
$\mathcal{X}$	Set of feature data.
$\mathcal{S}$	Set of data, prior to feature extraction.
$\mathcal{F}$	Feature extraction process.
C	Classifier.
R	Pattern recognizer, which includes feature extraction and classification.
$a(\cdot)$	Decision rule.
$\ell(\cdot)$	Loss or cost assigned to a token.
$\mathcal{L}$	Overall loss or Expected loss of the recognizer.
$\mathcal{L}_{\boldsymbol{x}}$	Overall loss or Expected loss of the classifier.
$\mathcal{L}_N$	Empirical loss for a data set of size N.
$\mathrm{p}(\cdot)$	Probability density function.
$\mathrm{p}(\cdot;\Lambda)$	Parameterized probability density function with parameter Λ.
$\Pr(\cdot)$	Probability function.
$\Pr(\cdot;\Lambda)$	Parameterized probability function.
$\mathcal{E}$	Expected error.
$\boldsymbol{v}^{\mathrm{T}}$	transpose of vector $\boldsymbol{v}$.
$\mathcal{R}$	Feature space partioning by a classifier.
$\mathcal{R}_j$	j-th region from the partioning made by a classifier.
$\boldsymbol{\Sigma}_{\boldsymbol{s}}$	Covariance matrix in the signal space.
$\boldsymbol{\Sigma}_{\boldsymbol{x}}$	Covariance matrix in the feature space.

$\boldsymbol{\Sigma}_W$	Within class covariance matrix.
$\boldsymbol{\Sigma}_B$	Between class covariance matrix.
$\boldsymbol{\Sigma}_T$	Total covariance matrix.
$g_i(\boldsymbol{x}; \Lambda_i)$	Discriminant function of category C_i.
$\boldsymbol{g}(\boldsymbol{x}; \Lambda)$	decision vector.
	$\boldsymbol{g}(\boldsymbol{x}; \Lambda) = [g_1(\boldsymbol{x}; \Lambda), g_2(\boldsymbol{x}; \Lambda), ..., g_M(\boldsymbol{x}; \Lambda)]^{\mathrm{T}}$.
$\boldsymbol{x}_1^T$	sequence of vector $\boldsymbol{x}$ of length T.
$\boldsymbol{x}_t$	Feature vector at frame-time t.
$\boldsymbol{s}_t$	signal vector (prior to feature extraction)
	at frame-time t.
$\mathcal{M}$	Model of a phoneme as represented by a finite state machine
	(HMM or prototype-based).
$x_{t,f}$	f-th component of feature-vector $\mathbf{x}_t$.
Λ	Parameter set of the classifier.
Λ_i	Parameter set of the category C_i.
Θ	Parameter set of the feature extractor.
Φ	Parameter set of the overall recognizer.
$\mathbf{c}$	A cepstrum vector.
$w_{i,f}$	Weigthing of frequency f by the i-th filter.
α_i	Gain parameter of the i-th filter.
β_i	Bandwidth parameter of the i-th filter.
γ_i	Center frequency of i-th filter.

Contents

List of Symbols iii

1 Introduction 1

1.1 Background . . . 2
1.2 An Example . . . 3
1.3 Speech Recognition . . . 4
1.3.1 Model of a speech recognizer . . . 4
1.3.2 Statistical pattern recognition . . . 5
1.3.3 Signal representation . . . 6
1.3.4 Acoustic phonetics . . . 6
1.3.5 Acoustic modeling . . . 7
1.4 Feature Extraction in Pattern Recognition . . . 8
1.4.1 Expertise-based Feature Extraction . . . 9
1.4.2 Embedded Feature Extraction . . . 10
1.5 Report Goal . . . 10
1.6 Report Overview . . . 12
1.6.1 Part I: review of basic concepts . . . 12
1.6.2 Part II: formalization . . . 12
1.6.3 Part III: applications . . . 13

I Review of Basic Concepts 15

2 Feature Extraction in Statistical Pattern Recognition 17

2.1 Introduction . . . 18
2.2 Formulation of Statistical Pattern Recognizer . . . 19
2.2.1 Recognizer formalization . . . 20
2.2.2 Feature extractor formalization . . . 20
2.2.3 Classifier formalization . . . 20
2.3 Bayesian Decision Theory for Optimum Recognizer Design . . . 21
2.3.1 The minimum error rate classifier . . . 21

2.3.2 Minimum error criteria for feature extractor design . . . 23
2.4 Statistical Feature Extraction . . . 24
2.5 Non-Discriminant Feature Extraction Techniques . . . 24
2.5.1 The curse of dimensionality . . . 25
2.5.2 Data normalization . . . 25
2.5.3 Principal component analysis . . . 26
2.5.4 Independent component analysis . . . 27
2.6 Discriminant Feature Extraction Techniques . . . 28
2.6.1 Feature selection . . . 29
2.6.2 Separability criteria for discriminant analysis . . . 29
2.6.3 Linear discriminant analysis . . . 30
2.7 The Classification Paradigm . . . 33
2.7.1 Discriminant functions for classifier design . . . 34
2.8 Probability Estimation-based Training . . . 35
2.8.1 Maximum likelihood estimation . . . 36
2.9 Discriminative Objective Functions . . . 37
2.9.1 Mean-squared error criterion . . . 38
2.9.2 Maximum mutual information . . . 40
2.10 Conclusion . . . 42

3 Discriminant Features Improve Maximum Likelihood-based HMM Performance: a Simple Experiment 43
3.1 Motivation . . . 44
3.2 Hidden Markov Modeling of Speech . . . 44
3.2.1 The theory of hidden Markov models . . . 45
3.2.2 The Forward algorithm . . . 46
3.2.3 The Viterbi algorithm . . . 47
3.2.4 The Forward-Backward algorithm . . . 48
3.2.5 HMM application to speech recognition . . . 49
3.3 Connectionist approaches . . . 50
3.3.1 Neural Networks for speech recognition . . . 52
3.4 TDNN-Features for HMM-based Phoneme Recognition . . . 52
3.4.1 TDNN architecture for continuous speech recognition . . . 53
3.4.2 Database . . . 56
3.4.3 Experimental procedure . . . 56
3.4.4 Label generation of HMM training . . . 57
3.4.5 HMM phoneme models . . . 58
3.4.6 VQ codebook generation . . . 58
3.4.7 Fuzzy vector quantization . . . 63

3.4.8 Preliminary experiment on the /b, d, g/ task . . . 63
3.4.9 Recognition of all Japanese phonemes . . . 66
3.4.10 Discussion . . . 70

II Formalization **71**

4 Discriminative Feature Extraction for Minimum Error **73**
4.1 Introduction . . . 74
4.2 Feature Extraction Based on Minimum Classification Error . . . 75
4.2.1 Formalization . . . 75
4.2.2 Minimum Classification Error estimation . . . 76
4.3 MCE Estimation and Bayes Error Rate . . . 81
4.3.1 Approximating the Bayes error . . . 82
4.3.2 Effect of loss smoothing . . . 83
4.4 Generalized Probabilistic Descent . . . 84
4.4.1 Probabilistic Descent Theorem . . . 85
4.5 Application of MCE to Speech Processing . . . 87
4.5.1 Application to speaker recognition . . . 87
4.5.2 Application to word spotting . . . 87
4.5.3 Application to speech recognition . . . 88
4.6 Links between MCE and Classical Discriminative Objective Functions . . . 88
4.6.1 MCE and MSE . . . 88
4.6.2 MCE and CFM . . . 90
4.6.3 MCE and MMI . . . 90
4.7 Joint Optimization of Feature Extraction and Classification . . . 92
4.7.1 Subspace methods . . . 92
4.7.2 Minimum Error Learning Subspace . . . 93
4.7.3 Feature extraction properties of Neural Networks . . . 93
4.7.4 Feature selection based on performance . . . 95
4.8 Application of DFE to Speech Recognition . . . 96
4.8.1 Shared feature extractor . . . 96
4.8.2 Category-dependent feature extractors . . . 97
4.9 Conclusion . . . 100

5 Expected loss and empirical loss **103**
5.1 Introduction . . . 104
5.2 Simple links between the empirical loss and the expected loss . . . 106
5.2.1 Law of Large Numbers . . . 106
5.2.2 Hoeffding inequality . . . 107

5.3 Estimating a Minimum of the Expected Lost . . . 107
5.3.1 Uniform convergence in Probability . . . 107
5.3.2 Vapnik and Chervonenski dimension . . . 108
5.3.3 Uniform convergence of the empirical loss . . . 109
5.3.4 Use of VC dimension within MCE . . . 110
5.4 Comparing DFE and Classical MCE/GPD . . . 111
5.4.1 Empirical error rate . . . 111
5.4.2 VC-dimension . . . 112
5.5 Conclusion . . . 112

6 Optimization of DFE-based Recognizers **113**
6.1 Introduction . . . 114
6.2 Choice of Loss Function in MCE . . . 115
6.2.1 Classical MCE losses . . . 115
6.2.2 MCE loss as a distribution function . . . 117
6.3 Minimizing the empirical loss by GPD . . . 119
6.3.1 Choice of the learning rate . . . 119
6.4 Modular Generalized Probabilistic Descent method . . . 123
6.4.1 MGPD simulation . . . 124
6.5 Incremental Generalized Probabilistic Descent Method . . . 126
6.5.1 Learning rates update . . . 127
6.5.2 Adjustment memory update . . . 127
6.5.3 Summary of the algorithm . . . 128
6.5.4 IGPD simulation on classifier training . . . 129
6.5.5 IGPD simulation on DFE-based recognizer . . . 133
6.6 Discussion . . . 136
6.7 Conclusion . . . 137

III Applications **139**

7 Discriminative Feature Extraction Applied to Lifter Design **141**
7.1 Introduction . . . 142
7.2 Liftering-based Speech Recognition . . . 142
7.2.1 Speech production model . . . 142
7.2.2 Cepstrum liftering application to speech . . . 144
7.3 DFE-based Lifter Design . . . 147
7.3.1 Recognizer structure . . . 147
7.3.2 Training process . . . 148
7.4 Speech Data . . . 149

7.5 Results and Discussion . . . 150
7.5.1 Baseline system . . . 153
7.5.2 Effect of initial lifter in DFE training . . . 153
7.5.3 Lifter analysis . . . 154
7.5.4 Comparison with statistically based lifter . . . 156
7.5.5 Comparing MCE and MSE . . . 156
7.5.6 MCE-based lifter and MSE-based lifter . . . 157
7.6 Discussion . . . 158
7.7 Conclusion . . . 160

8 Discriminative Feature Extraction Applied to Filter Bank Design 161
8.1 Introduction . . . 162
8.2 Filter Bank Modeling of Speech . . . 162
8.2.1 Filter bank-based feature extractor . . . 163
8.2.2 Parameter selection for filter bank design . . . 164
8.3 Recognizer Structure . . . 166
8.3.1 Classifier structure . . . 166
8.4 DFE implementation . . . 168
8.4.1 Classifier parameter adjustment . . . 170
8.4.2 Filter Bank optimization scheme . . . 170
8.5 System initialization . . . 173
8.6 Vowel Fragment Recognition Task . . . 174
8.6.1 Task . . . 174
8.6.2 Experimental settings . . . 175
8.6.3 Results: general comments . . . 175
8.6.4 Detailed analysis of the filter bank . . . 182
8.6.5 Classifier and filter bank interaction analysis . . . 184
8.7 Directory Assistance Task . . . 185
8.7.1 Task . . . 185
8.7.2 Experimental settings . . . 185
8.7.3 Results . . . 186
8.8 Conclusion . . . 189

9 Discriminative Feature Extraction Applied to FFT-based Cepstral Parameters 191
9.1 Introduction . . . 192
9.2 Filter Bank-based Cepstrum . . . 193
9.3 DFE-based Design . . . 194
9.4 Classifier Structure . . . 196
9.4.1 Segmental k-means for ML estimation . . . 197
9.4.2 Structure-dependent misclassification measure . . . 198

9.5 DFE Optimization . . . 198
9.5.1 String-level optimization . . . 199
9.5.2 Frame-level optimization . . . 200
9.5.3 Classifier's parameter adjustment . . . 201
9.5.4 Filter bank adjustment . . . 202
9.5.5 Embedding dynamic features . . . 202
9.6 Vowel Segment Recognition . . . 204
9.6.1 Task and classifier structure . . . 204
9.6.2 Experimental conditions and results . . . 204
9.6.3 General results . . . 205
9.6.4 Optimized filter bank analysis . . . 205
9.7 Directory Assistance Task . . . 211
9.7.1 Task . . . 211
9.7.2 Experimental settings . . . 212
9.7.3 System initialization . . . 212
9.7.4 Results . . . 212
9.8 Isolated Letter Recognition . . . 213
9.8.1 Recognizer structure . . . 214
9.8.2 ML-estimation using DFCC . . . 214
9.8.3 DFE-training . . . 217
9.8.4 Using Dynamic Features . . . 219
9.9 Conclusion . . . 221

10 Conclusion **223**
10.1 Summary of the Report . . . 224
10.2 Further extensions . . . 227
10.2.1 Application to large vocabulary . . . 227
10.2.2 Accurate optimization . . . 228
10.3 Few Potential Applications . . . 228
10.3.1 DFE application to time-frequency resolution . . . 228
10.3.2 DFE for noise adaptation . . . 229
10.3.3 DFE within LPC-based model . . . 230
10.3.4 DFE within digital filter design . . . 230
10.4 Final Discussion . . . 230

A Fundamentals of speech recognition **233**
A.1 What is Speech . . . 233
A.2 Various Views of Speech . . . 234
A.2.1 Speech production . . . 234
A.2.2 Speech perception . . . 234

A.2.3 Acoustic modeling 234
A.3 Speech Recognition by Machine 234
A.3.1 Speech representation 235
A.4 Reduction Techniques 236
A.4.1 Vector quantization 236
A.4.2 Time-normalization 236
A.5 Speech decoding 236
A.5.1 Pattern matching 237
A.5.2 Dynamic time warping (DTW) 237
A.5.3 Statistical decision rule 238
A.6 Language Modeling 239

B Back-propagation for classification **241**

C Filter bank optimization by DFE **243**

List of Tables

3.1 Number of VQ training samples. 60
3.2 Training and testing data of the $/b, d, g/$ task. 64
3.3 Recognition rate based Fuzzy-VQ on the $/b, d, g/$ task using TDNN-features. 64
3.4 Comparison of recognition rates in the $/b, d, g/$ task. 65
3.5 Recognition result for all phonemes using Fuzzy VQ. 67
3.6 Recognition rates within phonemic-categories: comparison using Fuzzy VQ. 68
3.7 Recognition rate within phonemic-categories: comparison using Hard-VQ. 69
3.8 Average recognition rates for the 4 feature types 69
7.1 Error rates for the baseline systems and the DFE-trained systems. 154
7.2 Recognition rates for the Japanese vowels task using the MCE criterion and the MSE criterion. 157
9.1 Experimental results using various filter bank cepstral parameters in terms of error rates (%) in the vowel recognition task for I=16 and Q = 15 (representation I). 206
9.2 Experimental results using Mel-based DFCC in terms of error rates in the vowel recognition task for Q=20 and L= 10 (representation II). 206
9.3 Experimental results of ATR names recognition task using string-level training and frame-level training for various cepstral features. 213
9.4 Recognition rate on ISOLET and E-SET using MLE with various mixture components for MFCC and preset DFCC. The preset DFCC has been obtained with using a 5-mixtures-Gaussian-pdf-HMM. 218
9.5 Recognition rate on ISOLET and E-SET using MLE, classical MCE, DFE-S (refers to MCE optimization of previously derived DFCC) and DFE-J (refers to joint optimization) with static filter bank cepstral coefficients. 219
9.6 Recognition rate on ISOLET and E-SET for DFE-S, DFE-J and classical MCE using delta parameters. ΔDFCC-R refers to calculating delta parameters using the regression formula on previously optimized static DFCC. ΔDFCC-R refers to delta parameters, which are derived by embedding the regression formula within the DFE optimization process. 220

List of Figures

1.1 Feature-points of /a/ and /i/ in the two-dimensional formant space. Note that these are fictional formant values for illustration purpose. 3

1.2 General model of the speech recognition process by machine. 5

2.1 Model of a Recognizer. The Recognizer can be viewed as a modular system in which the feature extractor and the classifier are intertwined. 19

2.2 Estimation of probabilities density functions for two classes. Discriminative approaches focus on estimating Bayes region boundaries. 35

3.1 Example of Hidden Markov Model with three states and loops. 45

3.2 The basic unit of a Neural Network without threshold. 51

3.3 The basic TDNN unit. The TDNN unit introduces delay within each input so as to keep track of past events. 53

3.4 Large-phonetic TDNN composed of TDNN subnets aimed at discriminating within each phonemic class. Each subnet is characterized by each own set of delays and number of weights (from [Sawai, 1991]). 55

3.5 The architecture of the recognition process (from [Sugiyama and Biem, 1991]). 56

3.6 The HMM structure for phonemic modeling. 58

3.7 VQ partitioning into regions. A region is represented by a centroid. 59

3.8 VQ distortion as a function of the codebook size. 59

3.9 Values of VQ codes (32 codes). 61

3.10 Values of VQ codes (128 codes). 62

4.1 Model of a DFE-based Recognizer. The Recognizer can be viewed as a modular system in which the feature extractor is parameterized. 76

4.2 The MCE loss function is a continuous approximation of the 0-1 loss function. 80

4.3 General model of speech recognition system design by DFE. 96

4.4 Optimal optimization of a pattern recognizer by DFE. 100

5.1 Empirical error rate and empirical loss as a function of epochs for a letter recognition task using classical MCE with a deterministic gradient optimization. 105

6.1 Various loss functions and their derivatives. 116

6.2 Error-function-based losses and derivatives. The histogram represents the misclassification measures of a database of 150 samples. **Top figures:** 1 mixture loss function and its derivative estimated by EM. **Bottom figures:** 3 mixtures loss function and its derivative estimated by EM. 118
6.3 Different types of learning rates. 121
6.4 Error surface and minimization. The error surface is due to one-dimensional data generated by computer simulating two sources of Gaussian densities centered at -0.5 and 0.3 with a standard deviation of 0.5. Each class is represented by its means, referred to as "class1" and "class2". **Top:** 0-1 loss-based error surface. **Middle:** Error surface using a sigmoid approximation of the error and the curve traced by the estimated configurations during the deterministic gradient descent learning. **Bottom:** Same as middle but with a curve which shows the evolution of the GPD learning. 122
6.5 Classical GPD optimization scheme and MGPD optimization scheme for a fixed number of iterations. **Top**: The performance of the system versus the learning rate ϵ using GPD. **Bottom**: the performance of the system versus the ratio between classifier learning rate ε and feature extractor learning rate ρ in MGPD. 125
6.6 Performances versus learning parameters in the classical MCE-based recognition task using the same learning rate for all parameters of the classifier. Figure (a) illustrates GPD performance versus the initial learning rate. Figure (b) illustrates IGPD-A performance versus the meta-learning rate. Figure (c) illustrates IGPD-B performance versus the meta-learning rate. 130
6.7 Performances versus learning parameters in the MCE-based recognition task using a class-dependent learning rate. Figure (a): IGPD-A performance versus the meta-learning rate. Figure (b): IGPD-B performance versus the meta-learning rate. 132
6.8 Performance versus the number of epochs for GPD and IGPD in classifier optimization. IGPD uses a class-dependent learning rate. 133
6.9 Performances versus learning parameters in the DFE-based center adjustment using a shared learning rate in the classifier. Figure (a) illustrates performance of GPD application. Figure (b) illustrates performance of MGPD. Figure (c) illustrates IGPD-A performance versus the meta-learning rate. Figure (d) illustrates IGPD-B performance versus the meta-learning rate. 134
6.10 Performances versus learning parameter in the DFE-based vowel recognition task using a class-dependent learning rate. Figure (a) illustrates IGPD-A performance versus the meta-learning rate. Figure (b) illustrates IGPD-B performance versus the meta-learning rate. 135
6.11 Performance versus the number of epochs for GPD, MGPD and IGPD in overall recognizer optimization. IGPD uses a class-dependent learning rate. 136
7.1 The liftering process. Liftering can be viewed as multiplying a cepstral coefficient by a weight 144

7.2 Recognizer structure consisting of a three-layer perceptron classifier with a liftering module. 148
7.3 Baseline rectangular lifters. 151
7.4 Error rates versus the number of hidden nodes. **Upper:** baseline system without liftering (with all pass lifter). **Bottom:** uniformly unitialized DFE-trained system. 152
7.5 **Top figure:** a typical DFE-designed lifter issued from the uniform initialization. **Upper-top:** the entire view. **Bottom-top:** the same lifter with a focus on the lower quefrency region. **Bottom Figure:** a typical DFE-designed lifter issued from the 16-length rectangular initialization. **Upper:** the entire view. **Bottom:** the same lifter with a focus on the lower quefrency region. 155
7.6 Normalized inverse variance over the training set. Normalization is performed such that the maximum value is equal to 1. 156
7.7 MSE-based lifter. 158
7.8 MCE-based liftered power spectrum and MSE-based liftered power spectrum of a vowel frame. 159
8.1 *G-type* training. The center frequencies, bandwidth and gain of the i-th filter can be optimized by adjusting the parameters γ_i, β_i and α_i 165
8.2 Block diagram of a modular speech recognizer consisting of a front-end filter bank-based feature extractor and a back-end multi-prototype distance classifier. 166
8.3 Prototype-based minimum error classifier structure. 167
8.4 Initial filter bank aligned in the Mel scale and the Linear scale. 174
8.5 Error rates versus classifier size for various configurations when using an initial filter bank aligned on the Mel scale in the vowel recognition task. 177
8.6 Error rates versus classifier size for various configurations when using an initial filter bank aligned on the Linear-frequency scale in the vowel recognition task. 178
8.7 Resulting filter banks after DFE optimization of various filter bank parameters in the context of 1 prototype/per vowel. 180
8.8 Formant location in the training data aiming at vowel recognition. 181
8.9 Filter bank modification versus the number of prototypes in the classifier. The degree of modification (FBMOD) is a equal to the average distance across all parameter of the model to the original model. 184
8.10 Experimental results of the directory assistance task for various adjusted parameters. With a 95% confidence interval the the histogram corresponds to the following error rates. Baseline MLE: 7.9%+− 0.0495. Gc-training: 7.1% +− 0.0471. Gb-training: 6.1% +− 0.0439. Gg-training: 5.3% +− 0.0411. I-type training: 9.7% +− 0.05431. . . 187
8.11 Experimental results of Mel-scale based filter bank optimization task in the directory assistance task. 188
9.1 Top: Mel log spectrum correlation matrix. Bottom: MFCC correlation matrix. 195
9.2 Block diagram of DFE-based cepstrum optimization. 196

9.3 String-level optimization using a normalized misclassification measure. When using a normalized misclassification the gradient of the loss for models that belong to both paths do not cancel out 200
9.4 Frame-level optimization using frame-based normalized misclassification measure. Only models that do not belong to both path are updated 201
9.5 **Left**: Resulting Mel-scale based filter bank center optimization task for I = 16 and Q = 16 (representation I). **Right**: Resulting Mel-scale based filter bank center optimization task for I = 20 and Q = 10 (representation II). Each channel is plotted versus its modified center. 207
9.6 **Left**: Resulting Mel-scale based filter bank bandwidth optimization task for I = 16 and Q = 15 (representation I). **Right**: Resulting Mel-scale based filter bank bandwidth optimization task for I = 20 and Q = 10 (representation II). The CBW is plotted versus the center frequency in each channel. 208
9.7 **Left**: Resulting Mel-scale based filter bank gain optimization task for I = 16, Q = 15 (representation I). **Right**: Resulting Mel-scale based filter bank gain optimization task for I = 20, Q = 10 (representation II). The values of α_i are plotted against the center frequencies in the channels. 209
9.8 Resulting Mel-scale based filter bank weighting optimization. **Left**: Resulting Mel-scale based filter bank weighting optimization task for I= 16, Q = 15 (representation I). **Right**: Resulting Mel-scale based filter bank weighting optimization task for I = 20, Q = 10 (representation II). 210
9.9 Resulting Mel-scale based filter bank CBG optimization task. **Left**: Resulting Mel-scale based filter bank center-bandwidth-gain optimization task for I = 16, Q = 15 (representation I). **Right**: Resulting Mel-scale based filter bank center-bandwidth-gain optimization task for I = 20, Q = 10 (representation II). 210
9.10 Initial and Optimized filter bank using the joint MLE/DFE optimization scheme in the ISOLET task. 216
9.11 DFCC performance on ISOLET. The classifier is untrained by MCE but re-estimated at each trial by MLE using DFE-estimated features 217
A.1 Principle of DTW 237

Chapter 1

Introduction

Man is a meeting-point of various stages of reality

-Rudolf Eucken-

1.1 Background

Will it be possible to talk to your computer ? This question summarizes the paradigm of automatic speech recognition.

The building of a machine which simulates the human perception process has been, and still is, the dream of many researchers worldwide. In addition to the straightforward economic opportunities that would result from speech recognition applications, such as voice dialing, automatic dictation, and more sophisticated human-machine interactions, a computer that can perceive human speech would also be useful for the handicapped. Automatic transcription of TV program will be available for the deaf and people without arms will find themselves suddenly capable of performing any basic task that usually requires the use of hands. Obviously, using our voice in place of our hands will have revolutionary effects on our way of living.

However, from the design of a simple "phonetic typewriter" to the realization of a "perceptual" machine, various difficulties are in the way to realizing these goals. The most fundamental and certainly the most difficult task is to find out what "perception" is. The mystery is that, we use our perceptual abilities on a daily basis without being able to understand this fundamental concept. The debate over whether this "self-perception" is possible is beyond the scope of this report. However, since it is in the human nature to try to simulate nature, pattern recognition, which means for instance, recognizing a shape, or being able to distinguish a dog from a cat, or one person voice from another one, is one aspect of these human capabilities that has been to target of many simulation attempts by computers.

The human perception process is an astounding pattern recognizer. It seems therefore natural to study the baseline process which permits a given person to recognize thousands of words, even when uttered in a very noisy environment. However, pattern recognition research, based on perceptual simulation, is limited due to the fact that the physiological and the biological aspects of the human perception process can not be investigated *in vivo*. The field of psychoacoustics, which considers the human being as a black box and tries to comprehend the inherent perceptual processes by analyzing responses to selected stimuli, sometimes produces conflicting perceptual models. So far, a good model of human perception is thus still awaited.

Instead of waiting for the final theory of "what cognition is", researchers have preferred the use of mathematical tools in the pattern recognition field, assuming that despite the lack of a clear perceptual theory, a few problems may be solved by classical mathematical models. In particular, pattern recognition through classification is one of the problems that a computer should be able to solve. Classification, that is, the assignment of events or objects to prescribed categories, is the first step toward the building of a perceptual machine. There is a wide area of literature covering the subject that has led to a theoretical formulation of pattern recognition and proposed solutions to particular problems. Speech recognition by machine can be viewed as one particular aspect of a more general machine-based theory of pattern recognition.

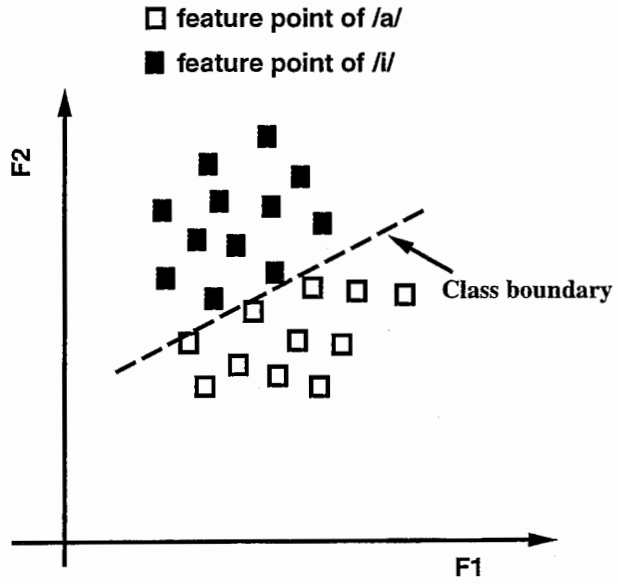


Figure 1.1: Feature-points of /a/ and /i/ in the two-dimensional formant space. Note that these are fictional formant values for illustration purpose.

1.2 An Example

To illustrate the point, let us suppose that we want to design a machine that can distinguish between two sounds, for instance the two vowels /a/, /i/. First, we need a transducer (in this case a microphone), that converts the sound waves into an electrical signal that can be stored for further precessing. From the waveform coming out of the transducer, we need to extract some special parameters which are characteristic of the two sounds and remove unwanted parameters. Here, expertise comes to our help because it has been established, due to investigations in speech production and perception, that vowels are characterized by specific peaks in their spectrum called *formants* and that vowels are distinguished by the values of their formants. Thus, it suffices to extract, for example, the first and second formants, F1 and F2 respectively, from the waveform and use these parameters as being representative of the sounds. This process is called *feature extraction.* The output of the feature extraction process (here the first and second formant values, which could be stored in a feature-vector for convenience), are passed on to the classification stage, which makes the final decision regarding the identity of the sounds.

We can see that the speech waveforms are now represented by points in a two-dimensional feature-space spanned by the values of F1 and F2. The classifier acts on the feature space by finding the boundary which separates the point belonging to /a/ from those belonging to /i/ and thus, devises a *classification rule.* For instance, as illustrated in Fig. 1.1, the classifier would consider as belonging to /a/ all feature-points that fall below the boundary line while others are classified as /i/. Clearly, the job of the classifier is much easier if the feature-points of /a/ and

/i/ are well separated in the feature-space. This illustrates the importance of features in a pattern classification problem.

The example above has shown three main elements that form the basis of a pattern recognizer: the transducer, the feature extractor and the classifier. The transducer (or receptor), takes the input and converts it into a form suitable for processing. The feature extractor (or pre-processor, attribute detector, property filter), extracts the relevant information and the classifier uses this information to assign the input to one of the prescribed categories. This is the basic model of a pattern recognition which does not depend on the framework of application.

1.3 Speech Recognition

Speech recognition is one aspect of the human capability of processing various types of information. The basic model of the pattern recognizer, described above, can be used for recognizing acoustic patterns. However, due the complex nature of the speech signal, this basic model is usually adapted for this particular application.

1.3.1 Model of a speech recognizer

Within this report, we consider the model illustrated in Fig. 1.2.

This model is composed of a front-end processing module and a back-end processing module. The front-end processing module takes an input from the environment. This input $\mathcal{I}$ is obtained by a transducer and is passed trough a signal measurement phase which produces a form S of the signal, suitable for further processing. S can be considered as the practical input to the recognizer, which first transforms S into a representation X by the feature extraction module. X can be considered to be a compact representation of both S and $\mathcal{I}$ and is the input to the evaluation (or classification) process. For illustration, $\mathcal{I}$ is the sound pressure, represented as a waveform, and S is its digital representation, either in the time-domain or spectral domain. X is thus the feature representation (filter bank outputs, cepstral coefficients, ...).

The above illustration characterizes most speech recognition schemes. In general, however, S and X are represented in different ways. Usually, the representation of S and X falls into one of the category below.

R-1 *continuous and sequential representation.* This is the case when the signal is a waveform or a sequence of spectral vectors.

R-2 *Discrete and symbolic representation.* This is the conceptual level, in which the signal is represented as an abstraction. For instance, an utterance is represented by its transcription. The symbolic representation is the input to most speech *understanding* systems.

R-3 *summarized representation*, which is between the two previous levels.

In this report, S and X are of the form **R-1**, whereas the output the recognizer is of the form **R-2**.

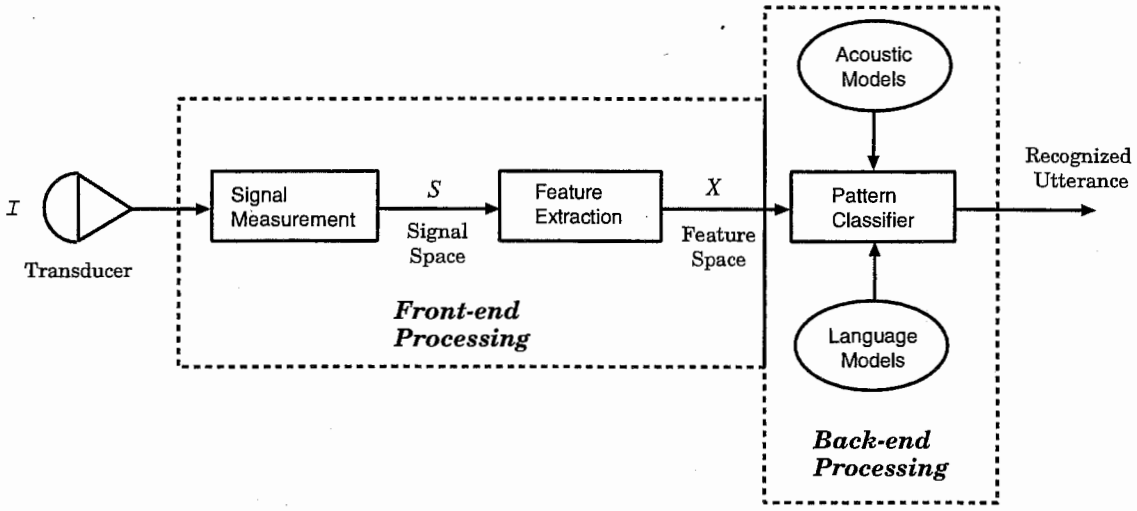


Figure 1.2: General model of the speech recognition process by machine.

1.3.2 Statistical pattern recognition

The X-space is the input space of the pattern classifier. As said before, pattern recognition is an essential aspect of human intelligence, and can be viewed as associating continuous and sequential representation with symbolic representation (modeled as category or class). The human pattern recognizer is an information processing scheme, based on overall judgment which deals in an impressive manner with the uncertainty and ambiguity encountered in the real world. The human pattern recognizer comprises an essential aspect of logic, which enables knowledge to be acquired by *induction* and *deduction.*

The British philosopher David Hume, in his *Treatise on Human Nature* published in 1739, argued that knowledge is acquired from experience. In 1787, this idea was partly refuted by the German Philosopher Immanuel Kant in his *Critique of Pure Reason*, in which he stated that although all knowledge begins with experience, it does all arise out of experience. It seems, however, that most designers of pattern recognition systems are faithful disciples of David Hume, since "knowledge" acquisition is usually simulated by an artificial pattern recognizer, in which the training of the recognizer is done in a supervised manner by using examples, through an algorithm which lets the recognizer learn the correspondence between a pattern and a category. Due to ambiguity and uncertainty, added to the fact that in a real world situation, patterns are noisy, the correspondence between a pattern and a category can be expressed in terms of *probability or statistics.* The Bayesian decision-theoretic approach, which assigns a cost to a given decision, is the fundamental framework of statistical pattern recognition. That is, the recognizer makes a decision to assign a pattern to a category based on 1) the feature representation of the pattern and 2) the statistics of the category. This is the framework of *statistical pattern recognition*, which is the basis of most state-of-the-art speech recognition systems.

1.3.3 Signal representation

In speech recognition, the signal S is time-dependent. That is, a particular speech utterance is simply represented as a waveform $s(t)$, which is digitized and presented in the form $s(n)$, where n refers to a sample. The speech signal is usually assumed to be quasi-stationary within a 10 to 30 ms interval, which enables the use of an analysis window. Shifting the window along the speech utterance results in the representation X, where $X = \{\boldsymbol{x}_1, ...\boldsymbol{x}_t, ...\boldsymbol{x}_T\}$ is a sequence of frame-vector $\boldsymbol{x}_t$, and T is the length of the sequence.

In most speech recognizers, $\boldsymbol{x}_t$ contains short time spectral features, which means that time-domain features are rarely used in most state-of-the-art speech recognition systems. This is due to the fact that 1) acoustic contexts are transcribed into the frequency domain (that is why it is possible for a human expert to read a spectrogram) and 2) it has been shown that the human auditory system performs some sort of spectral analysis [More, 1986]. The drawback of this approach is that the temporal resolution is usually lost, which has led to the search for a general time-frequency analysis method, such as the Wiener-transform and the Wavelet transform.

1.3.4 Acoustic phonetics

Linguists hold that any natural spoken language is based on a finite set of elementary sounds called *phonemes*. A phoneme is a linguistic unit which escapes clear definition. It is mainly characterized by its effect. That is, replacing a phoneme within a word changes the sound of the word. Each language has a specific set of phonemes which can be described by Jacobson's distinctive feature theory [Jacobson *et al.*, 1961]. The Jacobson theory characterizes each phoneme using binary features, such as the presence or absence of voicing, nasality, or the constricting of some place in the vocal tract and has been used in speech recognition [R. de Mori and Laface, 1984]. This means that articulatory parameters determine the phonemic type.

The acoustic representation of the phoneme varies highly with the speaker and the context of the phoneme (place within the utterance, neighboring phonemes), meaning that the phonemic-acoustic correspondence is not a one-to-one mapping: the same phoneme displays various acoustic representations according to the speaker, the context and the background noise. This lack of a one-to-one correspondence is perhaps the fundamental problem in speech recognition. Phonemes are usually classified into two broad classes: vowels and consonants.

Vowels

Vowels are one of the most interesting linguistic categories in speech recognition. It can be argued that accurate speech recognition heavily depends on accurate vowel recognition.

Vowels provide a good framework for the analysis of a given feature representation. During the production of a vowel sounds the vocal tract is essentially fixed and is excited by a quasi-periodic pulse caused by the vibration of the vocal cords. The rate of vibration determines the fundamental frequency of the vowel sound whose perceptual effect is known as the *pitch*. The vowel type is

determined by the shape of the vocal tract and the position of the jaws, tongue and lips. In some languages, such as French, the velum also contributes to the vowel sound.

An important observation is that the shape of the vocal tract determines the corresponding vocal-tract filter and consequently, the corresponding frequencies of resonance. Vocal-tract frequencies of resonance are what was previously referred to as *formants*. Formants appear as peaks in the vowel spectrum and are considered to be one of the most reliable features for vowel recognition.

Consonants

Broadly defined, consonants are non-vowel sounds. A more precise definition categorizes consonants according to the manner of articulation (the degree of constriction in the vocal tract, the position of the tongue, the nasal opening) and the place of articulation (the location of the obstruction within the vocal tract). Stops, such as $/p/, /t/, /k/, /b/, /d/, /g/$, are produced by a total obstruction of the vocal tract. The fricatives $(/s/, /sh/, /h/)$, are produced by a narrow constriction of the vocal tract. Nasals $(/m/, /n/)$ are also produced by constriction of some point of the vocal tract. Liquids $(/l/, /r/)$ and glides $(/j/, /w/$, also called semi-vowels) are characterized by a gliding transition between adjacent phonemes. Affricates $(/ch/, /ts/)$ can be seen as a combination of fricatives and stops [Flanagan, 1972; Fant, 1973; Rabiner and Juang, 1993].

In general, it is not an easy task to derive features for each specific consonant-class, which makes the knowledge-based approach difficult to be carried out in practice.

1.3.5 Acoustic modeling

In speech recognition, the back-end decision relies on pre-stored knowledge, which, in the approach taken in this report, is stored in the form of acoustic models or templates, as shown in Fig. 1.2. In most state-of-the-art speech recognizers, a language model, which is a model of the inherent structure of the language in use, helps in the final decoding of the utterance. This model of speech recognition represents most speech recognition systems that rely on a phonetically-based approach to speech recognition. The acoustic-phonetic approach to speech recognition has the following characteristics.

- Each phoneme is represented by a parameterized model $\mathcal{M}_i$. For P phonemes in the language, there are P phonemic models $\mathcal{M} = \{\mathcal{M}_1, ...\mathcal{M}_i, ...\mathcal{M}_P\}$, which are used in classification stage.

- The structure of the model is a finite state automaton. Thus, $\mathcal{M}_i = \{\mathcal{M}_{i,1}, ..., \mathcal{M}_{i,q}, ...\mathcal{M}_{i,Q_i}\}$, where $\mathcal{M}_{i,q}$ denotes the model parameters in the state q of model $\mathcal{M}_i$, and where Q_i is the number of states in $\mathcal{M}_i$.

- A phoneme model is constructed using exemplar data.

- The final categorization is determined by the task. For word recognition, the number of classes is the same as the number of words in the vocabulary and for phoneme recognition, it is the number of phonemes.

In the above framework, an important point is the use of a similarity measure between a feature-vector of an utterance and a state within a model. The overall utterance is decoded by performing a global measure between the utterance and the states of all phonemic models, and the best match is chosen. Usually, the global measure is performed by Dynamic Programming, based on a local similarity measure. The similarity measure and the type of model determine the framework of the recognition system. Most similarity measures belong to one of the following cases:

- A model for each category is reduced to an exemplar utterance and the similarity measure is a distance (Euclidean, Mahalanobis, Itakura-Saito). This is case is illustrated by the classical Dynamic Time Warping-based (DTW) pattern matching approach.

- The parameters in a model state are prototype feature-vectors and the similarity measure to a state is an L_p-norm of distances to the prototype feature-vectors within the state. This is the prototype-based recognizer.

- Each model state is associated with a probability function and the parameters of the state are simply the parameters of the probability function. The similarity measure of a given feature-vector is simply the output of the probability function for the particular feature-vector. This framework is widely known under the Hidden Markov Model(HMM) instantiation.

Most acoustic modeling schemes are a simple variant of the three categories presented above, which are characterized by the same fundamental structure, namely a finite state automaton, but with difference in similarity measures. For instance, in the last case, with a probability as a similarity measure, the probability can be discrete (Discrete Hidden Markov Models), modeled by a mixture of probability density functions (Continuous Hidden Markov Models) [Rabiner, 1989], or implemented by an Artificial Neural Network (Hybrid NN/HMMs) [Morgan and Bourlard, 1990; Haffner *et al.*, 1991].

Within this report, all acoustic-modeling approaches which fall into the above categories are considered essentially equivalent because the structural architecture is fundamentally the same. In particular, the prototype-based classifier was used in most of the experiments described in this report, because of the flexibility in choosing any distance measure as a state-distance measure.

1.4 Feature Extraction in Pattern Recognition

In this report, the focus is on the feature extractor and the classifier, and in particular, on their interaction. In general, feature extractor design is more difficult: the same classifier structure can be used for a wide range of task whereas the feature extractor design depends on the problem at hand. Usually, the feature extractor designer relies on *a priori knowledge* (as in the example shown

above), the statistics of the data or both. On the other hand, the same classifier's architecture is used across different tasks. This is one factor that has led to the separate design of classifiers and feature extractors with two schools of thought: one which focuses on optimizing the feature extractor and one which focuses on designing the classifier.

The construction of the feature extractor falls into one of the following schemes.

S-1 *Expertise-based Feature Extraction* which relies on knowledge or statistical analysis about what constitutes the relevant information within the input pattern S. The knowledge usually derived from previous investigations in the area of expertise.

S-2 *Embedded Feature Extraction*, where the feature extractor is considered part of the recognizer and designed jointly with the back-end classifier.

S-1 is the conventional approach to feature extraction in most pattern recognition tasks, and speech recognition in particular. The design is done by a procedural scheme (in the form of an algorithm) based on a priori knowledge or statistics of the data, which are supposed to be optimum in some sense. In **S-2**, a family of feature extractors is considered in the form of parameterized feature extractors and the optimum feature extraction process is obtained by optimizing the feature extractor parameters using training data.

1.4.1 Expertise-based Feature Extraction

The Expertise-based Feature Extraction scheme (EXFE) is the conventional way to perform feature extraction in the speech recognition framework. The feature extraction is done through various sub-tasks to obtain the final representation and then this representation is used to optimize the classifier. In this framework, the feature extractor and the classifier are designed according to separate criteria.

The Expertise-based Feature Extraction scheme suffers from the following shortcomings:

- The EXFE scheme needs some form of knowledge concerning the task. In speech recognition, this knowledge usually comes from research regarding speech production (articulatory phonetics), perception (acoustic phonetics), and spectral analysis. This knowledge is far from complete, resulting in the sub-optimality of the EXFE method.

- EXFE is only specified for certain tasks, and thus cannot be easily adapted to general application. For instance, features that are believed to be efficient in recognizing voiced sounds are inappropriate for capturing unvoiced characteristics (e.g., LPC-based models).

- EXFE is usually done through a sequence of procedures. Thus, errors at each step are accumulated and cannot be corrected at the final stage.

- The optimization done at each step does not guarantee optimization of the overall process.

- Optimization of the feature extractor is usually done without considering the classification stage, meaning that the overall recognition system is far from optimal: the design of the classifier is based on a criterion different from the criterion used in the design of the feature extractor.

The EXFE scheme is the standard feature extraction method in the speech recognition field. Although standard systems have been effective to some extent, they are undoubtedly characterized by the shortcomings listed above. Some of these shortcomings may be overcome by the Embedded Feature Extraction scheme.

1.4.2 Embedded Feature Extraction

The Embedded Feature Extraction (EMFE) scheme adjusts the feature extraction based on some criterion to be optimized. The feature extractor is "trained" by means of exemplar data. This assumes a parameterized form of feature extraction. Consequently, for the method to be applicable to a given task, the feature extraction should have some degree of freedom, subject to being represented by a few parameters within the system.

The Embedded Feature Extraction scheme is able to overcome some limitations encountered in EXFE methods. The EMFE scheme makes it possible to achieve optimality (at least locally) of the overall process for an appropriate choice of criterion to be optimized. EMFE can be applied to any task, assuming that a parameterized form of feature extraction is available. More importantly, it enables a better match of the feature extraction process to the classification stage. Also, EMFE allows the implementation of an adaptive feature extraction process, in which the feature extraction is adaptively optimized. Clearly, adaptive feature extraction inherits the advantages of adaptive systems, such as adaptability to the real world environment, asymptotic optimality, and other general capabilities.

EMFE applications have mainly been found within the framework of Artificial Neural Networks (ANN), simply because the architecture of such systems embeds feature extraction and classification. Subspace methods (SM) also provides a similar framework. However, for standard speech recognizers, the EMFE approach has raised little interest. This may be due to one of the following reasons. 1) early work on speech recognition had focused on a knowledge-based approach (e.g., spectrogram reading) in deriving speech features and later research on the matter, by ignorance or laziness, has followed the same path. 2) EMFE may be computationally costly, especially for large vocabulary recognition tasks 3) standard feature extraction techniques have been satisfactory to some extent.

1.5 Report Goal

The goal of this report is to introduce a formalism for Embedded Feature Extraction based on minimizing the system's errors through discriminative training and present application to selected speech feature extraction frameworks.

In the speech recognition field, feature extraction can be carried out in numerous ways. State-of-the art systems uses features based on an estimation of the spectrum. Across various spectrum estimation techniques, Linear Predictive Coding (LPC) and Discrete Fourier Transform (DFT) are the most popularized techniques. LPC spectrum analysis originated from speech coding and was extensively used in the 70's and early 80's [Makhoul, 1975]. However, the trend nowadays is towards DFT-based analysis, performed through the Fast-Fourier Transform (FFT) algorithm. This may be due to the FFT better immunity to noise and the possibility to implement a perceptual scale by the warping of the frequency through simple weighting of the DFT bins. In this report, the current trend is followed. That is, in all experiments in this work, the features are based on FFT.

The focus is therefore to re-design FFT-based feature extraction techniques by means of Embedded Feature Extraction (EMFE) techniques. The criterion for optimization is simply the same criterion used in back-end classification, within the statistical pattern recognition framework. Among various methods available to implement a statistical pattern recognizer, here, the focus is to achieve the recognition of the input pattern through discriminative training aiming at minimum error. The discriminative training approach has known considerable success recently through the Minimum Classification Error/Generalized Probabilistic Descent method (MCE/GPD) formalism [Juang and Katagiri, 1992a], which is the framework of our work throughout this report. Discriminative Feature Extraction (DFE) optimizes the feature extractor through a discriminative training criterion aiming at minimum classification error.

The immediate consequence of the Discriminative Feature Extraction approach is that the pattern recognizer is designed in such a manner that the feature extractor is matched to the classifier, both aiming at a single purpose, which is to minimize the errors occurring in the back-end classification process. This approach departs from the common practice in which the feature extractor is fixed and the focus is on the classification stage, which completely overlooks the interaction between the features and the classification process.

Another concern in this report is to improve over expertise based on empirical knowledge or physiological motivations by making use of data as a basic source of our knowledge. The goal is not to resolve the problem of finding general features given a recognizer structure, but rather to question the use of separate criteria in the selection of a given parameterization method, given a classifier structure.

Consequently, throughout this report, the resulting features are compared to standard knowledge concerning the task. There is no reason why statistically optimized features and knowledge-based features should agree. However, it can be argued that the two frameworks can influence each other. For instance, knowledge-based feature extraction can be used as a starting point for feature optimization and analysis of the EMFE-based features can provide new insight in speech analysis.

1.6 Report Overview

As described above, the goal of this report is to introduce a framework of achieving joint optimization of the front-end processing system with the back-end classification system aiming at minimizing the error rate of the overall recognizer. This is the Discriminative Feature Extraction (DFE) scheme, which is described through a theoretical formulation, followed by practical applications in selected speech recognition tasks.

1.6.1 Part I: review of basic concepts

Basic concepts of statistical pattern recognition, the paradigm of feature extraction and standard speech recognition techniques are reviewed in the first part of report, which is composed of chapter 2 to chapter 3. Basic concepts describing the field of automatic speech recognition are also reviewed in Appendix A.

In the first part of chapter 2, the concept of minimum error is reviewed through the Bayes theoretic-decision approach and its consequences for optimal feature extraction are outlined. In the second part of chapter 2, it is argued that two approaches can be adopted depending on the choice of a design criterion: criteria that focus on estimating the probability distribution or criteria which directly consider the discrimination of an input pattern. Thus, classical error criteria, such as the Maximum Likelihood estimation (MLE), the Mean-Squared Error (MSE) and the Maximum Mutual Information (MMI), are reviewed in light of their discriminant capacities.

Chapter 3 studies the usefulness of discriminant features in the conventional recognizer design by describing an experiment in which discriminant features are derived by the MSE criterion and the acoustic models are estimated by the MLE criterion. Contrasting MSE-derived discriminant features with classical cepstral features shows how making use of discriminant features can significantly improve performance, even in the classical framework of recognizer design. This chapter can also be viewed as an introduction to speech recognition techniques.

1.6.2 Part II: formalization

The second part of the report, which comprises chapters 4 through 6, is concerned with the theoretical formulation of the Discriminative Feature Extraction method.

The first part of chapter 4 introduces the concept of Discriminative Feature Extraction as a useful extension of MCE/GPD formalism of discriminative training applied to feature extraction. Consequently, the MCE criterion is reviewed and the GPD algorithm is described. It is shown how its use directly minimizes the error rate of a recognition system. Links between the MCE criterion and the Bayes error are discussed, providing a framework of comparison between the MCE criterion and other discriminant criteria, such as MMI, CFM, and MSE. The second part of chapter 4 discusses links between the DFE approach and other approaches, such as Artificial Neural Network and Subspace Methods, which, similar to the DFE method, aim at the integration of the feature extractor and the classifier.

Chapter 5 discusses the asymptotic behavior of a DFE-based recognizer and, in particular, the link between the use of a finite number of data in practical application of DFE and minimization of the probability of error. The MCE criterion is to minimize the expected loss. However, in practical applications, only the empirical loss, defined over a finite set of data, is accessible. Consequently, the problem of minimizing the expected loss, based on a finite data set is discussed. In particular, this chapter studies a reasonable criteria, for which a minimum of the empirical loss is a good estimate of the minimum of the expected loss.

Chapter 6 is concerned with the algorithmic part of the DFE process. In this chapter, various optimization methodologies within the DFE framework are described. In particular, the IGPD and MGPD algorithms are suggested as alternative to GPD. Also, a method for an appropriate choice of the MCE loss function is proposed.

1.6.3 Part III: applications

The third part of the report, that is, chapters 7 to 9, focuses on DFE experiments within the context of various speech-oriented feature extraction techniques.

In chapter 7, the DFE method is applied to cepstrum liftering. Cepstrum liftering is a standard speech feature extraction process that was extensively studied in the 1980s. DFE-derived lifters are compared with classical lifter design.

In chapter 8, DFE is applied to the re-design an FFT-based filter bank. Most speech parameterization relies an filter bank to extract meaningful spectral parameters. However, the selection of filter bank parameters, such as center frequencies or bandwidths, has a direct influence on the efficiency of the filter bank. Consequently, DFE is applied to the optimization of various filter bank parameters in the context of vowel recognition and word recognition tasks.

Finally, in chapter 9, the DFE-approach is applied to readjusting Mel frequency Cepstral Coefficients (MFCC), which constitutes the most widely used speech parameterization techniques in the speech recognition area.

Chapter 10 presents the summary of the report in English and Chapter 11 presents a summary in French.

Part I

Review of Basic Concepts

Chapter 2

Feature Extraction in Statistical Pattern Recognition

Sensational new breakthrough
in Improbability Physics

-Douglas Adams-

In this chapter, we review the formulation of a statistical pattern recognizer. Pattern recognition based on statistical methods requires two processes: a feature extraction process and a classification process. Feature extraction is the process whose goal is to find salient parameters which characterize a given category and form a feature space. Classification *is the class-label assignment process for each feature pattern produced by the feature extraction process. A recognizer is then composed of a feature extractor and a classifier. Optimal recognizer design can be realized within the Bayes decision-theoretic approach.*

2.1 Introduction

In this chapter, we review the formulation of statistical pattern recognition. It was seen in the introduction that designing a recognition system requires the processing of data in some way before classifying them. This processing step was termed *feature extraction*, because it is used to ease the classifying task by extracting key components from the input data, which are called *features*. Then, a classifier is supposed to perform classification by detecting the label (i.e., class) of the corresponding features.

Feature extraction is perhaps one of the fundamental problem underlying the pattern recognition field because it is the first step encountered when building up a recognizer. As emphasized previously, the purpose of the feature extractor is to identify, within the data, what information is important for performing accurate classification. The feature extraction process is expected to discard the information irrelevant to the task while keeping the right information. For instance, in designing a word recognition system, the feature extractor may discard information such as speaker identity, speaker stress, speaking rate, which are present in the speech waveform, but not related to the recognition of the word. However, in a speaker recognition task, information about the identity of the speaker is the useful information in contrast to the meaning of the utterance.

It is postulated that the following properties are required for a good feature extractor:

- The features must be compact so as to enable fast analysis in real time applications. Dimensionality reduction is also required due to an unexpected problem called the "curse of dimensionality". This is the *compactness* property of the feature extractor.

- The features must contain the maximum relevant information, which means minimizing the loss of discriminant information. This is the *maximization* property of the feature extractor.

- The feature extractor must discard information irrelevant to the classification task. That is, the feature extractor must produce "clean" features according to the task at hand. This ensures the robustness of the recognizer to "contaminated" data. This is the *cleanness* property of the feature extractor.

The above properties should be realized within the task at hand and describe the *ideal* feature extractor according to this task. In practice, as we shall see, the feature extractor may be far from this ideal. Furthermore, in our knowledge, there is no method of estimating how a designed feature extractor approximates this ideal feature extractor. This is unfortunate, because if such a method was available, it would open the way to devising an algorithm which iteratively improves a given feature extractor towards the ideal model.

In most pattern recognition approaches, feature extraction design falls into two categories:

- Expertise-based Feature Extraction (EXFE).

- Embedded Feature Extraction (EMFE).

EMFE is based on optimizing a criterion while EXFE is generally implemented by means of an algorithm. In this chapter, an overview of EXFE techniques is presented. Most EXFE are based on expertise and/or data statistics. Expertise means empirical knowledge of the task at hand due to scientific or natural information. Statistical analysis uses statistics on the exemplar data in order to determine the invariance and/or the discriminant information between classes.

Features derived from expert knowledge of the task are highly dependent on the current available knowledge. This knowledge is usually incomplete, meaning that the features may lack the "necessary information" for the task. Furthermore, expertise-based feature extractors can only be used within the specific task at hand. Thus, in speech recognition, features based on a human auditory system, may not be of any use in image recognition. In contrast, statistically-based feature extraction methods display a wide variety of standard algorithms, which do not depend on a specific task, but only require appropriate data. Thus, image and speech processing may share similar dimensionality reduction techniques. The first section of this chapter is spent in reviewing standard statistically-based feature extraction methods.

After features are obtained, their accuracy is tested in the classification stage. The optimal criterion for testing the accuracy of the feature extraction process is the probability of error of the classifier. In statistical pattern recognition, the probability of error can be theoretically tackled within the Bayesian framework. However, in practical pattern recognition applications, there is no direct access to the probability of error, hence, it is estimated by the classifier error rate on a given set of *testing* data. The second part of this chapter describes the main framework for classifier design. Classifier design is usually done via optimization methods: a parameterized architecture is chosen and a search for the right parameters is done through optimization of an error criterion.

2.2 Formulation of Statistical Pattern Recognizer

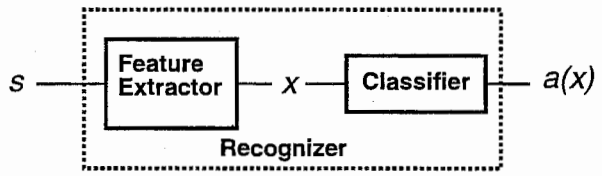


Figure 2.1: Model of a Recognizer. The Recognizer can be viewed as a modular system in which the feature extractor and the classifier are intertwined.

The model of a recognizer is illustrated in Fig 2.1. It is a modular system, in which the feature extractor collaborates with the classifier. We are given an input space $\boldsymbol{\Omega}_s$. Each element $\boldsymbol{s}$ of $\boldsymbol{\Omega}_s$ belongs to a certain category (or class[1]) among a series of M categories $\{C_j\}_{j\in\{1,\ldots,M\}}$. That is, $\boldsymbol{\Omega}_s$ is a labeled space. Formally, the label represents the fact of belonging to one of the classes of $\{C_j\}_{j\in\{1,\ldots,M\}}$. $\boldsymbol{s}$ is either a vector or a *sequence* of vectors of fixed dimension d. This is based on

[1]We shall use the term class or category in describing input data of the same label.

the assumption that in most pattern recognition, a numerical representation of the input data can always be realized.

2.2.1 Recognizer formalization

For each input-datum $\boldsymbol{s}$, the goal of the pattern recognizer is to find the label of $\boldsymbol{s}$, which means recognizing to which category the pattern belongs. Consequently, the recognition task is reduced to a classification task. In speech recognition, as briefly seen in the introduction, the recognition involves more post-processing after the classification of acoustics units. However, for simplicity, we assume throughout this chapter that recognition is reduced to classification. The pattern recognition process can be represented as a mapping:

$$\mathsf{R} \ : \ \boldsymbol{\Omega}_s \rightarrow \mathcal{C} = \{C_j\}_{j\in\{1,\ldots,M\}}, \tag{2.1}$$

which assigns each input data $\boldsymbol{s}$ of $\boldsymbol{\Omega}_s$ to its corresponding class.

2.2.2 Feature extractor formalization

Before recognition, the feature extraction transforms the input data $\boldsymbol{s}$ into a suitable-for-classification representation $\boldsymbol{x}$, which bears the necessary information required to accurately classify $\boldsymbol{s}$. Thus, the feature extractor process $\mathcal{F}$, is also a mapping:

$$\mathcal{F} \ : \ \boldsymbol{\Omega}_s \rightarrow \boldsymbol{\Omega}_x, \tag{2.2}$$

where $\boldsymbol{\Omega}_x = \mathcal{F}(\boldsymbol{\Omega}_s)$ is the space spanned by the transformed input data. $\boldsymbol{\Omega}_x$ is referred to as the *feature space* and inherits the labeling information of $\boldsymbol{\Omega}_s$. That is, each feature-sample $\boldsymbol{x} = \mathcal{F}(\boldsymbol{s})$ has the same label as $\boldsymbol{s}$.

2.2.3 Classifier formalization

The classifier $\mathrm{C} = (a|\mathcal{F})$ is a decision rule $a(\cdot)$, conditioned on the feature space $\boldsymbol{\Omega}_x$, produced by $\mathcal{F}$. $a(\cdot)$ describes the assignment of a feature-sample $\boldsymbol{x}$ into one the classes C_j. Hence, it is represented as

$$a \ : \ \boldsymbol{\Omega}_x \rightarrow \mathcal{C} = \{C_j\}_{j\in\{1,\ldots,M\}}. \tag{2.3}$$

(2.3) is equivalent to the partitioning of the feature space $\boldsymbol{\Omega}_x$ into M regions $\mathcal{R} = \{\mathcal{R}_j\}_{j\in\{1,\ldots,M\}}$, where

$$\mathcal{R}_j = \{\boldsymbol{x} \ \in \ \boldsymbol{\Omega}_x, \ \ a(\boldsymbol{x}) = C_j\}. \tag{2.4}$$

Given a feature extractor $\mathcal{F}$, a classifier is uniquely characterized by its decision rule $a(\cdot)$, which yields the regions R_j. In later sections, the classifier C is referred to by its decision rule $a(\cdot)$, when there is no confusion about the feature space on which classification is performed.

Let $\mathcal{R}(\mathrm{C}) = \mathcal{R}(a|\mathcal{F})$ represents the partitioning derived by the classifier $a(\cdot)$, given the feature extractor $\mathcal{F}$. We have the following definition.

Definition 1 *Given $\mathcal{F}$, two classifiers $a_1(\cdot)$ and $a_2(\cdot)$ are equivalent in $\mathcal{F}$ if and only if $\mathcal{R}(a_1|\mathcal{F}) = \mathcal{R}(a_2|\mathcal{F})$.*

2.3 Bayesian Decision Theory for Optimum Recognizer Design

When designing a recognizer, one would expect the recognizer to make the smallest number of errors in the classification stage. Therefore, the selection criteria for pattern recognition design should be the minimization of the probability of error or error rate of the recognizer. Bayes decision theory provides a framework for optimal classification formulation. This section describes the Bayesian decision theory and shows how it relates to optimal recognizer design. For simplicity, a pre-designed feature extractor $\boldsymbol{x} = \mathcal{F}(\boldsymbol{s})$ is assumed, and focus is on formalizing the probability of error of the classifier.

We have seen that the classifier design is related to the choice of the decision rule $a(\cdot)$. One method to devise such a rule is to make use of decision theory and, in particular, Bayesian decision theory. Decision theory, which is related to game theory, was pioneered in the 30s, by Neyman and Pearson [Neyman and Pearson, 1933] in their work regarding hypothesis testing and the use of the probability of error as selection criteria for classifiers. A general formalization was made by Wald [Wald, 1950], which introduced the notion of *loss* and *risk* in the selection of the decision rule. Bayes decision theory is formulated as follows.

Let $\ell(a(\boldsymbol{x}) = C_j|C_k)$ be the loss incurred when assigning the feature vector $\boldsymbol{x}$ belonging to category C_k to category C_j. The expected loss, given this loss definition, is

$$R(a(\boldsymbol{x}) = C_j|\boldsymbol{x}) = \sum_{k}^{M} \ell(a(\boldsymbol{x}) = C_j|C_k)\,\Pr(C_k|\boldsymbol{x}), \tag{2.5}$$

with $\Pr(C_k|\boldsymbol{x})$ being the *a posteriori* probability of category C_k, given $\boldsymbol{x}$. In Bayes terminology, this expected loss is called the conditional risk incurred in the classification of $\boldsymbol{x}$ [Duda and Hart, 1973]. The optimal classifier should minimize the overall risk defined as

$$\mathcal{L} = \int_{\boldsymbol{\Omega}_x} R(a(\boldsymbol{x})|\boldsymbol{x})\,\Pr(a(\boldsymbol{x})|\boldsymbol{x})\mathrm{p}(\boldsymbol{x})d\boldsymbol{x}, \tag{2.6}$$

where $\mathrm{p}(\boldsymbol{x}) = \sum_{k=1}^{M} \Pr(C_k)\mathrm{p}(\boldsymbol{x}|C_k)$ is the probability density function of $\boldsymbol{x}$ and $\Pr(C_k)$ the *a priori* probability of C_k. Clearly, the risk is minimized if $R(a(\boldsymbol{x}) = C_k|\boldsymbol{x})$ is the smallest for every feature-sample $\boldsymbol{x}$. Thus, the general form of the Bayes rule is

$$a(\boldsymbol{x}) = C_j \qquad \text{if} \qquad R(a(\boldsymbol{x}) = C_j|\boldsymbol{x}) < R(a(\boldsymbol{x}) = C_k|\boldsymbol{x}) \;\text{ for all } k \neq j. \tag{2.7}$$

That is, assign $\boldsymbol{x}$ to the category which yields the smallest risk. The classifier performing the selection rule of (2.7) is referred to as the Bayes minimum risk classifier.

2.3.1 The minimum error rate classifier

Given a feature-sample $\boldsymbol{x}$, one would expect a good classifier to minimize the error made in assigning a class label to $\boldsymbol{x}$. Statistically, it means minimizing the average probability of error, which is the

error rate of the classifier. In Bayesian decision theory, the error rate is defined along the following lines [Duda and Hart, 1973]. The errors are numbered by using a loss of following form:

$$\ell(a(\boldsymbol{x}) = C_j | C_k) = \begin{cases} 1 & j \neq k \\ 0 & j = k. \end{cases} \tag{2.8}$$

This loss assigns the value of 1 to an incorrect decision while a correct decision incurs no loss. Consequently, the risk in classifying $\boldsymbol{x}$ as C_j is reduced to

$$R(C_j|\boldsymbol{x}) = 1 - \Pr(C_j|\boldsymbol{x}) \tag{2.9}$$

and the Bayes decision rule to minimize this risk has the following form:

$$a(\boldsymbol{x}) = C_j \qquad \text{if} \qquad \Pr(C_j|\boldsymbol{x}) > \Pr(C_k|\boldsymbol{x}) \quad \text{for all } k \neq j. \tag{2.10}$$

In this context, the overall risk $\mathcal{L}$ is simply the probability of error defined as

$$\mathcal{E}_{Bayes}[\mathcal{F}] = \sum_{k=1}^{M} \int_{\boldsymbol{\Omega}_x} \Pr(C_k|\boldsymbol{x}) \mathbf{1}(\boldsymbol{x} \in C_k) \mathbf{1}(\Pr(C_k|\boldsymbol{x}) \neq \max_j \Pr(C_j|\boldsymbol{x}) \mathrm{p}(\boldsymbol{x}) d\boldsymbol{x}. \tag{2.11}$$

where $\mathbf{1}(\cdot)$ is the indicator function. That is,

$$\mathbf{1}(\mathcal{P}) \quad = \quad \begin{cases} 1 & \text{if } \mathcal{P} \text{ is true} \\ 0 & \text{otherwise.} \end{cases} \tag{2.12}$$

Note that the probability of error is a functional of the feature extractor $\mathcal{F}$. Since the feature space $\boldsymbol{\Omega}_x$ is determined by $\mathcal{F}$, the probability of error may be expressed in the form $\mathcal{E}_{Bayes}[\boldsymbol{\Omega}_x]$.

The decision rule of (2.10) is widely know as the "maximum a posteriori" (MAP) rule. This rule ensures that the classifier minimizes the probability of error. $\mathcal{E}_{Bayes}[\mathcal{F}]$ is the lowest error rate achievable by a classifier when using the feature extractor $\mathcal{F}$. Implementation of the MAP rule requires knowledge of the a posteriori probabilities $\Pr(C_k|\boldsymbol{x})$, which are usually not available directly. A classical method is to make use of the Bayes rule

$$\Pr(C_j|\boldsymbol{x}) = \frac{\Pr(C_j)\mathrm{p}(\boldsymbol{x}|C_j)}{\mathrm{p}(\boldsymbol{x})}, \tag{2.13}$$

and then implement the MAP rule in the following form:

$$a_{Bayes}(\boldsymbol{x}) = C_j \qquad \text{if} \qquad \Pr(C_j)\mathrm{p}(\boldsymbol{x}|C_j) > \Pr(C_k)\mathrm{p}(\boldsymbol{x}|C_k) \quad \text{for all } k \neq j, \tag{2.14}$$

which makes use of the prior probability $\Pr(C_k)$ and the class-conditional probability densities $\mathrm{p}(\boldsymbol{x}|C_k)$ in the optimal decision rule. Observing that $\Pr(C_j)\mathrm{p}(\boldsymbol{x}|C_j) = \Pr(\boldsymbol{x}, C_j)$ and that $\mathrm{p}(\boldsymbol{x})$ does not have a contribution in the decision process, the following decision rule are equivalent to (2.14):

$$a_{Bayes}(\boldsymbol{x}) = C_j \qquad \text{if} \qquad \Pr(\boldsymbol{x}, C_j) > \Pr(\boldsymbol{x}, C_k) \quad \text{for all } k \neq j. \tag{2.15}$$

The classifier C_{Bayes} performing the decision rule of (2.14) is referred to as *the Bayes classifier* and its error rate is the lowest error rate achievable on a given set of features. $\mathcal{R}_{Bayes} = \mathcal{R}(a_{Bayes}(\cdot)|\mathcal{F})$ refers to the partioning of the Bayes classifier, given the feature extractor $\mathcal{F}$.

2.3.2 Minimum error criteria for feature extractor design

Given a set of features (i.e., a pre-defined feature extraction $\mathcal{F}$), the Bayes classifier's error rate is the smallest error rate achievable on this feature set. This provides a framework for the optimal design of the classifier and the feature extractor.

Ideal feature extractor for minimum-error

The map rule, as given in equation (2.14), sets up the framework for ideal feature extractor design. That is, it suffices to choose as features the a posteriori probabilities $\Pr(C_k|\boldsymbol{s})$, thus producing a feature vector of dimensionality M. The dimensionality of this feature vector can be further reduced by observing that

$$\sum_{k=1}^{M} \Pr(C_k|\boldsymbol{s}) = 1,$$

thus, producing a feature vector of dimension $M-1$. The classifier is reduced to a simple "max" operator on the feature vector. This is the *ideal feature set for classification* [Fukunaga, 1972]. Note that the above feature extractor is realized without information loss. The above example illustrates the fact that when the classifier structure can be extremely simplified by making use of a "good" feature extractor.

Bayes error estimation for feature extractor design

Let us suppose that the Bayes error rate can be estimated on any feature set. Let us further make the assumption that the input data are noiseless (which is rather unrealistic in the speech recognition context). Since, any feature extraction method does not add new information in the input data, the best feature extractor is thus the identity function $\mathcal{F}_{\mathcal{I}}$ [Fukunaga, 1972] (no feature extraction algorithm is performed on the input-data). That is, given any feature extractor $\mathcal{F}$,

$$\mathcal{E}_{Bayes}[\mathcal{F}] \geq \mathcal{E}_{Bayes}[\mathcal{F}_{\mathcal{I}}]. \tag{2.16}$$

(2.16) may signify that raw input data should be fed to the Bayes classifier in order to achieve the smallest possible Bayes error. This approach can be taken if the input data are of lower dimension. For higher dimensional data, direct estimation of the Bayes error is usually infeasible. Furthermore, phenomena, such as the curse of dimensionality (as we shall see in later sections) requires the input data to be passed through a feature extraction process. The best feature extractor is therefore the one which yields the smallest Bayes error. That is, given competing feature extraction processes $\mathcal{F}_j$, for $j = \{1, \ldots, K\}$,

$$\text{choose } \mathcal{F}_k \qquad \text{if } k = \arg\min_j \mathcal{E}_{Bayes}[\mathcal{F}_j]. \tag{2.17}$$

(2.17) provides a straightforward method for optimal feature extraction design, provided that there is access to the Bayes classifier performance. This means re-designing the feature extractor after each test along the lines provided by the Bayes minimum error criteria, and discarding the "bad" feature extractor. In practice, this approach cannot be carried out simply because the Bayes error

is usually not available and should be estimated. The accuracy of the estimation depends on the training features, which leads to a vicious circle [Duda and Hart, 1973]. Furthermore, in certain applications such as speech recognition, estimating the Bayes error costs the design of an overall system. Consequently, the feature extractor is usually designed by other means, mainly based on EXFE techniques, as described in the next section.

2.4 Statistical Feature Extraction

In the previous section, it was seen that the Bayes error is the best criteria to use in the design of a feature extractor. In most cases encountered, such an approach cannot be carried out either because of the dimensionality of the data or because an estimate of the Bayes error is not available. In that case, one would like to maximize the "quality" of features, even when a direct estimate of the Bayes error is not available. Various literatures [Duda and Hart, 1973; Fukunaga, 1972], describe standard techniques for pre-processing data and performing feature extraction, without direct access to the Bayes error estimate, but only by making use of the information derived by performing statistical analysis of exemplar data $\mathcal{S}$, drawn from the input space $\boldsymbol{\Omega}_s$, according to a certain probability $\Pr(\boldsymbol{s})$. These methods are referred to as statistical feature extraction and are part of the EXFE techniques.

Statistically-based feature extraction provides a wide range of methods which can be applied across various pattern recognition tasks, and which are sometimes quite different from each other. Statistically-derived EXFE can further be sub-divided into *non-discriminant* feature extraction processes and discriminant feature extraction processes. Non-discriminant feature extraction means that the input data $\boldsymbol{s}$ is treated without explicit knowledge of the class information of the data. It is a general process that can be run on the input data for simple purposes, such as dimensionality reduction or normalization.

Usually, the non-discriminant part and discriminant part are intertwined, without clear boundaries between them. Very often also, the feature extraction is simply reduced to the non-discriminant part. Even if this approach is clearly sub-optimal, it may be satisfactory for simple classification problems.

2.5 Non-Discriminant Feature Extraction Techniques

This section describes a few non-discriminant feature extraction approaches, viewed as a form of pre-processing that are used in various pattern recognizers, such as image processing, character recognition and speech processing. As said before, pre-processing is a sub-process within the feature extraction process, which does not assume the class-information of the data and whose purpose is mainly to reduce the dimensionality of the data and/or perform some sort of data normalization. Thus, pre-processing enables the achievement of a *compact* feature extractor. Within this section, the standard methods that are applied across various pattern recognizers are reviewed.

Suppose we are given a data set $\mathcal{S} = \{s_n\}$ of size N, for $n = \{1, \ldots, N\}$, where $\mathcal{S}$ is drawn from $\boldsymbol{\Omega}_s$. The feature extractor transforms $\mathcal{S}$ by performing the mapping $\mathcal{X} = \mathcal{F}(\mathcal{S})$, such that the resulting features must have a lower-dimensionality without loss of the general specificity of the data set $\mathcal{S}$. The maximization property of the feature extractor requires that the features should include as much information as possible. However, reducing the feature dimensionality is advisable for two main reasons. First, a lower-dimensional feature reduces the complexity of the back-end system and is suitable for real-time processing. Second, the curse of dimensionality puts a boundary on the dimension of the feature-vector, given a data set.

2.5.1 The curse of dimensionality

When designing a recognizer, one is given a set of design data taken from the input space $\boldsymbol{\Omega}_s$. Given the task at hand, one is tempted to include as much information as possible in the features so as to produce as good a feature as possible. For instance, in the vowel recognition task described in the introduction, it may be useful to choose as features the formant frequencies, the energies of the signal, the relative energies per band, the shift rate of the formant frequencies, the number of zero crossings per time unit, and more.

In practice, this approach is not efficient because the link between the quantity of training data and the dimension of the data influences the "sparcity" of the data [Bellman, 1961a]. It has been noted in various classification tasks that the performance of a system increases with the dimension p of the feature-sample and starts decreasing when p reaches a certain value p_{th}, which depends on the given set of data and is empirically determined. Consequently, the feature extractor designer is constrained to choose $p < p_{th}$. This phenomenon is known as the curse of dimensionality and can intuitively be grasped by the observation that the size of the data set must compensate for the higher dimension of the data.

2.5.2 Data normalization

It may happen that the range of values within the data vectors $\boldsymbol{s}$ is rather large. In that case, it is necessary to perform a kind of data rescaling, which puts data within a specific range. Among various methods available, we here describe a simple data normalization process. For an input data $\boldsymbol{s}_n = [s_{n,1}, s_{n,2}, \ldots, s_{n,d}]^{\mathrm{T}}$, a typical rescaling process performs a normalizing of each data $\boldsymbol{s}_n$, so that the data may have uniform variance. Let $\overline{\boldsymbol{s}}$ and $\boldsymbol{\Sigma}_s = (\Sigma_{ij})$ be the mean and variance of the data set,

$$\overline{\boldsymbol{s}} = \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{s}_n, \tag{2.18}$$

$$\boldsymbol{\Sigma}_s = \frac{1}{N-1} \sum_{n=1}^{N} (\boldsymbol{s}_n - \overline{\boldsymbol{s}})(\boldsymbol{s}_n - \overline{\boldsymbol{s}})^{\mathrm{T}} \tag{2.19}$$

for $n \in \{1, \ldots, N\}$, where $\boldsymbol{s}_n$ is the n-th input pattern. A simple feature normalization can be obtained by

$$x_i = \frac{s_{n,i} - \overline{s}_i}{\Sigma_{ii}}, \tag{2.20}$$

producing a given feature set which has zero means and a unit standard deviation.

The procedure above is a simple implementation of a more general linear transformation process known as "whitening", which performs

$$\boldsymbol{x}_n = \boldsymbol{\Lambda}^{-1/2} \boldsymbol{T}^{\mathrm{T}} (\boldsymbol{s}_n - \overline{\boldsymbol{s}}) \tag{2.21}$$

where $\boldsymbol{T}$ is the matrix form by the eigenvectors of $\boldsymbol{\Sigma}_s$ and $\boldsymbol{\Lambda}$ is the diagonal matrix of the eigenvalues [Fukunaga, 1972].

2.5.3 Principal component analysis

Linear transformation of the data is a standard technique for data pre-processing. The feature extraction $\mathcal{F}$ process is represented as matrix $\boldsymbol{T}$, which performs the following transformation:

$$\boldsymbol{x} = \boldsymbol{T}\boldsymbol{s}, \tag{2.22}$$

where $\boldsymbol{x} = [x_1, x_2, \ldots, x_p]^{\mathrm{T}}$ is the corresponding feature-vector to the input vector $\boldsymbol{s}$.

To keep as much information as possible while reducing the dimensionality of the data by the transformation $\boldsymbol{T}$, one interesting criteria is that the resulting features should be as less correlated as possible. Correlated features means that somehow the same information is duplicated within the input data. Consequently, producing decorrelated features realizes the squeezing of meaningful information into a few parameters. An estimate of the degree of correlation is given by the covariance matrix of the data.

Let $\boldsymbol{\Sigma_S} = (\sigma^s_{ij})$ be the covariance matrix of the input data. The dependence between the data-component s_i and s_j is estimated by σ^s_{ij}. $\sigma^s_{ij} = 0$, for $i \neq j$ means that the features of index i and j are independent, thus carrying non-redundant information. The "importance" of the information carried by the component s_i is measured by σ^s_{ii}. The ideal case is to find a transformation which results in a diagonal covariance matrix. Such a transformation can be obtained by the principal component analysis (PCA). PCA is a special case of a broader feature transformation method called the Karhunen-Loevé expansion method. Hence, it is sometimes referred to as the *discrete Karhunen-Loevé* expansion. The PCA method is described below.

Let $\boldsymbol{\Sigma_x}$ be the covariance matrix of the corresponding features set. $\boldsymbol{\Sigma_S}$ and $\boldsymbol{\Sigma_x}$ are linked by the following relation:

$$\boldsymbol{T}\boldsymbol{\Sigma_S}\boldsymbol{T}^{\mathrm{T}} = \boldsymbol{\Sigma_x}. \tag{2.23}$$

Since $\boldsymbol{\Sigma_S}$ is a positive definite matrix, its eigenvectors form a basis of the original space spanned by $\boldsymbol{s}$. For $\boldsymbol{\Sigma_x}$ to be a diagonal matrix, simply choose as $\boldsymbol{T}$ the matrix which is formed by the eigenvectors of $\boldsymbol{\Sigma_S}$. The PCA procedure is as follows:

1. Find the eigenvalue λ_i of the covariance matrix $\boldsymbol{\Sigma_S}$ by finding the root of the characteristic polynomial defined as

$$\mathrm{P}(\lambda) = \det\{\boldsymbol{\Sigma_S} - \lambda \boldsymbol{I}\}$$

where $\boldsymbol{I}$ is the unit matrix of the same dimension as $\boldsymbol{\Sigma_S}$.

2. Find the corresponding eigenvectors $\boldsymbol{v}_i$ by solving in $\boldsymbol{v}$, the equation $\boldsymbol{\Sigma_S v} = \lambda_i \boldsymbol{v}$, for $i \in \{1, \ldots, d\}$.

3. Replace each $\boldsymbol{v}_i$ by $\frac{\boldsymbol{v}_i}{\|\boldsymbol{v}_i\|}$ so that they form an orthonormal base.

4. The transform matrix is simply formed from the eigenvectors: $\boldsymbol{T} = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_p]$.

The eigenvalues λ_i are the variances of the features x_i. A higher variance is an estimate of the "usefulness" of the information carried by the corresponding features. Consequently, a dimensionality reduction is achieved by only selecting the p eigenvectors, corresponding to the p highest eigenvalues. This process can be viewed as a projection of the features onto the space spanned by the chosen eigenvectors.

The geometrical meaning of the principal component analysis process is that it realizes a rotation of the system of coordinates so that the new system is aligned along the direction of maximum variances of the data, which is represented by the eigenvectors. It has been shown [Fukunaga, 1972] that principal component analysis is the best projection, in the Mean Squared Error sense or the Minimum Entropy sense (if the data obey a normal distribution), to a lower dimensional feature space.

In speech recognition, for better performance, various type of features are usually included in the feature vector, which is likely to lead to higher computational cost. In such cases, the PCA transformation has been used to reduce the dimensionality of the speech feature vectors and to obtain less correlated features [Rajaseharan and Doddington, 1985]. A comparative study made by Paliwal [Paliwal, 1992] has shown that in certain cases the PCA transformation is capable of increasing performance.

2.5.4 Independent component analysis

In the derivation of the PCA transformation, only second-order statistics (covariance matrix) are used. Second-order statistics are appropriate when data obey a Gaussian distribution. However, for non-Gaussian data, useful information is derived from higher order statistics [Fukunaga, 1972]. Using PCA in this case is not the optimal approach for dimensionality reduction. This has led to the use of non-linear PCA techniques, usually used within the framework of artificial neural network [Bishop, 1995].

Recently, the Independent Component Analysis (ICA) has been proposed as an alternative to PCA, well-suited to non-Gaussian data [Comon, 1994]. The ICA transformation focuses on expressing the input data as a linear combination of statistically independent components, and has

been shown to be a more meaningful representation than PCA [Karhunen *et al.*, 1997]. The main drawback of the ICA technique is that, unlike the PCA transformation, estimation of the basis vectors is not straightforward. ICA has been mainly applied to signal source separation and image processing. See [Comon, 1994] for further readings regarding PCA.

2.6 Discriminant Feature Extraction Techniques

In the previous section, the feature extraction process was reduced to a pre-processing system whose role was to normalize and/or reduce the dimensionality without consideration of the classification process. In this section, the feature extractor tries to find the right parameters which may be useful for accurate classification. Thus, the feature extractor takes the form $\mathcal{F}(\boldsymbol{s}|C)$, where C represent knowledge of the classification task to be performed. As said before, the information given by C sometimes takes the form of expertise and/or is derived trough statistical analysis of exemplar data $\mathcal{S}$ taken from the original input space $\boldsymbol{\Omega}_s$, according to a given probability $\mathrm{p}(\boldsymbol{s})$.

Incorporation of expertise in the design of the feature extractor highly depends on the framework, within which, the pattern recognizer is performed. In this section, the focus is on the use of data statistics for including class knowledge in the design of a feature extractor. In particular, when using a linear transformation as feature extractor.

A straightforward manner to incorporate class knowledge is to let this knowledge be taken from the classification process. This also provides a straightforward way to *test* the accuracy of the feature extraction process and select the most appropriate one, since the *probability of error* is the best criteria for choosing a feature extractor. However, in general, an analytical form of the probability of error as a function of the number of features, is not available, which would allow a straightforward optimal design of the feature extraction process. The probability of error must be estimated by 1) test samples and 2) a classifier. The classifier is first optimized on a training set, such as $\mathcal{S}$. Then, the percentage of misclassified test samples, called the classifier *empirical error rate*, is taken as an estimate of the probability of error. For the test to be valid, the training and testing set must be statistically independent.

Estimating the probability of error is sometimes prohibitive: one should first design a classifier structure on which to test the data and rerun the process when the feature set is not satisfactory. Instead, indirected estimation is usually performed, based on the data set, by defining "class-separability criteria", which ensures that the features are somehow "clustered" in the feature space. The *a priori* assumption is that the clustered features are somehow easy to be classified. Below, the main framework of linear feature extraction which uses a "class-separability" measure based on the data set is reviewed. It is supposed that we are given a data set $\mathcal{S} = \{\mathcal{S}_j\}$, for $j \in \{1, \ldots, M\}$, where $\mathcal{S}_j \in \boldsymbol{\Omega}_s$ is labeled as belonging to C_j. M is the number of classes. C_j contains N_j samples, and the number of data in the whole data set is $N = \sum_{j=1}^{M} N_j$. The feature extractor is required to transform the data set $\mathcal{S}$ into a feature set $\mathcal{X} = \{\mathcal{X}_j\}$, for $j \in \{1, \ldots, M\}$, where $\mathcal{X}_j = \mathcal{F}(\mathcal{S}_j)$. We suppose for simplicity that the dimensionality of $\mathcal{S}$ is d and the dimensionality of $\mathcal{X}$ is p.

2.6.1 Feature selection

Separability-measure-based feature extraction design uses a cost J to select the best feature extractor. The cost is assumed to provide an estimation of the corresponding feature "quality". Again the best features are those that yield the lowest probability of error when using an equivalent Bayes classifier. However, the estimate of the probability of error may be extremely prohibitive: one has to estimate the posterior probability in the given feature set using either a classifier or any non-parametric method available. Instead, separability criteria based on available data maybe a faster test.

Having an available separability criterion J may enable one to perform a *feature selection* process. "Feature selection" is an alternative term which is widely encountered in the pattern recognition area the term feature selection is sometimes used without serious concern regarding their meaning). Feature selection usually refers to the process of reducing the dimension of the feature vectors by simple truncation or a pick up-and-discard approach: from the original feature-vectors, discard feature-values believed to be irrelevant to recognition. In brief, given an input pattern of dimension d, feature selection chooses the "best" subset of size p, from the set of features, so as to maximize a criteria J.

Feature selection usefulness is met when one tries to minimize the overall cost of acquiring a pattern. Consequently, a mechanical procedure which automatically extracts the p necessary features out of d is usually carried out. Andrews [Andrews, 1968] derives a mechanical process which picks up features at random and chooses them statistically. This method was not successful because the probability of finding good features is extremely low. Moreover, this method is computationally expensive: Cover and Van Campenhout [Cover and Van Campenhout, 1977] have shown that to determine the p features of d input patterns, one needs to examine all possible subsets of size p. Some heuristics may be used but with no guarantee of optimality. [Devijvier and Kittler, 1982] provides a good survey on search algorithms for feature selection.

The feature selection approach is usually taken within problems where data are redundant and when one does not need sophisticated data pre-processing. However, for most pattern recognition tasks involving sophisticated data-processing as well as a high number of data, such a simple approach to feature extraction is not efficient.

2.6.2 Separability criteria for discriminant analysis

Statistical discriminant analysis makes use of specific notions such as the total covariance matrix, within-class covariance matrix, and between-class covariance matrix to describe the clustering and scattering properties of a given set of features.

Let the sample mean and covariance of feature set $\mathcal{X}_j$ be given by

$$\overline{\boldsymbol{x}}_j = \frac{1}{N_j} \sum_{\boldsymbol{x} \in \mathcal{X}_j} \boldsymbol{x} \tag{2.24}$$

$$\boldsymbol{\Sigma}_j = \sum_{\boldsymbol{x}\in\mathcal{X}_j} (\boldsymbol{x}-\overline{\boldsymbol{x}}_j)(\boldsymbol{x}-\overline{\boldsymbol{x}}_j)^{\mathrm{T}}. \tag{2.25}$$

The total covariance matrix is simply the variance across the whole feature set and is defined as

$$\boldsymbol{\Sigma}_T = \frac{1}{N}\sum_{\boldsymbol{s}}(\boldsymbol{x}-\overline{\boldsymbol{x}})(\boldsymbol{x}-\overline{\boldsymbol{x}})^{\mathrm{T}}, \tag{2.26}$$

where $\overline{\boldsymbol{x}}$ is the mean over the whole set of data $\mathcal{X}$. The within-class covariance matrix attempts to describe the scatter of features within their specific classes, and is defined as

$$\boldsymbol{\Sigma}_W = \sum_{j=1}^{M} \boldsymbol{\Sigma}_j. \tag{2.27}$$

Contrary to the within-class covariance matrix, the between-class covariance matrix expresses the scatter of the group/class of features, where a group/class of features is represented by its mean. The between-class covariance matrix is defined as

$$\boldsymbol{\Sigma}_B = \boldsymbol{\Sigma}_T - \boldsymbol{\Sigma}_W \tag{2.28}$$

$$= \sum_{j=1}^{M} N_j(\boldsymbol{x}_j-\overline{\boldsymbol{x}})(\boldsymbol{x}_j-\overline{\boldsymbol{x}})^{\mathrm{T}} \tag{2.29}$$

$$\tag{2.30}$$

In discriminant analysis, various criteria, which are functions of the above matrices, are used to grasp the "quality" of data in terms of separability. These criteria are supposed to be an increasing function of the "size" of the between-class covariance matrix while decreasing with the "size" of the within-class covariance matrix. "size" refers to a scalar function of the corresponding matrices such as the determinant or the trace. Typical criteria, as provided by [Fukunaga, 1972], are listed below.

1. $J_1 = \mathrm{Tr}(\boldsymbol{\Sigma}_b^{-1}\boldsymbol{\Sigma}_a)$
2. $J_2 = \frac{\det(\boldsymbol{\Sigma}_a)}{\det(\boldsymbol{\Sigma}_b)}$
3. $J_3 = \mathrm{Tr}(\boldsymbol{\Sigma}_a) - \mu\mathrm{Tr}(\boldsymbol{\Sigma}_b)$, where μ is a Lagrange multiplier.
4. $J_4 = \frac{\mathrm{Tr}(\boldsymbol{\Sigma}_a)}{\mathrm{Tr}(\boldsymbol{\Sigma}_b)}$,

where $\{\boldsymbol{\Sigma}_a, \boldsymbol{\Sigma}_b\}$ is one of the pairs $\{\boldsymbol{\Sigma}_B, \boldsymbol{\Sigma}_W\}$, $\{\boldsymbol{\Sigma}_B, \boldsymbol{\Sigma}_T\}$, and $\{\boldsymbol{\Sigma}_W, \boldsymbol{\Sigma}_T\}$.

2.6.3 Linear discriminant analysis

It was shown that the PCA transformation finds the direction of maximum variance. However, the PCA transformation does not consider discrimination. For classification, we are more interested in a linear transformation which performs discrimination. Such a transformation should preserve the class separability of the data and find a suitable system of coordinates in which the class are presented as clusters.

Linear discriminant analysis (LDA), which is described here, transforms the input data $\mathcal{S}$, using a linear transformation $\boldsymbol{T}$ which maximizes one of the above J criteria in the feature space.

A simple case

Let us suppose momentarily that $\boldsymbol{T}$ is simply a projection onto a one-dimensional space. The purpose is to find a one-dimensional subspace which realizes separability of the original data, according to a criterion to be defined later. The data-vector $\boldsymbol{s}$ is transformed into

$$x = \boldsymbol{T}\boldsymbol{s}. \tag{2.31}$$

The sample mean and covariance of $\mathcal{S}_j$ are given by

$$\overline{\boldsymbol{s}}_j = \frac{1}{N_j} \sum_{\boldsymbol{s} \in \mathcal{S}_j} \boldsymbol{s}, \tag{2.32}$$

$$\boldsymbol{\Sigma}_j(\boldsymbol{s}) = \sum_{\boldsymbol{s} \in \mathcal{S}_j} (\boldsymbol{s} - \overline{\boldsymbol{s}}_j)(\boldsymbol{s} - \overline{\boldsymbol{s}}_j)^{\mathrm{T}}. \tag{2.33}$$

For clarity, let us further simplify the problem into a 2-class problem. In the ideal case, two categories are separable if their probability density functions do not highly overlap. In a one-dimensional case, this can be achieved if the mean $\overline{x}_1 = \boldsymbol{T}\overline{\boldsymbol{s}}_1$ and $\overline{x}_2 = \boldsymbol{T}\overline{\boldsymbol{s}}_2$ of the two classes, in feature space, are further apart and/or the data are not too scaterred. i.e, relatively small variances σ_1^2 and σ_2^2.

An estimate of this class separability is given by the *Fisher's linear discriminant*, defined by

$$J = \frac{(\overline{x}_1 - \overline{x}_2)^2}{\sigma_1^2 + \sigma_1^2}. \tag{2.34}$$

The greater the Fisher's discriminant, the higher the class separability as expressed by the corresponding features. Let us express the Fisher's discriminant as a function of $\boldsymbol{T}$. It is straightforward that

$$(\overline{x}_1 - \overline{x}_2)^2 = \boldsymbol{T}\boldsymbol{\Sigma}_B(\boldsymbol{s})\boldsymbol{T}^{\mathrm{T}}, \tag{2.35}$$

$$\sigma_1^2 + \sigma_2^2 = \boldsymbol{T}\boldsymbol{\Sigma}_W(\boldsymbol{s})\boldsymbol{T}^{\mathrm{T}} \tag{2.36}$$

where

$$\boldsymbol{\Sigma}_B(\boldsymbol{s}) = (\overline{\boldsymbol{s}}_1 - \overline{\boldsymbol{s}}_2)(\overline{\boldsymbol{s}}_1 - \overline{\boldsymbol{s}}_2)^{\mathrm{T}}, \tag{2.37}$$

$$\boldsymbol{\Sigma}_W(\boldsymbol{s}) = \boldsymbol{\Sigma}_1(\boldsymbol{s}) + \boldsymbol{\Sigma}_2(\boldsymbol{s}), \tag{2.38}$$

are the between-class covariance matrix and within-class covariance matrix, respectively.

Consequently, the Fisher linear discriminant criterion takes the form

$$J(\boldsymbol{T}) = \frac{\boldsymbol{T}\boldsymbol{\Sigma}_B(\boldsymbol{s})\boldsymbol{T}^{\mathrm{T}}}{\boldsymbol{T}\boldsymbol{\Sigma}_W(\boldsymbol{s})\boldsymbol{T}^{\mathrm{T}}}, \tag{2.39}$$

which is the classical expression of a generalized Raleigh quotient, quite well-known in mathematical physics [Duda and Hart, 1973]. It is of common knowledge that J is maximized if $\boldsymbol{T}$ satisfies

$$\boldsymbol{\Sigma}_B(\boldsymbol{s})\boldsymbol{T} = \lambda \boldsymbol{\Sigma}_W(\boldsymbol{s})\boldsymbol{T} \tag{2.40}$$

for a parameter λ. If $\boldsymbol{\Sigma}_W(\boldsymbol{s})$ is nonsingular, (2.40) is widely known in the form

$$\boldsymbol{\Sigma}_W^{-1}(\boldsymbol{s})\boldsymbol{\Sigma}_B(\boldsymbol{s})\boldsymbol{T} = \lambda\boldsymbol{T}, \tag{2.41}$$

which can be re-written as

$$\boldsymbol{\Sigma}_W^{-1}(\boldsymbol{s})(\overline{\boldsymbol{s}_1} - \overline{\boldsymbol{s}_2}) = \tilde{\lambda}\boldsymbol{T}, \tag{2.42}$$

since $\boldsymbol{\Sigma}_B(\boldsymbol{s})\boldsymbol{T}$ is in the same direction as $(\overline{\boldsymbol{s}}_1 - \overline{\boldsymbol{s}}_2)$. (2.42) means that the solution is the line-space of direction of $\boldsymbol{\Sigma}_W^{-1}(\boldsymbol{s})(\overline{\boldsymbol{s}}_1 - \overline{\boldsymbol{s}}_2)$, which is a classical result.

Multiple-class linear discriminant analysis

Here we consider the case of an $M(>2)$-class problem. If $\boldsymbol{T}$ is the projection into a one-dimensional-space, the Fisher's discriminant (known in this context as the F ratio), is expressed as

$$F = \frac{\frac{1}{M-1}\sum_{j=1}^{M}(x_j - \overline{x})^2}{\frac{1}{M}\sum_{j=1}^{M}\sigma_j^2} \tag{2.43}$$

$$= \frac{\text{Variance of the means}}{\text{Mean of variances}} \tag{2.44}$$

where $\overline{x}_j$ and σ_j represents the mean and variance of class $\mathcal{X}_j$ and the mean over the whole data set, respectively.

The F ratio is clearly sufficient for evaluating a single feature. When more than one dimension is involved, which is usually the case in more complex problems, the F ratio is far from being satisfactory. One method may be to rank each feature using the F ratio and then pick the best ones, based on this ranking. However, this method is rather risky since the basic assumption is that features are uncorrelated. Decorrelated features may be achieved by the PCA technique in an earlier stage. However, it has been shown that the PCA technique does not maximize class separability. A criterion somehow equivalent to the F ratio is clearly in need. This criterion is provided by simply extending the notions of within-class covariance matrix and between-class covariance matrix to higher dimensional features.

It is straightforward to show that, in the feature space, the within-class covariance matrix $\boldsymbol{\Sigma}_W(\boldsymbol{x})$ and between-class covariance matrix $\boldsymbol{\Sigma}_B(\boldsymbol{x})$ are given by

$$\boldsymbol{\Sigma}_W(\boldsymbol{x}) = \boldsymbol{T}\boldsymbol{\Sigma}_W(\boldsymbol{s})\boldsymbol{T}^{\mathrm{T}}, \tag{2.45}$$

$$\boldsymbol{\Sigma}_B(\boldsymbol{x}) = \boldsymbol{T}\boldsymbol{\Sigma}_B(\boldsymbol{s})\boldsymbol{T}^{\mathrm{T}}. \tag{2.46}$$

Again, $\boldsymbol{\Sigma}_W(\boldsymbol{s})$ and $\boldsymbol{\Sigma}_B(\boldsymbol{s})$ are the within-class covariance matrix and the between-class covariance matrix in the data space.

Among the various candidate criteria that could be optimized, the J_1 criteria, which was defined as

$$J_1(\boldsymbol{T}) = \mathrm{Tr}(\boldsymbol{\Sigma}_W^{-1}(\boldsymbol{x})\boldsymbol{\Sigma}_B(\boldsymbol{x})) \quad (2.47)$$

$$= \mathrm{Tr}\Big\{\left(\boldsymbol{T}\boldsymbol{\Sigma}_W(\boldsymbol{s})\boldsymbol{T}^{\mathrm{T}}\right)^{-1}\left(\boldsymbol{T}\boldsymbol{\Sigma}_B(\boldsymbol{s})\boldsymbol{T}^{\mathrm{T}}\right)\Big\}, \quad (2.48)$$

is usually a convenient choice because of the invariant properties of the trace function across linear transformation. In that case, the optimal transformation $\boldsymbol{T}$ is given by the eigenvalues of the matrix $\boldsymbol{\Sigma}_W^{-1}(\boldsymbol{x})\boldsymbol{\Sigma}_B(\boldsymbol{x})$ [Fukunaga, 1972]. Again, this is a classic result.

LDA has been extensively used in speech recognition as a convenient means to enhance the speech representation and reduce the dimensionality. This framework was initiated by [Hunt and Lefèbvre, 1986] for reducing the dimensionality of the feature vectors derived from a cochlear model. The method was later applied to transform the output of a filter bank based on the Mel scale (a perceptual frequency scale which attempts to mimic the Human discrimination of pitch). The technique, dubbed IMELDA for Integrated Mel-scale representation using LDA, was shown to perform better than competing speech representations [Hunt and Lefèbvre, 1989].

2.7 The Classification Paradigm

Given a feature extractor $\mathcal{F}$, the optimal classifier is the Bayes classifier. Consequently, the goal of classifier design is to achieve an equivalent Bayes classifier. This equivalence can be measured numerically by the error rate of the system, by performing a comparison with the Bayes minimum error rate.

Estimation of the Bayes error rate can be done via parametric approaches or non-parametric approaches from available design samples. The latter is essentially useful when the feature is of low-dimension. The Parzen window or k-nearest-neighbor classifier [Duda and Hart, 1973] are widely used non-parametric approaches to error rate estimation. For higher dimensional features or when the data set is processed on-line, the k-nearest-neighbor classifier and the Parzen window estimation can no longer be carried out. Instead, in most cases, a parameterized structure is chosen as a classifier and the estimation of the parameters is done through design data. Bayes error estimation, through classifier design, can be divided into two groups:

- Methods that rely on estimating a posteriori probabilities $\Pr(C_j|\boldsymbol{x})$.
- Methods that focus on estimating the Bayes' classifier region $\mathcal{R}_{Bayes}$.

The first group relies on the design samples $\mathcal{X}_j$ of class C_j to estimate a posteriori probabilities $\Pr(C_j|\boldsymbol{x})$. This training method constitutes the family of probability estimation techniques. The second group is the family of discriminative training approaches which relies on the observation that accurate estimation of the Bayes classifier region can result into the implementation of a classifier equivalent to the Bayes classifier, even though a posteriori probabilities are unknown. The

observation is that application of the Bayes classifier decision in (2.10) only requires knowledge of the class-menbership of the token but not knowledge of the class properties. Discriminative training approaches tries to estimate the Bayes region $\mathcal{R}_j$ by making use of information from all competing classes, thus, estimating the boundaries between all classes C_k and C_j for $k \neq j$. Discriminative training, which is one of the focus of this report is best described in chapter 4 through the eyes of a recent framework called the Minimum Classification Error/Generalized Probabilistic Descent method (MCE/GPD).

2.7.1 Discriminant functions for classifier design

Implementation of the Bayes classifier requires the knowledge of prior probabilities and class-conditional probabilities. However, in practical pattern recognition system, those probabilities are rarely known, meaning that the Bayes rule cannot be used directly. However, if knowledge of the situation is available in the form of *design data*, those data can be used to estimate the classifier. A convenient way to represent the classifier is to make use of the concept of discriminant function [Duda and Hart, 1973].

The classifier C_Λ is parameterized by a parameter set Λ and characterized by a set of discriminant functions $\{g_i(\boldsymbol{x};\Lambda)\}_{i\in\{1,...,M\}}$. $g_i(\boldsymbol{x};\Lambda)$ indicates the degree to which a given token $\boldsymbol{x}$ belongs to class C_i. The corresponding decision rule is

$$a(\boldsymbol{x};\Lambda) = C_i \text{ if } g_i(\boldsymbol{x};\Lambda) = \max_k g_k(\boldsymbol{x};\Lambda). \tag{2.49}$$

Λ represents the set of parameters that has to be adjusted to achieve the efficiency of the decision rule. That is, tuning of the classifier is achieved by choosing Λ so as to optimize a criterion believed to lead to optimal classification. For convenience, we gather the discrimination functions into a *discriminant vector* $\boldsymbol{g}(\boldsymbol{x};\Lambda) = [g_1(\boldsymbol{x};\Lambda), g_2(\boldsymbol{x};\Lambda), ..., g_M(\boldsymbol{x};\Lambda)]^{\mathrm{T}}$.

As said earlier, two approaches are usually taken for estimating the classifier. The first approach uses $g_i(\boldsymbol{x};\Lambda)$ as an estimate of the *a posteriori* probability function of C_i, given $\boldsymbol{x}$. This approach focuses on *modeling* each category. In this case, the decision of (2.49) leads to a minimum error classifier, if the true probability is obtained [Duda and Hart, 1973; Fukunaga, 1972].Due to substantial development of estimation techniques, this approach is widely used in pattern recognition systems. Again, for this approach to be valid, two requirements should met 1) the form of probabilities should be known. That is, the model should be in complete agreement with the source 2) design resources should be available to achieve unbiased estimation. In most practical situations however, the form of the probabilities distribution is unknown.

The second method uses $g_i(\boldsymbol{x};\Lambda)$ as a *possibility* measure; the discriminant function is not required to be a probability but simply a means to achieve a class-membership decision. This approach was extensively studied in the 1960's, using linear discriminant function classifiers [Duda and Hart, 1973]. More recently, distances to references of a given category [McDermott and Katagiri, 1991], or log-likelihood have been used as discriminant functions.

The second approach is also practical for implementing a discriminative training of the classifier. Discriminative training focuses on finding differences between categories instead of trying to fit a model to a given category. Discriminative training is based on the observation that *the application of the Bayes rule only requires knowledge of the category which has the maximum a posteriori probability but not the actual value of this probability.* Within this approach, the concept of loss is usually used for evaluating the performance of the system. Design is done through the minimization of the loss function.

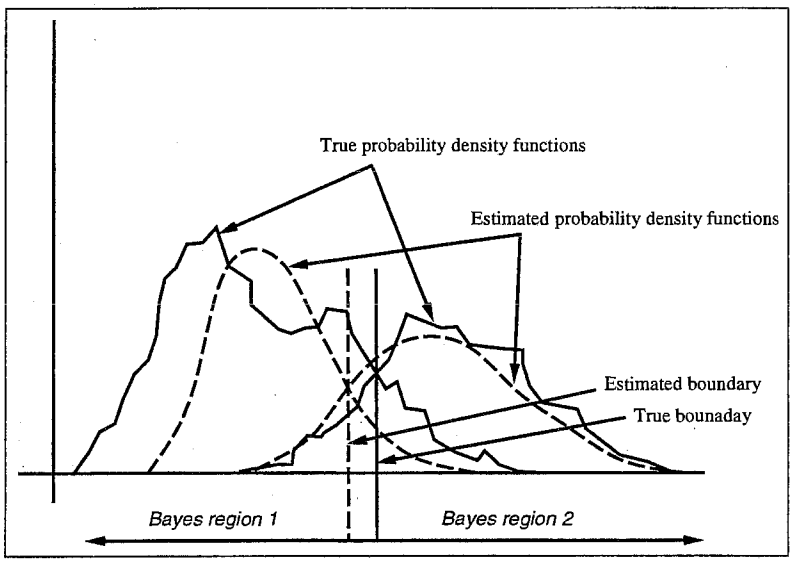


Figure 2.2: Estimation of probabilities density functions for two classes. Discriminative approaches focus on estimating Bayes region boundaries.

Fig. 2.2 illustrates the two approaches. In this figure, estimating the probabilities failed to accurately separate the two classes due the difficulty to accurately modeling the probability densities. The discriminative training approach attempts to directly estimate the boundary between the two Bayes regions, which can be achieved without the discrimination functions being of a probability form.

2.8 Probability Estimation-based Training

Probability estimation techniques try to estimate a posteriori probabilities

$$\Pr(C_j|\boldsymbol{x}) = \frac{\Pr(C_j)\mathrm{p}(\boldsymbol{x}|C_j)}{\mathrm{p}(\boldsymbol{x})}.$$

Since $\mathrm{p}(\boldsymbol{x})$ does not contribute to the Bayes decision, focus is on estimating $\Pr(C_j)$ and $\mathrm{p}(\boldsymbol{x}|C_j)$. In general, prior probabilities $\Pr(C_j)$ can easily be tackled. In speech recognition for instance, the prior probabilities are given by a language model, which provides the prior probability of words, in a word-based recognition model or phoneme in the phoneme-based recognition model.

Estimation of the class-conditional densities $p(\boldsymbol{x}|C_j)$ is more challenging and can be approached through parametric and non-parametric approaches. The non-parametric techniques try to estimate $p(\boldsymbol{x}|C_j)$ as a *function*, using design samples, while the parametric techniques assume a model of the probability function and then estimate the parameters of the model. The most popular non-parametric techniques are those based on Parzen window estimation and k-nearest neighbor. Duda [Duda and Hart, 1973] and Fukunaga [Fukunaga, 1972] provide a good coverage of these techniques. Here, we focus on parametric estimation techniques.

Probability density estimation derives an error function E, which is function of the discrimination functions g_i and whose minimization yields the discrimination function to estimate the class-condition probabilities. Maximum Likelihood Estimation (MLE), which is the most popular probability estimating method is described below. Other methods include Bayesian inference, which considers the classifier parameters Λ as a random variable and takes into account higher order statistics (at least seconf order) of the models. Bayesian inference has been applied to speech recognition and speaker adaptation [Gauvain and Lee, 1991; Huo and Lee, 1997].

2.8.1 Maximum likelihood estimation

Maximum likelihood estimation (MLE) is perhaps the most popular approach to class-conditional probability estimation. Its use is widespread in the speech recognition community because, coupled with the Expectation-Maximization algorithm (EM), it offers a simple and efficient method to classifier design. Here, the MLE philosophy is described. For more details concerning MLE, the reader is referred to [Duda and Hart, 1973; Fukunaga, 1972]. In the MLE estimation, the form of the class-probability is usually assumed, and only the parameters of the model (i.e., mean, variance) are unknown. However, use of discrete probabilities is also reported in the litterature [Fukunaga, 1972]. Discrete probability distributions do not need for an assumed form of the probabilities, which is indeed a risky assumption, but do require a large amount of training data for accurate estimation. Nowadays, however, in the speech recognition area, a form of the probability density function is usually assumed. Mixtures of Laplacian probability functions and mixtures of Gaussian probability functions have been extensively used [Huang *et al.*, 1990].

Let the unknown parameters of the class C_i be represented as Λ_i. The class-conditional probability density function is thus $p(\boldsymbol{x}|C_i;\Lambda_i)$. Consequently, the discriminant function are defined as

$$g_i(\boldsymbol{x};\Lambda_i) = p(\boldsymbol{x}|C_j;\Lambda_j)\Pr(C_j). \tag{2.50}$$

Given the set of designed data $\mathcal{X}_i = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_{N_i}\}$ for category C_i, the likelihood is defined as

$$\Pr(\mathcal{X}_i;\Lambda_i) = \prod_{k=1}^{M} p(\boldsymbol{x}_k|C_i;\Lambda_i). \tag{2.51}$$

This likelihood corresponds to the parameterized probability to producing the training vectors $\mathcal{X}_i$. Note that (2.51) relies on the assumption that the feature-data are independent from each other. The maximum likelihood estimate of these parameters is the value of Λ_i which maximizes the

likelihood $\Pr(\mathcal{X}_i; \Lambda_i)$. In practice, for easier manipulation, it is more convenient to consider the negative log of the likelihood as the error function to minimize, which gives

$$\mathrm{MLE}_{N_i}(\Lambda_i) = -\log(\Pr(\mathcal{X}_i; \Lambda_i)) = -\sum_{k=1}^{N_i} \log(\mathrm{p}(\boldsymbol{x}_k|C_i; \Lambda_i)). \tag{2.52}$$

$\mathrm{MLE}_{N_i}(\Lambda_i)$ is the empirical objective function, defined on a finite number of data, to be minimized. The MLE-based objective function is calculated over the whole feature space and is a functional of the parameter of category C_i:

$$\mathrm{MLE}(\Lambda_i) = -\int_{\boldsymbol{\Omega}_x} \log(\mathrm{p}(\boldsymbol{x}|C_i; \Lambda_i)\mathrm{p}(\boldsymbol{x})d\boldsymbol{x}. \tag{2.53}$$

Nadas [Nadas, 1983] has shown that if the assumptions made in the model about the form of the true data distribution are correct and enough training data is available, then the maximum likelihood estimate is the best estimate of the true parameters.

In speech recognition, the Expectation-Maximization (EM) re-estimation technique [Dempster *et al.*, 1977], is a widely used approach of MLE implementation, in particular when estimating mixture densities in the states of a hidden Markov model. In this context, the EM re-estimation takes the form of the Baum-Welch algorithm, or the Viterbi algorithm (see Chapter 3) [Lee, 1989; Huang *et al.*, 1990; Rabiner, 1989].

It is useful to compare maximum likelihood estimation with the discriminative training approach, which is the basis for most of the work presented in this report. The essential aspect of MLE is that it is targeted at *modeling* the data categories. In contrast, discriminative training is concerned with *separating* the categories.

MLE can generate an optimal classifier if the correct form of the probability is known and enough training data is available. In that situation, MLE can perfectly estimate the densities, thus enabling direct implementation of the Bayes decision rule. However, when the form of the distributions is not known (which is usually the case in practical applications), or when the amount of training data is insufficient, the resulting estimated model is not guaranteed to produce an equivalent Bayes classifier.

2.9 Discriminative Objective Functions

Although estimation of the posteriori probabilities could be done through statistical algorithms, the certainty of the optimality of the estimation is not guaranteed. For instance, in most cases encountered in the speech recognition field, the form of the probability density functions is not known and design data are sometimes limited.

Instead, given the set of design samples to be classified, we can focus on minimizing an objective function targeting accurate class separation. If class separation is achieved, (i.e., the class boundaries are accurately estimated), the estimated classifier may generate the same partitioning as the optimal Bayes classifier even though the corresponding probabilities functions are unknown.

A way to achieve this is to work out a minimizable objective function whose optimization implements discriminative training. Various objective functions have been proposed in the literature, including Mean-Squared-Error (MSE), Maximum Mutual Information (MMI) and the Kullback-Leiber divergence measure. Our focus is on the recently proposed Minimum Classification Error learning, which is explained in the Chapter 4. Here, we overview the MSE and MMI criteria.

2.9.1 Mean-squared error criterion

The MSE criterion is a widely error criterion in signal processing. An interesting feature of the MSE criterion is that it is both a discriminative training approach and a probability estimation procedure.

MSE as discriminant objective function

The discriminative properties of MSE are as follows. Each class C_i is represented by a vector $\boldsymbol{t}$ in the space spanned by the decision vector $\boldsymbol{g}(\boldsymbol{x};\Lambda)$. The MSE-based error, which corresponds to an input feature-pattern $\boldsymbol{x}$ (belonging to C_k), is

$$\ell_{MSE}(\boldsymbol{x};\Lambda) = \| \boldsymbol{g}(\boldsymbol{x};\Lambda) - \boldsymbol{t}_k \|^2 \tag{2.54}$$

$$= \sum_{i=1}^{M} (g_i(\boldsymbol{x};\Lambda) - t_{ki})^2, \tag{2.55}$$

where $\boldsymbol{t}_k = [t_{k1}, t_{k2}, ...t_{kM}]^{\mathrm{T}}$ is the target-vector, given the input feature $\boldsymbol{x}$. The empirical MSE-based objective function over the whole training set is defined as

$$\mathrm{MSE}_N(\Lambda) = \frac{1}{N} \sum_{n=1}^{N} \ell_{MSE}(\boldsymbol{x}_n;\Lambda). \tag{2.56}$$

The objective function is the expected MSE-based error over the feature-space. Its definition is

$$\mathrm{MSE}(\Lambda) = \int\int \ell_{MSE}(\boldsymbol{x};\Lambda)\mathrm{p}(\boldsymbol{x},C)d\boldsymbol{x}dC. \tag{2.57}$$

The MSE criterion can be viewed as a regression of class vector representatives $\boldsymbol{t}_k$ with the feature-vector $\boldsymbol{x}$. Thus, if the family of vector $\boldsymbol{t}_k$ are taken to be an orthonormal base, i.e.,

$$\boldsymbol{t}_j{}^{\mathrm{T}}\boldsymbol{t}_k = \delta_{jk}$$

then the MSE criterion can be viewed as creating clusters around the point $\boldsymbol{t}_k$, in the space spanned by $\boldsymbol{g}(\boldsymbol{x};\Lambda)$, where each cluster corresponds to a specific class. Consequently, the MSE criterion is discriminant. More discussion related to the MSE's discriminative capacities are provided in Chapter 4.

The mean squared error (MSE) criterion has found a widespread application in feed-forward neural network trained using "Back-propagation" [Hinton *et al.*, 1986]. Special architectures, well suited to speech processing have been proposed [Waibel *et al.*, 1987; Haffner *et al.*, 1991], with somehow better results than MLE trained classifiers.

MSE estimates a posteriori probabilities

The link between MSE and the Bayes error is indirect. However, it has been established that with sufficient parameters, minimizing the MSE makes $g_i(\boldsymbol{x};\Lambda)$ into an estimate of the a posteriori probabilities $\Pr(C_i|\boldsymbol{x})$ [Richard and Lippmann, 1991; Geman *et al.*, 1992]. The reasoning is the following: it is assumed that $N \rightarrow \infty$. The target corresponding to $g_i(\boldsymbol{x};\Lambda)$ is thus a random variable $\mathbf{t}_i$. The error in (2.57) is simply the expectation of the error and can be written as

$$\mathrm{MSE}(\Lambda) = \int\int \left[\sum_{i=1}^{M}(g_i(\boldsymbol{x};\Lambda)-\mathbf{t}_i)^2\right] \mathrm{p}(\mathbf{t}_i,\boldsymbol{x})d\mathbf{t}_i d\boldsymbol{x}. \tag{2.58}$$

It is straightforward to show that (2.58) can be re-written as

$$\mathrm{MSE}(\Lambda) = \int\int \left[\sum_{i=1}^{M}(g_i(\boldsymbol{x};\Lambda)- < \mathbf{t}_i|\boldsymbol{x} >)^2\right] \mathrm{p}(\mathbf{t}_i|\boldsymbol{x})\mathrm{p}(\boldsymbol{x})d\mathbf{t}_i d\boldsymbol{x} \tag{2.59}$$

$$+\int\int \left[\sum_{i=1}^{M}(< \mathbf{t}_i{}^2|\boldsymbol{x} > - < \mathbf{t}_i|\boldsymbol{x} >)\mathrm{p}(\mathbf{t}_i|\boldsymbol{x})\right] \mathrm{p}(\boldsymbol{x})d\mathbf{t}_i d\boldsymbol{x}. \tag{2.60}$$

From Eq. (2.60), we can see that $\mathrm{MSE}(\Lambda)$ is minimized for parameters Λ^*, yielding

$$g_i(\boldsymbol{x};\Lambda^*) =< \mathbf{t}_i|\boldsymbol{x} >, \text{ for } i = \{1,\ldots,M\}. \tag{2.61}$$

$< \mathbf{t}_i|\boldsymbol{x} >$ and $< \mathbf{t}_i{}^2|\boldsymbol{x} >$ are defined as

$$< \mathbf{t}_i|\boldsymbol{x} > = \int \mathbf{t}_i \mathrm{p}(\mathbf{t}_i|\boldsymbol{x})d\mathbf{t}_i \tag{2.62}$$

$$< \mathbf{t}_i{}^2|\boldsymbol{x} > = \int \mathbf{t}_i{}^2 \mathrm{p}(\mathbf{t}_i|\boldsymbol{x})d\mathbf{t}_i. \tag{2.63}$$

The term $< \mathbf{t}_i|\boldsymbol{x} >$ is the conditional expectation of the target, given a feature vector $\boldsymbol{x}$. For a classification problem, in which a target $\mathbf{t}_i$ is assigned to a discrimination function, the conditional expectation of the target has the form

$$< \mathbf{t}_i|\boldsymbol{x} > = \sum_{k=1}^{M} \mathbf{t}_i \Pr(\mathbf{t}_k|\boldsymbol{x}). \tag{2.64}$$

Now, choosing

$$\mathbf{t}_i = \delta_{ik} \text{ for } \boldsymbol{x} \in C_k, \tag{2.65}$$

where δ_{ik} is the Kronecker symbol, yields $< \mathbf{t}_i|\boldsymbol{x} >= \Pr(\mathbf{t}_i|\boldsymbol{x})$, meaning that MSE is minimized if the output $g_i(\boldsymbol{x};\Lambda)$ is equal to $\Pr(C_i|\boldsymbol{x})$. The same result is valid if the error is the cross-entropy error criterion [Bishop, 1995].

Note that the MSE estimation of a posteriori probability requires that

- There is an unlimited number of training samples
- There is a sufficient number of training parameters Λ. That is, the classifier has sufficient complexity to learn the form of probability.

The MSE criterion has been mainly applied in Artificial Neural Networks (ANN) as classifier, even though it is a general framework for classifier design. In an ideal situation, an ANN trained using MSE may learn the posterior probabilities $P(C_k|\boldsymbol{x})$ and be able to implement the Bayes decision rule perfectly if it has a enough weight (e.g., hidden nodes).

2.9.2 Maximum mutual information

The maximum mutual information (MMI) criterion has been proposed as an alternative to overcome some of the problems associated with MLE, i.e., MLE with an incorrect probability form. The use of MMI has produced improvements in recognition accuracy in several speech recognition systems using hidden Markov models or feed-forward neural networks [Bahl *et al.*, 1986; Brown, 1987; Normandin, 1991; Haffner, 1994].

The MMI criterion derives from information theory, originally proposed by Shannon. The mutual information between a feature-vector $\boldsymbol{x}$ and a category C_k is defined as

$$I_k(\boldsymbol{x};\Lambda) = \log \frac{\Pr(C_k,\boldsymbol{x};\Lambda)}{\Pr(C_k;\Lambda)\Pr(\boldsymbol{x};\Lambda)} \tag{2.66}$$

$$= \log \frac{\mathrm{p}(\boldsymbol{x}|C_k;\Lambda)}{\Pr(\boldsymbol{x};\Lambda)}. \tag{2.67}$$

Let the random variable $\boldsymbol{C}$ represents a category, given the random variable $\boldsymbol{X}$ (representing a feature-vector), conditioned on the parameter set Λ. Let $\boldsymbol{I}$ be the random variable representing the mutual information between $\boldsymbol{X}$ and $\boldsymbol{C}$. The MMI criteria attempts to find a value for Λ that provides as much information as possible about the class random variable $\boldsymbol{C}$, given the input pattern random variable $\boldsymbol{X}$. The MMI's objective function is thus defined as the expected value of $\boldsymbol{I}$. That is,

$$\mathrm{MMI}(\Lambda) = \int\int \boldsymbol{I}(\boldsymbol{x};\Lambda)\mathrm{p}(\boldsymbol{x},C)d\boldsymbol{x}dC \tag{2.68}$$

$$= \sum_{\boldsymbol{x},C} \Pr(\boldsymbol{C}=C,\boldsymbol{X}=\boldsymbol{x})\log\frac{\Pr(\boldsymbol{C}=C,\boldsymbol{X}=\boldsymbol{x};\Lambda)}{\Pr(\boldsymbol{C}=C;\Lambda)\Pr(\boldsymbol{X}=\boldsymbol{x};\Lambda)} \tag{2.69}$$

$$= \sum_{i=1}^{M}\Pr(C_k)\int_{C_k}\log\frac{\mathrm{p}(\boldsymbol{x}|C_k;\Lambda)}{\Pr(\boldsymbol{x};\Lambda)}\mathrm{p}(\boldsymbol{x}|C_k)d\boldsymbol{x}. \tag{2.70}$$

$\mathrm{MMI}(\Lambda)$ is known as the *mutual information* between $\boldsymbol{C}$ and $\boldsymbol{X}$ and is a symmetric function of $\boldsymbol{C}$ and $\boldsymbol{X}$.

MMI and conditional entropy

The entropy of a random variable $\boldsymbol{C}$ is defined as:

$$H(\boldsymbol{C};\Lambda) = -\sum_{C}\Pr(\boldsymbol{C}=C;\Lambda)\log\Pr(\boldsymbol{C}=C;\Lambda). \tag{2.71}$$

The entropy can be viewed as a measure of the uncertainty in the random event $\boldsymbol{C}$, conditioned by the parameters Λ. A similar measure of the uncertainty in the random event $\boldsymbol{C}$ given the outcome of another random event $\boldsymbol{X}$ is the conditional entropy of $\boldsymbol{C}$ given $\boldsymbol{X}$:

$$H(\boldsymbol{C}|\boldsymbol{X};\Lambda) = -\sum_{C,\boldsymbol{x}} \Pr(\boldsymbol{C}=C, \boldsymbol{X}=\boldsymbol{x};\Lambda) \log \Pr(\boldsymbol{C}=C|\boldsymbol{X}=\boldsymbol{x};\Lambda). \tag{2.72}$$

The difference between the two entropies above measures the amount of information provided by $\boldsymbol{X}$ about $\boldsymbol{C}$, which is simply the MMI criterion. That is,

$$\text{MMI}(\Lambda) \;=\; H(\boldsymbol{C};\Lambda) - H(\boldsymbol{C}|\boldsymbol{X};\Lambda).$$

We note in passing that $H(\boldsymbol{C}|\boldsymbol{X};\Lambda)$ can also be expressed as a *Cross Entropy* to be minimized [Haffner, 1994].

In speech recognition, the acoustic model that determines $H(\boldsymbol{C};\Lambda)$ is usually given. Thus, minimizing the conditional entropy is equivalent to maximizing mutual information $\boldsymbol{I}(\boldsymbol{X};\Lambda)$ [Brown, 1987]. Usually, the true probability $\Pr(C,\boldsymbol{x})$ is unknown, so MMI(Λ) is maximized by choosing Λ which yields the maximum value of $I_k(\boldsymbol{x};\Lambda)$ for all $\boldsymbol{x}$ and C_k.

Contrasting the error function MMI(Λ) with the maximum likelihood criterion (2.52) shows the difference between maximum mutual information and maximum likelihood. While MLE is only concerned with maximizing the class-dependent *a posteriori* probability $\Pr(C_k|\boldsymbol{x};\Lambda_k)$, MMI maximizes the *difference* between $P(\boldsymbol{x}|C_k;\Lambda)$ and the density probability $\mathrm{p}(\boldsymbol{x};\Lambda) = \sum_k^M \Pr(C_k)\mathrm{p}(\boldsymbol{x}|C_k;\Lambda)$, enabling the maximum mutual information criterion to be "discriminative". The advantage of the MMI approach comes from the fact that even if the assumed form of the probabilities is incorrect, maximizing the model-based mutual information between the feature $\boldsymbol{x}$ and classes C may result in an equivalent Bayes classifier [Brown, 1987]. Nadas [Nadas *et al.*, 1988] has shown that in some cases the use of MMI even with an incorrect form of probability converges to an optimal solution given sufficient training data, whereas the use of MLE does not. However, in practice, maximizing the MMI criterion is considerably more fastidious than maximizing the likelihood. This is due to to consideration of all the task categories, not just the correct category, for each training token. Typical optimization methods include gradient-based search, even though an EM-like re-estimation procedure has been proposed and used [Normandin, 1991].

A limitation of MMI training is that there is still no a direct link between optimizing the MMI criterion and minimizing the probability of classification error, which is the goal of optimal classifier design. This is especially evident when using an incorrect model. Indeed, as shown above for MLE, there are situations where although a classifier is efficient enough to separate the categories optimally, training the same classifier using MMI fails to produce the optimal solution, even with large amounts of training data [Gopalakrishnan *et al.*, 1988]. The fact that MMI is maximal when the true probabilities are learned suggests that the approach shares the limitations of MLE and that the discrimination power of the MMI learning is far from sufficient, since as said before, the classifier does not need to learn the true probabilities to implement the Bayes rule.

2.10 Conclusion

We have seen in this chapter that in classical pattern recognition, feature extraction is based on expertise or data statistics. From the output of the feature extractor, an equivalent classifier to the Bayes classifier is estimated. Design of classifiers is usually achieved through parametric techniques. A classifier structure is assumed and a search for the right parameters is done by optimization of an objective function. Given a pre-chosen feature extractor, the optimal classifier, that is, the one which achieves the lowest error rate (probability of error) is the Bayes classifier. The Bayes classifier assigns the incoming feature vector $\boldsymbol{x}$ to the category C_j of highest a posteriori probability $\Pr(C_j|\boldsymbol{x})$. The Bayes error is the ideal criterion for testing the validity of the designed feature extractor. However, in most cases, the Bayes error is not available and must be estimated through parametric or non-parametric methods of classifier design.

Bayes error estimation can be divided into two strategies: those that rely on direct estimation of the posteriori probabilities, such as the Maximum Likelihood Estimation (MLE) and those that focus on estimating the Bayes regions. Even if the MLE is an efficient way to estimate these probabilities, given a preset form of the distribution, in most cases, the form of the distribution is unknown, which makes the MLE approach impractical for achieving minimum error. The alternative is the concept of discriminative training, which focuses on estimating the boundaries dividing the categories. Several approaches can be used within the framework of discriminative training. The most popular approaches use the Mean-Squared Error (MSE) or Maximum Mutual Information (MMI). However, both approaches, albeit being discriminant are not a monotonic function of the probability of error.

Estimation of the Bayes error is usually prohibitive or unrealisable. Consequently, most feature extractor designs rely on a priori knowledge or make use of a separability criterion on the input data. In this approach, there is a clear mismatch in the target of the objective function between the feature extractor and the classifier, meaning that the overall recognizer is not optimal. The feature extractor should minimize the information loss in the process, which can be realized when the feature extraction design criterion is the probability of error. A feature extraction which aims at the probability of error can be designed by the Embedded Feature Extraction Scheme using an appropriate error criterion.

Chapter 3

Discriminant Features Improve Maximum Likelihood-based HMM Performance: a Simple Experiment

He created Man and taught him articulate speech

-The Koran-

The goal of this chapter is to show that simple, discriminant features can improve the performance of a system, even in the context of a non-discriminant criterion for acoustic modeling. It describes an experiment in which the MSE criterion is used to derive discriminant features for acoustic modeling based on MLE. This chapter also serves as an introduction to classical speech recognition techniques.

3.1 Motivation

In Chapter 2, it was assumed that the recognition system is reduced to classification. However, speech recognition is more than a simple classification task, even if the classification of acoustic speech units is a necessary step to the recognition of the whole utterance. In most state-of-the art speech recognition systems, hidden Markov modeling has been the framework, within which, both steps (that is, classification of acoustic units and utterance recognition), are linked in an elegant formalism: a hidden Markov model (HMM) enables both the classification of speech sounds at the frame level while its structure permits to capture the dynamic of the speech sounds and performs a recognition scheme based on the Bayes decision-theoretic framework.

Hidden Markov models of speech units are robust in coping with coarticulation, which is an important obstacle to continuous speech recognition. However, they have shown a tendency in misclassifying acoustically similar speech units. This is partly due to the use of Maximum Likelihood Estimation on non-discriminant features, which are unable to discriminate acoustically similar sounds. Consequently, using discriminant features in the HMM framework is one method for overcoming this deficiency.

An approach for deriving such features is to make use of embedded feature extraction techniques, based on a discriminant criterion. Due to the proven discriminant capabilities of a feed-forward Neural Network for static pattern classification, the approach here is to use the Time-Delay Neural Network (TDNN) structure as a parameterized feature extractor, trained using the Minimum Squared Error Criterion (MSE). The resulting features are then used to estimate HMM by Maximum Likelihood. This approach was originally proposed in [Sugiyama and Biem, 1991; Biem and Sugiyama, 1992].

The proposed recognizer is as follows: segmental acoustic units are classified by a Neural Network, which provides the discrimination features required in the recognition stage, based on HMM. To fully understand the process and motivation, the use of HMM in modeling speech signals is first reviewed, followed by a description of the Time-Delay Neural Network. Lastly, the experimental procedure is explained in details.

The aim of this chapter is twofold: show the utility of discriminative features in the conventional framework of recognizer design, based on the widely-known Maximum Likelihood Estimation (MLE) approach for estimating acoustic models, and serve as a practical introduction to state-of-the art speech recognition techniques.

3.2 Hidden Markov Modeling of Speech

Hidden Markov modeling is a standard stochastic technique for modeling time sequences. Originally, HMMs were applied within mean-stream statistics, before their potential application to speech recognition was discovered and proposed [Jelinek, 1972]. Since then, hidden Markov modeling is the most extensively used technique for acoustic modeling of speech. Here, the theory of

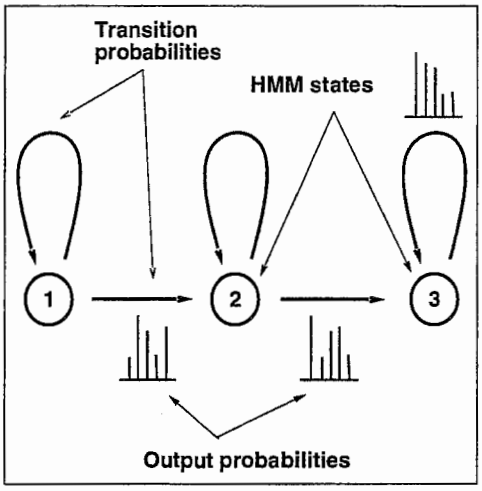


Figure 3.1: Example of Hidden Markov Model with three states and loops.

HMMs is described, in particular discrete HMMs, and discuss its application to speech recognition.

3.2.1 The theory of hidden Markov models

A discrete hidden Markov model is a collection of states with the finite alphabet of input observation symbols [Rabiner, 1989]. Each state is linked with others via a set of transitions, where a transition is characterized by two sets of probabilities: the probability of taking this transition and the probability distribution to emit one symbol from the finite alphabet of symbols. Concretely, an HMM is specified by :

- The number of states in the model Q, which comprises an initial state S_I, and a final state S_F.
- The alphabet size P, which is the number of discrete symbols in the alphabet.
- The set of transition probabilities $A = (a_{ij})$ where a_{ij} is the probability of taking the transition from state S_i to state S_j.
- The symbol probability distribution for each transition $i \rightarrow j$. That is, a given matrix $B = (b_{ij}(k))$ where $b_{ij}(k)$ is the probability of emitting the symbol k when taking the transition $i \rightarrow j$.

The Fig. 3.1 shows an example of an HMM with 3 states.

The above HMM definition concerns the discrete case (use of discrete probabilities). Hence, the use of an alphabet of discrete symbols. Most systems nowadays use continuous Hidden Markov Models (CHMM) where the output probabilities are probability density functions. The main advantage of discrete HMMs over continuous HMMs is that there is no assumption to be made concerning the form of the probability distributions [Brown, 1987]. However, mixed results have

been reported when comparing the two approaches. For instance, Brown [Brown, 1987] has shown that discrete HMMs performed better than continuous HMMs in the E-set task. On the other hand, Gupta [Gupta *et al.*, 1987], reported high performance of continuous HMMs in a large vocabulary (60.000) word recognition task. It seems that for small-size classification tasks, discrete HMMs is more efficient.

Stated briefly, an HMM is a Finite State Machine (FSM), whose state transition is a 1st-order Markov process and which generates a sequence of symbols at each transition. In speech processing, the observations are a sequence of frame-vectors $\boldsymbol{x}_1^T = \{\boldsymbol{x}_1, \boldsymbol{x}_2, .., \boldsymbol{x}_T\}$, which are the output of the feature extractor. The key concept is to find the HMM structure, which has generated a given sequence (at least likely to have done so). Let S_t be the state occupied by $\boldsymbol{x}_t$, where t refers to the time-sequence in the frame domain. A and B are then defined as :

$$a_{ij} = \Pr(S_{t+1} = j | S_t = i), \tag{3.1}$$

$$b_{ij}(k) = \Pr(\boldsymbol{x}_t = k | S_{t+1} = j, S_t = i), \tag{3.2}$$

in which the states $S = (S_t)$ are hidden and only $\boldsymbol{x}_1^T$, the sequence of output symbols are observable. It goes without saying that the parameters A, B must satisfy the following properties, as they are probabilistic distributions,

$$a_{ij} \geq 0, \tag{3.3}$$

$$b_{ij}(k) \geq 0, \tag{3.4}$$

$$\sum_j a_{ij} = 1, \tag{3.5}$$

$$\sum_k b_{ij}(k) = 1, \tag{3.6}$$

as well as the Markov assumption:

$$\Pr(S_t = j | S_{t-1} = i, S_{t-2} = k, ...) = \Pr(S_t = j | S_{t-1} = i). \tag{3.7}$$

The above Markov assumption describes a first order Markov chain, in which the values of S_t lies between 1 and Q.

3.2.2 The Forward algorithm

Given an HMM and a sequence of observations, the Forward algorithm is a method to effectively compute the probability of the model to producing the sequence of observations. It is a recursive procedure that is described below.

Let us consider an HMM $\mathcal{M} = (Q, P, A, B)$ and the set of observations $\boldsymbol{x}_1^T = \{\boldsymbol{x}_1, \boldsymbol{x}_2, .., \boldsymbol{x}_T\}$ of length T. To compute the probability of this HMM producing the observation $\boldsymbol{x}_1^T$, we must sum the probabilities over all paths of length T, where a path Ψ is a sequence of states $S_1, ..., S_T$,

$$\Pr(\boldsymbol{x}_1^T | \mathcal{M}) = \sum_{on\ all\ paths\ \Psi} \Pr(\boldsymbol{x}_1^T | \Psi) \Pr(\Psi | \mathcal{M}). \tag{3.8}$$

The forward algorithm provides a recursive procedure which enables a faster computation of (3.8) than the obvious straightforward computation of the summed individual path probabilities. The process is as follows. Consider the *forward variable* $\alpha_t(i)$ defined as

$$\alpha_t(i) = P(\boldsymbol{x}_1^T, S_t = i|\mathcal{M}). \tag{3.9}$$

$\alpha_t(i)$ is the probability that the Markov process is in state i at time t, after generating the observations $\boldsymbol{x}_1^T$ and is computed recursively as follows::

1. $\alpha_0(S_I) = 1.$
   $\alpha_0(i) = 0$ if $i \neq S_I$ (initial state).

2. $\alpha_{t+1}(j) = \sum_i \alpha_t(i) a_{ij} b_{ij}(\boldsymbol{x}_t)$ for $t > 0.$

Consequently,

$$\Pr(\boldsymbol{x}_1^T|\mathcal{M}) = \alpha_T(S_f),$$

which is the probability of the model $\mathcal{M}$ producing the sequence of observations $\boldsymbol{x}_1^T$, where $\alpha_T(S_f)$ is computed in the last state. The forward algorithm is the key for implementing the decision rule of the HMM classifier. Given a sequence of observations, the forward algorithm enables to choose among competing HMM, the one with the highest probability of producing the given observations. By using the Bayes'rule,

$$\Pr(\mathcal{M}|\boldsymbol{x}_1^T) = \frac{\Pr(\boldsymbol{x}_1^T|\mathcal{M})\Pr(\mathcal{M})}{\Pr(\boldsymbol{x}_1^T)},$$

one can implement the Bayes classifier, based on the assumption that $\Pr(\mathcal{M}|\boldsymbol{x}_1^T) = \Pr(\omega|\boldsymbol{x}_1^T)$, where ω is the acoustic unit being modeled by $\mathcal{M}$.

3.2.3 The Viterbi algorithm

Given a set of observations, generated by an HMM, one would like to know which state sequence has produced the given observations. The forward algorithm, however, does not provide this state sequence since the probability is computed over all possible state paths. Instead, the Viterbi algorithm [Bellman, 1961b; Viterbi, 1967], which performs a dynamic programming along the states of the models, is used to compute the single most likely state sequence that produced the given observations. The procedure is as follows.

Let us define:

$$v_t(i) = \max_{\text{on all path of length } t} \Pr(\boldsymbol{x}_1, ..., \boldsymbol{x}_t, S_1, S_2, ..., S_t = i|\mathcal{M}), \tag{3.10}$$

where $v_t(i)$ is the highest probability along a single path of length t, after generating the first t observations and ending in state S_i. $v_t(i)$ is also computed recursively. The state i which maximizes (3.10) is kept using a function φ_t defined below. The procedure is as follows:

1. $v_0(0) = 1$.
   $v_0(i) = 0$ for $i \neq 0$.

2. $v_t(j) = \max_i(v_{t-1}(i)a_{ij}b_{ij}(\boldsymbol{x}_t))$ for $t > 0$.
   $\varphi_t(j) = \arg\max_i(v_{t-1}(i)a_{ij})$ for $t > 0$.

3. $S_T = \arg\max_i(v_T(i))$.

The most likely path sequence is then obtained by backtracking, i.e.,

$$S_t = \varphi_{t+1}(S_{t+1}).$$

The Viterbi algorithm is used for segmentation and for recognition, when the use of the forward algorithm is time-consuming.

3.2.4 The Forward-Backward algorithm

The Forward-Backward algorithm, also known as the Baum-Welch algorithm, enables to re-estimate the HMM parameters from the training data. i.e., to optimize the model parameters so that the model matches the observed data, in the Maximum Likelihood sense. That is, given an HMM $\mathcal{M}$, the Forward-Backward algorithm produces a better model $\bar{\mathcal{M}}$, such that

$$\Pr(\boldsymbol{x}_1^T|\bar{\mathcal{M}}) \geq \Pr(\boldsymbol{x}_1^T|\mathcal{M}),$$

which then enables an iterative re-estimation procedure to be carried out.

Let us consider $\alpha_t(i)$, defined above, and the *backward variable* $\beta_t(i)$ defined as:

$$\beta_t(i) = \Pr(\boldsymbol{x}_{t+1}, \boldsymbol{x}_{t+2}, ..., \boldsymbol{x}_T|S_t = i, \mathcal{M}). \tag{3.11}$$

$\beta_t(i)$ is the probability that the model is in state i and will generate the *future* sequence $\boldsymbol{x}_{t+1}, ..., \boldsymbol{x}_T$. $\beta_t(i)$ can be computed recursively as follows:

1. $\beta_T(i) = 1$ if $i = S_f$.
   else $\beta_T(i) = 0$.

2. $\beta_t(i) = \sum_j \beta_{t+1}(j)a_{ij}b_{ij}(\boldsymbol{x}_{t+1})$.

Let us define:

$$\gamma_t(i,j) = \Pr(S_t = i, S_{t+1} = j|\boldsymbol{x}_1^T, \mathcal{M}),$$

the probability of being in state i at time t and in state j at time $t+1$.

$$\gamma_t(i,j) = \frac{\alpha_t(i)a_{ij}b_{ij}(\boldsymbol{x}_{t+1})\beta_{t+1}(j)}{\Pr(\boldsymbol{x}_1^T|\mathcal{M})} \tag{3.12}$$

can be viewed as the probability of taking the transition $i \rightarrow j$ during the observation sequences $\boldsymbol{x}_1^T$. Consequently, the expected number of times for taking the transition $i \rightarrow j$ during the observation sequence $\boldsymbol{x}_1^T$ is simply

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

Given some initial parameters, the following re-estimation formulas are used to update the parameters at each iteration:

- $\bar{a}_{ij} = \dfrac{\sum_{t=1}^{T} \gamma_t(i,j)}{\sum_{t=1}^{T} \sum_{k} \gamma_t(i,k)} = \dfrac{\textit{expected number of transitions from } i}{\textit{expected number of transitions from } i \rightarrow j}.$
- $\bar{b}_{ij}(k) = \dfrac{\sum_{\boldsymbol{x}_t = k} \gamma_t(i,j)}{\sum_{t=1}^{T} \gamma_t(i,j)} = \dfrac{\textit{expected number of times of taking the transition } i \rightarrow j \textit{ and emitting } k}{\textit{expected number of times of taking the transition } i \rightarrow j}.$

The above formulas produce a new HMM $\bar{\mathcal{M}} = (\bar{A}, \bar{B}, P, Q)$ where $\bar{A} = (\bar{a}_{ij})$ and $\bar{B} = (\bar{b}_{ij}(k))$. An important theorem proved by Baum [Baum, 1972] states that :

$$\Pr(\boldsymbol{x}_1^T | \bar{\mathcal{M}}) \geq \Pr(\boldsymbol{x}_1^T | \mathcal{M}).$$

That is, the likelihood is maximized at each iteration, which leads to an efficient way to generate the a "good" model, in the Maximum Likelihood sense, from the training data. The procedure for training the HMMs is thus as follows:

1. Guess an initial set of parameters.
2. Compute $\bar{a}_{ij}$ and $\bar{b}_{ij}(k)$ from this set according to the re-estimation formulas.
3. Replace a by $\bar{a}$ and b by $\bar{b}$
4. Stop if the likelihood of the observation do not change according to some threshold.

3.2.5 HMM application to speech recognition

In acoustic-phonetic modeling approach to speech recognition, the standard method is to assign a model to each phone or phoneme and, in the case of discrete HMMs, build a codebook for each phone/phonemes. In the latter approach, each frame of the speech wave is represented (in the feature extractor domain) by the closest prototype from the codebook, in the sense of a pre-defined distance (the prototypes of the codebook has the same dimension with the incoming feature-vectors). The codebook is produced by a Vector Quantization algorithm, which selects a set of prototypes from the training data, according to a minimum distortion criteria [Linde *et al.*, 1980].

Even if the vector quantization process does introduce some noise (quantization noise), it has been shown that little accuracy is lost [Shikano *et al.*, 1986].

The vector quantizer performs in the feature-space. That is, the raw speech is first converted into multi-dimensional real feature vectors. Many different speech features are used in HMM. In the early 80s, when discrete HMMs were the most widely spread modeling technique, the most common parameters were Linear Predictive Coding (LPC) of speech, filter-bank or LPC-based cepstrum coefficients, first order differential cepstrum coefficients, power, and first order differential power [Sugiyama, 1981]. Various ways are offered to use all the knowledge brought by these features. One method, adopted by Furui [Furui, 1986] is to put different feature types into multi-dimensional vectors and find the suited composite metric distance to produce the codebook. Another method, proposed by Gupta [Gupta *et al.*, 1987] is to build different codebooks for each kind of parameters and train the HMM using several codebooks. In this case, there is no longer a single homogeneous symbol in the HMM alphabet, but a vector of different symbols. The recursive computation of the forward variable is replaced as follows:

$$\alpha_{t+1}(j) = \sum_i \alpha_t(i) a_{ij} \prod_c b_{ij}(\boldsymbol{x}_t^{(c)}),$$

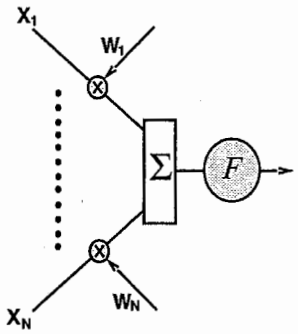
where c refers to a codebook label. The output probability of emitting multiple symbols is then computed as the product of the probability of producing each symbol. The underlying assumption is that different features are statistically independent, which is indeed a risky assumption in HMM modeling. The use of a composite distance may reduce the distortion, but using several codebooks leads to a higher computational cost.

For recognition, the Forward algorithm is used (the model with highest probability is chosen) or the Viterbi algorithm, when faced with computational problems. If sub-word units (such as phonemes) are used to train HMM, then the recognition of a word is just a concatenation of the sub-word models. The efficiency of HMM comes from the fact that, there is no special care about dealing with segments boundaries, since across segmental information are absorbed in the states of the models, which enables to deal with the coarticulation problem in an efficient manner.

3.3 Connectionist approaches

Connectionist methods are the most recent approach to speech recognition. Said briefly, knowledge, constraints, and procedures are distributed across many simple computing units. Such systems are usually named Neural Networks because of the similarity with the nervous system. That is, Neural Nets (NN) have been studied with the motivation of mimicking the nervous system. Concretely, NN are simply an amount set of computational units linked together in such a way that they provide a distributed processing system [Hinton *et al.*, 1986]. They have been used for various problems in speech recognition and image recognition.

The basic unit or neuron model of a Neural Networks is shown in the Fig 3.2. Given a set of



A Neural Network Unit

$$y = F\left(\sum_{i=0}^{n} w_i x_i\right)$$

Figure 3.2: The basic unit of a Neural Network without threshold.

input unit $(x_i)_{i\in\{1,\ldots,n\}}$, the NN unit output y is defined as

$$y = F(\sum_{i=0}^{n} w_i x_i - \theta),$$

where F is a the transfer function of the unit and θ is a threshold. That is, the NN unit computes the weighted sum of the inputs and pass the result through a non-linear function, usually a sigmoid function (see Chapter 6 for the definition of a sigmoid). The NN unit can be viewed is a classifier, which splits the input space into two subspaces.

There are different types of Neural Networks, depending on the architecture as well as on the learning algorithm used. The concern here is about feed-forward networks usually called multi-layer perceptrons (MLP). The MLP architecture includes an input layer, one or more hidden layers, and an output layer. For classification, the output layer consists of M nodes which correspond to an M-class problem. For recognition, generally, the largest output is considered as the main product of the network. The discrimination function corresponding to class C_k is simply the weighted sum of the outputs of the previous layers. That is,

$$g_k(\boldsymbol{x}, \Lambda) = F\left(\sum_{j=0}^{M} w_{kj} y_j(\boldsymbol{x}; \Lambda')\right). \tag{3.13}$$

Again, F is the transfer function, $y_j(\boldsymbol{x}; \Lambda')$ is the output of j-th node of the hidden layer feeding the output layer, w_{kj} is its connection to the output node of C_k. Λ' represents the set of the other weights. Usually, an NN is trained in a supervised manner. That is, by mean of the an objective function which describes the error. The most widespread error criteria is certainly the some of squared-error on the training data:

$$\mathrm{MSE}(\Lambda) = \frac{1}{2}\sum_{\boldsymbol{x}}\sum_{j=1}^{M}(g_j(\boldsymbol{x}; \Lambda) - t_j(\boldsymbol{x}))^2 \tag{3.14}$$

where N is the total number of training data and $t_j(\boldsymbol{x})$ is the target corresponding to class C_j , when input $\boldsymbol{x}$ is presented to the NN. Again, for a classification problem,

$$t_j(\boldsymbol{x}) = \delta_{jk} \qquad \text{for } \boldsymbol{x} \in C_k$$

where δ_{jk} is the Kronecker notation.

The learning process is usually based on gradient search, in which, the weights are adjusted according to the errors so as to produce the desired output. This process is widely know as back-propagation because the derivatives of the error are back propagated through the network and the weights are adjusted accordingly [Hinton *et al.*, 1986] (see Appendix B).

3.3.1 Neural Networks for speech recognition

In speech recognition, the classification properties of Neural Nets are used to discriminate speech units. For accurate speech modeling, various problems must be overcome when choosing the architecture of the network to use. The first decision concerns the choice of the number of nodes. A layered network must have sufficient interconnections between units, according to the task, so as to be able to approximate complex decision regions [Lipmann, 1987]. However, the number of weights shall be sufficiently smaller than the amount of training data, so as to force the Neural Network to encode the training data by extracting regularities. This is usually done by an empirical study and tradeoff between the required accuracies and the number of data available.

For speech recognition applications, the network should be able to find the relationships between time-dependent events as well as be able to discover the invariance across those events. Among various proposed Neural Network architectures for dealing with these problems, the recurrent Neural Network [Robinson and Fallside, 1990] and the Time-delay Neural [Waibel *et al.*, 1987] explicitly consider time-dependency within their architecture.

3.4 TDNN-Features for HMM-based Phoneme Recognition

It was shown in chapter 2 that a classifier trained with MSE can estimate a posteriori probabilities and that, given an M-class problem, the a posteriori probabilities are the best features for classification. Hence, local acoustic properties can be reduced to a few number of parameters, that are discriminant since the features gather information about competing categories.

Based on the above assumption, the method presented here is the use of the Time-Delay Neural Networks (TDNN), trained with MSE, as a feature extractor for hidden Markov modeling. That is, the HMM theory is applied to modeling feature-vector sequences produced by the TDNN. Pattern classification using TDNN has proved to be a powerful way to classify speech units [Sawai *et al.*, 1989]. Sawai's results for large vocabulary have confirmed previous TDNN-based experiments [Waibel *et al.*, 1987], which showed a good discrimination ability for the stop consonants /*b*/, /*d*/ and /*g*/ and also high performance on the recognition of all phonemes.

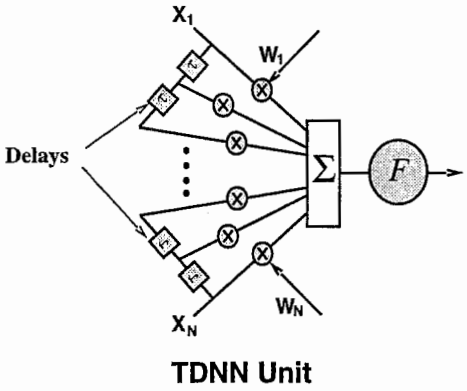


Figure 3.3: The basic TDNN unit. The TDNN unit introduces delay within each input so as to keep track of past events.

MSE-based TDNN training can be viewed as implementing an embedded feature extraction scheme, which results in two advantages: compactness of information and accurate acoustic modeling. First, a description of the TDNN architecture is needed to fully understand the proposed approach.

3.4.1 TDNN architecture for continuous speech recognition

The TDNN architecture described in this section is the same as in [Sawai *et al.*, 1989] and was aimed at scanning Japanese phonemes. That is, the overall architecture was optimized to discriminate phonemes. Accurate phoneme accuracies is a big step toward realizing a robust speech recognizer. The TDNN description is as follows.

The TDNN basic unit is a modified version of the standard basic unit of a standard Neural Network (see Fig. 3.3). The basic unit, used in many neural networks, computes the weighted sum of its inputs and then passes this sum through a non-linear function (a threshold or a sigmoid function)[Lipmann, 1987]. However, the TDNN's unit is modified by introducing several delays D_1 through D_N, which are assigned to each input unit. As shown in the Fig. 3.3, a delay is also characterized by its own weight, which enables to track past events. For instance, given for $N = 2$ and 16 number of input units, 48 weights are required to compute the weighted sum of the input corresponding to current, previous and previous-to-previous acoustic event. Throughout this chapter, a sigmoid function, is used as a non-linear transfer function.

Large phonetic TDNN

The TDNN used in the experiments of this chapter was composed of subnets and is fully described in [Miyatake *et al.*, 1990; Sawai, 1991]. It is a 4 layered, modular structure in which each subnet was trained to recognize a set of confusable phonemes. For example, one subnet was trained to recognize the stop consonants /*b*/, /*d*/, /*g*/ (a BDGnet), another subnet was trained to recognize

the voiceless stops /p/, /t/, /k/. In total, the TDNN was composed of subnets trained to recognize the voiced stops /b/, /d/, /g/, the voiceless stops /p/, /t/, /k/, the nasals /m/, /n/ and the syllabic nasal (denoted here as /N/), the fricatives /s/, /sh/, /h/ and /z/, the affricates /ch/, /ts/, the liquids and glides /r/, /w/, /y/, the set of vowels /a/, /i/, /u/, /e/, /o/ and the silence represented by /Q/. The network was specially designed to spot the phonemic contain of a given utterance, which may result into a vocabulary-free, large vocabulary recognition system [Miyatake *et al.*, 1990]

Each subnet was trained only within each respective phonemic group and without knowledge of other groups, using shifted training tokens to enhance the time-shift invariance properties of the network. After training each subnet, the whole network has to be trained for improving the discrimination capabilities. The training was performed after concatenating the subnets one after another and running a fast back-propagation algorithm [Haffner *et al.*, 1989]. The resulting architecture is the Large-phonetic TDNN (LTDNN) as shown in Fig. 3.4. This figure is taken from [Sawai, 1991] and shows the characteristic of each subnet in term of number of weights and delays. The input layer is composed of 240 units (16 Mel spectral coefficients times 15 frames). Each node in the input layer has 3 delays (represented by the hashed region). The boxes in the first layer represents the TDNN subnets, each trained to discriminate within its specific phonemic class. It can be seen that all subnets in the first hidden layer comprises 8 nodes with 13 delays and nodes in the second hidden layer has 9 delays with the number of nodes depending on the subnet. The third and last hidden layer has 24 nodes and 5 delays and is role is to gather and "glue" the different subnets. Each node in this layer is linked with all the nodes of the previous layer. The output layer is comprised of 24 nodes with each node being connected to only one node of the third hidden layer. The final decision is made by summing the output of the second hidden layer over the duration of the speech utterance.

The output of the LTDNN is a vector $\boldsymbol{o} = [o_1, o_2, ..., o_{24}]^{\mathrm{T}}$, of 24 dimension, where o_i is the output of the LTDNN, which corresponds to the i-th phone-category. This output is simply a "compact" representation of the information gathered by the third hidden layer. Consequently, the output vectors enable to represent the acoustic information into few, discriminant parameters. This approach can be related to the one proposed in [Bimbot *et al.*, 1990], in which the TDNN are use to discriminate phonetic features.

TDNN training

For optimization, the usual back propagation algorithm was carried out within each subnet. As said before, the back propagation is simply a gradient descent of the mean-squared error as a function of the weights (see Appendix B). The weights were randomly initialized with small values so that they range between -1 and 1. The output of all the units were computed, starting at the input layer and working forward to the output layer. The output was compared to the desired output and the derivatives of the errors were back propagated through the network. This procedure was repeated many times until the network produced the desired output.

Since the LTDNN had to cope with speech through its delays, the usual back-propagation

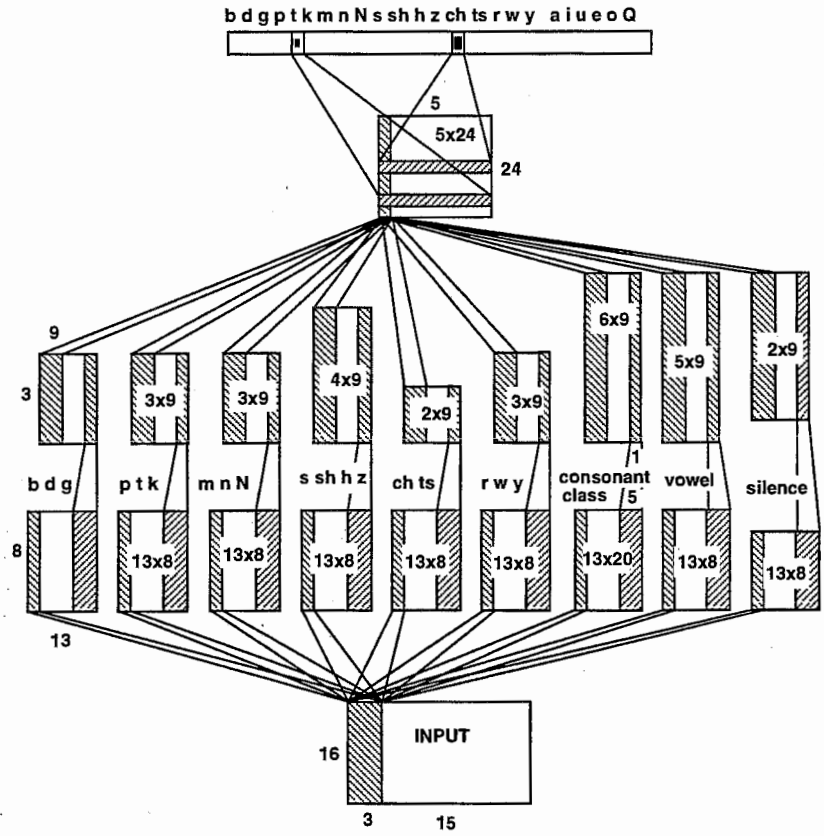


Figure 3.4: Large-phonetic TDNN composed of TDNN subnets aimed at discriminating within each phonemic class. Each subnet is characterized by each own set of delays and number of weights (from [Sawai, 1991]).

procedure were slightly modified. Conceptually, the back-propagation procedure is applied to speech pattern that are stepped through in time. For the LTDNN, each collection of the unit is duplicated for each one frame-shift in time and then the back-propagation is applied to all the shifted copy as if they are separate events. This yields different error derivatives, corresponding to each time-shifted connections. Moreover, instead of changing the weights on the time-shifted connections separately, each weight is updated by the average of all corresponding time-delayed weight changes. This process was applied to all connections and all time-shift, which in practice needed many iterations to reach convergence.

3.4.2 Database

The speech data was taken from a large vocabulary database of 5240 common Japanese words uttered in isolation by one male native speaker in a soundproof booth and digitized at a 12kHz sampling rate. The database was split into two sets: the training set (2620 utterances) and the testing set (2620 utterances). The entire database was phonetically labeled by hand and used to extract the center segment corresponding to a given phoneme.

3.4.3 Experimental procedure

The speech wave is preprocessed by the LTDNN and the result is a 24 dimensional vector whose elements are the activation values of the output units of the optimized LTDNN. Consequently, each value in the vectors represents an activation of the output unit of the LTDNN. After collecting the outputs corresponding to all utterance in the database, these vectors are used as features for HMM (codebook production and HMM training). This is the feature extraction phase which makes use of the optimized LTDNN for scanning the phonemes contained a word [Sawai *et al.*, 1989].

The procedure (block diagram) is shown in Fig. 3.5 and is as follows.

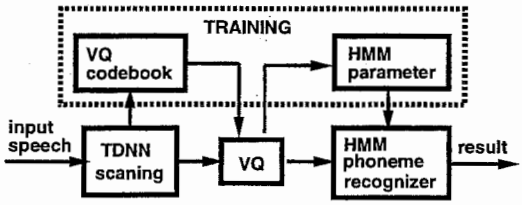


Figure 3.5: The architecture of the recognition process (from [Sugiyama and Biem, 1991]).

1. Sample the speech utterances at 12 kHz. All utterances consist of words spoken by a male professional announcer (5240 words in total).

2. Compute 256 FFT coefficients every 5 ms. Average by every two frames, which gives a frame rate of 10 ms.

3. Compute 16 Mel-scale spectrum coefficients from the FFT coefficients. The Mel-scale is a frequency scale, which tries to approximate human perception of frequency.

4. Normalize all the values between -1 and +1 among 15 frames.

5. In case of LTDNN training, pass 15 frames of 16 spectrum coefficients as input in the LTDNN network (which makes use of information about data labeling). Consequently, the input to the LTDNN is 240 (16 times 15) dimensional vectors.

6. Collect the set of training features and the testing features from the LTDNN output, corresponding to the whole set of utterance.

7. Process the sequence of features vectors according to the phoneme labeling information.

8. Train the HMM by the available discriminant features.

Stated briefly, the raw speech, taken from the database (one male speaker), was first sampled at 12kHz, Hamming-windowed and 256 FFT coefficients were computed every 5 ms. Two consecutive frames are then averaged to produce 256 averaged FFT coefficients. From these FFT coefficients, the Mel-scale spectrum coefficients (see chapter 8) are computed and 15 frames (one frame is equivalent to 16 Mel-scale coefficients) are passed on to the LTDNN as an input token. All the coefficients of an input token were normalized between -1 and +1 by subtracting from each coefficient the average coefficient energy computed over all 15 frames of an input token. To enhance the sensibility to time-shift, a sequence of overlapping tokens are derived from a segment, by moving the token centers from the start of the hand-labeled segment boundary to the end of the segment boundary. This process is fully explained in [Miyatake *et al.*, 1990].

Finally, the output was obtained by summing the last units of the second layer over time. The normalized values were used as input to the LTDNN and the output of the LTDNN was collected in the form of sequence of vectors of 24 dimensions per token.

After LTDNN optimization, the output of the LTDNN for an utterance was collected. Repeating this process over the whole database created the feature set to be used for acoustic modeling by HMM.

3.4.4 Label generation of HMM training

The LTDNN has been optimized for phonemic discrimination. However, the feature-vector sequence, output of the LTDNN over the word utterance, most be labeled to extract the phonemic segment for HMM training.

An output vector of the LTDNN is equivalent to 150 ms of speech wave. For a word utterance, the window is shifted every 10 ms and starts at the beginning of the word, which includes some silence. The features file containing the features-vectors scanned from the LTDNN output units, should processed to extract the right sequence of frames, corresponding to a given phoneme. This is done using the simple formulas:

$$f_1 = \left(\frac{b-a}{s} + \frac{w}{2s}\right) \tag{3.15}$$

$$f_2 = \left(\frac{e-a}{s} + \frac{w}{2s}\right) \tag{3.16}$$

where f_1 and f_2 are the indexes of the first and the last frame respectively (in the LTDNN output domain), corresponding to the phonemic segment. b and e refer to the beginning and end of the speech segment corresponding to the phoneme in the speech waveform; a is the end of the silence within the utterance and w is the size of the LTDNN window (150 ms) with s being window shift size (5 ms).

3.4.5 HMM phoneme models

An important choice concerns the topology of the Hidden Markov Model and the tying of probability distributions [Rabiner, 1989]. For phoneme modeling, at least two states are needed to capture the beginning and end of a phonemic sound. Two more states were added to model the middle part of the sounds, which is usually characterized by highly varying spectral characteristics. Consequently, each phoneme-HMM (discrete model) consisted of 4 states (including the last state), with loops in the three first states. The architecture is shown in Fig. 3.6.

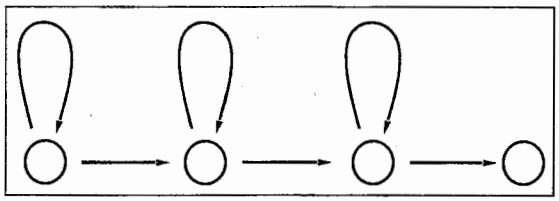


Figure 3.6: The HMM structure for phonemic modeling.

Before parameter estimation, the transition probabilities were initialized to be equal to 0.5 (uniform transition probabilities) and the initial values of the observation probabilities were set equal to the number of observations of one code divided by the total number of codes. The Baum-Welch algorithm was used to train the models. During training, the probabilities approach zero when running the Forward computation and usually after hundred frames or so, these probabilities may underflow the computer floating point representation. To solve this problem, probabilities values were simply represented by their logarithms.

3.4.6 VQ codebook generation

Again, the Vector Quantization technique codes each frame of the speech data into a pre-stored reference vector. The set of reference vectors is called a codebook. In other words, all the speech frames are represented by this codebook. For computational ease, the number of codes in the codebook is, in many cases, a power of $2(2^b)$ such that b is the number of bits per vector. Obviously, speech recognition performance depends on the "quality" of the codebook. Generally the codebook is generated by a splitting procedure that minimizes the total distortion. The algorithm used to produce the codebook is the LBG (Linde, Buzo, Gray) [Linde *et al.*, 1980], which splits the training data into 2,4,..., 256 partitions, generating a centroid for each partition as the average of the vectors in the partition. The clustering process is shown in Fig. 3.7. A centroid represents a region of the feature-space, and all feature-vectors of a given region are represented by the centroid of this region.

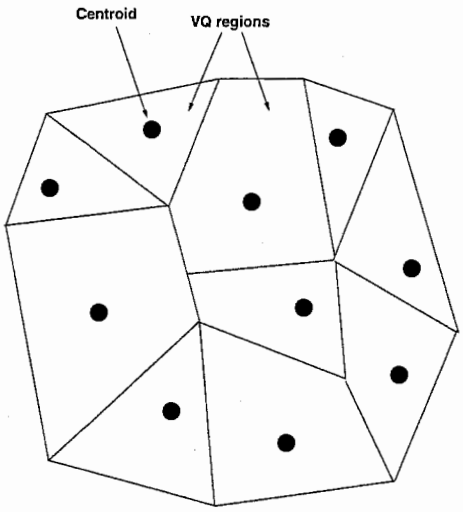


Figure 3.7: VQ partitioning into regions. A region is represented by a centroid.

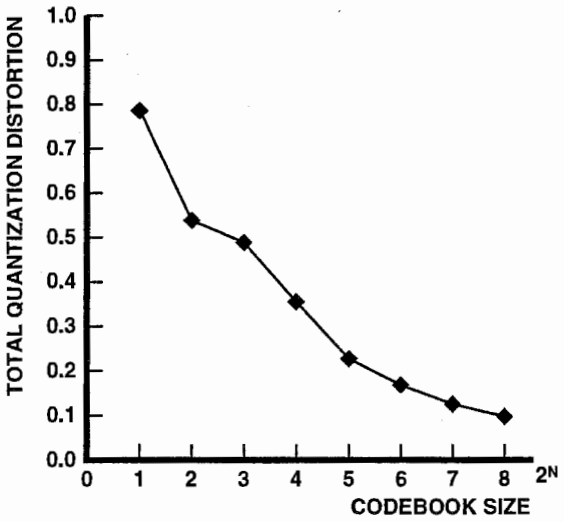


Figure 3.8: VQ distortion as a function of the codebook size.

As displayed in Table 3.1, the codebook was processed from 216 common Japanese words, taken from a special database. This database was different from the database used for training and testing.

Table 3.1: Number of VQ training samples.

training words	216 words
training frames	18281 frames

Several codebooks of different sizes (64,128,256) were generated in order to observe the recognition rate as a function of the codebook size. The metric associated with the codebook production model is simply the Euclidean distance:

$$D(v_1, v_2) = \sum_{i=1}^{24} (v_1(i) - v_2(i))^2$$

where $v(i)$ is the i-th element of the vector that corresponds to the activation of the i-th phoneme. We note in passing that this metric is similar to the metric used within the MSE criteria.

The target in producing the codebook is to minimize the average distortion over the training set. The average distortion as a function of the number iterations is displayed in the Fig. 3.8. In this figure, it can be seen that average distortion is a decreasing function of the codebook size.

One of the shortcoming of the VQ procedure is the fact that the algorithm does not take into account the probability distribution of the data: some categories may be over-represented while others are not. For instance, knowing that the Japanese language is composed of 23 phonemes, it may seem that a codebook of size 32 is sufficient to have a representation of each phoneme in the language. However, this is not the case, when we examine the 32-sized codebook in Fig. 3.9. In this figure, each row corresponds to a prototype vector, in which the dot/column position relates to the corresponding phonemes. The first prototype corresponds to the activation of the /m/. There is no prototype for /d/ or /p/, which may be due to the rather limited number of /p/ and /d/ examples in the database. Increasing the codebook size from 32 to 64 enables to represent all the Japanese phonemes. For more efficiency, codebooks of size 128 and 256 were also generated (Fig. 3.10). Generally, the codebooks are redundant, which is due to lack of consideration for the probability density functions of the data in the VQ algorithm.

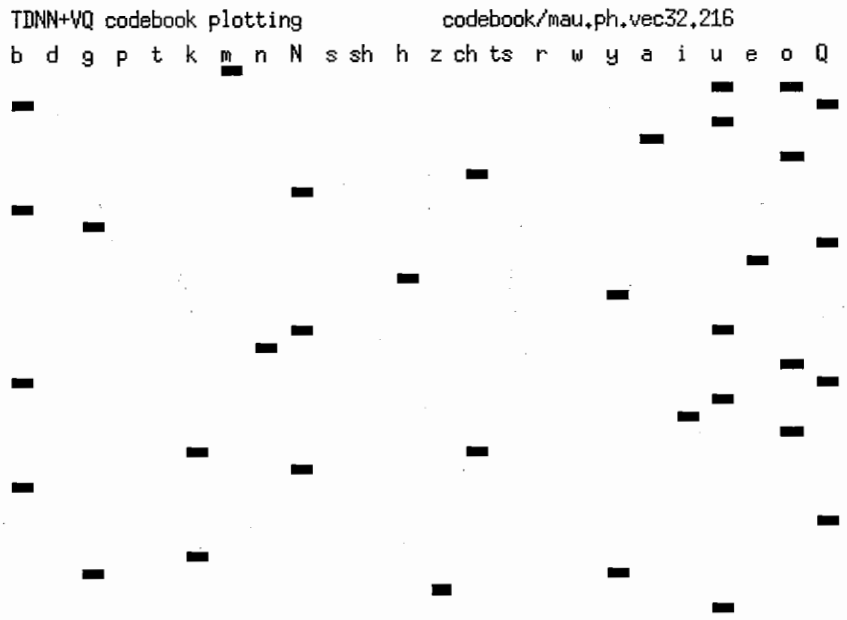


Figure 3.9: Values of VQ codes (32 codes).

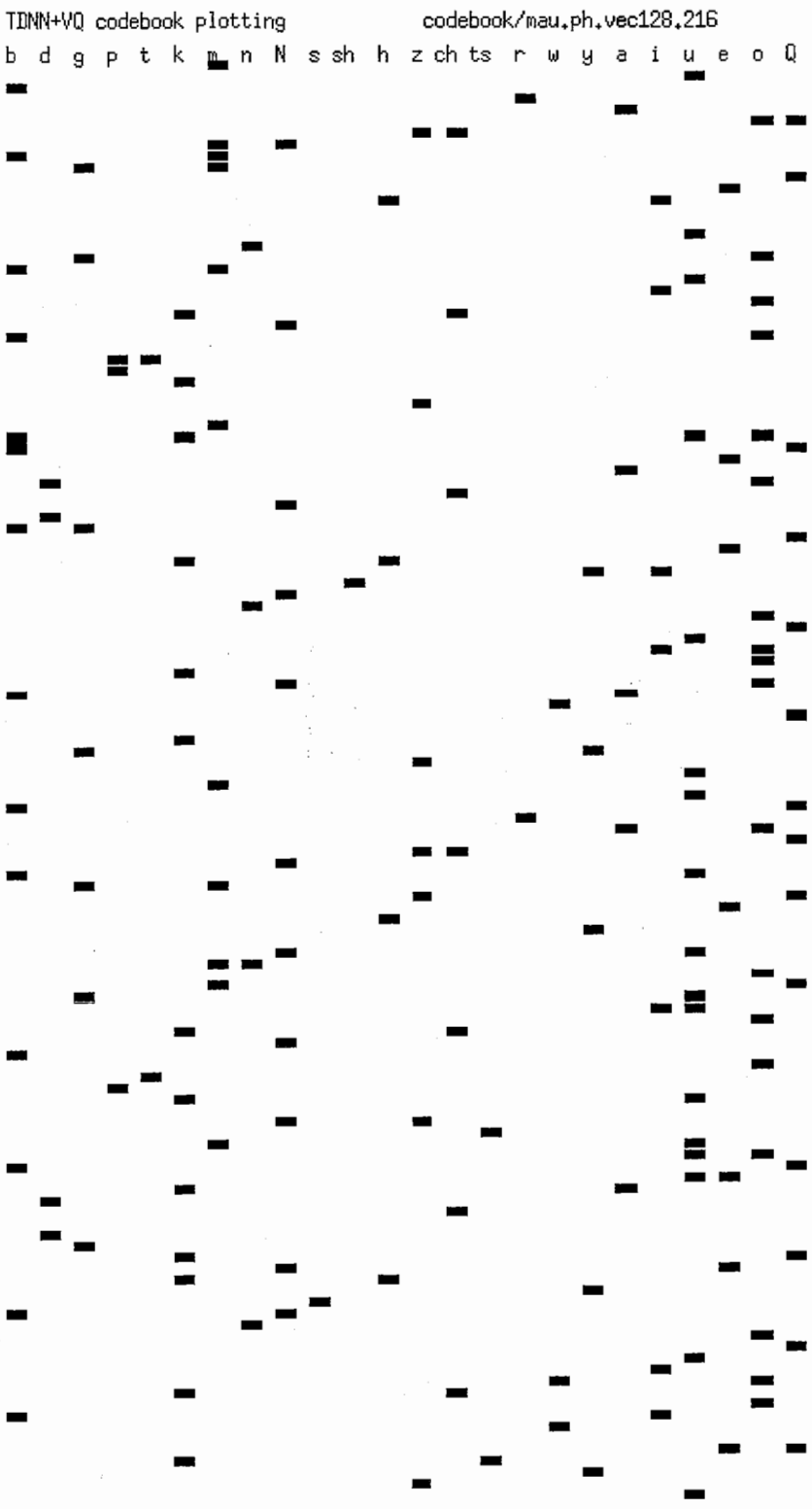


Figure 3.10: Values of VQ codes (128 codes).

3.4.7 Fuzzy vector quantization

The Fuzzy Vector Quantization (FVQ) [Tseng *et al.*, 1987] is a variant of the Vector Quantization (VQ) algorithm that produces the observation alphabet used in HMM. The Vector Quantization algorithm stores a set of reference vectors (codewords): $\nu_1, \nu_2, ..., \nu_P$ and those vectors are used as alphabet in the HMM. In standard use of VQ, a speech input vector x_i is replaced by to one of the ν_j using a pre-chosen distance $d(\boldsymbol{x}_i, \nu_j)$. In contrast, the FVQ maps an input vector $\boldsymbol{x}_i$ into an mass vector $O_i = (m_{i1}, m_{i2}, ..., m_{iP})$ where,

$$m_{ij} = \left[\sum_{k=1}^{G} \left(\frac{d(x_i, \nu_j)}{d(x_i, \nu_k)} \right)^{\frac{1}{F-1}} \right]^{-1}.$$

The vector O_i minimizes the function:

$$\sum_{i=1}^{T} \sum_{j=1}^{G} m_{ij}^{F} d(x_i, \nu_j),$$

thus taking into account distances to competing reference vectors. G is the number of fuzzy and $F > 1$ is a constant called the degree of fuzziness. When $F = 1$, the decision is known as Hard (Hard VQ). The Fuzzy vector quantization is an attempt to "soften" the vector quantization approach by enabling competing prototypes to be included in the matching distance. In contrast to the standard VQ technique, the relative number of prototypes belonging to a given category are likely to influence the final decoding. Consequently, the Fuzzy VQ can viewed as an attempt to take into account the probability distribution of the data within the VQ process.

Let us consider a set of observations $\boldsymbol{x}_1^T$. According to the FVQ algorithm, each $\boldsymbol{x}_t$ is now a probabilistic mass vector. Consequently, the Forward algorithm and the Forward Backward algorithm are modified by replacing $b_{ij}(\boldsymbol{x}_t)$ with:

$$\omega_{ij}(\boldsymbol{x}_t) = \sum_{k=1}^{G} m_{tk} b_{ij}(\boldsymbol{x}_t),$$

which is the probability of observing $\boldsymbol{x}_t$ when taking the transition $i \rightarrow j$.

3.4.8 Preliminary experiment on the /b, d, g/ task

The system was first tested on the recognition of /b, d, g/ phonemes. The same /b, d, g/ task (and database) on which the TDNN became known for in [Waibel *et al.*, 1987]. The /b, d, g/ sounds are categorized as voiced stops and characterized by a short duration, albeit bearing spectral peak in their structures. They bear acoustically similar structures which makes them difficult to discriminate by Maximum Likelihood-based approaches.

Table 3.2 shows the number of token used for training the 3 HMM. Note that the testing and training conditions are rather well balanced.

Table 3.2: Training and testing data of the $/b, d, g/$ task.

Phoneme	number of tokens		number of frames	
	Training	Testing	Training	Testing
b	227	218	1632	1558
d	202	179	1475	1330
g	259	251	1877	1785

VQ distortion and recognition rate

The Fuzzy-VQ (FZVQ) algorithm and the Hard-VQ (HDVQ) were both implemented. For the FZVQ, we chose the number 1.6 as the fuzziness value, setting the number of fuzzy to 6. The results, as a function of a codebook size, are shown in Table 3.3.

Table 3.3: Recognition rate based Fuzzy-VQ on the $/b, d, g/$ task using TDNN-features.

Iterations = 7
Fuzziness = 1.6
Number of fuzzy = 6

codebook size	b		d		g		Average rec. rate	
	FZVQ	HDVQ	FZVQ	HDVQ	FZVQ	HDVQ	FZVQ	HDVQ
64	98.6	97.7	96.6	96.6	95.6	95.6	96.9	96.6
128	98.2	98.2	97.2	97.2	95.6	96.0	97.0	97.1
256	98.6	97.7	97.2	96.6	96.0	96.8	97.3	97.0

The recognition performance differs according to the algorithm in used (HDVQ or FZVQ), even though both methods display rather competing results.

Contrasting the performance against the size of the codebook is useful to comprehend the utility of the codebook size. The lowest average distortion was produced in the codebook of size 256. However, it can be observed that the highest recognition rate, when using HDVQ, occurred with the codebook of size 128, which confirms the fact the recognition rate is not an monotonic function of the distortion. The same analysis is valid when using FVZQ in recognizing /b/. This actually make sense, if one keeps in mind that reducing the distortion is not directly link with error minimization. The highest average recognition rate (97.27%) is achieved by FZVQ, with a codebook size of 256. On average, these results show the effectiveness of FZVQ in contrast to the HDVQ.

Comparison with standard features

Here, the performance of standard features on the same task is examined. Standard features consist of 3 different speech parameterization methods: the first is the Weighted Likelihood Ratio (WLR)

(using a 12-order cepstrum and autocorrelation coefficients) with a codebook of size 256, the second is the power with a codebook size of 64, and the third is the Δcepstral (differential cepstrum) with codebook size 256. That is, the resulting features vectors is composed of 37 (12 + 12 + 12 + 1) parameters. Each codebook was generated from the same 216 words used in the LTDNN-generated codebook.

The WLR-based distance measure [Sugiyama, 1981] has been developed to be sensitive to spectral peak such as formants. The distance is defined as

$$d^2{}_{wlr} = \sum_{i=1}^{12}(r_i{}^{(f)} - r_i{}^{(r)})(c_i{}^{(f)} - c_i{}^{(r)}), \tag{3.17}$$

where r_i is an autocorrelation feature and c_i is a cepstral feature. The labels f and r correspond to an input frame and a prototype of the codebook, respectively. The delta cepstral parameters try to capture frame transition dynamics. It is an attempt to overcome the Markov assumption which states that the output probability of the frame depends only on the current state whereas it is acknowledged that frame spectral characteristics depend on neighboring frames [Furui, 1986]. The delta cepstral distance is simply an Euclidean distance between delta cepstra. That is,

$$d^2{}_{\Delta cep} = \sum_{i=1}^{12}(\Delta c_i{}^{(f)} - \Delta c_i{}^{(r)})^2. \tag{3.18}$$

Similar to the $d^2{}_{\Delta cep}$, the power distance $d^2{}_{pow}$ is also an Euclidean distance between the power of the two corresponding frames.

Since a feature vector is composed of non-homogeneous parameters, a proper distance need to be defined. As proposed by Furui [Furui, 1986], the distance used here was defined as

$$d = td^2{}_{wlr} + (1-t)d^2{}_{\Delta cep} + d^2{}_{\Delta cep}, \tag{3.19}$$

where t is a controlling parameter, here chosen to be equal to 0.5. The features presented above focus on trying to capture "interesting phenomena" within the waveform and uses a priori knowledge, while TDNN-features focus on discrimination.

Table 3.4: Comparison of recognition rates in the $/b, d, g/$ task.

HDVQ: Hard-VQ with 256 size codebook
FZVQ: Fuzzy-VQ with 256 size codebook

Features	Feature size	b	d	g	Average rate
LTDNN-features (HDVQ)	24	97.7	96.6	96.8	97.0
LTDNN-features (FZVQ)	24	98.6	97.2	96.0	97.3
Cepstral features	37	95.6	97.2	94.4	95.6
Mel-energies on LTDNN	16	98.2	98.3	99.6	98.7

Table 3.4 shows the comparative result of different features. Clearly, even with the lowest dimension, discriminant features achieves better performance than classical features. TDNN results

are also displayed on the same table. As expected the TDNN performance is the best, which seems to confirm that the MSE-trained TDNN displays higher discrimination than MLE-trained HMM. A fairer comparison with the TDNN should be done by using the output of the third layer as features, which enables a comparison of the TDNN last layer's classifier ability with the HMM classifier, using the same feature type. Again, the aim of this investigation is about continuous speech recognition, which is better tackled within the HMM framework than with Neural Network. The results obtained on the $/b, d, g/$ task suggest that it is possible to use the discriminant power of the TDNN (in the form of discriminant features), to improve the HMM performance at the acoustic level.

3.4.9 Recognition of all Japanese phonemes

Encouraged by the results from this first experiment, the system was applied to additional consonant clusters, which cover the entire phoneme set for the Japanese language. Each of the phoneme clusters was treated in a manner similar to the phonemes $/b/, /d/$ and $/g/$. TDNN results were kindly provided by Dr. H. Sawai.

The recognition rates are shown in Table 3.5, which displays the results of LTDNN-features using the Fuzzy-VQ-based algorithm and cepstral features for all the Japanese phonemes. Clearly, the discriminant features, extracted from the LTDNN, achieve a higher score than conventional cepstral coefficients for all phonemes except $/r/$, which may be due an insufficient number of $/r/$ representatives in the codebook. However, generally, these results confirm the ones obtained in the smaller $/b, d, g/$ task and provide a rather simple example of how making use of discriminant features can significantly improve the performance of a speech recognizer.

Comparison with LTDNN

Although the prime goal was to improve the recognition accuracies of the HMM at the acoustic level, a simple comparison with the LTDNN accuracies on the same task is examined. Using LTDNN's classification result as features for HMM, can be viewed as a form of post-processing of the output of the LTDNN. The main reason for this comparison is to investigate how this post-processing scheme compares to the result without post-processing.

Here, recognition rates calculated within each phonemic cluster is contrasted with the performance of the system that considers all phonemes as a possible recognition candidates. These recognition schemes are motivated by the training of the LTDNN, which, as shown previously, was trained using subnets aiming at discriminating within each phonemic class. The performances are shown in Table 3.7 for Hard-VQ and in Table 3.6 for Fuzzy-VQ.

In Table 3.6, the LTDNN-HMM system shows a higher recognition rate on the nasals $/m/, /n/, /N/$, the fricatives $/s/, /sh/, /h/, /z/$ and the vowels $/a/, /i/, /u/, /e/, /o/$ than the LTDNN. This is a rather surprising result because post-processing is expected to introduce a non-optimality within the system since the LTDNN-HMM system was not *globally* optimized under the same criteria (as

Table 3.5: Recognition result for all phonemes using Fuzzy VQ.

phoneme	LTDNN-features (size 24)			Cepstral features (size 37)	
	errors /token	recognition rate(%)	average rate(%)	recognition rate(%)	average rate(%)
b	8/218	96.3		88.5	
d	9/179	95.0	92.6	97.2	88.3
g	34/251	86.5		79.4	
p	3/33	90.9		60.0	
t	29/400	92.8	94.3	94.5	81.9
k	3/400	99.3		91.4	
m	6/400	98.5		75.5	
n	0/260	100.0	99.5	86.4	85.9
N	0/151	100.0		92.2	
s	0/400	100.0		95.2	
sh	0/310	100.0	100.0	99.4	88.3
h	0/214	100.0		91.3	
z	0/116	100.0		67.4	
ch	1/134	99.3	99.4	97.2	97.2
ts	1/212	99.5		97.2	
r	130/400	67.5		78.8	
w	0/72	100.0	88.9	98.7	92.3
y	1/159	99.4		99.4	
a	1/400	99.8		98.2	
i	4/122	96.7		84.2	
u	7/223	96.2	98.4	72.7	89.4
e	4/400	99.0		94.7	
o	1/400	99.8		97.2	

done within the LTDNN structure) and thus should be largely less optimal than the LTDNN. As expected, however, the LTDNN has the average best performing over the whole set.

Contrasting HZVQ and HDVQ, the recognition results using Hard-VQ HMM are shown in Table 3.7 and show that the Fuzzy-VQ algorithm performed better than the standard VQ algorithm in average. This is similar to the result obtained on the smaller $/b, d, g/$ task.

At last, an average comparison of LTDNN -features, cepstral features and the LTDNN is summarized in Table 3.8.

Discriminant features provides higher recognition rate than the conventional cepstral parameters (7.1% increase in recognition rate) as well as a better dimensionality reduction in both approaches (HDVQ and FZVQ), which again, clearly confirms the importance of using discriminant features in pattern recognition task.

Table 3.6: Recognition rates within phonemic-categories: comparison using Fuzzy VQ.

phoneme	LTDNN-HMM-FZVQ			LTDNN
	errors /token	recognition rate(%)	average rate(%)	average rate(%))
b	3/218	98.6		
d	5/179	97.2	97.2	98.6
g	10/251	96.0		
p	2/33	93.9		
t	24/400	94.0	95.8	98.7
k	2/400	99.5		
m	3/400	99.3		
n	0/260	100.0	99.7	96.6
N	0/151	100.0		
s	0/400	100.0		
sh	0/310	100.0	100.0	99.3
h	0/214	100.0		
z	0/116	100.0		
ch	0/134	100.0	99.7	100.0
ts	1/212	99.5		
r	120/400	70.0		
w	0/72	100.0	90.0	99.9
y	0/159	100.0		
a	1/400	99.8		
i	1/122	99.2		
u	4/223	98.2	99.3	98.6
e	2/400	99.5		
o	0/400	100.0		

Table 3.7: Recognition rate within phonemic-categories: comparison using Hard-VQ.

phoneme	LTDNN-HMM-HDVQ			LTDNN
	errors /token	recognition rate(%)	average rate(%)	average rate(%)
b	8/218	96.3		
d	9/179	95.0	92.7	98.6
g	33/251	86.9		
p	5/31	84.8		
t	30/400	92.5	92.2	98.7
k	3/400	99.3		
m	11/400	97.3		
n	0/260	100.	98.6	96.6
N	2/151	98.7		
s	0/400	100.0		
sh	0/310	100.0	99.4	99.3
h	1/214	99.5		
z	2/116	98.3		
ch	1/134	99.3	99.4	100.0
ts	1/212	99.5		
r	132/400	67.0		
w	0/72	100.0	83.0	99.9
y	4/159	100.0		
a	1/400	99.8		
i	3/122	97.5		
u	8/223	96.4	98.5	98.6
e	4/400	99.0		
o	1/400	99.8		

Table 3.8: Average recognition rates for the 4 feature types

LTDNN-features (HDVQ)	95.6
LTDNN-features (FZVQ)	96.1
Cepstral features (FZVQ)	89.0
Mel energies (LTDNN)	98.6

3.4.10 Discussion

The main goal in this chapter was to show how simple discriminant features can be used to improve performance, in the conventional framework of speech recognizer design. It seems obvious to use time derivatives in order to take into account the high variability of speech in the time axis. This experiment shows that performance highly depends on the quality of features. The features used here were produced through an MSE-based feature extraction system and consisted of few data in the sense that LTDNN-generated vector is composed of values ranging between 0 and 1. It was managed to represent one speech unit (a phoneme) by just a few values, even though the performance still depends on the codebook size of the discrete HMM.

This research was primarily aimed at continuous speech recognition. Again, the HMM has shown an ability in coping with the variability of speech in time axis. The performance of the HMM in phoneme recognition was improved by making use of discriminant features, which is a first step towards achieving better performance in continuous speech.

The proposed system can also be viewed an as integration of the Neural Network and HMM under a system which takes advantage of both systems. Several approaches have been proposed to try to combine HMM and NN while keeping the advantages of the two systems in order to improve overall speech recognition performance. Some studies have to do with the connectionist interpretation of HMM algorithms [Niles and Silverman, 1990; Bridle, 1990; Young, 1990]. Others, as in the approach described here, have tried to use the discriminant power of NN for accurate recognition by HMM [Morgan and Bourlard, 1990; Katagiri and Lee, 1990; Weiye and Compornelle, 1990]. [Morgan and Bourlard, 1990] and [Weiye and Compornelle, 1990] have used NN to estimate frame probabilities within HMM states. Katagiri and Lee [Katagiri and Lee, 1990] have replaced the standard use of a codebook by a discriminant codebook generated by Linear-Vector Quantization. Even if the above approaches have achieved reasonable results to some extent, the prime goal in this chapter, was to introduce a rather simple method to derive discriminant features and show how they can significantly improve recognition performance at the acoustic level: the focus is on the front-end.

One drawback, so to say, within this framework is that the criterion on which the feature extraction is optimized (MSE) is not the same criterion used to optimize the HMM classifier (MLE). An optimal approach is to optimize the LTDNN and the HMM together under a single criterion as was done by Bengio [Bengio *et al.*, 1995]. This can also be carried out under the framework proposed by Bottou and Gallinari [Bottou and Gallinari, 1991], which describes a back-propagation-like algorithm for optimizing different system structures under a single criterion. However, an appropriate optimization criterion must be chosen, which can reduce the errors of the system more directly than MLE or MSE. Chapter 4 proposes such a criterion and describes a methodology for extracting discriminative features in a more efficient manner.

Part II

Formalization

Chapter 4

Discriminative Feature Extraction for Minimum Error

Everything should be made as simple as possible but not simpler

-Albert Einstein-

*Practical recognizers (*recognition *systems) are modular systems, consisting of a feature extractor (*feature extraction *module) and a classifier (*classification *module). The process of feature extraction is traditionally based on some objective that is different from the minimization of recognition errors. This chapter introduces the formalism for Discriminative Feature Extraction. The Discriminative Feature Extraction is described as a framework for designing the overall speech recognizer in a unified manner, so as to minimize the errors of the overall recognition system.*

4.1 Introduction

Feature extraction is a process which converts each input pattern to salient features, while classification is a process which associates a class label to the input pattern as represented in the feature space. Finding the right features for best classification results has been a debate over years. Statistics-based methods such as principal component analysis and the Karhunen-Loeve expansion have been extensively studied in a wide range of pattern recognition applications [Duda and Hart, 1973]. Scientific knowledge has also been well utilized. For instance, in the speech recognition area, features that bear psychoacoustic or physiological significance based on studies in the humans auditory system and hearing process have often been discussed [Seneff, 1986; Ghitza, 1991; Pathasarathy and Cooker, 1992; Shroeter and Sondhi, 1994]. Other mathematical models such as linear predictive coding (LPC) have received equal enthusiasm in speech analysis and representation [Itakura, 1975]. As seen in the first chapter, conventional features have to deal with the fact that there is little or no design consistency between the feature extraction and the classification task.

Given the observation space $\boldsymbol{\Omega}_s$ and $\boldsymbol{s} \in \boldsymbol{\Omega}_s$, a conventional feature extractors $\mathcal{F}_c$, performs $\boldsymbol{x} = \mathcal{F}_c(\boldsymbol{s})$, based on a criterion which is different from the classification criterion.

Briefly stated, conventional approaches imply that

1. There is no optimal interaction between feature extractor and classifier because the feature extraction criteria may not be consistent with the error criteria used in selection of the classifier.

2. Knowing that the probability of error is the best estimate of a performance of a system, classical error criteria have no direct link with minimum-error classification.

Although the conventional way has led to successful results to some extent, it is obvious that there is still plenty of room for improvement. Due to the lack of interaction between the feature extractor design and the classifier design, the conventional way does not guarantee that a resulting feature representation is the best for the post-end classification process.

For optimal pattern recognition achievement, feature extractor and classifier have to be chosen so that *both* participate in the optimal labeling of the original data: the feature extractor should be selected based on its discriminant abilities, given the data which could be realized by a supervised choice of the feature extractor, given the data. An objective function directly related to the error rate is the ideal criterion to select the feature extractor as well as the classifier, leading to an optimal recognizer.

In this chapter, a framework for the design of practical and transparent recognizer in which the feature extractor is optimally matched to the classifier, is described. This approach has been referred to as *Discriminative Feature Extraction* (DFE) [Biem and Katagiri, 1993b; Katagiri *et al.*, 1993] since its basis is a recent discriminative learning methodology, called the Minimum Classification Error/Generalized Probabilistic Descent method (MCE/GPD) [Katagiri *et al.*, 1990;

Juang and Katagiri, 1992a]. The DFE concept is simple and quite natural; that is, assuming that the "best" features, *given a task and a recognizer structure*, are the ones that yield the lowest recognition (classification) error rate, DFE embeds all the adjustable parameters of the feature extractor and the classifier in a unified *smooth* (at least the first differentiable with respect to the adjustable recognizer parameters) functional form that is suitable for a practical gradient-based optimization algorithm, and searches for optimal parameter values through the GPD algorithm [Biem *et al.*, 1993; Biem *et al.*, 1997].

The first section of this chapter is concerned with the formalism of the DFE method as an extension to the MCE/GPD's theory of discriminative training applied to feature optimization. Consequently, the MCE criterion is described in details and compared to the Bayes error. The GPD is introduced as an efficient optimization technique. Comparison between the MCE criterion and other discriminative criteria is given in the last section.

4.2 Feature Extraction Based on Minimum Classification Error

In this section, the Discrimination Feature Extraction (DFE) approach is introduced as a joint optimization of the feature extraction and the classifier by discriminative training aiming at minimum error.

4.2.1 Formalization

The pattern recognizer is assumed to be a modular system consisting of a front-end feature extractor and a back-end classifier. The first step is to parameterize the feature extractor with a set of parameter Θ. Let $\boldsymbol{s}$ be the measured signal or observed pattern. The feature extractor with parameter Θ is a function which maps $\boldsymbol{s}$ to a corresponding feature pattern $\boldsymbol{x}$; i.e., $\mathcal{F}_\Theta(\boldsymbol{s}) = \boldsymbol{x}$. The feature pattern $\boldsymbol{x}$ is the input accepted by the classifier $a(\cdot)$ which operates under the following decision rule

$$a(\boldsymbol{x}) = C_i \qquad \text{if} \qquad i = \arg\max_j g_j(\boldsymbol{x}; \Lambda), \tag{4.1}$$

where $g_j(\boldsymbol{x}; \Lambda)$ is the discriminant function which indicates the degree to which $\boldsymbol{x}$ belongs to C_j and is defined by the classifier parameter (set) Λ. In certain context (e.g, distance-based classifier), the decision rule should be based on the minimum value of the discrimination function. However, to avoid confusion, the formulation of (4.1) is kept.

As said before, the feature extractor $\mathcal{F}_\Theta$ is traditionally determined by empirical means independent of the classifier design $\{g_j(\boldsymbol{x}; \Lambda)\}$. In other words, the parameter Θ, is selected independently of the classification task. The purpose here is to integrate the function of feature extraction with the classifier function. Obviously, substituting $\boldsymbol{x}$ by $\mathcal{F}_\Theta(\boldsymbol{s})$ in (4.1), provides equivalent integrated classifier design defined by a set of discriminant functions $\{g_j(\mathcal{F}_\Theta(\cdot); \Lambda)\}$ in which $g_j(\mathcal{F}_\Theta(\boldsymbol{s}); \Lambda) = g_j(\boldsymbol{s}; \Phi)$ with equivalent parameter set Φ. That means working directly with a recognizer defined by a super parameter set Φ . This is likely to be more effective without assuming

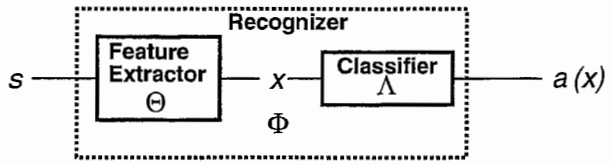


Figure 4.1: Model of a DFE-based Recognizer. The Recognizer can be viewed as a modular system in which the feature extractor is parameterized.

separability $\Phi = \{\Theta, \Lambda\}$.

In the DFE framework, the *recognizer* makes the classification decision by the rule

$$a(\boldsymbol{s}) = C_i \qquad \text{if} \qquad i = \arg\max_j g_j(\boldsymbol{s}; \Phi), \tag{4.2}$$

where $g_j(\boldsymbol{s}; \Phi)$ is the discriminant function which indicates the degree to which $\boldsymbol{s}$ belongs to C_j and is defined by the super-parameter (set) Φ. Obviously, the decision framework of (4.2) permits the feature extractor to participate in the classification decision, enabling the feature to be *discriminant.* Hence, the name Discriminative Feature Extraction (DFE) was chosen to describe this process. For clarity, the assumption of a modular design is maintained throughout this report. i.e., $\Phi = \{\Theta, \Lambda\}$, Φ refers to the overall recognizer, Θ represents the feature extractor and Λ refers to the classifier. Fig. 4.1 illustrates this modular approach to pattern recognition in the DFE framework.

Contrary to conventional methods, features in the DFE framework are generated while focusing on the final performance of the recognizer, assuming an interaction between classification and feature extraction. This ensures that the designed recognizer will itself map an observation input to a more suitable feature pattern for an easier classification.

The DFE formalism can be regarded as if the classifier directly acts on the input space, with a discriminant function of the form $\{g_j(\mathcal{F}_\Theta(\cdot); \Lambda)\}$, which helps reducing its complexity. The implication of substituting Λ by Φ on the Bayes error is that the Bayes error is now defined on the input-space $\boldsymbol{\Omega}_s$, which enables to control the loss of information during the feature extraction process.

The first question arisen is which parametric family of feature extraction functions is of interest. The answer to this question relies on our empirical knowledge, which helps determining the degree of freedom in the design of the feature extractor structure. In most of the cases, this degree of freedom can be represented by a set of free parameters in the feature extraction, which are subject to optimization.

4.2.2 Minimum Classification Error estimation

The key contribution of the DFE training is to make possible the joint optimization of the trainable parameters of the feature extractor and the classifier for a single objective function. The feature extraction operation $\mathcal{F}(\cdot)$ is embedded in the discriminant function $g_i(\cdot)$, which enables this joint

optimization by back-propagating the derivative of the objective function to the feature extractor according to the chain rule of differentiation calculus.

The natural question concern the choice of the objective function to be considered. Again, as seen before, the best criterion for the pattern recognition design is the probability of error or error rate. Hence, error minimization seems to be the natural target in the design of the pattern recognition.

In chapter 1, various design strategies were reviewed. In this chapter, the recently proposed Minimum Classification Error/Generalized Probabilistic Descent method, widely known as MCE/GPD is introduced, since MCE/GPD has been the main framework for DFE implementation.

MCE/GPD refers to minimizing the Minimum Classification error (MCE) objective function within the family of the discriminative training approaches known as the Generalized Probabilistic Descent methods (GPD). GPD was originally proposed by Katagiri in [Katagiri *et al.*, 1991b] as a framework for realizing discriminative training by use of the probabilistic descent theorem proposed by Amari [Amari, 1967]. Amari's probabilistic descent was concerned with the classification of static pattern which was rather impractical for dynamic pattern such as speech. [Katagiri *et al.*, 1991b] extended the probabilistic descent method to the Generalized Probabilistic Descent suitable to handle dynamic patterns. The new framework provided 1) a new probability measure for dynamic patterns 2) a functional form for discrimination, which makes use of the discrimination functions 3) a cost (or lost) function that approximates the empirical error rate 4) an optimization procedure with proof of convergence. In a latter paper by Juang and Katagiri [Juang and Katagiri, 1992a], it was shown that GPD asymptotically approximates the Bayes error. The method was dubbed Minimum Classification Error (MCE), with the term GPD reserved for the optimization scheme. Although, the two terms GPD and MCE are sometimes used interchangeably, throughout this report, the latest terminology is followed. That is, MCE refers to the objective function and GPD to the training scheme.

Minimization of the error rate requires somehow that a tractable form of the error rate be available and suitable to minimization by standard method such as gradient descent. However, as seen before, the 0-1 loss $\ell(a(\boldsymbol{x}) = C_j|C_k)$ is an intrinsically discontinuous classification loss: classification is either correct or incorrect, which makes gradient-based methods inapplicable.

In an earlier attempt to directly minimize the error rate, corrective training [Bahl *et al.*, 1988] has been proposed as an error correction strategy designed to deal with the misclassification problem encountered in the MLE-based approach. In corrective training, the parameters of the competing models are "corrected" heuristically if the likelihood of the correct model is not the best. Even if clear gains have been obtained comparatively to MLE, the lack of a rigorous theoretical framework is a fundamental problem in this approach.

The MCE's response to error minimization is simply to use a smooth approximation to the zero-one classification loss function, which is close to zero when no error is detected and close to one when there is an error. Again, such a loss function is referred to as MCE loss and its minimization is thus a direct way of minimizing the actual number of misclassifications (empirical error rate).

Asymptotically, as is shown below, minimizing the smooth zero-one loss function yields a classifier equivalent to the Bayes classifier [Juang and Katagiri, 1992a].

Misclassification measure

The first step in defining the smooth 0-1 error function is the notion of *misclassification measure.* The misclassification measure is a value whose *sign* indicates the correct or incorrectness of a classification decision. Consequently, for a given classifier parameter set Λ and a given feature pattern $\boldsymbol{x}$, the misclassification measure should depend on the relative values of the discriminant functions. In a first attempt to define such a measure, Amari [Amari, 1967], proposed the following misclassification measure, for a feature pattern $\boldsymbol{x}$ of category C_k:

$$d_k(\boldsymbol{x};\Lambda) = \frac{1}{N_k} \sum_{j \in \mathcal{A}_k} (g_j(\boldsymbol{x},\Lambda) - g_k(\boldsymbol{x};\Lambda)). \tag{4.3}$$

where the set $\mathcal{A}_k = \{j | g_j(\boldsymbol{x};\Lambda) > g_k(\boldsymbol{x};\Lambda)\}$ consists of the N_k categories whose discriminant function value is higher than that of the correct category C_k. d_k is assumed to be zero when the set $\mathcal{A}_k$ is empty. This misclassification is clearly motivated by the decision rule of (4.1). Its value is positive for incorrect classifications, and zero otherwise. This misclassification measure, however, have few shortcomings. 1) For a 2-class problem, in the context of simple distance-based classifier, Amari's misclassification measure is smooth with regard to the recognizer parameter Λ. However, in most cases, the set $\mathcal{A}_k$ is not fixed and changes according to Λ, thus within a learning algorithm which updates Λ, Amari's discrimination measure displays discontinuity 2) this misclassification measure does not provide any gradual information concerning "how bad" a given decision is, which might be desirable in certain circumstances.

Consequently, Katagiri in [Katagiri *et al.*, 1991a; Katagiri *et al.*, 1991b] and Juang and Katagiri [Juang and Katagiri, 1992a] suggested the following misclassification measure:

$$d_k(\boldsymbol{x};\Lambda) = -g_k(\boldsymbol{x};\Lambda) + \left[\frac{1}{M-1} \sum_{j \neq k} g_j(\boldsymbol{x};\Lambda)^{\nu} \right]^{\frac{1}{\nu}}, \tag{4.4}$$

for a feature pattern $\boldsymbol{x}$ that belongs to category C_k and is presented to the classifier during training. M is the number of categories involved and ν is a positive constant.

Misclassification measure in the DFE framework

In the DFE framework,

$$d_k(\boldsymbol{x};\Lambda) = d_k(\mathcal{F}_\Theta(\boldsymbol{s});\Lambda) = d_k(\boldsymbol{s};\Phi),$$

which gives the following expression for the misclassification measure:

$$d_k(\boldsymbol{s};\Phi) \;=\; -g_k(\boldsymbol{s};\Phi) + g_{\overline{k}}(\boldsymbol{s};\Phi), \tag{4.5}$$

where $g_{\overline{k}}(\boldsymbol{s};\Phi)$ represents the competing discriminant function to category C_k.

Various MCE approaches can be implemented by the selection of $g_{\overline{k}}(\boldsymbol{s};\Phi)$ as shown below.

Soft MCE

The general form as proposed by Katagiri in [Katagiri *et al.*, 1991b; Katagiri *et al.*, 1991a] is

$$g_{\overline{k}}(\boldsymbol{s};\Phi) = \left[\frac{1}{M-1}\sum_{j\neq k} g_j(\boldsymbol{s};\Phi)^{\nu}\right]^{\frac{1}{\nu}}.$$

For $\nu = 1$, $g_{\overline{k}}(\boldsymbol{s};\Phi)$ is simply the average over the competing categories to the correct category. Thus, learning estimates the boundaries between the correct category and other categories as represented by their average, which might be an appropriate misclassification measure when the number of training data is relatively small, and a form of smoothing is required. This form of misclassification measure is said to be "soft" and is referred to as *soft MCE*.

For a large ν, $g_{\overline{k}}(\boldsymbol{s};\Phi)$ is close to the value of the discriminant function of the best incorrect category. That is,

$$g_{\overline{k}}(\boldsymbol{s};\Phi) \approx \max_{j\neq k} g_j(\boldsymbol{s};\Phi), \tag{4.6}$$

which is the *strict MCE* case.

Strict MCE

The strict MCE attempts to discriminate against the most competing category. The competing category is thus defined as

$$g_{\overline{k}}(\boldsymbol{s};\Phi) = \max_{j\neq k} g_j(\boldsymbol{s};\Phi).$$

In this implementation, learning occurs for the two closest competing models i.e., a direct comparison of the correct category with the best matching incorrect category (which is thus represented by the subscript $\overline{k}$) in a manner similar to a two-class problem. Consequently, during training, MCE estimates the classification regions by comparing pair of competing models. Clearly, the sign of this misclassification measure directly reflects the classification decision for all classes.

The strict MCE case is close to the 0-1 classification scheme, since comparison with the most competing incorrect category directly evaluates the wrongness or goodness of a classification decision. In this context, the misclassification measure is a monotonic function of the error. However, in practice, the strict MCE case can also result into a model that overfits the training data. That is, a good performance on the training may result in poor performance on the testing set. This is the well-known generalization problem, which may be alleviated by the soft MCE scheme. By controlling ν, one is given various degree of discriminative training implementations, which shows the flexibility of the MCE-based discriminative training formulation.

MCE loss function

It was seen that for a feature pattern $\boldsymbol{x}$, the minimum error loss is

$$\ell(\boldsymbol{x};\Lambda) = \begin{cases} 1 & \text{if } \boldsymbol{x} \text{ is incorrectly classified} \\ 0 & \text{otherwise.} \end{cases} \tag{4.7}$$

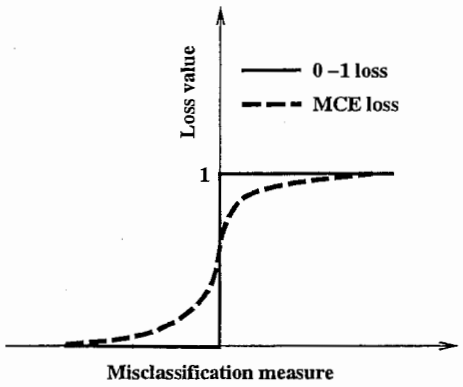


Figure 4.2: The MCE loss function is a continuous approximation of the 0-1 loss function.

The MCE approach approximates this ideal loss by a smooth, differential function and then apply this function to the misclassification. That is, for an input feature-data $\boldsymbol{x}$ belonging to category C_k, the MCE loss is defined as

$$\ell_k(\boldsymbol{x};\Lambda) = \ell(d_k(\boldsymbol{x};\Lambda)). \tag{4.8}$$

This loss definition, which depends on the feature-data $\boldsymbol{x}$ and the classifier Λ, directly embeds the classification decision through the use of the misclassification measure. The concept of the MCE loss is illustrated in Fig.4.2. The MCE loss is a continuous approximation of the minimum-error loss function as derived from the Bayes decision-theoretic approach.

Extension of MCE to DFE is simply made by replacing the feature-data $\boldsymbol{x}$ by its corresponding input-data $\boldsymbol{s}$. That is, in DFE framework, the loss is defined as

$$\begin{aligned} \ell_k(\boldsymbol{s};\Phi) &= \ell(d_k(\mathcal{F}_\Theta(\boldsymbol{s});\Lambda)) && (4.9)\\ &= \ell(d_k(\boldsymbol{s};\Phi)). && (4.10) \end{aligned}$$

The above DFE loss embeds both the classification decision and the feature extraction, enabling DFE to gather the overall recognizer resources for a single objective. The subscript k in defining the loss signifies that the loss can be made class-dependent. A general form of the loss is defined as

$$\ell(\boldsymbol{s};\Phi) = \sum_{k=1}^{M} \ell_k(\boldsymbol{s};\Phi)\mathbf{1}(\boldsymbol{s}\in C_k). \tag{4.11}$$

Objective function

The objective function of DFE is the expected loss:

$$\begin{aligned} \mathcal{L}(\Phi) &= E_{\boldsymbol{S}}\left[\ell(\boldsymbol{s};\Phi)\right] && (4.12)\\ &= \sum_{k}^{M} \Pr(C_k) \int_{\boldsymbol{\Omega}_s} \ell(\boldsymbol{s};\Phi)\mathrm{p}(\boldsymbol{s}|C_k)d\boldsymbol{s}, && (4.13) \end{aligned}$$

where $\Pr(C_k)$ and $\mathrm{p}(\boldsymbol{s}|C_k)$ are the class a priori and conditional probabilities, respectively. Minimization of $\mathcal{L}(\Phi)$ is the target of the DFE's estimation. That is, finding the parameter $\Phi^\star$ such that

$$\Phi^\star = \arg\min_\Phi \mathcal{L}(\Phi). \tag{4.14}$$

The uniqueness of a set of parameter satisfying (4.14) depends on the structure of the recognizer. e.g., the set $\{g_j(\boldsymbol{s};\Phi)\}$ and the input space $\boldsymbol{\Omega}_s$.

Note that, in contrast to classical use of MCE, the DFE's objective function is defined over the original input space and is a functional of the recognizer parameter set whereas the classical MCE's objective function is restricted to the classifier parameter set and depends on a particular feature extractor. The classical MCE's objective function is

$$\mathcal{L}_{\boldsymbol{x}}(\Lambda) = E_{\boldsymbol{x}}\left[\ell(\boldsymbol{x};\Lambda)\right] \tag{4.15}$$

$$= \sum_k^M \Pr(C_k) \int_{\boldsymbol{\Omega}_x} \ell(\boldsymbol{x};\Lambda)\mathrm{p}(\boldsymbol{x}|C_k)d\boldsymbol{x}. \tag{4.16}$$

4.3 MCE Estimation and Bayes Error Rate

This section investigates the link between MCE's estimation and the Bayes error rate. For simplicity, the classical MCE formulation is used by expressing both MCE and Bayes error rate on the feature space $\boldsymbol{\Omega}_x$.

The error rate of the Bayes minimum-error classifier resulting from the maximum a posteriori rule is

$$\mathcal{E}_{Bayes}[\boldsymbol{\Omega}_x] = \sum_{k=1}^M \int_{\Omega_x^{(k)}} \Pr(\boldsymbol{x}, C_k)\mathbf{1}(\boldsymbol{x} \in C_k)d\boldsymbol{x}, \tag{4.17}$$

where

$$\Omega_x^{(k)} = \{\boldsymbol{x} \in \boldsymbol{\Omega}_x | \Pr(\boldsymbol{x}, C_k) \neq \max_j \Pr(\boldsymbol{x}, C_j)\}.$$

This error rate can be re-written as

$$\mathcal{E}_{Bayes}[\boldsymbol{\Omega}_x] = \sum_{k=1}^M \int_{\boldsymbol{\Omega}_x} \Pr(\boldsymbol{x}, C_k)\mathbf{1}(\boldsymbol{x} \in C_k)\ell_{Bayes}(\boldsymbol{x}, C_k)d\boldsymbol{x}, \tag{4.18}$$

in which $\ell_{Bayes}(\boldsymbol{x}, C_k)$ is the 0-1 cost incurred for a correct/incorrect classification i.e.,

$$\ell_{Bayes}(\boldsymbol{x}, C_k) = \begin{cases} 0 & \text{if } \Pr(\boldsymbol{x}, C_k) = \max_j \Pr(\boldsymbol{x}, C_j) \\ 1 & \text{otherwise.} \end{cases}$$

In the MCE formalism, $\ell_{Bayes}(\boldsymbol{x}, C_k)$ is implemented as the 0-1 step function applied to the misclassification measure of the Bayes minimum-error classifier. That is,

$$\ell_{Bayes}(\boldsymbol{x}, C_k) = 1\left(d_k(\boldsymbol{x}) > 0\right),$$

where

$$d_k(\boldsymbol{x}) = -\Pr(\boldsymbol{x}, C_k) + \max_{j, j\neq k} \Pr(\boldsymbol{x}, C_j),$$

4.3.1 Approximating the Bayes error

In practice, the joint probabilities $\Pr(\boldsymbol{x}, C_k)$ are rarely known and the classifier relies on its discriminant function $g_k(\boldsymbol{x}; \Lambda)$ to estimate the Bayes classifier discriminant function. Thus, in practice, the discriminant measure of a feature-pattern $\boldsymbol{x}$ of class C_k is expressed as

$$d_k(\boldsymbol{x}; \Lambda) = -g_k(\boldsymbol{x}; \Lambda) + \max_{j, j \neq k} g_j(\boldsymbol{x}; \Lambda). \tag{4.19}$$

The above formalism of the misclassification error is adopted, instead of the general formalism in (4.4) because it permits a direct comparison between competing class. Let us express the Bayes classifier error rate in the form

$$\mathcal{E}_{Bayes}[\boldsymbol{\Omega}_x] = \sum_{k=1}^{M} \int_{\boldsymbol{\Omega}_x} \Pr(\boldsymbol{x}, C_k) \mathbf{1}(\boldsymbol{x} \in C_k) \mathbf{1}\left(\Pr(\boldsymbol{x}, C_k) \neq \max_j \Pr(\boldsymbol{x}, C_j)\right) d\boldsymbol{x}, \tag{4.20}$$

which is the smallest error possible given the classification task and the set of features $\boldsymbol{\Omega}_x$. Given a classifier of parameter set Λ, the error rate of the classifier, conditioned by use of discriminant functions $\{g_k(\boldsymbol{x}; \Lambda)\}$, is

$$\begin{aligned}
\mathcal{E}_\Lambda[\boldsymbol{\Omega}_x] &= \sum_{k=1}^{M} \int_{\boldsymbol{\Omega}_x} \Pr(\boldsymbol{x}, C_k) \mathbf{1}(\boldsymbol{x} \in C_k) \mathbf{1}\left(g_k(\boldsymbol{x}; \Lambda) \neq \max_j g_j(\boldsymbol{x}; \Lambda)\right) d\boldsymbol{x} \\
&\approx \sum_{k=1}^{M} \int_{\boldsymbol{\Omega}_x} \Pr(\boldsymbol{x}, C_k) \mathbf{1}(\boldsymbol{x} \in C_k) \ell(d_k(\boldsymbol{x}; \Lambda)) d\boldsymbol{x} \qquad (4.21) \\
&= \mathcal{L}_{\boldsymbol{x}}(\Lambda). \qquad (4.22)
\end{aligned}$$

The loss $\ell(d_k(\boldsymbol{x}; \Lambda))$ approximates $\mathbf{1}\left(g_k(\boldsymbol{x}; \Lambda) \neq \max_j g_j(\boldsymbol{x}; \Lambda)\right)$ and the approximation can be made arbitrary closer by controlling the smoothness of the loss as seen above [Juang and Katagiri, 1992a]. That is, $\mathcal{E}_\Lambda[\boldsymbol{\Omega}_x]$ can be made arbitrary closer to $\mathcal{E}_{Bayes}[\boldsymbol{\Omega}_x]$ provided that

$$\mathbf{1}\left(g_k(\boldsymbol{x}; \Lambda) \neq \max_j g_j(\boldsymbol{x}; \Lambda)\right) = \mathbf{1}\left(\Pr(\boldsymbol{x}, C_k) \neq \max_j \Pr(\boldsymbol{x}, C_i)\right) \tag{4.23}$$

for all $\boldsymbol{x}$ and for a certain parameter set Λ. Note that (4.23) does not require that $g_k(\boldsymbol{x}; \Lambda) = \Pr(\boldsymbol{x}, C_k)$, and thus makes it possible to realize an equivalent Bayes classifier without probability estimation. Consequently, the target of MCE is simply to estimate the sign

$$-\Pr(\boldsymbol{x}, C_k) + \max_{j, j \neq k} \Pr(\boldsymbol{x}, C_i),$$

which might be easier than estimation of $\Pr(\boldsymbol{x}, C_k)$.

The MCE objective function $\mathcal{L}_{\boldsymbol{x}}(\Lambda)$, defined in (4.16) is almost equal to $\mathcal{E}_\Lambda[\boldsymbol{\Omega}_x]$. There is equality if $\ell(d_k(\boldsymbol{x}; \Lambda)) = 0$ for $d_k(\boldsymbol{x}; \Lambda) < 0$, which can be realized for an appropriate choice of the loss function. By definition, we have the following inequality

$$\mathcal{L}_{\boldsymbol{x}}(\Lambda) \approx \mathcal{E}_\Lambda[\boldsymbol{\Omega}_x] \geq \mathcal{E}_{Bayes}[\boldsymbol{\Omega}_x]. \tag{4.24}$$

The above relation shows that the minimum value of $\mathcal{E}_\Lambda[\boldsymbol{\Omega}_x]$ is the Bayes error $\mathcal{E}_{Bayes}[\boldsymbol{\Omega}_x]$. Thus, minimizing $\mathcal{L}_{\boldsymbol{x}}(\Lambda)$, with respect to Λ is equivalent to letting $\mathcal{E}_\Lambda[\boldsymbol{\Omega}_x]$ getting closer to $\mathcal{E}_{Bayes}[\boldsymbol{\Omega}_x]$,

with equality if (4.23) is realized. Consequently, the MCE framework enables the design of an equivalent Bayes classifier in a more direct manner. Estimation of the Bayes decision rule, is done through minimization of an objective function which approximates the Bayes classifier error rate. MCE is therefore a reasonable target for implementing the Discriminative Feature Extraction approach.

The monotonicity of the loss is an important propriety, within the MCE framework because it enables the MCE criterion to be a monotonic function of the errors as expressed through the misclassification measure. Note that the monotonicity of the loss is only guaranteed in the context of the strict MCE.

4.3.2 Effect of loss smoothing

In the MCE framework, a discrete 0-1 loss function is replaced by a smooth approximation. This can be viewed as replacing the minimum-error risk by a different risk, given a feature-space, which seems to contradict the original target of achieving minimum error. This section tries to find out the effect of this smoothing on the minimum-error goal achievement, which means investigating the link between the Bayes error rate $\mathcal{E}_{Bayes}[\boldsymbol{\Omega}_x]$ and the Bayes risk $\mathcal{E}_{\Lambda}[\boldsymbol{\Omega}_x]$, which makes use of the smooth 0-1 cost function. Again, for simplicity, the Bayes error rate is expressed on a fixed feature set $\boldsymbol{\Omega}_x$ and assume for a fixed set of classifier parameters Λ. It is supposed that the probabilities $\Pr(\boldsymbol{x}, C_k)$ are known and that for a class C_k, the misclassification measure is $d_k(\boldsymbol{x}) = -\Pr(\boldsymbol{x}, C_k) + \max_{j, j \neq k} \Pr(\boldsymbol{x}, C_j)$.

In the MCE framework, $\ell_{Bayes}(\boldsymbol{x}, C_k)$ is approximated by $\ell_{\Lambda}(\boldsymbol{x}, C_k) = \ell(d_k(\boldsymbol{x}; \Lambda))$ through a classifier of parameter set Λ. Suppose $\ell_{Bayes}(\boldsymbol{x}, C_k)$ is replaced by a smooth version such as

$$\ell_{\Lambda}(\boldsymbol{x}, C_k) = \alpha(\boldsymbol{x}, \Lambda) \ell_{Bayes}(\boldsymbol{x}, C_k) + \beta(\boldsymbol{x}, \Lambda) \tag{4.25}$$

where the term $\alpha(\boldsymbol{x}, \Lambda) \neq 0$, which depends on the classifier parameters Λ and on the input token $\boldsymbol{x}$, assigns a gradual value to the correct/incorrect decision represented by $\ell_{Bayes}(\boldsymbol{x}, C_k)$. Such a loss is referred to as a *Minimum error consistent loss* or simply *consistent loss* when there is no risk of confusion. The Bayes risk, corresponding to the smooth 0-1 cost function is

$$\mathcal{E}_{\Lambda}[\boldsymbol{\Omega}_x] = \sum_{k=1}^{M} \int_{\boldsymbol{\Omega}_x} \Pr(\boldsymbol{x}, C_k) \mathbf{1}(\boldsymbol{x} \in C_k) \ell_{\Lambda}(\boldsymbol{x}, C_k) d\boldsymbol{x}. \tag{4.26}$$

For simplicity, let us suppose that $\beta(\boldsymbol{x}, \Lambda) = 0$ for all $\boldsymbol{x}$ and Λ, which can always be done by simple translation of the loss form in (4.25). (4.26) can be re-written as

$$\mathcal{E}_{\Lambda}[\boldsymbol{\Omega}_x] = \sum_{k=1}^{M} \int_{\boldsymbol{\Omega}_x} \Pr(\boldsymbol{x}, C_k) \mathbf{1}(\boldsymbol{x} \in C_k) \alpha(\boldsymbol{x}, \Lambda) \ell_{Bayes}(\boldsymbol{x}, C_k) d\boldsymbol{x}. \tag{4.27}$$

Let us further assume that $\alpha(\boldsymbol{x}, \Lambda) \neq 0$ is smooth enough such that there is a 1-differentiable one-to-one mapping f_{Λ} satisfying

$$\left| \frac{\partial f_{\Lambda}(\boldsymbol{x})}{\partial \boldsymbol{x}} \right| = \alpha(\boldsymbol{x}, \Lambda), \tag{4.28}$$

where $\left|\frac{\partial f_\Lambda(\boldsymbol{x})}{\partial \boldsymbol{x}}\right|$ denotes the Jacobian of the function $\boldsymbol{z} = f_\Lambda(\boldsymbol{x})$. The smooth Bayes risk in (4.27) can then be re-written as

$$\mathcal{E}_\Lambda[\boldsymbol{\Omega}_x] = \sum_{k=1}^{M} \int_{f_\Lambda(\boldsymbol{\Omega}_x)} \Pr\left(f_\Lambda^{-1}(\boldsymbol{z}), C_k\right) \mathbf{1}\left(f_\Lambda^{-1}(\boldsymbol{z}) \in C_k\right) \ell_{Bayes}\left(f_\Lambda^{-1}(\boldsymbol{z}), C_k\right) d\boldsymbol{z}, \tag{4.29}$$

which gives

$$\begin{aligned}
\mathcal{E}_\Lambda[\boldsymbol{\Omega}_x] &= \sum_{k=1}^{M} \int_{f_\Lambda(\boldsymbol{\Omega}_x)} \Pr\left((\boldsymbol{z}, f_\Lambda(C_k)\right) \mathbf{1}\left(\boldsymbol{z} \in f_\Lambda(C_k)\right) \ell_{Bayes}\left((\boldsymbol{z}, f_\Lambda(C_k)\right) d\boldsymbol{z} && (4.30)\\
&= \sum_{k=1}^{M} \int_{f_\Lambda(\boldsymbol{\Omega}_x)} \Pr(\boldsymbol{z}, C_k) \mathbf{1}(\boldsymbol{z} \in C_k) \ell_{Bayes}(\boldsymbol{z}, C_k) d\boldsymbol{z} && (4.31)\\
&= \mathcal{E}_{Bayes}[f_\Lambda(\boldsymbol{\Omega}_x)]. && (4.32)
\end{aligned}$$

$f_\Lambda(C_k) = C_k$ since $\boldsymbol{z}$ and $\boldsymbol{x}$ have the same label. f_Λ is basically a feature extraction process unto the space $\boldsymbol{\Omega}_{\boldsymbol{z}} = f_\Lambda(\boldsymbol{\Omega}_x)$. In words, (4.32) means that if (4.28) is fulfilled for a consistent loss function, the MCE objective function on the feature space $\boldsymbol{\Omega}_x$ is equal to the error rate of the Bayes minimum error classifier on a new set of features $\Omega_{\boldsymbol{z}} = f_\Lambda(\boldsymbol{\Omega}_x)$, where the feature extractor f_Λ is entirely determined by the smoothness of the loss and the parameters Λ (the classifier structure). Clearly, if $\alpha(\boldsymbol{x}, \Lambda)$ is equal to one, both error rate become equal. The degree of discrepancy between the two Bayes risk is controlled by the smoothness, more precisely, by the degree of approximation of the 0-1 loss and the optimization scheme, which set the parameter Λ.

Consequently, MCE offers the possibility to achieve the Bayes classifier performance by selecting Λ such that $f_\Lambda(\boldsymbol{\Omega}_x) = (\boldsymbol{\Omega}_x)$, given an optimal estimation of the Bayes classifier discriminant function $d_k(\boldsymbol{x}) = -\Pr(\boldsymbol{x}, C_k) + \max_{j, j \neq k} \Pr(\boldsymbol{x}, C_j)$.

4.4 Generalized Probabilistic Descent

The target in the DFE paradigm is to find the optimal values of both Θ and Λ. Thus, DFE treats both Θ and Λ as adjustable parameters, while the original MCE implementation treats only Λ as adjustable parameters for recognition. Any optimization method could be used for this purpose. Here, the Generalized Probabilistic Descent method is described as an adaptive, descent optimization which updates the recognizer parameters Θ and Λ every time a training sample is presented. Again, the Generalized Probabilistic Descent (GPD) was first described in [Katagiri *et al.*, 1990], with further development made in [Juang and Katagiri, 1992a; Juang and Katagiri, 1992b].

The training aims at reducing the ultimate error measure.i.e., the expected loss

$$\begin{aligned}
\mathcal{L}(\Phi) &= E_{\boldsymbol{S}}\left[\ell(\boldsymbol{s}; \Phi)\right] && (4.33)\\
&= \sum_{k}^{M} P(C_k) \int_{\boldsymbol{\Omega}_s} \ell(\boldsymbol{s}; \Phi) p(\boldsymbol{s}|C_k) d\boldsymbol{s} && (4.34)
\end{aligned}$$

where it is assumed that the expectation for our observation sample space $\boldsymbol{\Omega}_s$ exists. Note that in the DFE framework, the objective function is simply a function of the overall recognizer parameter in contrast with the pure MCE case, where it depends on both the classifier parameter set and a preset feature-space. Again, $P(C_k)$ and $p(\boldsymbol{s}|C_k)$ are the class *a priori* and class-conditional densities probabilities. Usually, these probabilities are unknown and must be estimated from a body of training data. GPD belongs to the family of stochastic gradient descent methods, which are sometimes referred to as "on-line back-propagation" in the Neural Network terminology. The key contribution of stochastic gradient algorithms is the fact that the objective function in (4.34) can be minimized without having the knowledge of the probabilities $P(C_k)$ and p$(\boldsymbol{s}|C_k)$. The particularity of the GPD algorithm, in contrast to other stochastic-based gradient descent methods, is its focus on generalizing the Probabilistic Descent theorem of Amari [Amari, 1967] to a general, smoother form, applicable to variable-length sequence of pattern.

4.4.1 Probabilistic Descent Theorem

In the late 60s, Amari [Amari, 1967] introduced the Probabilistic Descent Theorem (PD), which was originally applied to static pattern. Again, the overall loss $\mathcal{L}(\Phi)$, which is a functional defined over the input space as the expectation of the local loss is

$$\mathcal{L}(\Phi) = E[\ell(\boldsymbol{s};\Phi)]. \tag{4.35}$$

The recognizer parameter Φ are adjusted according to

$$\Phi_{\mathcal{T}+1} = \Phi_{\mathcal{T}} + \delta\Phi_{\mathcal{T}}, \tag{4.36}$$

where $\delta\Phi = \delta\Phi(\boldsymbol{s}, C_k, \Phi_{\mathcal{T}})$ is the correcting term, which depends on the input pattern $\boldsymbol{s}$, its label C_k, and the current parameter $\Phi_{\mathcal{T}}$.

Amari's probabilistic descent theorem, in the DFE paradigm, is as follows:

Theorem 1 *Given $\boldsymbol{s} \in C_k$, if the parameter adjustment $\delta\Phi(\boldsymbol{s}, C_k, \Phi_{\mathcal{T}})$ is of the form*

$$\delta\Phi(\boldsymbol{s}, C_k, \Phi) = -\epsilon \boldsymbol{U} \nabla \ell(\boldsymbol{s}, \Phi), \tag{4.37}$$

where ϵ is a small positive number and $\boldsymbol{U}$ is a positive definite matrix, then we have

$$E[\delta\mathcal{L}(\Phi)] \leq 0. \tag{4.38}$$

Proof:

First note that

$$E[\nabla\ell(\boldsymbol{s},\Phi)] = E[\delta\Phi(\boldsymbol{s}, C_k, \Phi)].\nabla\mathcal{L}(\Phi). \tag{4.39}$$

From equation (4.37), we have

$$E[\delta\Phi(\boldsymbol{s}, C_k, \Phi_{\mathcal{T}})] = -\epsilon \boldsymbol{U} \nabla E[\ell(\boldsymbol{s};\Phi_{\mathcal{T}})] = \epsilon \boldsymbol{U} \nabla \mathcal{L}(\Phi) \tag{4.40}$$

which gives,

$$E[\nabla \ell(\boldsymbol{s}; \Phi)] = \nabla E \ell(\boldsymbol{s}, \Phi)] = \nabla \mathcal{L}(\Phi),$$

and then

$$E[\delta\Phi(\boldsymbol{s}, C_k, \Phi_\tau)] = -\epsilon \boldsymbol{U} \nabla \mathcal{L}(\Phi). \tag{4.41}$$

Injecting the above expression into (4.39) gives,

$$E[\delta \mathcal{L}(\Phi_\tau)] = -\epsilon \boldsymbol{U} \nabla \mathcal{L}(\Phi). \nabla \mathcal{L}(\Phi) = -\epsilon \nabla \mathcal{L}(\Phi)^{\mathrm{T}} \boldsymbol{U} \nabla \mathcal{L}(\Phi) \leq 0. \tag{4.42}$$

The basic idea behind the above proof is to choose the small increment $\delta\Phi$ such that a first order Taylor expansion is valid. To see that, let us express $\ell(\boldsymbol{s}, \Phi)$ into its Taylor expansion.

$$\ell(\boldsymbol{s}; \Phi + \delta\Phi) = \ell(\boldsymbol{s}; \Phi) + \nabla \ell(\boldsymbol{x}; \Phi)^{\mathrm{T}} \delta\Phi + \frac{1}{2} \delta\Phi^{\mathrm{T}} \boldsymbol{H}(\boldsymbol{x}, \Phi + \mu\delta\Phi)\delta\Phi \tag{4.43}$$

with $\boldsymbol{H}$ being the Jacobian of $\ell(\boldsymbol{s}; \Phi)$ with respect to Φ and $0 \leq \mu \leq 1$. Using,

$$\delta\Phi(\boldsymbol{s}, C_k, \Phi) = -\epsilon \nabla \ell(\boldsymbol{s}; \Phi)^{\mathrm{T}} \boldsymbol{U} \nabla \ell(\boldsymbol{s}; \Phi),$$

we have

$$\ell(\boldsymbol{s}; \Phi + \delta\Phi) = \ell(\boldsymbol{s}; \Phi) - \epsilon \parallel \nabla \ell(\boldsymbol{s}; \Phi) \parallel^2 + \epsilon^2 R_2(\boldsymbol{s}, \mu), \tag{4.44}$$

where $R_2(\boldsymbol{s}, \mu)$ is the Lagrange remainder and $\|\|^2$ denotes the expression $\ell(\boldsymbol{s}; \Phi)^{\mathrm{T}} \nabla \ell(\boldsymbol{s}; \Phi)$, which is a norm. Thus, the above theorem is based on neglecting the remainder and letting

$$E[\ell(\boldsymbol{s}; \Phi + \delta\Phi)] \approx E[\ell(\boldsymbol{s}; \Phi)] - E\left[\parallel \nabla \ell(\boldsymbol{s}; \Phi) \parallel^2\right], \tag{4.45}$$

which decreases at each iteration.

Theorem 2 (Probabilistic Descent) *Assume that ϵ_τ is a positive time series satisfying*

$$i) \sum_{\tau=1}^{\infty} \epsilon_\tau \rightarrow \infty, \tag{4.46}$$

and

$$ii) \sum_{\tau=1}^{\infty} \epsilon_\tau^2 < \infty. \tag{4.47}$$

Then, updating Φ according to

$$\Phi[\tau + 1] = \Phi[\tau] + \delta\Phi[\tau], \tag{4.48}$$

with

$$\delta\Phi(\boldsymbol{s}, C_k, \Phi_\tau) = -\epsilon_\tau \boldsymbol{U} \nabla \ell(\boldsymbol{s}, \Phi_\tau), \tag{4.49}$$

will produce a sequence of parameter vectors Φ_τ that converges to a local minimum of $\mathcal{L}(\Phi)$, with the probability of one.

A classic proof of this result, which makes use of the Doob's theory of Martingales, can be found in [Doob, 1953; Fu, 1968]. Bottou in [Bottou, 1991] also furnishes detailed explanations. The above theorem provides sufficient conditions for a stochastic gradient descent algorithm to converge. The objective function does not have to be of the MCE form. Condition i) means that the GPD algorithm will indefinitely learn as long new training data are fed to the recognizer. ii) implies that ϵ_T is a decreasing sequence that converge to zero. A simple example of a training rate sequence satisfying i) and ii) is $\epsilon_T = \frac{1}{T}$. Stated briefly, the probabilistic descent theorem then guarantees that asymptotically (infinite repetition), the recognizer is optimal. Due to the randomness introduced by the token-by-token learning phenomenon, the GPD algorithm is more likely to avoid local minima than the deterministic gradient.

The important point of this result is that, even though the expected loss $\mathcal{L}(\Phi)$ is not directly calculated, it can be minimized by using the derivative of the local loss $\ell(\boldsymbol{s};\Phi)$ for each incoming example $\boldsymbol{s}$.

4.5 Application of MCE to Speech Processing

As shown above, the goal of MCE is to directly minimize the error rate of the system. In most MCE applications, the optimization scheme is carried out by the Generalized Probabilistic Descent algorithm, which, as described previously, belongs to the family of stochastic descent training algorithms. Application of MCE/GPD for various recognizer structures with emphasis on Neural Network was described in [Katagiri *et al.*, 1991a] before detailed formalization in [Juang and Katagiri, 1992a].

4.5.1 Application to speaker recognition

In speech processing, speaker recognition refers to the process of guessing the identity of the talker. This process is usually classified into speaker identification (identification of the speaker identity) or speaker verification (verifying a claimed identity). MCE has been applied to both. The input to the system is a test utterance and the output of the system is the speaker identity. Application of MCE to speaker recognition is reported in [Liu *et al.*, 1995].

For reducing the difficulty encountered in multi-speaker-based speech recognition systems, a speaker adaptation process is sometimes carried out. This consists in mapping the speaker voices to a "standard speaker". Application of MCE to speaker mapping was made by Sugiyama and Kurinami [Sugiyama and Kurinami, 1992] using NN. [Matsui and Furui, 1995] describe the application of MCE to speaker adaptation using HMM.

4.5.2 Application to word spotting

Word spotting consists in detecting the presence of a word within a flow of speech. Thus, it includes both detecting the word occurrence and taking a decision regarding the identity of the

word. Usually, detection is made by making a comparison with a threshold, which in the conventional framework of design, is determined empirically. The Minimum SPotting Error learning (MSPE), suggested in [Komori and Katagiri, 1995], introduces a framework in which the threshold is optimized by MCE.

4.5.3 Application to speech recognition

MCE has been extensively applied to speech recognition using various architectures. Those applications can easily be found in the speech literature. Few early works on MCE application to speech are presented. See [McDermott, 1997] for further detailed explanations.

Use of MCE/GPD within a dynamic-programming based recognition framework was described by Komori and Katagiri [Komori and Katagiri, 1992] and Chang and Juang [Chang and Juang, 1992]. McDermott and Katagiri [McDermott and Katagiri, 1992] implemented MCE/GPD within a prototype-based speech recognition framework. Hidden Markov models implementation was done by Rainton and Sagayama [Rainton and Sagayama, 1992] and Chou and colleagues [Chou *et al.*, 1992].

Again, the common ground in all MCE implementations is to optimize the classifier so as to directly minimize the classification error rate.

4.6 Links between MCE and Classical Discriminative Objective Functions

Various discriminative training approaches have been proposed in the literature. Unlike MCE, most of them are not a monotonic function of the classification error even if a relation with the MCE formulation of discriminative training can still be investigated. MCE devises an elegant framework for discriminative training, which includes some other discriminative training methods as special cases. Among various implementations, only 3 discriminative objective functions, namely, Maximum Mutual Information (MMI), Classification Figure of Merit (CFM) and Minimum Squared Error criterion (MSE), are discussed.

4.6.1 MCE and MSE

As was stated in Chapter 2, the MSE criterion is a well-studied error criterion with main properties available in the literature [Duda and Hart, 1973]. Our concern is mainly to show the relationship with the MCE criterion. Let $\boldsymbol{g}(\boldsymbol{x};\Lambda) = [g_1(\boldsymbol{x};\Lambda), ..., g_M(\boldsymbol{x};\Lambda)]^{\mathrm{T}}$ be the decision vector. The MSE criterion is defined as

$$\ell_{MSE}(\boldsymbol{x};\Lambda) = \| \boldsymbol{g}(\boldsymbol{x};\Lambda) - \boldsymbol{t}(\boldsymbol{x}) \|^2 \tag{4.50}$$

$$= \sum_{j=1}^{M} (g_j(\boldsymbol{x};\Lambda) - t_j(\boldsymbol{x}))^2 \tag{4.51}$$

where $\boldsymbol{t}(\boldsymbol{x}) = [t_1(\boldsymbol{x}), ..., t_M(\boldsymbol{x})]^{\mathrm{T}}$ is the target vector corresponding for input $\boldsymbol{x}$. For a classification problem,

$$t_j(\boldsymbol{x}) = \delta_{jk} \qquad \text{for } \boldsymbol{x} \in C_k, \tag{4.52}$$

where δ_{jk} is the Kronecker notation. This means that the target vectors $\boldsymbol{t}(\boldsymbol{x})$ are an orthonormal base of the space spanned by the decision vectors $\boldsymbol{g}(\boldsymbol{x}; \Lambda)$. For convenience, the target vector basis of C_k is referred to as $\boldsymbol{t}_k$. Each class is thus represented by one of the $\boldsymbol{t}_k$.

Geometrical interpretation of MCE and MSE

The geometrical interpretation of the MSE as shown in eq. (4.51) is that MSE tries to create clusters around the vector-points $\boldsymbol{t}_k$. The decision vector $\boldsymbol{g}(\boldsymbol{x}; \Lambda)$ of feature-vector $\boldsymbol{x}$ of category C_k is made similar, in the sense the Euclidean norm, to $\boldsymbol{t}_k$, which represents category C_k in the space spanned by the decision vectors $\boldsymbol{g}(\boldsymbol{x}; \Lambda)$.

Making use of the target vectors $\boldsymbol{t}_k$, the MCE criterion, expressed through the misclassification measure, can be re-written as

$$\begin{aligned} d_k(\boldsymbol{x}; \Lambda) &= -\boldsymbol{g}(\boldsymbol{x}; \Lambda)^{\mathrm{T}} \boldsymbol{t}_k + \max_{j, j \neq k} \boldsymbol{g}(\boldsymbol{x}; \Lambda)^{\mathrm{T}} \boldsymbol{t}_j & (4.53) \\ &= -\boldsymbol{g}(\boldsymbol{x}; \Lambda)^{\mathrm{T}} \boldsymbol{t}_k + \boldsymbol{g}(\boldsymbol{x}; \Lambda)^{\mathrm{T}} \boldsymbol{t}_{\overline{k}} & (4.54) \end{aligned}$$

where $\boldsymbol{t}_{\overline{k}}$ refers to the target vector of the best incorrect category, given $\boldsymbol{x}$. The MCE loss is a difference of dot products involving the representatives of the two competing classes. Expanding the dot product gives

$$\begin{aligned} d_k(\boldsymbol{x}; \Lambda) &= - \parallel \boldsymbol{g}(\boldsymbol{x}; \Lambda) \parallel \parallel \boldsymbol{t}_k \parallel \cos \alpha_k + \parallel \boldsymbol{g}(\boldsymbol{x}; \Lambda) \parallel \parallel \boldsymbol{t}_{\overline{k}} \parallel \cos \alpha_{\overline{k}} & (4.55) \\ &= \parallel \boldsymbol{g}(\boldsymbol{x}; \Lambda) \parallel \left(-\cos \alpha_k + \cos \alpha_{\overline{k}} \right), & (4.56) \end{aligned}$$

where α_k and $\alpha_{\overline{k}}$ refer to the angle between $\boldsymbol{g}(\boldsymbol{x}; \Lambda)$ and $\boldsymbol{t}_k$ and to $\boldsymbol{g}(\boldsymbol{x}; \Lambda)$ and $\boldsymbol{t}_{\overline{k}}$, respectively. Minimizing the MCE loss is thus equivalent to letting $\cos \alpha_k \rightarrow 1$ and/or $\cos \alpha_{\overline{k}} \rightarrow -1$. That is, minimize the angle between $\boldsymbol{g}(\boldsymbol{x}; \Lambda)$ and $\boldsymbol{t}_k$ and/or maximize the angle between $\boldsymbol{g}(\boldsymbol{x}; \Lambda)$ and $\boldsymbol{t}_{\overline{k}}$.

Thus, MCE discrimination relies on "direction" optimization whereas MSE focuses on "norm" optimization.

MSE's discriminant term

For a feature data $\boldsymbol{x}$ belonging to C_k and assuming the target function given in (4.52), the MSE criterion can be re-written as

$$\ell_{MSE}(\boldsymbol{x}; \Lambda) = (g_k(\boldsymbol{x}; \Lambda) - 1)^2 + \sum_{j=1, j \neq k}^{M} g_j(\boldsymbol{x}; \Lambda)^2, \tag{4.57}$$

which, by adding and subtracting $\frac{1}{2}$, gives

$$\ell_{MSE}(\boldsymbol{x}; \Lambda) = \left(g_k(\boldsymbol{x}; \Lambda) - \frac{1}{2} - \frac{1}{2} \right)^2 + \sum_{j=1, j \neq k}^{M} \left(g_j(\boldsymbol{x}; \Lambda) - \frac{1}{2} + \frac{1}{2} \right)^2 \tag{4.58}$$

$$= -\left(g_k(\boldsymbol{x};\Lambda) - \frac{1}{2}\right) + \sum_{j=1,j\neq k}^{M}\left(g_j(\boldsymbol{x};\Lambda) - \frac{1}{2}\right) + \sum_{j=1}^{M}\left(g_j(\boldsymbol{x};\Lambda) - \frac{1}{2}\right)^2 + \frac{M}{4}.$$

Let $g_j'(\boldsymbol{x};\Lambda) = g_j(\boldsymbol{x};\Lambda) - \frac{1}{2}$. According to (4.59), the MSE criterion can be written as

$$\ell_{MSE}(\boldsymbol{s};\Lambda) = \underbrace{-g_k'(\boldsymbol{x};\Lambda) + \sum_{j=1,j\neq k}^{M} g_j'(\boldsymbol{x};\Lambda)}_{\text{discriminant term } d_k(\boldsymbol{x};\Lambda)} + \underbrace{\sum_{j=1}^{M} g_j'(\boldsymbol{x};\Lambda)^2 + \frac{M}{4}}_{\text{distortion term}}. \tag{4.59}$$

Equation (4.59) shows that the MSE criterion is composed of two terms: a discriminant term $d_k(\boldsymbol{x};\Lambda)$ and a distortion term. Because of this distortion term, the MSE criterion is less related to the minimum error criterion albeit being discriminant. For classification problems, the non-monotonicity of the MSE often leads to unoptimality. That is, the MSE criterion can display cases where the value of the MSE criterion is small for a given input data while the data is still being mis-recognized [Hampshire and Waibel, 1990].

4.6.2 MCE and CFM

The Classification Figure of Merit (CFM) was proposed for feedforward Neural Networks by Hampshire and Waibel [Hampshire and Waibel, 1990]. It is an attempt to bridge the gap between the MSE criterion and the non-monotonicity of MSE to the recognition errors. For a feature-data $\boldsymbol{x}$ of category C_k, the CFM criterion is defined as

$$\ell_{CFM}(\boldsymbol{x};\Lambda) = \frac{1}{M-1}\sum_{j=1,j\neq k}^{M} \text{sig}(\nabla_{kj}(\boldsymbol{x};\Lambda)), \tag{4.60}$$

where

$$\nabla_{kj}(\boldsymbol{x};\Lambda) = g_k(\boldsymbol{x};\Lambda) - g_j(\boldsymbol{x};\Lambda) \qquad \text{for } j = \{1,\ldots,M\};\ j \neq k$$

and sig is the sigmoid function [Hampshire and Waibel, 1990].

The CFM form of (4.60) is not a monotonic function of the errors for $M \geq 3$. Thus, a variant of CFM has been proposed, which considers the smallest $\nabla_{kj}(\boldsymbol{x};\Lambda)$ and neglects the other terms. This variant of CFM (CFM2) is the same as the strict MCE case when using a sigmoid function as the loss. Thus, it can be argued that CFM2 is a special case of MCE.

4.6.3 MCE and MMI

Similar to the MCE criterion, the MMI criterion does not focus on estimating the Bayes probabilities but rather tries to minimize the average uncertainty of a class-labels, given input feature-data [Brown, 1987]. For a feature-data $\boldsymbol{x}$ of category C_k, the mutual information $I_k(\boldsymbol{x};\Lambda)$ between the feature-data $\boldsymbol{x}$ and the category C_k, given the classifier parameter set Λ is defined as

$$I_k(\boldsymbol{x};\Lambda) = \log \mathrm{p}(\boldsymbol{x}|C_k;\Lambda) - \log\left(\sum_{j=1}^{M}\Pr(C_j;\Lambda)\mathrm{p}(\boldsymbol{x}|C_j;\Lambda)\right), \tag{4.61}$$

where $\mathrm{p}(\boldsymbol{x}|C_k;\Lambda)$ and $\Pr(C_j;\Lambda)$ are the class-conditional probabilities and a priori probabilities conditioned on the classifier parameter set Λ. However, the right side of (4.61) takes the summation over all classes, including the correct class, whereas MCE considers incorrect categories only. Thus, MMI, from the view point of MCE does not directly tries to separate the competing categories.

A different version of MMI, which considers the summation only over "confusing categories" has been proposed [Brown, 1987]. This approach may be closer to the MCE formulation. However, the general version of MMI is not directly related to the minimum error criterion, albeit being a discriminative training error criterion.

The link between MCE and MMI is as follows. First, the discrimination function of class C_j is defined as

$$g_j(\boldsymbol{x};\Lambda) = \log\left(\mathrm{p}(\boldsymbol{x}|C_j;\Lambda)\right).$$

Second, due to the use of log probabilities, the misclassification measure is defined as

$$d_k(\boldsymbol{x};\Lambda) = -g_k(\boldsymbol{x}) + \log\left(\frac{1}{M-1}\sum_{j=1,j\neq k}^{M} e^{\nu g_j(\boldsymbol{x};\Lambda)}\right)^{\nu} \tag{4.62}$$

where ν is a positive parameter. It is straightforward to show that (4.62) can be re-written as

$$d_k(\boldsymbol{x};\Lambda) = \frac{1}{\nu}\left[\log\left(\sum_{j=1,j\neq k}^{M} e^{(g_j(\boldsymbol{x};\Lambda)-g_k(\boldsymbol{x};\Lambda))}\right) - \log(M-1)\right]. \tag{4.63}$$

For $\nu = 1$, assuming equi-probability of the a priori probability ($\Pr(C_j) = \frac{1}{M}$), we can express the mutual information as a function of the MCE loss using the discriminant measure of (4.62).

$$\begin{aligned} I_k(\boldsymbol{x};\Lambda) &= -\log\left(e^{(d_k(\boldsymbol{x};\Lambda)+\log(M-1))}+1\right)+\log(M) && (4.64)\\ &= -\log\left(e^{(h(\ell(d_k(\boldsymbol{x};\Lambda)))+\log(M-1))}+1\right)+\log(M), && (4.65)\end{aligned}$$

where $h(\cdot) = \ell(\cdot)^{-1}$, is the inverse function of the MCE loss, (which is defined since the mapping $d \rightarrow \ell(d)$ is a continuous one to one mapping). $h(\cdot)$ being a monotonic one-to-one mapping, it can be seen that the MMI cost is a monotonic function of a soft MCE cost function. Thus, minimizing MCE is equivalent to minimizing the MMI criterion under following assumptions 1) all classes have equal a priori probabilities 2) the discrimination measure considers averaging over incorrect categories. For MCE and MMI to lead to the same minima in the parameter space, the MCE loss shall simply be the identity function. That is, $h(d) = d$ in (4.65), which can be achieved by using a piece-wise linear function.

MMI is equivalent to soft MCE, thus MMI is not a monotonic function of the error that the system makes. As said earlier, Gopalahrishan [Gopalakrishnan *et al.*, 1988] described a case where, the MMI criterion, despite being able to separate categories, asymptotically failed to realize the optimal minimum error solution.

4.7 Joint Optimization of Feature Extraction and Classification

Several attempts have been made to integrate feature extraction and classification. Most of them are included in the ANN paradigm [Intrator, 1990; Morgan and Bourlard, 1990; Biem and Sugiyama, 1991; Lowe and Webb, 1991]. The subspace method (SM) is also a well-formulated, traditional approach [Oja, 1978]. However, these attempts, unlike the DFE approach, are often specific to a particular recognizer structure rather than based on a general framework of applicability. Furthermore, in most cases, *optimality* of the overall recognizer in terms of error minimization is not guaranteed.

4.7.1 Subspace methods

Recognition-oriented feature design has been most extensively studied in the Subspace Methods (SM) paradigm, especially for character and image recognition [Oja, 1978]. In SM, the recognition (classification) decision is made by measuring the angle between a pattern (vector) to recognize and a class model for every class. In particular, a recognizer based on this method is defined by (lower-dimensional) *class subspaces*, each assumed to represent the salient features of its corresponding class. There is no distinction between the feature extraction process and the classification process in this framework. Given an input sample, the recognizer computes the orthogonal projection of the input onto each class subspace and then classifies the pattern to the class giving the maximum projection value.

The performance of SM-based recognizers relies on the quality of the class subspaces. The most fundamental algorithms for designing the subspaces have been the Class Featuring Information Compression (CLAFIC) method [Watanabe *et al.*, 1967] and the Multiple Similarity Method (MSM) [Iijima, 1989], where each class subspace is designed by running Karhunen-Loève Transformation or Principal Component Analysis over the design data of its corresponding class. Obviously, this class-by-class design does not directly guarantee recognition error reduction: the recognition result of design samples is not reflected in the subspace design. These methods have been improved over time, aiming at increasing the recognition accuracy. CLAFIC has been extended to the Learning Subspace Method (LSM) [Kohonen *et al.*, 1979]; LSM was later reformed as the Compound Similarity Method (CSM). In these new versions, subspaces are trained iteratively (adaptively) according to the recognition result of each design sample; i.e., when an input design pattern is misrecognized, the subspace of the true class and that of the most likely but incorrect class are adjusted so that projection onto the true class subspace increases and projection onto the competing incorrect class subspace decreases. This discriminative iteration actually contributes towards improving the accuracy. However, the training mechanism remains intuitive, and its mathematical optimality in terms of error minimization has yet to be clarified.

4.7.2 Minimum Error Learning Subspace

Let us assume that similar to SM, a feature is represented by the subspaces of the original pattern (vector) space. Given this assumption, DFE can easily be applied to the design of an SM-based recognizer. However, in this framework, the class subspace works as a class model such as a set of prototypes. It is possible to formulate a discriminative training method for this recognizer structure by directly using MCE/GPD. The resulting method is Minimum-Error Learning Subspace (MELS), which can be viewed as an application of the DFE approach to subspace configuration [Watanabe *et al.*, 1996]. It has also been shown that the conventional LSM is an intuitive implementation of MELS [Watanabe and Katagiri, 1995]

4.7.3 Feature extraction properties of Neural Networks

Despite its prime use as a classifier, NN do provide an appreciable framework for feature extraction. For various NN-like architectures, such a Feed-Forward network or Radial Basis Functions (RBF) [Girosi and Poggio, 1989], the output of each layer can be viewed as features for the next layer meaning that the NN are simply a cascade of feature extractors, with the last layer acting as the classifier. This is the internal representation of the NN scheme [Rumelhart *et al.*, 1986]. The internal representation corresponds to discriminant features because the local feature extraction process is optimized while targeting the classification task.

Various feature extraction processes can be implemented by appropriate choice of the target function t_j.

Dimensionality reduction

Efficient dimensionality reduction can be achieved by Neural Network in a manner similar to principal component analysis. The method consists of squeezing the inputs unto a lower-dimensional representation within the hidden layers. A non-linear principal component analysis can be derived by minimizing the sum-of-squared error between the input and the output. That is, $(t_j(\boldsymbol{x}) = x_j)$ for $\boldsymbol{x} = [x_1, \ldots, x_d]^{\mathrm{T}}$, where p is the dimensional on the feature vector $\boldsymbol{x}$. The error function takes the form

$$\mathrm{MSE}(\Lambda) = \frac{1}{2} \sum_{\boldsymbol{x}} \sum_{j=1}^{d} (g_j(\boldsymbol{x}; \Lambda) - x_j)^2 \tag{4.66}$$

where x_j is the j-th component of $\boldsymbol{x}$ and p is the dimension of the input data. This form of input-to-input mapping is known as the auto-associative mapping. If the network consists of 3 layers, with one input layer of p nodes, and is at the minimum of the error function, the output of the input layer are simply the projection unto the p-dimensional subspace generated by the p first eigenvector of the input data [Bourlard and Kamp, 1988].

Discriminant analysis

It has been established that Neural Network performs some form of discriminant analysis at the output of the hidden layer which feeds the output layer, trained for classification task with the mean-squared error criterion.

Gallinari [Gallinari *et al.*, 1991] has shown that a Neural Network with linear units at the last hidden layer maximizes the cost

$$J = \frac{\det \boldsymbol{\Sigma}_B}{\det \boldsymbol{\Sigma}_W},$$

if trained for classification with MSE. Again, $\boldsymbol{\Sigma}_B$ and $\boldsymbol{\Sigma}_T$ are the between class covariance matrix and within class covariance matrix, in the space spanned by the output of the last hidden layer. Asoh and Otsu [Asoh and Otsu, 1989] empirically demonstrated that nonlinear nodes of the last hidden layer give greater values of

$$J = \mathrm{Tr}(\boldsymbol{\Sigma}_B \boldsymbol{\Sigma}_W{}^{-1})$$

when trained with the MSE criterion. Lowe and Webb [Lowe and Webb, 1991] extended Gallinari's results by introducing the weighted sum of error squared criterion which takes into account the prior probability of each class, leading to the general result that, minimizing the sum-of-squared criterion maximizes the criterion

$$J = \mathrm{Tr}(\boldsymbol{\Sigma}_B \boldsymbol{\Sigma}_T^{+})$$

at the last hidden units, where + represents the pseudo-inverse [Bishop, 1995].

NN-based feature representation approach is one example of Embedded Feature Extraction scheme. The essential shortcoming of the conventional NN approaches are 1) the design objective function are usually inconsistent with the minimum recognition error criterion. 2) the simple use of a general network structure, which mainly consists of weighted sum calculations, is often than insufficient to handle the complex nature of various kind of patterns such as speech utterances 3) most applications required already pre-processed data. Although, thanks to Kolmogorov' theorem, a Neural Network can theoretically perform any mapping [Girosi and Poggio, 1989; Kůrková, 1991; Lin and Unbehauen, 1993], still in practical applications, data should be passed through a pre-processing stage.

Use of Neural Network as feature extractor

In the previous chapter, a method in which an NN is used as feature extractor for discrete HMM was described. The NN and HMM, however, used different criteria for their optimization scheme, which is clearly far from the DFE approach presented here. A more efficient scheme would be to use the NN are a non-linear feature transformation, optimized jointly with the HMM classifier.

Joint optimization of NN and HMM, has been reported in the literature. In an early work by Bridle [Bridle and Dodd, 1991], joint optimization of an HMM, implemented within a connectionist interpretation of HMM, called Alphanet [Bridle, 1990], and a linear transformation were jointly

optimized aimed at speaker-dependent continuous phoneme recognition. However, this experiment did not show substantial improvement over the baseline approach.

Bengio and his colleagues in [Bengio *et al.*, 1990b; Bengio *et al.*, 1990a] have jointly optimized a set of NN and HMM for the recognition of plosive sounds using the MLE and MMI criterion. In their work, the feature extractor consists of three NN, of which two have been especially pre-trained to perform plosive recognition and the third network was deterministically preset to perform a PCA transformation. The results of this quite elaborate platform shows a clear improvement over baseline approaches, using the MMI criterion.

A simpler non-linear transformation of features has been investigated by [Johansen and Johnsen, 1994]. The current speech frame is appended to the output of an MLP, whose input consists of the current frame, the last two past frames and the next two future frames. The overall system is globally trained with the MMI criterion and achieved 22% improvement in recognition rate on a TIMIT broad-class phone recognition task.

4.7.4 Feature selection based on performance

DFE as described above, is not the first attempt to use the performance of the recognizer to value the quality of features.

Paliwal [Paliwal, 1992] made a study where each individual feature is ranked according the its performance on the training set aiming at recognizing telephone-based isolated alpha-digit in a multi-speaker recognition mode. For a p-dimensional feature vector, this experience is the same performing p tests and then choosing the best feature according to the ranking. Paliwal used a 38 dimensional feature set, composed of 12 LPC cepstral parameters, their first derivatives, and their second derivatives and made a comparative feature ranking based on the F-ratio and the error rate figures of merit which shows a clear discrepancy between F-ratio-valued features and their performance rankings. Furthermore, Paliwal showed that some manually selected feature subsets may outperform the F-ratio-based selection. This later result shows that, unlike the DFE approach, features ought to be uncorrelated for the ranking to be of any significance.

Similar to [Paliwal, 1992], Bochieri and Wilpon [Bocchieri and Wilpon, 1993] used a statistical information based on mismatch between transcribed output of the Viterbi alignment of HMM and the correct transcription of the utterance, aiming at phone recognition. This method was shown to be related to the between-class and within-class scatter measure computed for a single-component of the feature vector. The main drawback of this approach is its incapacity to dealing properly with deletions errors when applied to continuous speech.

Other works have directly attempted to apply linear discriminant analysis for improving acoustic modeling in speech recognition. Doddington [Doddington, 1989] has used a "confusion data model" at the state level (provided by a Viterbi alignment) to discriminate between true distribution and acoustically similar distribution across all the HMM states.

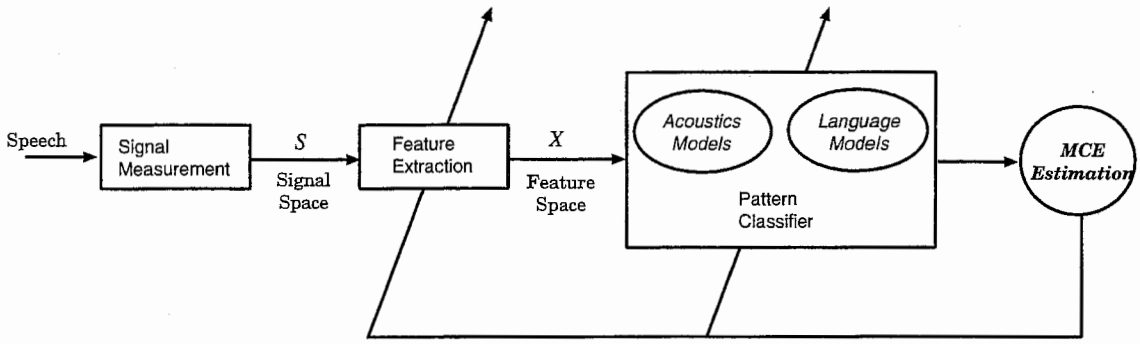


Figure 4.3: General model of speech recognition system design by DFE.

4.8 Application of DFE to Speech Recognition

Since the suggestion of the DFE approach was made in [Biem and Katagiri, 1993b; Biem *et al.*, 1993; Katagiri *et al.*, 1993] and applied to a particulars speech recognition tasks, the DFE method has enjoyed various implementations. Below, an overview of some works is presented.

In the context of speech recognition, the simple model of the pattern recognizer as defined previously should be adapted. The dynamics of the speech signal should be taken into account when optimizing the overall recognizer.

A speech recognizer design by DFE is illustrated in Fig. 4.3. The MCE criteria is used to jointly optimize the feature extractor and the acoustic models, which perform classification. The language model can be integrated in the design by relying on it during the computation of the discriminant functions. The feature extractor can take various forms. It can be parameterized LPC-based model, a filter-bank, a linear or an NN-based transformation or a combination of these.

In this report, the feature extraction is considered a standard speech parameterization process such as a filter bank so as to have a straightforward comparison with conventional speech parameterization process. Again, as have been suggested through various works, one can use a linear or a non-linear mapping as feature extractor. If the linear transformation is constrained to keep certain characteristics (for instance, positive-definiteness), DFE can be applied through a decomposition technique such as the singular value decomposition or the Cholesky factorization.

4.8.1 Shared feature extractor

Feature extraction is usually a shared module by all categories. That is, the output of the feature extraction is compared against the models of all categories. Here, some applications of DFE in this context are introduced.

Optimizing dynamic cepstrum lifters

Bachianni and Aikawa [Bachiani and Aikawa, 1994] have used the DFE method to optimize the dynamic cepstral lifters for HMM-based phone classification. The lifter were implemented with a TDNN, whose delays permits a straightforward implementation of the dynamic lifters.

Optimizing feature weighting

Hernando [Hernando *et al.*, 1995] and his colleagues have used a DFE-like approach to optimize the weighting given to various features in a continuous HMM framework.

Interesting work by [De la Torre *et al.*, 1996] has been done in this area. [De la Torre *et al.*, 1996] have applied various DFE strategies to optimizing the weighting of various feature streams for semi-continuous HMMs. They have proposed the Single Gaussian DFE (SGDFE), which presupposes a Gaussian model for each class, and perform a DFE run based on this supposition. Since the models are single Gaussians, this approach may help to reduce the computational burden in the DFE optimization. The estimated features can then be used for MLE or MCE training or as the starting point of an overall DFE process. Obviously in this approach, the feature extractor is not fully embedded in the recognition system. However, the results show a clear improvement over the baseline MCE and MLE. In [De la Torre *et al.*, 1997b], the same approach has been applied to feature reduction using a linear transformation. [De la Torre *et al.*, 1997a] shows application of DFE to speech recognition in noise, which shows the robustness of the DFE recognizer to noise.

Unlike the works presented in this report, or in [Biem and Katagiri, 1994; Biem *et al.*, 1995; Biem and Katagiri, 1997a; Bachiani and Aikawa, 1994], most research within the DFE framework, has been done using linear transformations or using NNs as a non-linear transformation, which is certainly the right approach for improving performance. However, this approach does not always permit an easy analysis of the resulting feature extractor nor set up the framework of comparison with fundamental speech parameterization methods, which is one main goal of this report.

4.8.2 Category-dependent feature extractors

Feature extraction can be viewed as a process that forms a metric in measuring a class membership of an input pattern. This is motivated by the fact that the definitions of class identity can be different from one class to another. In that situation, it may be appropriate to use a class-dependent feature extraction. That is, the feature representation does not have to be common to all classes; i.e., each class can have its feature extractor mapping, which is then viewed as a metric that is suitable for representing its identity as effectively as possible. Within the DFE framework, this approach enforces the discriminative capabilities to the feature extraction process.

A recognition decision rule of this framework then is as follows:

$$a(\boldsymbol{s}) = C_k \quad \text{if} \quad k = \arg\max_j g_j\left(\mathcal{F}_{\Theta_j}(\boldsymbol{s};\Lambda)\right), \tag{4.67}$$

where $\mathcal{F}_{\Theta_j}(\cdot)$ is a class-specific feature extractor. Θ_j is the parameter set of class C_j. Note that for each class the feature extractor is embedded in its corresponding discriminant function.i.e. $g_j(\cdot)$ [Watanabe *et al.*, 1996].

Discriminative metric design

Based on the above assertion, DFE was applied to designing a metric for each class. This method has been referred to as Discriminative Metric Design (DMD) [Watanabe *et al.*, 1995].

The DMD approach is similar to the SM case, which shows individuality of the class metric, though the feature representation is strictly limited to subspaces. Among many possibilities, DMD in particular was implemented by using quadratic discriminant functions [Watanabe *et al.*, 1995] and its utility shown in a comparative experiment against a multi-template Learning Vector Quantizer (LVQ).

Again, DMD is as a particular case of DFE, in which the feature extractor is reduced to a class-dependent metric and where focus is on the metric describing the matching between pattern [Watanabe *et al.*, 1996]. However, in practice this approach can lead to higher number of parameters, leading to poor performance if relatively few data are available.

Use of class-dependent, linear transformation

Most applications of class-dependent feature extraction have made use of a linear transformation as feature extractor. In the HMM framework, the linear transformation is usually assigned to HMM states, to capture local spectral events. This approach can be likened to the use of class-specific covariance matrix in which the covariance matrix is trained with MCE.

Euler in [Euler, 1995] has used the DFE-approach to optimize a linear transformation aiming at recognizing spelled letters by HMM. In the work of [Ayer *et al.*, 1993], an initial LDA transformation is optimized using a discriminative criteria for the recognition of the British Telecom Laboratories (BTL) E-set. The BTL E-set is defined as the set of the E-sounding letter {"B", "C", "D", "E", "G", "P", "T" and "V"} and considered to be a particularly difficult task. Ayer's result of 96% accuracy seems to the best result so far in this database.

Application of DFE on the TIMIT database was done by Rathinavelu and Deng [Rathinavelu and Deng, 1996] using HMM for classifying phonemes. In their work, they have used a state-based linear transformation, which also integrates dynamic features. The linear transformation is initially set to perform a DCT transform from the output of a Mel filter bank (see Chapter 8). Their results have shown a clear improvement over the baseline MLE and MCE approach . Using 5 mixtures context independent HMM, MLE achieved 66.6 %, MCE realized 80.51% and DFE gave 82.19%, which is one the best classification result achieved so far in the TIMIT database, using the reduced 39 phonemes set.

The approach described in [Watanabe and Katagiri, 1997] is quite similar to [Rathinavelu

and Deng, 1996] but applied to continuous speech recognition. The DMD (or DFE using class-dependent linear transformation) is here applied to the ATR 5240-word database and has shown an improvement over classical use of MCE (from 90.73% to 93.95% of phoneme accuracy).

Use of class-dependent, non-linear transformation

Use of NN as a non-linear feature extractor for HMM was investigated by [Rahim and Lee, 1996] aiming at speaker-independent telephone-based connected at digit recognition in which the input to the feature extractor are cepstrum coefficients. In this work, two approaches were investigated. The first approach makes use of a single NN as feature extractor for sub-word HMM. The second approach makes use of 12 NNs as a feature extractor assigned to each digit and to a noise model. This approach has shown a clear improvement over a baseline ML-based HMM and MCE-based HMM, with the best result being achieved in the context of 12 NNs. In this case, the DFE approach reduced the word and string error rate to 16% and 25% over MCE performance using HMM.

Is class-dependent DFE the best ?

Paliwal and colleagues in [Paliwal *et al.*, 1995] have made a comparative study of various implementations of DFE configurations. The task in their work was a simple classification of static pattern and the feature extraction was reduced to a linear transformation. The classifier was a distance-based classifier. The configurations that were investigated are: 1) MCE-optimization of the linear transformation while the classifier is untrained. 2) MCE-optimization of the linear transformation while the classifier is re-adjusted according to the previously trained transformation. 3) Classifier and linear transformation are optimized jointly by MCE. 4) same as (1) but with a class-dependent transformation. 5) Same as (3) with class-dependent linear transformation. The best result on the testing set was achieved by configuration (3) which is the simultaneous optimization of the overall recognizer, using a shared (single) feature extractor. However, it will be inappropriate to conclude from this experiment since this was a rather simple task with a limited number of training data. Nevertheless, these results point out the fact that in the framework of DFE implementation, making use of multiple or single feature extractor or single feature extractor should be carefully investigated according to the task and the recognizer structure as well as the number of training data available.

If enough data are available, for overall general optimality, a shared feature extractor can be used to optimize basic speech parameterization such as filter bank parameters for extracting overall discriminant parameters and the class-dependent transformation can then be used in the next stage to select class-specific informations. This optimal DFE design is illustrated in Fig. 4.4. This figure has been adapted from [Watanabe and Katagiri, 1997]).

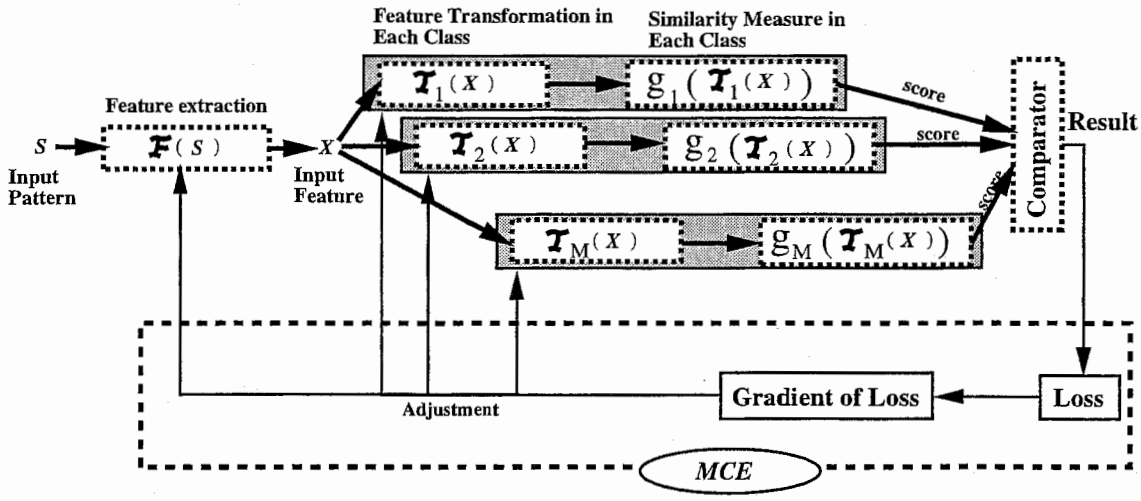


Figure 4.4: Optimal optimization of a pattern recognizer by DFE.

4.9 Conclusion

In this chapter, the pattern recognition concept was formalized and the discriminative training concept was introduced as an efficient design method for the overall recognizer. The concept of recognizer design assumes that the feature extractor takes the raw input data and creates a feature space on which classification is made.

The aim is to let a front-end feature extraction process collaborate with a classification process for direct minimization of the classification. This is realized through the representation of the overall recognizer as a modular system in which a parameterized feature extractor module and a parameterized classifier are subject to optimization for minimum error achievement using a discriminative training algorithm. This *Discriminative Feature Extraction* method can be viewed as an extended application of the Minimum Classification Error/Generalized Probabilistic Descent method introduced in [Juang and Katagiri, 1992a].

Consequently, in this chapter, the Minimum Classification Error/Generalized Probabilistic Descent (MCE/GPD) formalism, which have been shown to provide a framework of discriminative training aiming at minimum error, was described. GPD, which is a stochastic gradient descent technique, enables the minimization, up to a local minima, of the MCE objective function. MCE/GPD approach is to approximate the discrete 0-1 minimum error by a smooth, at least first differentiable form. Thus, the objective function within the MCE/GPD scheme is a close approximation to the actual error rate that the system makes. Minimizing the MCE criteria is therefore equivalent to a direct minimization of the error rate.

In the DFE framework, the feature extraction participates in the classification process and all system resources, that is, the feature extraction process and the classification process, are optimized jointly for the single purpose of reducing the misclassifications of the system. This ensures that the

feature extraction is optimally matched to the classification process, given a recognizer structure.

DFE can be applied within two schemes. A single feature extractor-based scheme, which may be suitable to extract the general discriminant properties of the task and a class-dependent feature extractor-based approach which add more the discriminant capabilities to the recognizer.

Chapter 5

Expected loss and empirical loss

Accuracy is a costly thing

-Jonas C. Nader-

In DFE, the target is to minimize the expected loss. The GPD process is bound to find at least a local minima of the expected loss for an infinite run of the algorithm. In practice, the recognizer is designed through a finite set of data. The designer ends up minimizing an empirical loss defined over the finite set of data, which is different from the initial target. In this chapter, we discuss the following issues: 1) what is the link between the empirical loss and the expected loss ? 2) is a minimum of the empirical loss "close" to a minimum of the expected loss ?

5.1 Introduction

The objective function of DFE is simply the *expected or ideal cost* $\mathcal{L}(\Phi)$ defined as the expectation of the loss over the observation space $\boldsymbol{\Omega}_s$:

$$\mathcal{L}(\Phi) = \sum_{k=1}^{M} \int_{\boldsymbol{\Omega}_s} \Pr(\boldsymbol{s}, C_k) 1(\boldsymbol{s} \in C_k) \ell(\boldsymbol{s}; \Phi) d\boldsymbol{s}, \tag{5.1}$$

which can be minimized using the GPD algorithm. That is, for each in coming input data, update the recognizer parameters according to

$$\Phi[\tau + 1] \quad = \quad \Phi[\tau] - \epsilon_\tau \boldsymbol{U} \nabla \ell(\boldsymbol{s}; \Phi), \tag{5.2}$$

where the learning rate should theoretically obey the stochastic constraints:

$$\sum_{\tau=1}^{\infty} \epsilon_\tau \quad = \quad \infty, \tag{5.3}$$

$$\sum_{\tau=1}^{\infty} \epsilon_\tau^2 \quad < \quad \infty, \tag{5.4}$$

for leading to the minimization of the expected loss.

In practice, one only has access to a *finite* set of training data and the classifier is designed through this set of data, which are drawn from the observation space $\boldsymbol{\Omega}_x$ according to a probability p($\boldsymbol{s}$). On the other hand, the GPD algorithm is only asymptotically efficient. That is, optimality is realized for an infinite number of examples, which is not possible in practice. This leads to the problem of minimizing the loss $\mathcal{L}(\Phi)$, using a finite set of data.

Let $\mathcal{S}$ be the set of training data available $\mathcal{S} = \{\boldsymbol{s}_n\}$ of size N, for $n = \{1, \ldots, N\}$. The feature extractor $\mathcal{F}(\boldsymbol{s}; \Theta)$ preprocesses $\mathcal{S}$ by performing the mapping $\mathcal{X} = \mathcal{F}(\mathcal{S}; \Theta)$ where $\mathcal{X} = \{\boldsymbol{x}_n\}$, $\boldsymbol{x}_n = \mathcal{F}(\boldsymbol{s}_n; \Theta)$ and Θ is the parameter set of the feature extractor.

An objective function based on the training set $\mathcal{S}$ of size N is the empirical loss function:

$$\mathcal{L}_N(\Phi) = \frac{1}{N} \sum_{k=1}^{N} \ell(\boldsymbol{s}_k; \Phi). \tag{5.5}$$

The empirical loss $\mathcal{L}_N(\Phi)$ is directly accessible and can be used as an indirect link to the expected lost value. In particular, within the MCE framework, the empirical loss yields values close to the empirical error rate. This can be easily seen from (5.5) by noting that $\ell(\boldsymbol{s}_k; \Phi)$ approximates the 0-1 lost. For system designers in the speech recognition society, the above observation is rather of importance because, the empirical error rate is usually chosen as an estimate of the performance of the system. Actually, in the speech recognition society, researchers compete in terms of error rate (meaning empirical error rate), over standard databases. It is a common observation that speech system designers sometimes end up over-training a particular method for a given database.

As indicated above, the empirical loss has values close to the empirical error rate's values of the recognizer on a given set of training data. Fig 5.1 illustrates the evolution of the empirical error rate

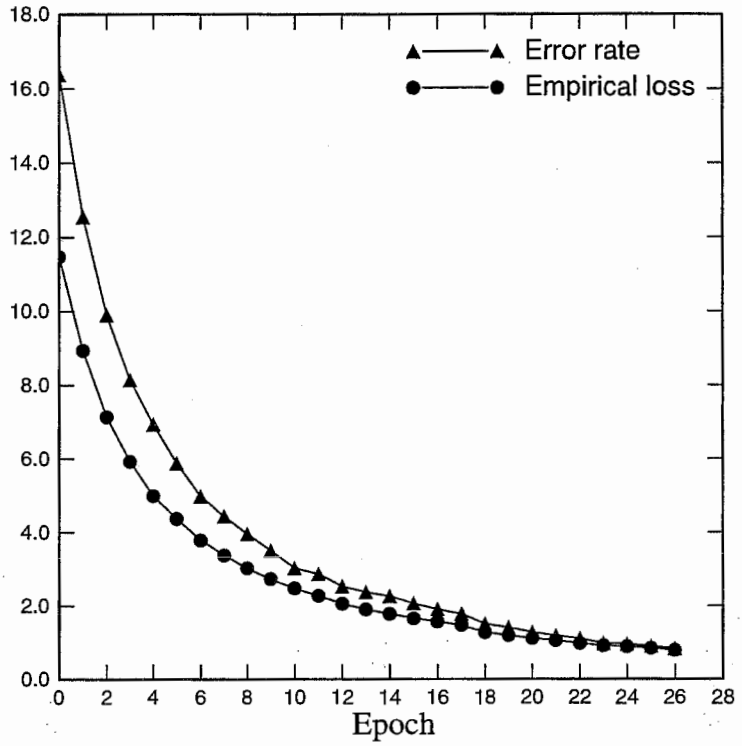


Figure 5.1: Empirical error rate and empirical loss as a function of epochs for a letter recognition task using classical MCE with a deterministic gradient optimization.

and the empirical loss $\mathcal{L}_N(\Phi)$ for a letter recognition task using classical MCE with a deterministic gradient descent. The loss function used here is a mixture of error functions defined in Chapter 6, section 6.2.2. The curves displays the same evolution during learning. The difference between the two curves is mainly due to positive misclassification measures values, as the loss function is almost equal to zero for negative misclassification measures. As learning goes on, there are fewer and fewer positive misclassification measures and, consequently, the two curves become closer in values.

Since the empirical error rate is usually taken as an estimate of the classifier error rate, the empirical loss is in turn an estimate of the classifier error rate. Given a training set of size N, the natural question that arises is how good is this estimate.

This chapter discusses the minimization of the expected lost given a finite number N of input data. Again, we have seen in Chapter 4 that the expected loss is an approximation of the probability of error of the recognizer. Now, assuming that this approximation, which depends on the choice of the loss function, is consistent with the minima of both functions, the focus is to examine the

validity of these estimated minima.

As previously stated, the goal is to minimize the expected lost, given the training set of size N and we only have access to the value of the empirical loss. So far in our knowledge, there is no algorithm, which, relying on a finite number of data, will ultimately minimize the expected loss. Thus, an inductive approach should be carried out. That is, focus is on minimizing the empirical loss and estimating the resulting probability of error, given the data set.

The reader is introduced to some mathematical statements establishing relationships between the empirical loss defined on a finite set of samples and the expected lost, followed by an investigation of the degree of confidence attached to these estimations. This investigation is mainly based on the pattern recognition theory developed by Vapnik [Vapnik, 1982].

5.2 Simple links between the empirical loss and the expected loss

Given a parameter set Φ, links between the empirical loss $\mathcal{L}_N(\Phi)$ and the expected loss $\mathcal{L}(\Phi)$ are investigated by considering the framework in which input training data are randomly chosen from the observation space $\boldsymbol{\Omega}_s$, according to a given probability distribution $\mathrm{p}(\boldsymbol{s})$. A data-vector can be represented by a random variable $\boldsymbol{s}$ (For simplicity, the same notation for a data vector $\boldsymbol{s}$ and the corresponding random variable is used.)

5.2.1 Law of Large Numbers

The first and obvious relation between the expected loss and empirical loss is given by the strong Law of Large Numbers. The strong Law of Large Numbers affirms the convergence in probability, of an averaging sum to its expectation when the number of samples increases. For a fixed Φ, given that $\boldsymbol{s}$ is a random variable, if we consider $\ell(\boldsymbol{s};\Phi)$ as a random variable, $\mathcal{L}(\Phi)$ is simply its expectation and $\mathcal{L}_N(\Phi)$ represents its averaging sum. For a fixed Φ, the strong Law of Large Numbers leads to the following assumption:

$$\Pr\left(\lim_{N\to+\infty} \mathcal{L}_N(\Phi) = \mathcal{L}(\Phi)\right) = 1, \text{ for any } \Phi. \tag{5.6}$$

The strong Law of Large Numbers means that increasing the number of tokens makes $\mathcal{L}_N(\Phi)$ to be close to $\mathcal{L}(\Phi)$, which is not a surprising result. The interesting fact is that (as we show later) within a degree of confidence, it is possible to let the values of $\mathcal{L}_N(\Phi)$ be close to the values of $\mathcal{L}(\Phi)$ with a certain precision by use of an appropriate choice of the number of tokens. This is certified by the weak Law of Large Numbers, which in this context, can be mathematically written as:

$$\forall \varepsilon > 0 \lim_{N\to+\infty} \Pr\{|\mathcal{L}_N(\Phi) - \mathcal{L}(\Phi)| > \varepsilon\} = 0, \text{ for any } \Phi. \tag{5.7}$$

Given any small positive number ε, the distance between $\mathcal{L}_N(\Phi)$ and $\mathcal{L}(\Phi)$ can be made smaller than ε by increasing the number of data. Unfortunately, the weak Law of Large Numbers does

not provide an estimate of the convergence rate. An upper bound of the right part of (5.7) as a function of the number of data N is needed. Inequalities, such as the Bienaimé-Chebyshev's, although providing such a bound, requires the estimation of the variance.

5.2.2 Hoeffding inequality

The Hoeffding inequality [Hoeffding, 1963] provides an upper bound which does not require variance estimation (the only assumption is a have a bounded random variable). In the MCE framework, considering that $0 \leq \ell(\boldsymbol{s}; \Phi) \leq 1$, the Hoeffding inequality gives

$$\forall \varepsilon > 0, \quad \Pr\{|\mathcal{L}_N(\Phi) - \mathcal{L}(\Phi)| > \varepsilon\} \leq 2e^{-2\varepsilon^2 N}, \text{ for any } \Phi. \tag{5.8}$$

The above equation could be rewritten more practically as

$$\Pr\{|\mathcal{L}_N(\Phi) - \mathcal{L}(\Phi)| \leq \varepsilon\} > 1 - 2e^{-2\varepsilon^2 N}, \text{ for any } \Phi. \tag{5.9}$$

The right term of (5.9) does not depend on Φ and decreases exponentially to 0 for larger N. Consequently, given a fixed Φ, $\mathcal{L}_N(\Phi)$ converges exponentially to $\mathcal{L}(\Phi)$ in probability. This result could then be used to estimate the value of the expected lost, given a parameter set Φ. Moreover, there is no relation whatsoever with the type of classifier used. In practice, this result could be used as follows. Suppose a system trained in the MCE framework using $N = 100$. Using (5.9), it is 86% probable ($1 - 2e^{-2\left(\frac{\varepsilon}{\tau}\right)^2 T} = 0.86$) that the value of the empirical loss is equal to the value of the ideal loss assuming a 10% error ($\varepsilon = 0.1$).

We shall be aware that the Hoeffding inequality shows convergence in probability. That is, in practice, $\mathcal{L}_N(\Phi)$ converges to $\mathcal{L}(\Phi)$ with fluctuations whose amplitudes depend on the task at hand as well the recognizer structure. The obvious question is whether minimizing the empirical loss results in a minimization of the expected loss. This question is not answered by the Hoeffding inequality, which only provides convergence results for a fixed value of Φ.

5.3 Estimating a Minimum of the Expected Lost

The main goal in the MCE estimation is to find at least a local minimum of $\mathcal{L}(\Phi)$. The Hoeffding inequality does not, however, give information about how close we are to the minima of the expected lost, given a estimated minima of the empirical lost: a parameter set Φ which yields a minimum value of the empirical lost will not automatically yields a value of the expected lost close to the minimum. However, we have the following proposition.

5.3.1 Uniform convergence in Probability

Proposition 1 *If $\mathcal{L}_N(\cdot)$ converge uniformly to $\mathcal{L}(\cdot)$ in probability. i.e.,*

$$\forall \varepsilon > 0 \lim_{N \to +\infty} \Pr\left\{\sup_{\Phi} |\mathcal{L}_N(\Phi) - \mathcal{L}(\Phi)| \geq \varepsilon\right\} = 0 \tag{5.10}$$

then a local minimum of the empirical loss converges in probability to a local minimum of the ideal loss as $N \rightarrow \infty$.

Proof:

To simplify we suppose, there is only one minima $\inf_\Phi \mathcal{L}(\Phi)$ of the expected lost[1]. $\forall\ \varepsilon$, there is a N such that $\varepsilon \geq \sup_\Phi |\mathcal{L}_N(\Phi) - \mathcal{L}(\Phi)|$. For such N, let us define Φ_e as $\mathcal{L}_N(\Phi_e) = \inf_\Phi \mathcal{L}_N(\Phi)$, a local minimum of the empirical loss function . It is straightforward that

$$|\mathcal{L}_N(\Phi_e) - \mathcal{L}(\Phi_e)| < \varepsilon$$

and

$$|\inf_\Phi \mathcal{L}_N(\Phi) - \inf_\Phi \mathcal{L}(\Phi)| \leq \varepsilon.$$

Hence

$$|\mathcal{L}(\Phi_e) - \inf_\Phi \mathcal{L}(\Phi)| \leq |\mathcal{L}(\Phi_e) - \mathcal{L}_N(\Phi_e)| + |\inf_\Phi \mathcal{L}_N - \inf_\Phi \mathcal{L}(\Phi)| \leq 2\varepsilon \tag{5.11}$$

□.

The meaning of the above proposition is that if the empirical loss approximates the expected lost in a uniform manner, then minimizing the empirical loss would be equivalent to finding a parameter set which is close to the minimum of the expected lost. The problem then is to find at least the sufficient conditions which realize the uniform convergence in probability of the empirical lost towards the expected lost. An interesting theory has been developed by Vapnik and Chervonenkis [Vapnik and Chervonenski, 1971] and Vapnik [Vapnik, 1982] which shows detailed convergence proofs concerning uniform convergence of a random variable towards its mean under certain conditions. We here provide a little study of this theory and present application within the MCE framework.

5.3.2 Vapnik and Chervonenski dimension

Let us consider a recognizer structure $\mathsf{R} = \{g_j(\boldsymbol{s}; \Phi)\}$. Let us assume that the loss function is a binary lost of the 0-1 form. Given a fixed architecture of the recognizer, a precise recognizer is characterized by a parameter set Φ. Thus, the overall recognizer can simply be represented by its binary loss function $\mathsf{R} = \{\ell_b(\boldsymbol{s}; \Phi)\}$, where $\ell_b(\boldsymbol{s}; \Phi)$ has value in $\{0, 1\}$. Clearly, a specific Φ defines a partioned space. The regions of this space represent a class of events. Let $\{\mathcal{R}(\Phi)\}$ be the partitioning provided by a given parameter set Φ. Given N data taken from the input space, there are at most 2^N separations or dichotomies that can be done on this data set. Let $d_\mathsf{R}(N)$ be the maximum number of separations or dichotomies of the data set realizable by the recognizer structure across all possible values of Φ. Vapnik and Chervonenkis [Vapnik and Chervonenski, 1971] have shown that there exist a number $d_{vc}(\mathsf{R})$, that can be finite or infinite, such that:

[1]Therefore, the proposition hold in the vicinity of a local minima.

$$\text{if } N \leq d_{vc}(\mathsf{R}) \qquad \text{then} \qquad d_{\mathsf{R}}(N) = 2^N \tag{5.12}$$

$$\text{if } N > d_{vc}(\mathsf{R}) \quad \text{then} d_{\mathsf{R}}(N) \leq \sum_{i=1}^{d_{vc}(\mathsf{R})} \binom{N}{i}. \tag{5.13}$$

$d_{vc}(\mathsf{R})$ is called the *capacity* or *VC dimension* of the recognizer R and is noted VCdim(R). Its value depends on the recognizer structure and the underlying feature space. In words, the VC-dimension of a recognizer structure is the maximum number of data taken from the input space, for which all possible dichotomies can be obtained, by using the architecture of R. Note that the above definition supposes that the final decision of the recognizer is of the 0-1 form.

For a more general decision scheme, that is, for a more general loss function $\ell(\boldsymbol{s}; \Phi)$, [Vapnik and Bottou, 1993] give the following definition. First, define a new binary loss function as

$$\ell_b(\boldsymbol{s}; \Phi) \quad = \quad \mathbf{1}((\ell(\boldsymbol{s}; \Phi) - \xi) \geq 0) \qquad \text{for } \xi \in \left[\inf_{\boldsymbol{s},\Phi} \ell(\boldsymbol{s}; \Phi), \sup_{\boldsymbol{s},\Phi} \ell(\boldsymbol{s}; \Phi)\right]. \tag{5.14}$$

The VC dimension corresponding to $\ell(\boldsymbol{s}; \Phi)$ is simply the VC dimension of the same recognizer when using $\ell_b(\boldsymbol{s}; \Phi)$. This generalizes the VC-dimension to any loss function. In the MCE framework, the value of ξ are contained in the interval $[0, 1]$.

An interesting result [Vapnik, 1982] is that for a finite value of VCdim(R) = h:

$$d_{\mathsf{R}}(N) \leq 1.5\frac{N^h}{h!} \qquad \text{for } N > h. \tag{5.15}$$

Examples of VC-dimension

There is no algorithm which computes the effective value of VCdim(R), given a classifier structure. A classical case is that the capacity of linear discriminators in an I dimensional space is equal to $I + 1$.

However, few results have been obtained for feed-forward architectures. Baum and Haussler [Baum and Haussler, 1989] have shown that a Multi-Layer Perceptron of N nodes and W weights has a capacity lower that $2W \log_2(eN)$, where e is the Neper number and that the VC-dimension of a completely connected two-layer feedforward network composed of I input units and L_1 first-layer units is no more than $2I\frac{L_1}{2}$. Bartlett [Bartlett, 1993] emphasized these results by showing that the VC-dimension of a feedforward architecture of I connections from the input units to the other units is greater that $I + 1$. Consequently, the VC-dimension of such an architecture increases linearly with the number of input units.

5.3.3 Uniform convergence of the empirical loss

For a recognizer of finite VC-dimension, an interesting theorem demonstrated by Vapnik [Vapnik, 1982] shows the uniform convergence of the empirical loss to the expected loss. Given recognizer R, the theorem, within the MCE framework, is as follows:

Theorem 1 (Vapnik and Chervonenski) *if VCdim(R) = h and N > h, then*

$$\Pr\left\{\sup_{\Phi}|\mathcal{L}_N(\Phi)-\mathcal{L}(\Phi))|\geq 2\sqrt{\frac{h(\ln\frac{2N}{h}+1)-\ln\frac{\nu}{9}}{N}}\right\}\leq\nu. \tag{5.16}$$

The proof of this theorem can be found in [Vapnik and Chervonenski, 1971] and [Vapnik, 1982]. Here, only a sketch of this proof is presented. First, Vapnik and Chervonenkis [Vapnik and Chervonenski, 1971] show the following result:

$$\Pr\left\{\sup_{\Phi}|\mathcal{L}_N(\Phi)-\mathcal{L}(\Phi)|\geq\varepsilon\right\}\leq 4d_{\mathsf{R}}(2N)e^{-\varepsilon^2N/8}\text{ for }N\geq 2/\varepsilon^2 \tag{5.17}$$

The above result is valid for any recognizer structure and does not require a finite VC dimension. Even if the right side of (5.17) is independant of Φ, its convergence rate remains quite unclear due the term $d_{\mathsf{R}}(2N)$. Consequently, a sufficient condition to achieve uniform convergence in probability of the empirical error rate to their expected loss is then to have a suitable bound of $d_{\mathsf{R}}(2N)$ by a polynomial such as $d_{\mathsf{R}}(N)\leq N^n+1$ for all N, which is the case when the classifier has a finite VC-dimension h as seen previously. Thus, if VCdim(R) is finite and according to (5.15), we have the following more practical bound:

$$P\left\{\sup_{\Phi}|\mathcal{L}_N(\Phi)-\mathcal{L}(\Phi)|\geq\varepsilon\right\}\leq 6\frac{(2N)^h}{h!}e^{-\varepsilon^2N/4}\qquad\text{for }N>h, \tag{5.18}$$

which, when using the Stirling formula ($\ln(h!)\simeq h\ln(h)-h$), gives the form written in the theorem.

Using the above theorem, we have the following estimate of a local minima of the expected lost.

$$\Pr\left\{\left|\mathcal{L}(\Phi_e)-\inf_{\Phi}\mathcal{L}(\Phi)\right|\leq D_{N,h,\nu}=4\sqrt{\frac{h(\ln\frac{2N}{h}+1)-\ln\frac{\nu}{9}}{N}}\right\}\geq 1-\nu\qquad\text{with }\Phi_e=\inf_{\Phi}\mathcal{L}_N(\Phi). \tag{5.19}$$

(5.19) means that, with probability $1-\nu$, a minima of the empirical loss is a minima of the expected loss, within an interval of confidence equal to $D_{N,h,\nu}$, which depends on the training size N, the capacity of the recognizer h and the degree of trust $1-\nu$. Given a certain trust $1-\nu$ and a capacity h, the interval of confidence $D_{N,h,\nu}$ can be decreased by increasing the training size N.

5.3.4 Use of VC dimension within MCE

The Vapnik and Chervonenski theorem affirms the uniform convergence of the MCE 's empirical loss to the MCE's expected lost when the number of tokens increases as soon as the capacity h of the recognizer R is of finite VC dimension. Consequently, within the MCE framework, for a recognizer of finite VC dimension, the following statement are true.

- A minimum value of the empirical loss is an estimate of a local minimum of the expected loss.
- The estimate can be improved by increasing the number of data
- The estimate can be improved by decreasing the capacity of the recognizer.

Within the MCE framework, the loss $\ell(\boldsymbol{s};\Phi)$ is bounded by 0 and 1. The only requirement therefore is to use a classifier of finite VC dimension. Again, according to (5.11), the following assumption is true for recognizer of VC dimension h, trained within the MCE framework:

$$\Pr\left\{\left|\mathcal{L}(\Phi_e)-\inf_{\Phi}\mathcal{L}(\Phi)\right|\leq 4\sqrt{\frac{h(\ln\frac{2N}{h}+1)-\ln\frac{\nu}{9}}{N}}=2\varepsilon\right\}\geq 1-\nu. \tag{5.20}$$

where Φ_e denotes a parameter set which yields a minimum value of the empirical loss and $\inf_{\Phi}\mathcal{L}(\Phi)$ is the closest minimum of the expected lost.

This result can be used as follows: suppose, we have reached a local minimum of the empirical loss represented by Φ_e using the N training tokens. Assuming that the capacity of our system is greater that h, we can affirm with probability $1-\nu$ that Φ_e realizes a minimum value of the expected lost assuming an error of ε, which means

$$\left|\mathcal{L}(\Phi_e)-\inf_{\Phi}\mathcal{L}(\Phi)\right|\leq\varepsilon$$

where

$$\varepsilon=2\sqrt{\frac{h(\ln\frac{2N}{h}+1)-\ln\frac{\nu}{9}}{N}}.$$

If the capacity h is unknown, a bound based on the Hoeffding inequality can be used, i.e.,

$$\varepsilon=2\sqrt{\frac{-\ln(\frac{\nu}{2})}{2N}}.$$

5.4 Comparing DFE and Classical MCE/GPD

5.4.1 Empirical error rate

MCE/GPD is equivalent to using DFE with a fixed feature extractor. Consequently, the empirical loss of the DFE should be smaller than MCE/GPD's after optimization. Let us formalize this rather obvious result. The empirical loss of the recognizer can be written as

$$\mathcal{L}_N(\Phi=\{\Theta,\Lambda\})=\frac{1}{N}\sum_{k=1}^{N}\ell(\boldsymbol{s}_k;\Theta,\Lambda). \tag{5.21}$$

DFE tries to find

$$\Phi^{\star} \;=\; \arg\min_{\Phi}\mathcal{L}_N(\Phi) \tag{5.22}$$

whereas classical MCE/GPD search for

$$\Lambda^{\star} \;=\; \arg\min_{\Lambda}\mathcal{L}_N(\Theta_c,\Lambda), \tag{5.23}$$

where Θ_c is a parameter set describing the conventional feature extraction scheme. It is straightforward that

$$\inf_{\{\Theta,\Lambda\}}\mathcal{L}_N(\{\Theta,\Lambda\})\leq\inf_{\Lambda}\mathcal{L}_N(\{\Theta_c,\Lambda\}) \tag{5.24}$$

for all N. Consequently, assuming accurate optimization, DFE always yields a smaller empirical error rate than classical MCE/GPD.

5.4.2 VC-dimension

Let $\mathsf{R}\{\Phi\}$ be a recognizer structure and $\mathrm{C}(\Lambda)$ its classifier module. Since $\Lambda \subset \Phi$, we have

$$\mathrm{VCdim}(\mathsf{R}) \geq \mathrm{VCdim}(\mathrm{C}). \tag{5.25}$$

That is, the VC dimension of a DFE-based recognizer is higher than the corresponding classifier. Thus, DFE requires more training data than classical MCE in order to achieve a good estimate of the minimum of the expected loss. This is intuitively true if one considers the fact that a DFE recognizer yields more parameters to train.

5.5 Conclusion

In this Chapter, use of a finite number of data in the design of a DFE based system was investigated. This study was motivated by the fact that the MCE criterion tries to minimize the expected loss defined over an infinite number of data. In practice, one only has access to the empirical loss defined over a finite set of data. Thus, the link between optimizing an empirical loss and the effect on the expected loss was investigated.

In light of the above, it can be argued that, in practice, when training a system under MCE, the only requirement is then to use a recognizer of finite VC dimension since the loss function is already bounded by 0 and 1. This ensures that a minimum of the empirical lost is a reasonable estimate of a minimum of the expected lost, with a certain degree of confidence. Given a trust $1 - \nu$, the lower the capacity of the recognizer, the faster the convergence of a minimum of the empirical loss to a minimum of the expected lost when the number of tokens increases: more training tokens are needed, for a system showing a high capacity to reach a given precision ε of closeness to the true minima.

Chapter 6

Optimization of DFE-based Recognizers

Ask and it will be given you
seek and you will find
knock and it will be opened to you.

-Jesus of Nazareth-

This chapter discusses various optimization methodologies that can be carried out within the DFE framework. The study concerns an optimal choice of the MCE loss function and optimization algorithms that can be used in practice. In particular, the Modular Generalized Probabilistic Descent (MGPD) and the Incremental Generalized Probabilistic Descent (IGPD) are proposed as alternatives to overcome the learning instability of the Generalized Probabilistic Descent method in the DFE framework.

6.1 Introduction

Learning in the MCE framework is of the utmost importance for accurate achievement of robust minimum-error oriented speech recognizers. Various learning methodologies have been adopted in the design of MCE-based speech recognizers, including simulated annealing [Imai and Ando, 1992], batch-based gradient descent [Rainton and Sagayama, 1992] and GPD-based optimization [McDermott and Katagiri, 1992; McDermott and Katagiri, 1994]. Gradient-based methods are the most widely used technique due to their efficiency in term of speed and the attractive formalism they carry. In particular, GPD provides an elegant theorem which guarantees accurate local optimization, given certain conditions on the choice of the learning rate and the positive definite matrix [Katagiri *et al.*, 1991b]. Again, GPD is only asymptotically efficient. That is, optimality is achieved for an infinite number of training examples, drawn according to a certain distribution.

One reason which may explain the difficulty in choosing the learning rate in GPD-based optimization, may be due to the saturation property of the loss function (the smooth approximation of the 0-1 cost). The chosen loss function usually displays an area where training does not occur even when the token is misclassified. In that situation, one may need to compensate this effect by a larger learning rate, which cannot be achieved by GPD since the learning rate depends only on the iteration index and decreases at each iteration [Katagiri *et al.*, 1990].

In practice, one must experimentally find the optimal learning parameters (learning rate and positive definite matrix) according to the task involved. The positive definite matrix is usually set to the unit matrix [McDermott and Katagiri, 1994], or a to diagonal matrix when different king of parameters are involved [Chou *et al.*, 1992]. However, these choices are far from optimality when a complex error surface is encountered. In particular, in the DFE framework, training is highly sensitive to an accurate tuning of the unit matrix because of various parameters sometimes having diverging influences on the direction of the gradient descent. Furthermore, the time spent in the tuning of the training parameters is proportional to both the complexity of the recognizer structure and the complexity of the task. In DFE-based speech recognizer, where the recognizer is expected to simultaneously learn the right features and accurately classify them, the number of trials to be tested before convergence increases with the structure of the feature extraction module. The complexity of the error surface adds to the burden of finding the right training parameters. That is, due to the complexity of the structure as well as the statistical distribution of speech patterns, the error surface is likely to have a larger number of local minima, in which a simple implementation of a gradient-descent based algorithm could be easily trapped.

One way to tackle the complexity of the error surface in the DFE framework is to make use of higher order derivatives. Higher order derivatives may provide local information about the slope of the error surface which can be used to locally tuned the learning rate [White, 1989]. A popular approach within this framework is to make use of second-order derivatives with respect to each recognizer parameter and then perform the Newton algorithm (or a derived-Newton algorithm) [Battiti, 1992]. However, computing second-order derivatives is extremely costly in computing time,

thus usually an algorithm that approximates these derivatives is usually carried out [Watrous, 1987; Becker and Le Cun, 1989].

Another approach is to iteratively adjust the learning rate. The learning rate adjustment is thus based on a memory of the previous learned examples which provides some insight about "how much" have been learned and "how much" remain to be learned. Furthermore, each parameters could be assigned it own learning rate [Jacobs, 1988], which may solve the problem of the optimum choice of the positive definite matrix encountered in the GPD paradigm. A private learning rate for each parameter is particularly attractive in the discriminative training framework, since it extends the discriminative learning to the the spaced spanned by the learning parameters, thus adding more discriminative power. However, since speech recognizers usually include a high number of parameters, assigning a learning rate for each parameter maybe computationally costly. A tradeoff is to let the parameters that model a category share a learning rate, which is then updated iteratively.

Here, two methods of training in the DFE framework are proposed: the modular Generalized Probabilistic Descent method (MGPD) and the Incremental Generalized Probabilistic Descent (IGPD). Evaluation of the proposed algorithms is described in the context of recognizing single vowel frame in a multi-speaker mode. The algorithms are compared against the classical use of GPD. For each algorithm, optimization of a classical MCE-based recognizer (only the classifier module is optimized) is also investigated. The recognizer is a simple distance classifier fully described in Chapter 8 and the feature extraction is a filter bank process, where the center frequency, the bandwidth and the gain of each filter are free parameters (see Chapter 8 for full description of the filter bank).

Before applying any learning paradigm, first a convenient MCE loss function shall be chosen. The first part of this chapter is concerned with the selection of an optimal loss function, given a task.

6.2 Choice of Loss Function in MCE

Various approximations of the smooth error count measure are given in the literature. Here, we describe few, chosen functions that were used throughout this report. A subsequent discriminative measure is simply represented by d, without the class-label subscript.

6.2.1 Classical MCE losses

Sigmoidal loss

One of the widely used 0-1 approximation is the sigmoid function. The sigmoid is widely used in feed-forward neural network as the neuron transfer function. Its shape and the shape of its derivative are both shown in the top-left of Fig. 6.1 and are defined as :

$$\ell(d) = \frac{1}{1+e^{-\alpha(d-\beta)}} \tag{6.1}$$

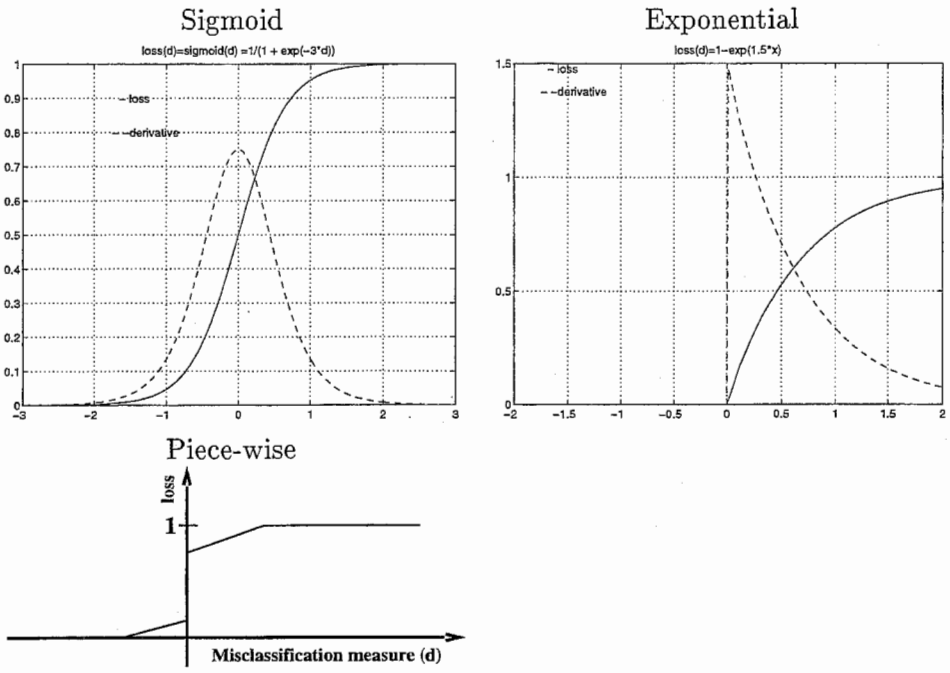


Figure 6.1: Various loss functions and their derivatives.

$$\ell'(d) \;=\; \alpha\ell(d)(1-\ell(d)) \tag{6.2}$$

with a large positive α. One can see that this loss approximates an ideal binary loss function well and is indefinitely continuous. The sigmoid loss focuses its learning to the value of d that are close to β. Thus, one has the freedom to control the region where the learning must occurs.

Exponential-based loss

The exponential-based approximation is not as widely used as the sigmoidal loss function but provides a good approximation to the 0-1 step wise function. Its shape is shown in the top-right of Fig. 6.1 and it is defined as

$$\ell(d) \;=\; (1-\exp(-\alpha d))\mathbf{1}(d>0), \qquad \alpha \geq 0 \tag{6.3}$$

$$\ell'(d) \;=\; \alpha\exp(-\alpha d)\mathbf{1}(d>0). \tag{6.4}$$

The exponential loss has a discontinuity at the origin. However, it is convenient means to derive an analytical form of the loss when d is a logarithm.

Piece-wise linear loss

The piece-wise linear loss is defined as

$$\ell(d) = \begin{cases} 0 & \text{if } d < -Q1 \\ (d+Q1)/(Q1+Q2) & \text{if } -Q1 \leq d < Q2 \\ 1 & \text{if } d \geq Q2. \end{cases} \tag{6.5}$$

and shown in Figure 6.1. This loss is discontinuous at a finite set of points, and thus can broadly be considered differentiable. The slope can be controlled by the parameters $Q1$ and $Q2$.

6.2.2 MCE loss as a distribution function

Most MCE studies have used the sigmoid as MCE loss function. Indeed, the sigmoid seems a convenient choice when the misclassification measures are mainly located around the zero area. Even in such case, one is still face with the choice the optimal parameter α. Here, we present a new MCE loss function, which is believed to be quite general and can be used in most tasks. Furthermore, is is shown how to automatically select the loss parameters given the data.

A recognizer structure Φ establishes a mapping $\boldsymbol{s} \rightarrow d_{\boldsymbol{s}}$, between a data and its corresponding misclassification measure $d_{\boldsymbol{s}}$. This induces a probability measure $\Pr(\cdot;\Phi)$ on the space spanned by the value of $d_{\boldsymbol{s}}$, which is the space of real numbers R. Having this in mind, $\ell(d_{\boldsymbol{s}})$ can be viewed as the distribution function of the probability measure $\Pr\{X < d_{\boldsymbol{s}_n};\Phi\}$ defined on R, where X is a random number on the real axis. This viewpoint is inspired by the fact the loss $\ell(\cdot)$ satisfy the requirements of a distribution function. That is,

- The loss has its values between 0 and 1.
- The loss is a monotonically increasing function.
- The loss limits are 0 and 1, for $d_{\boldsymbol{s}} \rightarrow -\infty$ and $d_{\boldsymbol{s}} \rightarrow \infty$, respectively.

With the above assumption in mind, the derivative of the loss is simply the probability density function.

Loss parameter estimation

Given that the loss function is viewed as a probability measure, the estimation of loss parameters can be done by any probability estimation technique available. The use of the EM algorithm is straightforward. In this context, the derivative of the loss can be modeled as a Gaussian mixture, which gives

$$\ell'(d) = \sum_{i=1}^{L} c_i \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(-\frac{(d-\mu_i)^2}{2\sigma_i^2}\right) \tag{6.6}$$

$$= \sum_{i=1}^{L} c_i \mathcal{N}(d, \mu_i, \sigma_i). \tag{6.7}$$

The corresponding loss is simply the anti-derivative function, which obeys the MCE requirement of a loss. The following loss is one such a loss:

$$\ell(d) = \sum_{i=1}^{L} c_i \left(\frac{1}{2}\mathrm{erf}\left(\frac{d-\mu_i}{\sqrt{2}\sigma_i}\right) + \frac{1}{2}\right). \tag{6.8}$$

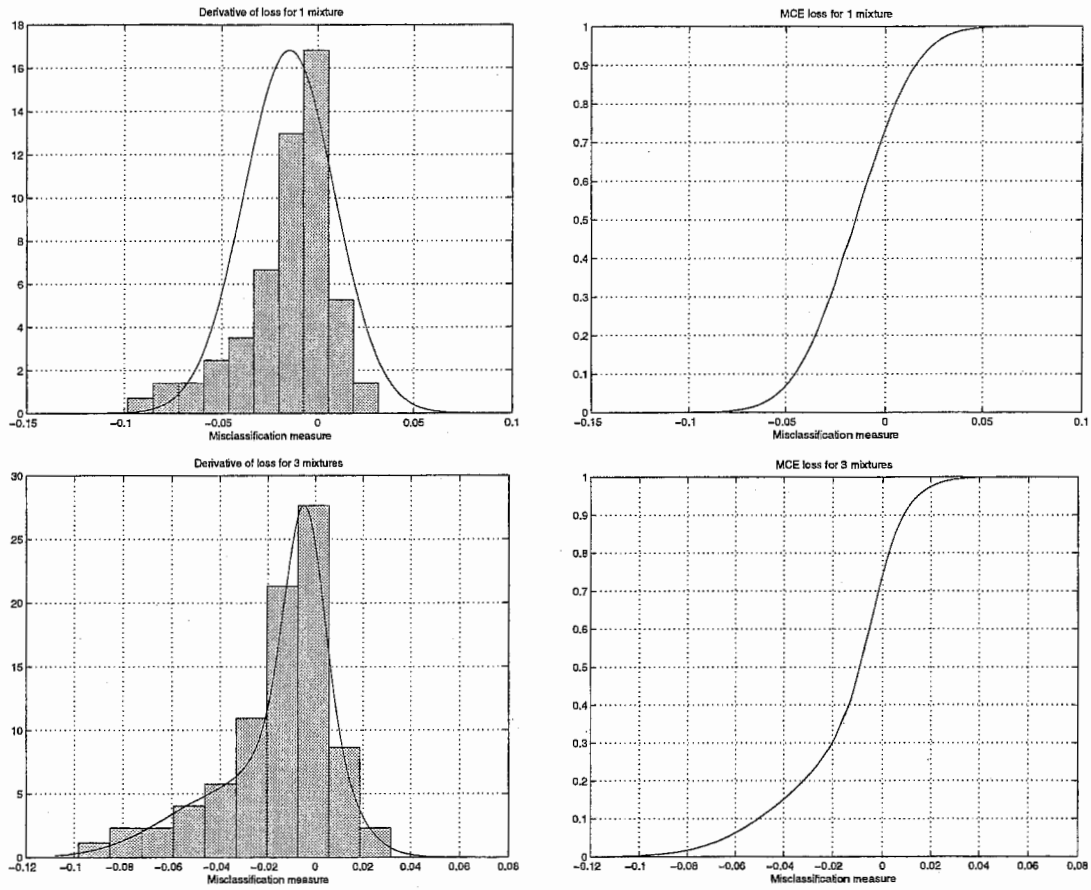


Figure 6.2: Error-function-based losses and derivatives. The histogram represents the misclassification measures of a database of 150 samples. **Top figures:** 1 mixture loss function and its derivative estimated by EM. **Bottom figures:** 3 mixtures loss function and its derivative estimated by EM.

where erf(·) is the widely known Error function and is defined as.

$$\text{erf}(d) = \frac{2}{\sqrt{\pi}} \int_0^d \exp(-y^2) dy. \tag{6.9}$$

The error-function-based loss basically shares the same properties with the sigmoid. The definition of (6.8) introduces the loss as mixture of L error functions, with parameters μ_i, σ_i and c_i, which control the location, the width and the contribution of each error function within the loss definition.

Fig. 6.2 shows examples of error-function-based losses estimated by the EM algorithm on the set of misclassification measures, for the one and the three mixture cases. The one mixture case is essentially equivalent to using a sigmoid. Using three mixtures generates an asymmetrical loss function which matches the data in the MLE sense. The MLE estimation automatically finds the right parameters of the loss, thus relieving the designer of the burden of loss tuning. In this specific example, the MLE estimation was carried over the whole data set. An alternative is to consider only positive misclassification measures, for learning to occur for misclassified tokens only. Note that, the process of loss estimation can be also be carried out sequentially during learning (for

instance, after an epoch).

6.3 Minimizing the empirical loss by GPD

After an appropriate choice of loss, minimization of the empirical loss $\mathcal{L}_N(\Phi)$, given a data set of size N can be carried out, by any method available. Minimization of $\mathcal{L}_N(\Phi)$ can be done by a simple gradient descent which performs the following iteration:

$$\Phi[\tau+1] = \Phi[\tau] - \epsilon \nabla \mathcal{L}_N \tag{6.10}$$

$$= \Phi[\tau] - \frac{\epsilon}{N} \sum_{\tau=1}^{N} \nabla \ell(\boldsymbol{s}_k; \Phi), \tag{6.11}$$

where $\Phi[\tau]$ denotes the current status of the parameters set at iteration τ and ϵ is the training rate. This iterative process, known as deterministic gradient descent method, has been widely applied in various Minimum Error-based classifier optimization including Neural Networks [Sugiyama and Kurinami, 1992] or Hidden Markov Modeling of Speech [Rainton and Sagayama, 1992]. However, performing the deterministic gradient descent could become impractical when the number of training samples N increases. Furthermore, the above gradient descent method, although efficient in minimizing the empirical error rate, does not have a clear link on the minimization of the expected loss, in contrast to the GPD algorithm. As seen in the previous chapter, minimization of the expected loss is guaranteed for a recognizer of a finite VC dimension.

6.3.1 Choice of the learning rate

In GPD-based gradient descent learning, performance usually depends on an accurate choice of the learning rate. In theory, as seen before, the learning should obey the stochastic constraints:

$$\sum_{\tau=1}^{\infty} \epsilon_\tau = \infty, \tag{6.12}$$

$$\sum_{\tau=1}^{\infty} \epsilon_\tau^2 < \infty, \tag{6.13}$$

for leading to the minimization of the expected loss .

In practice, infinite training is not possible. Similar to other MCE/GPD applications, DFE simply tries to reduce the classification errors over the given finite training pattern set by minimization of the empirical loss. The constraint learning rate of (6.13) should be somehow simulated for this finite case.

The basic ideas behind the above constraints are: 1) the training rate should be a decreasing function of the number of iterations 2) the training rate should never be null if there is an incoming data. Various training rate are here proposed based on the observation above. One classical example is to choose

$$\epsilon_\tau^{(1)} = \frac{\epsilon_o}{a\tau + b}, \tag{6.14}$$

where the parameters a and b are chosen such that

$$\epsilon_\tau^{(1)} = \epsilon_o \left(1 - \frac{\tau}{\mathcal{N}}\right) \tag{6.15}$$

for a finite number of iterations $\mathcal{N} = NP$ and P being the number of epochs. That is, P is the number of times the whole training set is presented to recognizer during the training process.

Sometimes, a fast learning is needed for the first iterations. Thus, Darken [Darken *et al.*, 1992], proposed the "search-then-converge learning rate", which is defined for an infinite number of iterations as

$$\epsilon_\tau^{(2)} = \epsilon_o \frac{1 + \frac{a}{\epsilon_o}\frac{\tau}{\tau_o}}{1 + \frac{a}{\epsilon_o}\frac{\tau}{\tau_o} + \tau\frac{\tau^2}{{\tau_o}^2}}. \tag{6.16}$$

This learning scheme obeys the following properties:

$$\epsilon_\tau^{(2)} \sim \epsilon_o \quad \text{when} \quad \tau \ll \tau_o \tag{6.17}$$

$$\epsilon_\tau^{(2)} \sim \frac{a}{\tau} \quad \text{when} \quad \tau \gg \tau_o \tag{6.18}$$

which means that the learning rate is approximately constant (equal to ϵ_o) for iterations which are much smaller that τ_o and decreases linearly at rate a for iterations which are far greater than τ_o. For a finite number of $\mathcal{N}$ iterations, this learning scheme could be approximated by

$$\epsilon_\tau^{(3)} = \frac{\epsilon_\tau^{(2)} - \epsilon_\mathcal{N}^{(2)}}{\epsilon_0^{(2)} - \epsilon_\mathcal{N}^{(2)}}. \tag{6.19}$$

Fig. 6.3 shows the behavior of the two types of learning for a fixed number of iterations.

Fig. 6.4 displays the empirical error surfaces using the 0-1 loss and its approximation by a sigmoid. Also shown in the figure are the curves along the surface during batch and GPD learning. The error surfaces are computed from a two-class problem that consists of classifying one-dimensional data. Data were generated artificially by computer, simulating two sources of overlapping Gaussian distributions with means -0.5 and 0.3 and a variance of 0.5. The parameters of the classifiers are thus simply a configuration of two values (referred to as class1 and class2 in the figure), representing each category. The error surface is generated by plotting the value of the overall error for each configuration. The minimization problem consists of finding the configuration which locates a valley of the error surface.

From this figure, it can be seen that the 0-1 loss-based error surface is piecewise discontinuous. The sigmoid-based error surface smoothes out the discontinuity of the 0-1 error surface thus permitting the use a gradient descent as optimization method. GPD or stochastic gradient descent-based learning is shown in the bottom figure and deterministic gradient descent is shown in the middle figure. The deterministic gradient descent proceeds by "jump" along the error surface in the direction of minimum gradient whereas the stochastic gradient descent proceeds by little steps.

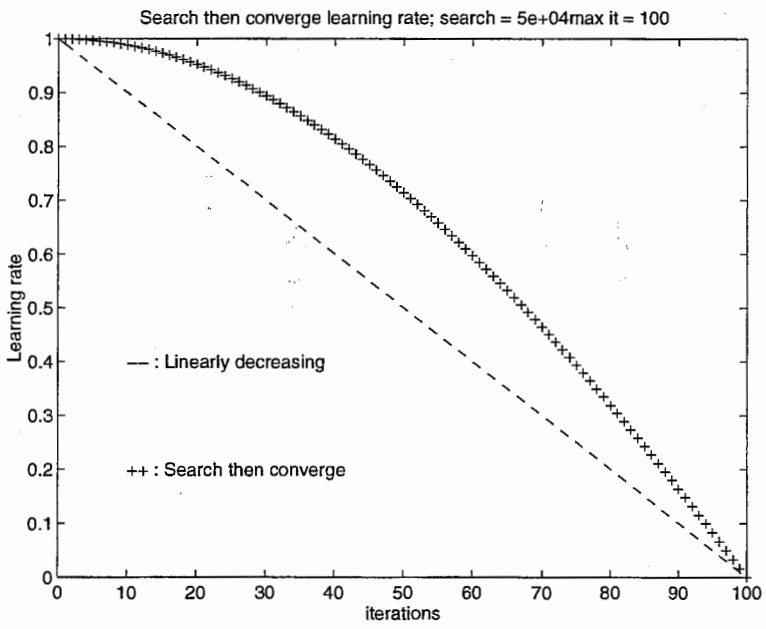


Figure 6.3: Different types of learning rates.

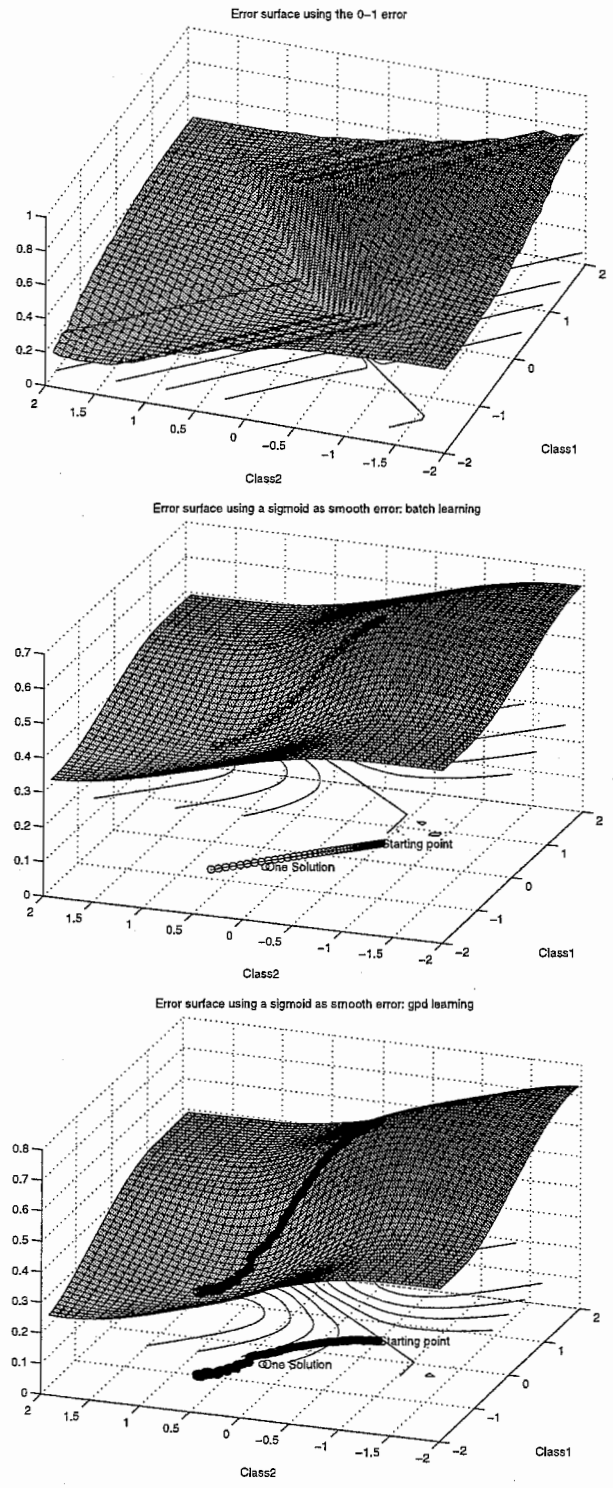


Figure 6.4: Error surface and minimization. The error surface is due to one-dimensional data generated by computer simulating two sources of Gaussian densities centered at -0.5 and 0.3 with a standard deviation of 0.5. Each class is represented by its means, referred to as "class1" and "class2". **Top:** 0-1 loss-based error surface. **Middle:** Error surface using a sigmoid approximation of the error and the curve traced by the estimated configurations during the deterministic gradient descent learning. **Bottom:** Same as middle but with a curve which shows the evolution of the GPD learning.

6.4 Modular Generalized Probabilistic Descent method

The learning rule of (5.2) carries in a compact form the interaction between the classifier's parameter and feature extractor's parameter. However, in practice, if both parameters are different in nature, a straightforward use of (5.2) may lead to instability (no immediate convergence). It may be useful, in light of better comprehension of the interaction of the feature extractor process to run a modular optimization scheme according to the type of parameters. This is done through an appropriate choice of the positive definite matrix $\boldsymbol{U}$ according the type of parameters to be optimized. That is, let $\boldsymbol{U} = [\boldsymbol{U}1, \boldsymbol{U}2]$ such that (5.2) is replaced by

$$\begin{aligned} \Lambda[\tau+1] &= \Lambda[\tau] - \varepsilon_\tau \boldsymbol{U}_1 \nabla_\Lambda \ell(\boldsymbol{s}; \Lambda[\tau], \Theta[\tau]) = \Lambda[\tau] - \varepsilon_\tau \boldsymbol{U}_1 \nabla_\Lambda \ell(\boldsymbol{x}; \Lambda[\tau]) \\ \Theta[\tau+1] &= \Theta[\tau] - \rho_\tau \boldsymbol{U}_2 \nabla_\Theta \ell(\boldsymbol{s}; \Lambda[\tau], \Theta[\tau]) = \Theta[\tau] - \rho_\tau \boldsymbol{U}_2 \nabla_{\boldsymbol{x}} \ell(\boldsymbol{x}; \Lambda[\tau])^{\mathrm{T}} \frac{\partial \mathcal{F}(\boldsymbol{s}; \Theta)}{\partial \Theta} \end{aligned} \quad (6.20)$$

where ε_τ and ρ_τ are small positive numbers, representing the classifier learning rate and the feature extractor learning rate, respectively. The two positive matrices are usually set to the unit matrix or set to match the sensitivity of each parameters to the corresponding learning rate within each module. Since, this version of GPD considers the modularity of the system, it is referred to as Modular GPD (MGPD) or selective optimization.

MGPD permits a more flexible application of GPD in the DFE framework and enables to use different convergence speed for the classifier parameters and the feature extractor parameters which would permit more adaptability and interaction between the two sets of parameters. The relation between ε_τ and ρ_τ is crucial for accurate learning. The use of $\rho_\tau = f(\varepsilon_\tau)$ where $f(\cdot)$ is a monotonic mapping called the *modulating function* is advisable. This is equivalent to an adaptive learning speed between the feature extractor and the classifier. However, the use of higher order functions might also be appropriate. Few suggestions are:

- $\rho_\tau = \alpha p_n(\varepsilon_\tau)$ where $\alpha < 1$.
- $\rho_\tau = \alpha \log(p_n(\varepsilon_\tau) + 1)$.

where $p_n(\varepsilon_\tau) = \varepsilon_\tau q_n(\varepsilon_\tau)$ and q_n is a n-th order polynomial with $n = 0$ or $n \geq 2$.

A variant of this training approach is to train separately, classifier and feature extractor (separate optimization). Within this framework, one can train first feature extractor, using the initial classifier parameters models and then update the classifier or vice-versa. This method is similar to the "relaxation" method [Culioli, 1994], in which an objective function is optimized sequentially along each axis.

If we assume that the parameter set of the classifier and the parameter set of the feature extractor are hyperplane generating the overall recognizer parameter space, modular optimization and separate optimization can be viewed as using different direction for reaching the minima of the error surface. The separate optimization optimizes along the hyperplanes whereas modular optimization optimizes across the hyperplanes. The two methods (modular optimization and separate optimization) are merely equivalent in term of convergence but may not lead to the same minima

in the parameter space for a complex error surface. In particular, as pointed out in [Culioli, 1994], for a non differentiable objective function, the separate optimization can easily be caught in the intersection line between the two hyperplanes without reaching the true minima.

The modular optimization is convenient in practice for a straightforward optimization of the overall recognizer. It enables the achievement of a more flexible interaction between the feature extractor and the classifier as well as accelerates the overall training convergence. That is, in this framework, one can control the convergence speed and convergence nature for each of the module. However, the modulating function $f(\cdot)$ which sets the link between the two sets of parameters during learning should be carefully chosen.

Separate optimization is a convenient way to use DFE on already optimized classifiers in order to optimize the feature extractor and enable one to easily retrain (or adapt) pre-designed status of either the feature extractor or the classifier. Also, separate optimization provides a convenient way to adaptively optimize the feature extractor as data becomes available. Note that although modules are optimized separately, this does not contradict the DFE concept, since the two modules are still using the same criterion for optimization, which is MCE.

6.4.1 MGPD simulation

A DFE-based speech recognizer requires an accurate tuning of parameters for optimality. Again, in classical MCE/GPD-based optimization scheme, the decreasing learning rate ϵ_τ, which is shared by both the feature extractor and the classifier, is chosen as

$$\epsilon_\tau = \epsilon\left(1 - \frac{\tau}{\mathcal{N}}\right)$$

where $\mathcal{N}$ is total number of iterations and ε is the learning at the beginning of the training set. Needless to say, ϵ_τ determines the performance of the system, given a fixed number of iterations $\mathcal{N}$. Classical GPD (that is, use of the same learning rate for feature extractor and classifier) is the standard and straightforward method.

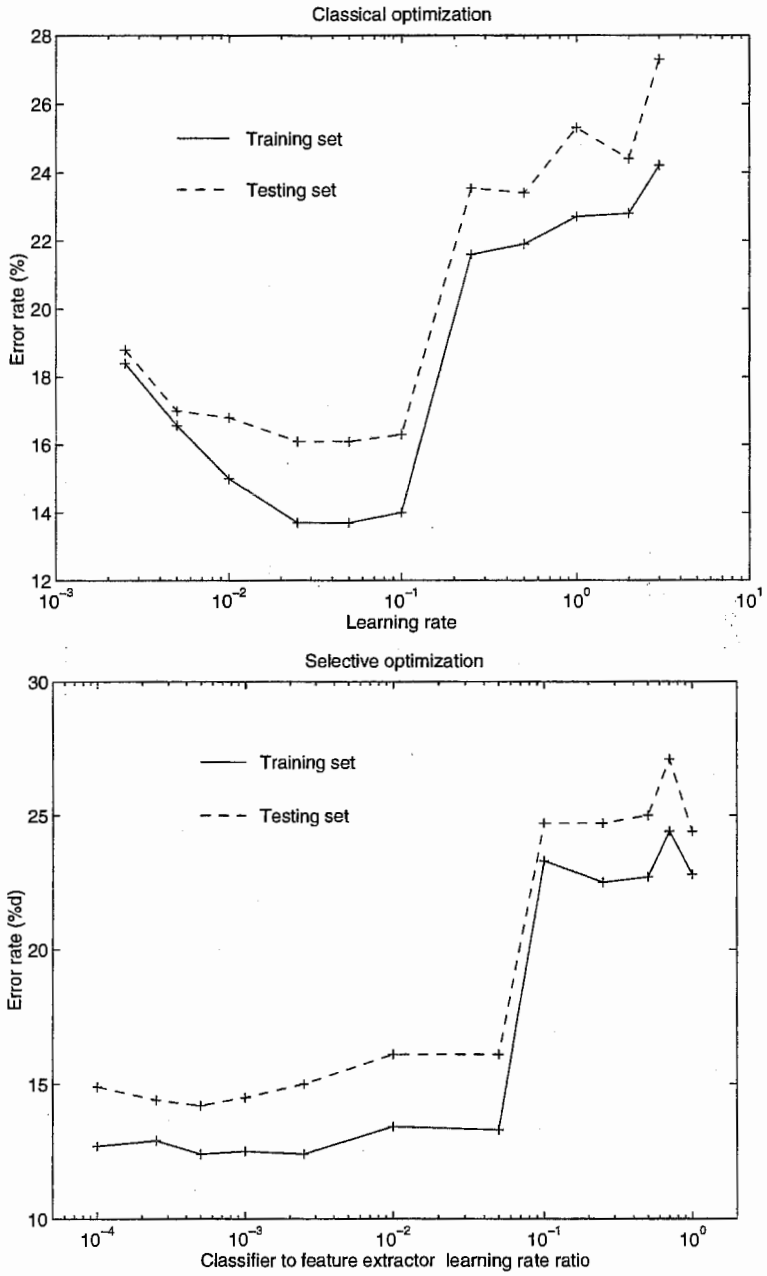


Figure 6.5: Classical GPD optimization scheme and MGPD optimization scheme for a fixed number of iterations. **Top**: The performance of the system versus the learning rate ϵ using GPD. **Bottom**: the performance of the system versus the ratio between classifier learning rate ε and feature extractor learning rate ρ in MGPD.

The way these two methods of optimization behave when adapting the center frequency, the bandwidth, and the gain of a filter bank-based feature extractor simultaneously, is shown in Figure 6.5. In the top of figure, the performance of the GPD is displayed as a function of the initial learning rate value (ϵ). The performance of the GPD is optimal for ϵ falling into the range of 0.1 to 0.01. On the bottom figure, performance of the MGPD is displayed. The figure displays the ratio $\frac{\varepsilon}{\rho}$ for the value of $\varepsilon = 1$. It can be seen that, the MGPD realizes better performance (i.e., lower error rate) in both the training set and the testing while showing a more stable curve across a wide range of the learning rate ratio. The solution consists of finding the accurate learning rate ratio between the feature extractor and the classifier. Moreover, the performance depends on the task as well as the optimized parameter. GPD can be viewed as special case of MGPD in which the classifier's and feature extractor's learning rate ratio is equal to 1.

The shortcoming of the MGPD approach derives from the fact that the interaction of the feature extraction and the classification is sometimes complex to analyze. Thus, a non neglecting number of experiments are required before one can find the optimal configuration of learning parameters ε_τ and ρ_τ. Furthermore, even the selected learning scheme does not guarantee optimality. The assurance that this minimum is a global one (or at least an acceptable one) depends on the shape of the error surface. That is, since the MCE error surface is not generally quadratic, there is no guarantee regarding the concave properties of the error surface, which could lead to local properties to be taken into account for speeding up convergence.

6.5 Incremental Generalized Probabilistic Descent Method

The Incremental Generalized Probabilistic Descent Method (IGPD) is here described as an alternative to MGPD. The approach is to iteratively adapt the training rate during the learning process. Let us suppose that the recognizer parameters Φ can be decomposed into a subset of parameters.i.e., $\Phi = \{\phi_1, \ldots, \phi_i, \ldots, \phi_I\}$. For sake of simplicity, let us assume that ϕ_i is a scalar. For instance, ϕ_i can be a component of a mean vector, or of a covariance matrix, within a state of model or a parameter of a feature extractor module (for example, the bandwidth or gain within a filter-bank channel). In the context of discriminative training, the purpose is to push or pull the models apart according to their contribution to the error. This is particularly the case when each category is represented by separate models or, in a case of the feature extractor, when each feature extractor parameter is contributing to a single feature-vector component.

Each parameter ϕ_i is assigned its learning rate ϵ_i, which controls its increment according to its contribution to the error. Since the target is minimum error, parameter-dependent learning rates extend the discrimination power to the space spanned by the learning rates. That is, each learning rate is updated by gradient descent at each iteration. Since the learning rate should stay positive, the adaptation is done through a transformation. Here, as in [Sutton, 1992], a logarithmic transformation is used. That is,

$$\epsilon_i = \exp(\nu_i), \tag{6.21}$$

which leads to the following adaptation rule for the learning rate:

$$\nu_i[\tau+1] = \nu_i[\tau] - \mu \frac{\partial \ell}{\partial \nu_i}, \tag{6.22}$$

μ is the meta-learning rate which controls the overall learning process. In (6.22), ℓ refers to the loss $\ell(\boldsymbol{s};\Phi)$ and is the notation used throughout this chapter. The meta-learning rate acts as the "learning rate" in the error surface spanned by the recognizer learning rates. The learning rate of each parameter is updated according to its contribution in reducing the error in the next step. Given, the updated learning rate, the recognizer parameters are adjusted by

$$\begin{aligned} \phi_i[\tau+1] &= \phi_i[\tau] - \exp(\nu_i[\tau+1]) \frac{\partial \ell}{\partial \phi_i} && (6.23) \\ &= \phi_i[\tau] - \epsilon_i[\tau+1] \frac{\partial \ell}{\partial \phi_i}. && (6.24) \end{aligned}$$

The key difference with GPD adjustment lies in the iterative optimization of the learning rates, which depend here on the current parameter update whereas in GPD, the learning rate depends on the current iteration, for a fixed positive definite matrix.

6.5.1 Learning rates update

The key to the optimization scheme is the calculus of $\frac{\partial \ell}{\partial \nu_i}$ for each recognizer parameter ϕ_i. The general chain-rule of calculus is

$$\begin{aligned} \frac{\partial \ell}{\partial \nu_i} &= \sum_k \frac{\partial \ell}{\partial \phi_k} \frac{\partial \phi_k}{\partial \nu_i} && (6.25) \\ &\approx \frac{\partial \ell}{\partial \phi_i} \frac{\partial \phi_i}{\partial \nu_i} && (6.26) \\ &\approx \frac{\partial \ell}{\partial \phi_i} m_i. && (6.27) \end{aligned}$$

The approximation of (6.26) assumes that each learning rate affects only its assigned parameter. This is not a very good approximation when parameters are shared among categories (Neural Network, for instance). However, (6.26) is a valid approximation when each category has its own models. (For instance, HMM-based speech recognizers).

According to (6.26), the update of the learning rates is proportional to the gradient of the loss and to the term m_i. $m_i = \frac{\partial \phi_i}{\partial \nu_i}$ is an additional term assigned to the adaptation of ϕ_i, which characterizes the variability of ϕ_i due the learning rate. Thus, this term memorizes the adjustment of the parameters ϕ_i. m_i is called *the adjustment memory* (AM). Its calculus is key to the overall learning process.

6.5.2 Adjustment memory update

The adjustment memory parameter (AM) is updated on an iteration basis based on the previously learned examples. The update rule is

$$m_i[\tau+1] = \frac{\partial \phi_i[\tau+1]}{\partial \nu_i} \tag{6.28}$$

$$= \frac{\partial \left(\phi_i[\tau] - \epsilon_i[\tau+1] \frac{\partial \ell}{\partial \phi_i} \right)}{\partial \nu_i} \tag{6.29}$$

$$= m_i[\tau] - \epsilon_i[\tau+1] \frac{\partial \ell}{\partial \phi_i} - \epsilon_i[\tau+1] \frac{\partial^2 \ell}{\partial \phi_i \partial \nu_i} . \tag{6.30}$$

The last term of (6.30) implies using second-order derivatives. Since, it is assumed that each learning rate influences only the adaptation rule of its assigned parameter,

$$\frac{\partial^2 \ell}{\partial \phi_i \partial \nu_i} = \sum_k \frac{\partial^2 \ell}{\partial \phi_i \partial \phi_k} \frac{\partial \phi_k}{\partial \nu_i} \tag{6.31}$$

$$\approx \frac{\partial^2 \ell}{\partial^2 \phi_i} \frac{\partial \phi_i}{\partial \nu_i} = \frac{\partial^2 \ell}{\partial^2 \phi_i} m_i. \tag{6.32}$$

Now, reconsidering the adaptation rule in (6.30) and using (6.32), we derive the adaptation of the memory parameter:

$$m_i[\tau+1] = m_i[\tau] \left(1 - \epsilon_i[\tau+1] \frac{\partial^2 \ell}{\partial^2 \phi_i} \right) - \epsilon_i[\tau+1] \frac{\partial \ell}{\partial \phi_i} . \tag{6.33}$$

Given a recognizer structure, the computation of the second derivative $\frac{\partial^2 \ell}{\partial^2 \phi_i}$ is computationally expensive. An approach could be to ignore the effect of the Hessian matrix altogether and update the AM terms using only the gradient of the loss versus the recognizer parameters. This approach leads to the following adjustment rule:

$$m_i[\tau+1] = m_i[\tau] - \epsilon_i[\tau+1] \frac{\partial \ell}{\partial \phi_i} . \tag{6.34}$$

which is the IGPD-A algorithm. In IGPD-A, the AM term is simply accumulated over the increments of the parameter categories.

Another approach is derived by considering that within the vicinity of a minimum, the Hessian is usually positive definite, thus the first term of the memory adaptation tends to decrease the memory values. This effect can be simulated by decaying linearly the first term of equation (6.33). In that case, the AM adjustment rule is

$$m_i[\tau+1] = m_i[\tau] \left(1 - \frac{1}{\mathcal{N}} \right) - \epsilon_i[\tau+1] \frac{\partial \ell}{\partial \phi_i} , \tag{6.35}$$

where $\mathcal{N}$ is the total number of iterations. This is IGPD-B algorithm.

6.5.3 Summary of the algorithm

Here, the IGPD algorithm is summarized. For generality, let $\boldsymbol{\phi}_i$ be a vector of parameters (e.g., a mean vector) and its learning rate ϵ_i (a scalar) and its AM parameter $\boldsymbol{m}_i$. $\boldsymbol{m}_i$ is a vector of the same size as $\boldsymbol{\phi}_i$. The gradient is

$$\frac{\partial \ell}{\partial \boldsymbol{\phi}_i} = \left[\frac{\partial \ell}{\partial \phi_{i,1}}, \ldots, \frac{\partial \ell}{\partial \phi_{i,l}}, \ldots, \frac{\partial \ell}{\partial \phi_{i,size(\boldsymbol{\phi}_i)}} \right]^{\mathrm{T}},$$

is also a vector of the same as $\boldsymbol{\phi}_i$.

The IGPD algorithm is

- Initialization:

  For each parameter $\boldsymbol{\phi}_i$, let $\boldsymbol{m}_i = 0$ and ν_i randomly initialized. $\mathcal{N}$ is the total number of iterations.

- For each input data $\boldsymbol{s}$, do

$$\nu_i[\tau+1] = \nu_i[\tau] - \mu \frac{\partial \ell}{\partial \boldsymbol{\phi}_i}^{\mathrm{T}} \boldsymbol{m}_i[\tau]$$

$$\epsilon_i[\tau+1] = \exp(\nu_i[\tau+1])$$

$$\boldsymbol{\phi}_i[\tau+1] = \boldsymbol{\phi}_i[\tau] - \epsilon_i[\tau+1] \frac{\partial \ell}{\partial \boldsymbol{\phi}_i}$$

$$\boldsymbol{m}_i[\tau+1] = \begin{cases} \boldsymbol{m}_i[\tau] - \epsilon_i[\tau+1] \dfrac{\partial \ell}{\partial \boldsymbol{\phi}_i} & \text{If IGPD-A} \\ \boldsymbol{m}_i[\tau]\left(1 - \frac{1}{\mathcal{N}}\right) - \epsilon_i[\tau+1] \dfrac{\partial \ell}{\partial \boldsymbol{\phi}_i} & \text{if IGPD-B} \end{cases}$$

endfor.

The relation between IGPD and GPD is the about the choice of the positive definite matrix. In IGPD, the positive definite matrix is simply the diagonal matrix with elements

$$(\epsilon_1[\tau+1], \ldots, \epsilon_i[\tau+1], \ldots,)$$

where it size equal to the number of parameters in the system. Thus, the IGPD provides a way to select the positive definite matrix iteratively based on local properties of the error surface. The designer is removed from the burden of learning rate tuning.

6.5.4 IGPD simulation on classifier training

Here, the IGPD algorithm is tested on optimizing the classifier while keeping the feature extraction fixed.

Shared learning rate

The learning rate is shared across all parameters of the classifiers. Fig. (6.6) illustrates the performance on the vowels recognition task of each learning paradigm when using a global learning rate: GPD, IGPD-A, IGPD-B, respectively.

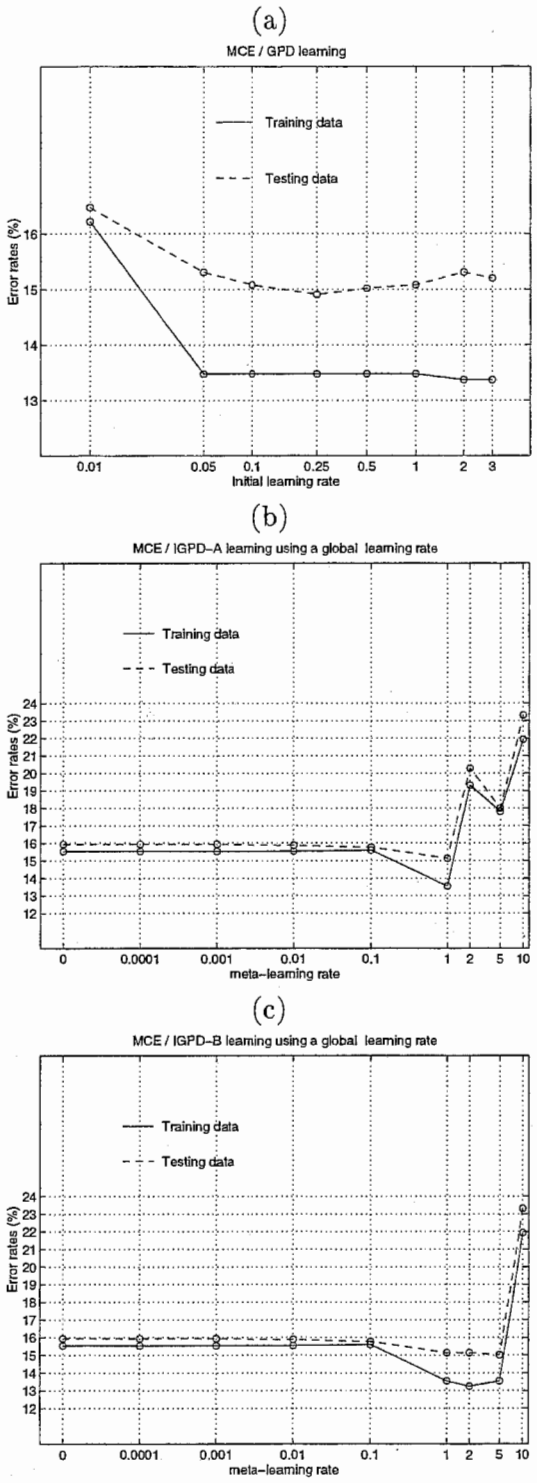


Figure 6.6: Performances versus learning parameters in the classical MCE-based recognition task using the same learning rate for all parameters of the classifier. Figure (a) illustrates GPD performance versus the initial learning rate. Figure (b) illustrates IGPD-A performance versus the meta-learning rate. Figure (c) illustrates IGPD-B performance versus the meta-learning rate.

GPD algorithm achieves a rather good and stable performance in the training set, even though the performance fluctuates on the testing set. The training curve displays a concave form with a rather wide gap between the performance on the training set and the performance on the testing set. This illustrates the sensitivity of DFE to over learning, since the gap between the two curves get wider as the learning rates increases.

The gap between the two curves is drastically reduced when running the IGPD algorithm. In particular, one can notice that the two training curves and the testing curve fluctuate in a synchronous manner, with a large plateau across different values of the meta-learning rate. IGPD-B algorithm show much more stability across meta-learning rate values.

Comparing the 3 algorithms in the context of MCE on classifier design, it seems that the IGPD algorithm, and in particular IGPD-B displays a more constant curve across meta-learning rate values, thus relieving the designer of the burden of learning rate tuning. However, the 3 algorithms shows rather close results on the testing set (around 15 %) even though GPD outperforms IGPD on the training set for various values of the learning rates (a large plateau around 13.5 % error rates).

Class-dependent learning rate

Each category is now assigned a private learning rate which is updated by the IGPD algorithms. For clarity, the various IGPD versions are referred to as IGPD-A-S and IGPD-B-S, for IGPD-A and IGPD-B, in this context.

Fig. 6.7 illustrates the results on the task for the two proposed algorithms. IGPD-A-S seems much more stable than IGPD-B-S although IGPD-B-S shows the best performance in the testing set (around 15.5%) for a particular value of the meta-learning rate. Unlike the GPD case, it seems that the performance using IGPD algorithms depends less on the tuning of the meta-learning rate.

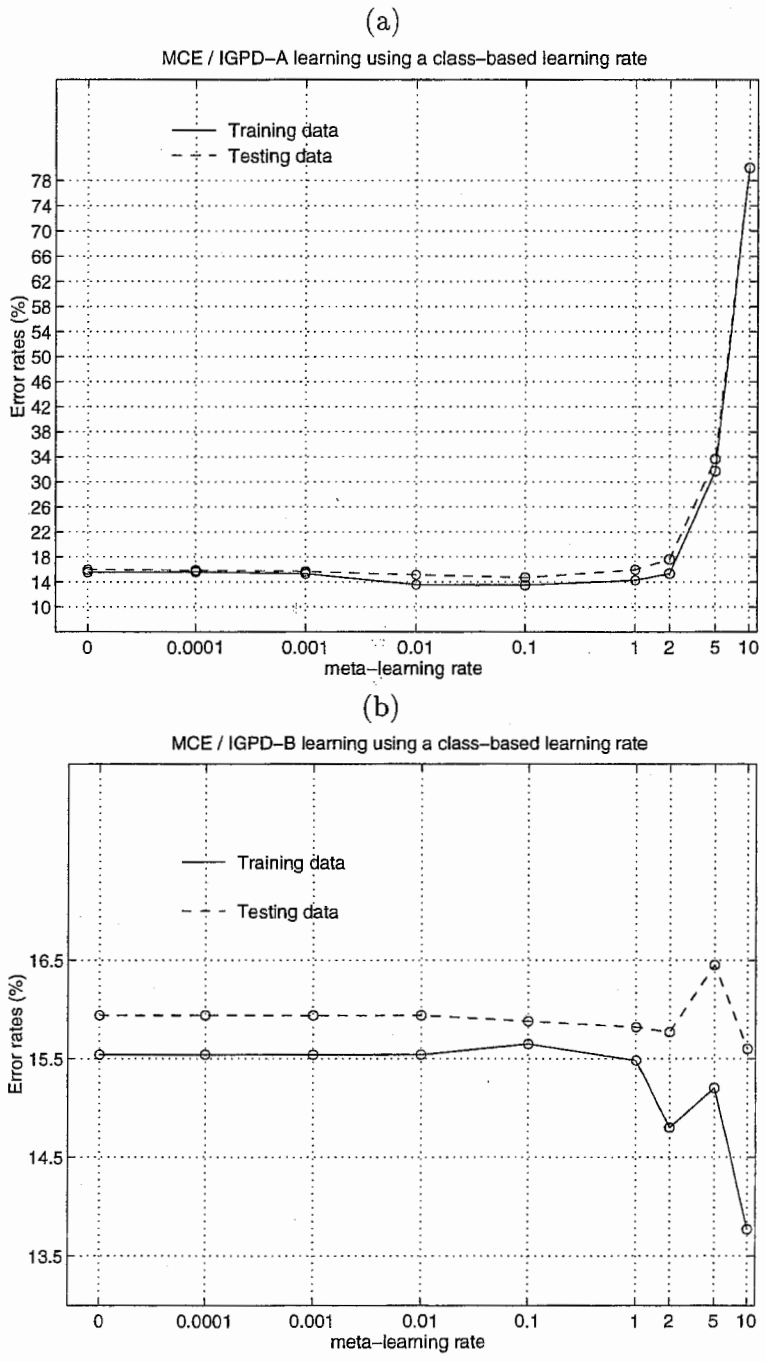


Figure 6.7: Performances versus learning parameters in the MCE-based recognition task using a class-dependent learning rate. Figure (a): IGPD-A performance versus the meta-learning rate. Figure (b): IGPD-B performance versus the meta-learning rate.

Asymptotic behavior

For given learning parameters, performance versus the number of iteration gives an idea about the convergence rate of each algorithm and the asymptotic behavior, when increasing the number of iterations.

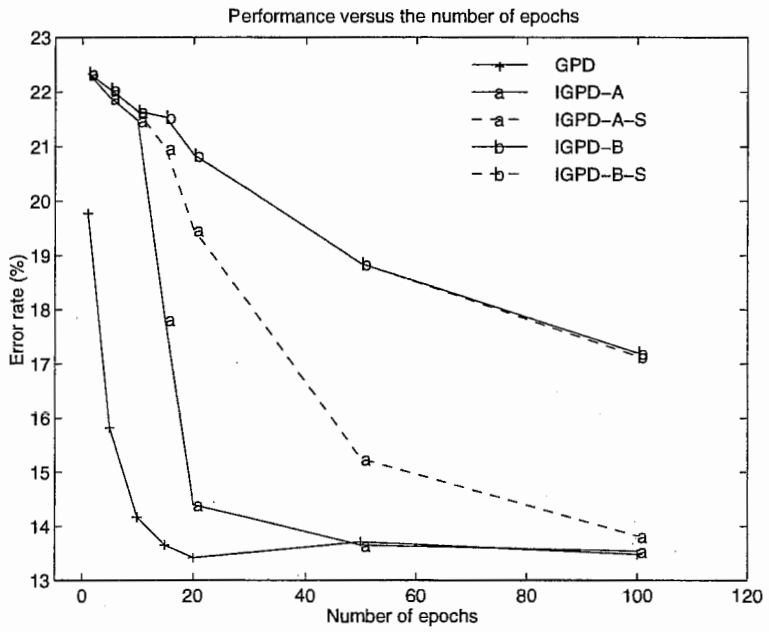


Figure 6.8: Performance versus the number of epochs for GPD and IGPD in classifier optimization. IGPD uses a class-dependent learning rate.

GPD error rate decrease is the fastest for the first iterations, followed by IGPD-A. The slow decrease in error rate of the IGPD algorithm may be due to the time spent during the first iterations in searching the right learning rate.

6.5.5 IGPD simulation on DFE-based recognizer

The application of IGPD to DFE-based recognizer is here described. In DFE, feature extractor parameters and classifier parameters are updated simultaneously for minimum error. Due to the complexity of the overall system, tuning of the learning rate can be time-consuming. Furthermore, as emphasized previously, a straightforward application of GPD does not always converge. IGPD should be useful in finding iteratively the learning parameters which permits the overall system learning. Here, the use of the IGPD algorithm on DFE-based recognizer is illustrated. The target is to adjust the center frequencies of a filter bank under a cepstral transformation. The center frequency of each channel is adjusted simultaneously with the classifier parameters. Each channel is assigned a private learning rate. That is, 16 learning rates which correspond to the 16 channels. The number of epoch is fixed at 300 and performance are studied by varying the learning parameters.

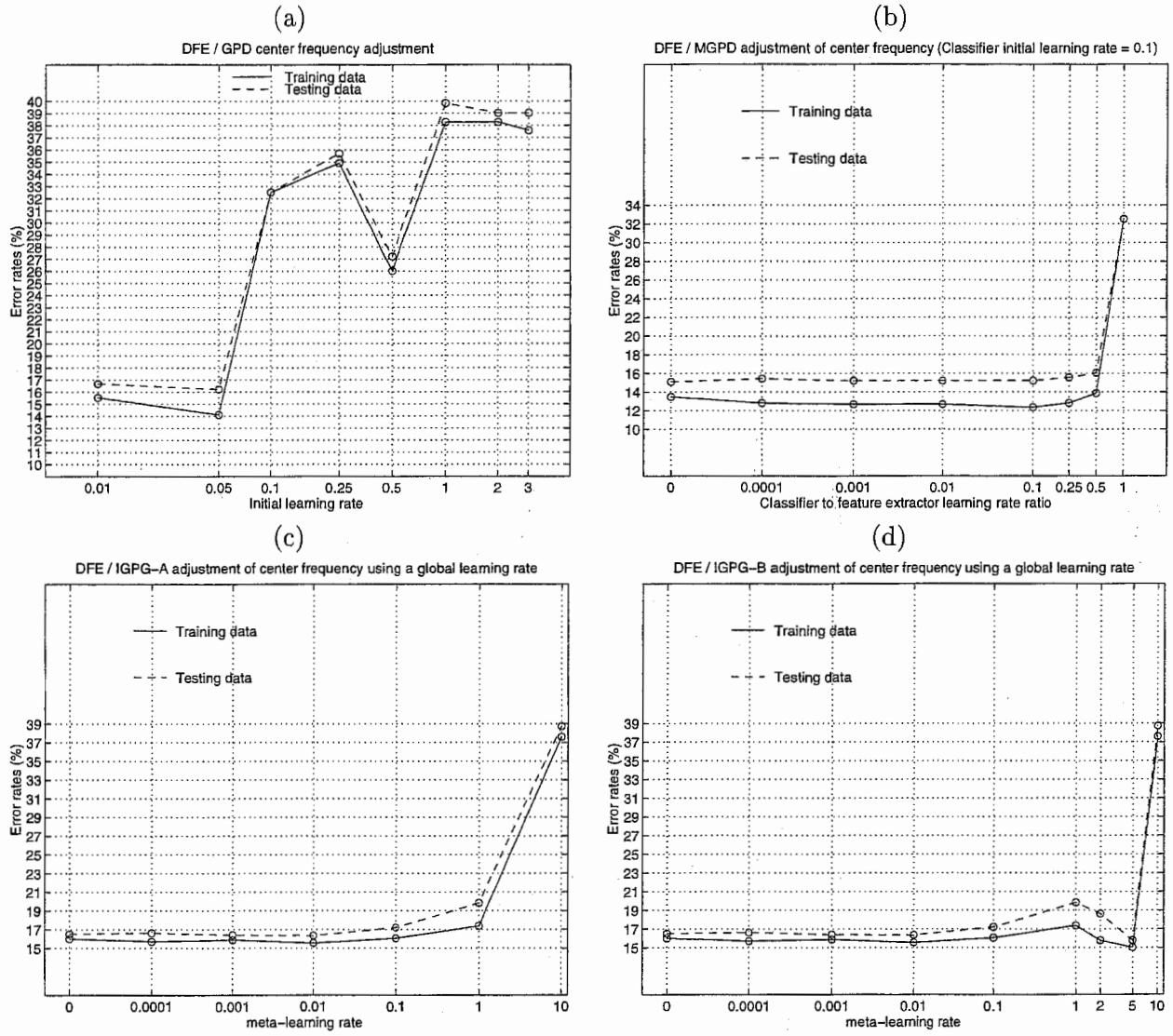


Figure 6.9: Performances versus learning parameters in the DFE-based center adjustment using a shared learning rate in the classifier. Figure (a) illustrates performance of GPD application. Figure (b) illustrates performance of MGPD. Figure (c) illustrates IGPD-A performance versus the meta-learning rate. Figure (d) illustrates IGPD-B performance versus the meta-learning rate.

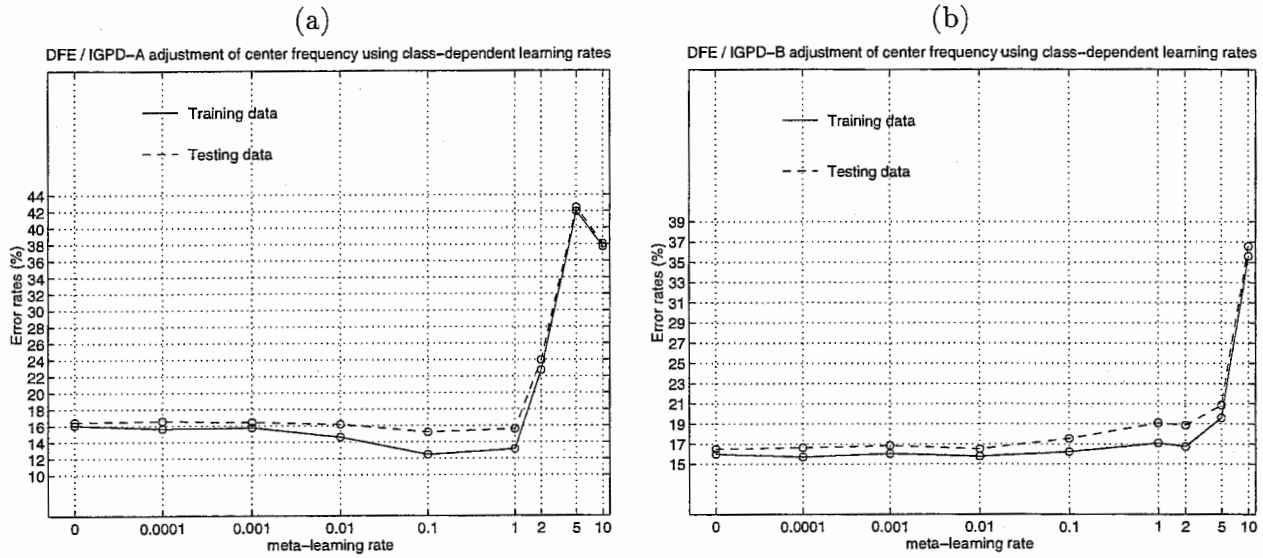


Figure 6.10: Performances versus learning parameter in the DFE-based vowel recognition task using a class-dependent learning rate. Figure (a) illustrates IGPD-A performance versus the meta-learning rate. Figure (b) illustrates IGPD-B performance versus the meta-learning rate.

Shared learning rate

Fig. 6.9 shows the results when adjusting the center frequencies using the 4 learning methodologies: GPD, MGPD, IGPD-A and IGPD-B. Here, the classifier parameter share the same learning rate.

The GPD curve shows many fluctuations making it harder to empirically find the right learning rates. MGPD and IGPD provides a more stable configuration. In particular, MGPD and IGPD shows rather similar results in the testing. However, it is clear that MGPD is extremely sensible to the learning configuration. For instance, a learning rate ratio of 1.0, which is equivalent to classical GPD, shows poor results.

GPD's best results in the testing is slightly higher than 16%, while MGPD shows around 15.5% across a wide range of filter learning ratios, similar to IGPD results. The best results is given IGPD-A for the meta-learning of 5 (15%).

Class-dependent learning rate

A private learning rate is assigned to each class and to the center frequency of each filter channel. Fig. 6.10 shows the results.

The two curves show a rather constant performance across meta-learning rate, although IGPD-A gives better results. The results seems higher than the one obtained when using a global learning rate.

Asymptotic behavior

Similar to the classical MCE study in the context of classifier optimization, we intend to grasp the behavior of the four algorithms, when increasing the number of epochs for fixed learning parameters. Performance versus the number of epochs provides an estimate of this asymptotic behavior. Here, the IGPD uses a class-dependent learning rate. Fig. 6.11 illustrates performance of GPD, MGPD,

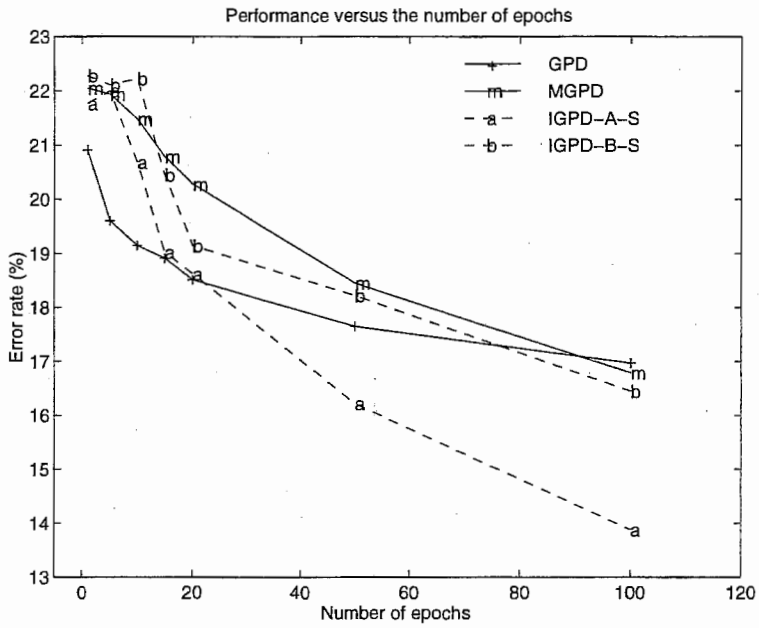


Figure 6.11: Performance versus the number of epochs for GPD, MGPD and IGPD in overall recognizer optimization. IGPD uses a class-dependent learning rate.

and IGPD versus the number of epochs in the DFE context. It is clear that IGPD-A converges faster in this case. Ultimately, as we have seen previously, the three algorithms give rather similar results for a higher number of epochs.

However, it is quite obvious that IGPD-A provides a faster convergence in the DFE framework. This maybe due to its ability to optimize the learning parameter, given a complex structure in a more efficient manner than classical GPD.

6.6 Discussion

In average, the IGPD and MPGPD performs better than the classical use of the GPD algorithm. Contrasting MGPD and IGPD is more difficult, given the above framework. The best algorithm, obviously depends on the classifier structure as well as the task.

6.7 Conclusion

In some practical application within the DFE approach, a straightforward use of the GPD algorithm may result in poor convergence due to the non-homogeneous nature of the DFE-based recognizer. This problem may be solved by appropriate choice of the positive definite matrix and learning rate. However, such a choice is usually time-consuming. In this chapter, it was proposed to consider the modular nature of the recognizer in the learning process by choosing a training parameters (learning rate and positive definite matrix) according to the module. This approach termed Modular Generalized Probabilistic Descent, although more adapted to the nature of the recognizer, added more learning parameters. The incremental Generalized Probabilistic Descent (IGPD), which performs a gradient descent on the space spanned by the learning parameters, was described as a method to overcome this deficiency.

Part III

Applications

Chapter 7

Discriminative Feature Extraction Applied to Lifter Design

To find out what is natural,
we must study specimen which retain their nature
and not those which have been corrupted

-Aristotle-

A cepstrum vector, which is computed by applying the inverse DFT to a logarithmic power spectrum of a speech fragment, has been shown to have information useful for classifying phonemes, particularly in its low-quefrency components. Cepstrum Liftering, which is equivalent to applying a low-pass lifter on the cepstrum is the tool for enhancing and/or removing undesirable cepstral component. Although the liftering process has been studied extensively, standard techniques are limited due to various mixed sources of information within the cepstrum coefficients. This chapter presents an application of the Discriminative Feature Extraction to lifter design that may help overcome some of these limitations.

7.1 Introduction

A standard model of the speech production system assumed that speech is generated through a vocal tract filter excited by a periodic excitation or noise (vocal cords). Based on this model, various attempts have been made to set apart the contribution of the vocal tract and the contribution of the source within the speech signal [Ohyama *et al.*, 1981a]. In particular, for voiced sound such as vowels, the sound category is mainly determined by the shape of the vocal, if the effects of the lips are neglected. Cepstrum coefficients, processed from the inverse Fourier transform of the logarithm of the spectrum carries the effect of the vocal tract and vocal cords in an additive manner.

Several investigations have been carried out to extract the discriminant information by the liftering process. Liftering, which is filtering in the quefrency domain (cepstral domain), has thus been studied extensively. Research has been done on accurate pitch estimation [Noll, 1964], or on a optimal lifter shape design, using either LPC-based spectrum [Juang *et al.*, 1986; Tohkura, 1987], or Fourier transform-based spectral representation within a statistical estimation framework [Ohyama *et al.*, 1981a]. Nevertheless, these techniques are limited due to various mixed sources of information over cepstrum coefficients; even lower quefrency terms, considered indicative of phoneme identity, possess various components characterizing the vocal-tract behavior which leads to a high statistical variation. Difficulties in separating these components for general purposes may be overcome through a task-oriented framework.

In place of the conventional setting, based on heuristics or experimental knowledge as shown above, a full-length lifter over a given cepstrum vector was prepared without explicit a priori assumptions, which enables any initial configuration [Biem and Katagiri, 1992a; Biem and Katagiri, 1992b]. This initial lifter is then adjusted by the DFE training, finally producing an optimal one, in the sense of minimizing the classification errors of the back-end classifier attached to the lifter . Experimental results on a vowel recognition task are presented, applying the proposed idea to the design a lifter shape over cepstrum coefficients using a Neural Network classifier [Biem and Katagiri, 1993b]. Comparison with standard approaches that include investigation in the time/frequency domain show how speech characteristics are extracted for recognition purposes.

7.2 Liftering-based Speech Recognition

7.2.1 Speech production model

A standard speech production model considers that the speech $s(t)$ is the output of the vocal tract impulse response $v(t)$, convolved with the source $u(t)$ [Fant, 1973]. That is,

$$s(t) = v(t) * u(t) \tag{7.1}$$

$$= \int_{-\infty}^{+\infty} v(t+\tau)u(\tau)d\tau. \tag{7.2}$$

This is the basic assumption of the model. Using homomorphic processing, it is possible to filter out the vocal tract contribution from the speech signal. For digital speech processing, the signal

$s(t)$ is represented as sequences of frame $s_1^T = \{\mathbf{s}_1, ..., \mathbf{s}_T\}$, where $\mathbf{s}_t = [s_{t,1}, ..., s_{t,d}]^{\mathrm{T}}$ is the frame at time t and the corresponding cepstrum vector is $\mathbf{c}_t = [c_t(0), ..., c_t(n), ..., c_t(L)]^{\mathrm{T}}$, where n refers to the quefrency.

Let $S(\omega)$ be the Fourier transform of the speech signal $s(t)$, the cepstral parameter $c(n)$ is defined as the inverse Fourier transform of the log spectrum. That is,

$$c(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \log S(\omega) \exp(j\omega n) d\omega. \tag{7.3}$$

n spans a domain called the quefrency domain. This term was apparently proposed by J. R Tukey to reflect the fact that inverse Fourier of the log spectrum does not take us back to the time domain but unto a new domain called "quefrency" [Parsons, 1987]. He also proposed the term "cepstrum" out of "spectrum" and "liftering", out of "filtering", for describing quefrency domain operations.

By homomorphic processing, taking the logarithm of the power spectrum gives

$$\log S(\omega) = \log U(\omega) + \log V(\omega),$$

where $U(\omega)$ and $V(\omega)$ are the Fourier transform of the excitation source and the vocal tract impulse response, respectively. It can be seen that the effect of the vocal tract and the vocal cord are additive within the log magnitude spectrum. Let $v(n)$ and $u(n)$ be the inverse Fourier transform of $\log V(\omega)$ and $\log U(\omega)$, respectively. The cepstrum coefficient $c(n)$ of the power spectrum is given by

$$c(n) = v(n) + u(n). \tag{7.4}$$

From the above equation, it is possible to remove out the source contribution $u(n)$ by appropriate processing. For cepstral analysis, the following two assumptions are made:

1. $u(t)$ is approximately a sequence of pulses. The distance T between two pulses is called the pitch period and determine the underlying pitch of the signal. Thus, $V(\omega)$ is basically a line spectrum, with corresponding cepstra $v(n)$, and is also a sequence of pulses spaced by TF_s, where F_s is the working sampling rate of the signal [Flanagan, 1972].

2. The vocal tract is a causal filter.

Speech production is a highly complex process, involving various articulators, whose global influence on the output speech is still under investigation. The two assumptions above are a gross attempt to simplify the study of such a complex problem. The first assumption is questionable for unvoiced speech such as fricatives and the second assumption neglects the effect of the nasal tract. Despite this simplified scheme, the above model has been proven to be quite useful in various speech processing system such as speech coding, speaker recognition and speech recognition.

Thus, based on the above model, lower quefrency terms of a cepstral representation contain vocal tract information and the spectral representation become smoother without the fine spectral structure due the pitch harmonics. Consequently, cepstral analysis has been used for various tasks, such as estimating the spectrum envelope of the speech spectrum, extracting the vocal tract transfer function, which is extremely useful, when the power spectrum is estimated by DFT techniques.

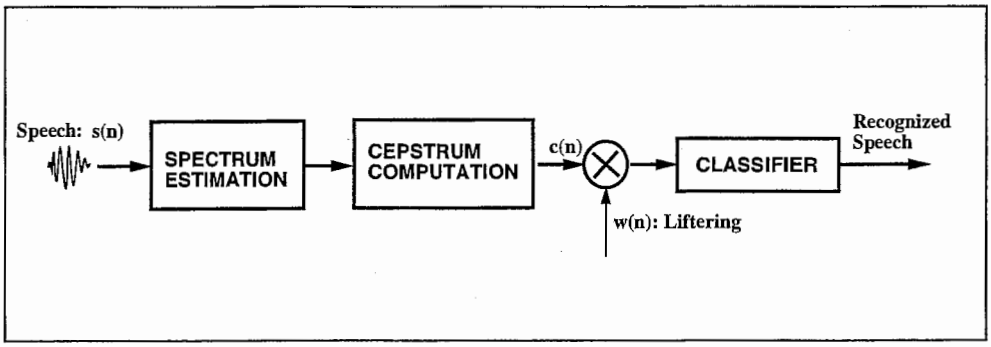


Figure 7.1: The liftering process. Liftering can be viewed as multiplying a cepstral coefficient by a weight

That is, the raw speech spectrum, produced through FFT techniques, is highly correlated and therefore contains redundant information which are useless and even inhibits efficient discrimination [Juang *et al.*, 1986]. Cepstrum-based speech processing thus permits the design of a smooth spectrum removing the component of the excitation source through liftering.

7.2.2 Cepstrum liftering application to speech

As said before, a cepstrum vector, which is computed by applying the inverse DFT to a logarithmic power spectrum of a speech fragment, has been shown to have information useful for classifying phonemes, particularly in its low-quefrency components. The lifter shape is therefore expected to be a low-pass one, designed to keep low quefrency values. However, since the low quefrency terms are usually highly variable due to speaker characteristics and vocal efforts, which may inhibit their discriminating capabilities, the liftering process is also expected to reduce the variance for better recognition.

Methods of designing a lifter, have been extensively investigated (e.g., [Ohyama *et al.*, 1981a; Tohkura, 1987]). Most approaches are based on the a priori speech production model, as described above or on the statistics of the data. The basic idea is illustrated in 7.1 and is as follows. Since the vocal tract is modeled as a causal filter with an finite impulse response, the cepstral coefficient are located in the lower quefrency domain. Consequently, highest quefrency values of $c(n)$ mainly correspond to $u(n)$. The vocal tract information can thus be extracted by a simple rectangular lifter, which performs

$$\tilde{c}(n) = w(n)c(n),$$

where

$$w(n) = \begin{cases} 1 & \text{if } n < n_c \\ 0 & \text{otherwise.} \end{cases} \tag{7.5}$$

n_c is the cut-off quefrency of the lifter and cepstral terms above the cut-off quefrency are considered useless for discrimination. The rectangular lifter is the baseline liftering process, far from being

satisfactory in certain contexts due to the required estimation of the value n_c. Moreover, not all cepstral values below n_c have discriminant information: they carry various information such as speaker identity, the vocal efforts and transmission characteristics. Thus, having equal weighting on those coefficients is not an optimal solution.

Triangular lifters

Most of the research in the 80s, done on liftering process, was made within the framework of finding a suitable cepstral distance measure [Paliwal, 1982; Tohkura, 1987; Hanson and Wakita, 1986; Juang *et al.*, 1986]. The focus was on estimating $w(n)$ such that the cepstral distance between two patterns $\mathbf{c}_t$ and $\mathbf{c}_r$, defined as

$$d(\mathbf{c}_t, \mathbf{c}_r) = \sum_{n=1}^{L} (w(n)c_t(n) - w(n)c_r(n))^2 \tag{7.6}$$

$$= \sum_{n=1}^{L} w(n)^2 (c_t(n) - w(n)c_r(n))^2, \tag{7.7}$$

yields better performance. Note that, in this framework, the liftering is embedded in the distance measure. One proposal [Paliwal, 1982; Tohkura, 1987] is to use an asymmetric triangular lifter of the form

$$w(n) = n, \text{ for } n < n_c.$$

This lifter was designed through two motivations: a statistical one and a physical one. The statistical motivation relies on the fact that cepstral coefficients (except for $c(0)$) has zero means and a standard deviation inversely proportional to the square of the coefficient index n [Juang *et al.*, 1986]. i.e.,

$$E\{c^2(n)\} \sim \frac{1}{n^2}.$$

Consequently, a variance normalization is obtained by choosing $w(n)$ to be the inverse of $\frac{1}{n}$. That is, $w(n) = n$. By so doing, the cepstral distance becomes,

$$d_w(c_t, c_r) = \sum_{n} (nc_t(n) - nc_r(n))^2 \tag{7.8}$$

$$= \sum_{n} n^2 (c_t(n) - c_r(n))^2. \tag{7.9}$$

This distance puts emphasis on higher quefrency terms. A variant is to use a specific cut-off quefrency for achieving better robustness. The physical meaning of (7.9) is as follows. Remember that

$$\log |S(\omega)| = \sum_{n=-\infty}^{\infty} c(n) \exp(-jn\omega). \tag{7.10}$$

Thus, taking the derivative of the log spectrum, with respect to the frequency ω, gives

$$\frac{\partial \log |S(\omega)|}{\partial \omega} = \sum_{n=-\infty}^{\infty} -jnc(n) \exp(-jn\omega) \tag{7.11}$$

and by using the Parseval theorem, we have

$$d_w(\mathbf{c}_t, \mathbf{c}_r) = \sum_{n}^{L}(n^2(c_t(n) - c_r(n))^2 \tag{7.12}$$

$$= \int_{-\pi}^{\pi} \left| \frac{\partial \log |S_t(\omega)|}{\partial \omega} - \frac{\partial \log |S_r(\omega)|}{\partial \omega} \right|^2, \tag{7.13}$$

where $S_t(\omega)$ and $S_r(\omega)$ correspond to the power spectrum of c_t and c_r, respectively. $d_w(\mathbf{c}_t, \mathbf{c}_r)$ corresponds to an L_2-norm between the differential log magnitude spectrum (spectral slopes). Since any fixed spectral slope is constant within the differential log magnitude, spectral peaks, such as formants are well preserved by this distance measure. When the power spectrum $S(\omega)$ is computed from an LPC model i.e., $S(z) = \frac{1}{A(z)}$, the distance $d_w(c_t, c_r)$ is referred to as the "root power sum" distance, because $nc(n)$ is equal to the sum of the n-th power of the root of $A(z)$ [Rabiner and Juang, 1993].

The triangular lifter has been used in various speech recognition tasks [Paliwal, 1982; Hanson and Wakita, 1986; Tohkura, 1987]. Paliwal [Paliwal, 1982] reported an increase in recognition rate from 91.4% to 92.7% on a vowel recognition. However, despite its powerful significance, the triangular lifter essential shortcoming is that, unless one uses a cut-off quefrency value of n_c, it does no de-emphasize the higher frequency domain, which result in less robustness when the system is performing under adverse conditions.

A more effective way to use the variance of the cepstrum is simply to use as lifter the inverse of the diagonal coefficients of the covariance matrix. This approach suggested by Tohkura [Tohkura, 1987] as a way to simplify the straightforward use of the Mahalanobis distance has proven to be quite useful for speech recognition. The obvious shortcoming of this approach is the fact that one needs to estimate the diagonal covariance matrix, which is extremely prohibitive in large vocabulary recognition systems.

Raised-sine lifter

An interesting study made by Juang [Juang *et al.*, 1986] has established that higher quefrency terms are extremely dependent on the power spectrum model, the signal analysis method (window position, system interaction with the excitation) and lower quefrency bears speaker characteristic (glottal shape, vocal cord duty cycles, speaking effort) as well as channel transmission information, which are undesirable for recognition. Consequently, Juang [Juang *et al.*, 1986] has suggested the use of a bandpass lifter of raised-sine shape. That is

$$w(n) = \begin{cases} 1 + h\sin\left(\frac{n\pi}{L}\right) & \text{if } n < L \\ 0 & \text{otherwise,} \end{cases} \tag{7.14}$$

where h is usually equal to $\frac{L}{2}$. The raised sine-lifter simulates the effect of the triangular lifter for $n \leq \frac{L}{2}$ while smoothly truncating higher coefficients. This result in enhancing some spectral peaks while suppressing the effect of the spectral tilt. The shortcoming of the raised-sine lifter is

that tuning of both h and L for achieving reasonable accuracy is required, which can extremely be time-consuming for practical speech recognition task.

Exponential lifter

Another used lifter is an exponential lifter proposed by [Junqua and Wakita, 1993], which is of the form

$$w(n) = n^g,$$

and tries to establish a continuum of choice between a no-liftering situation ($g = 0$) and the triangular lifter ($g = 1$).

It can be noted that most lifters, presented above are sub-optimal due to 1) no guarantee of optimality regarding the liftering parameter choice 2) no guarantee of minimum error achievement.

7.3 DFE-based Lifter Design

The liftering procedure provides a good framework for testing the Discriminative Feature Extraction process in terms of performance and analysis of the resulting lifter [Biem *et al.*, 1997]. Since liftering is a simple weighting process, it can be viewed as linear transformation of the cepstral coefficients. The DFE idea consists of representing the liftering process, within the recognizer structure and optimizing the overall recognizer using the MCE/GPD criterion. Any recognizer structure can be used for this purpose. Below, a Neural Network-based implementation is described.

DFE is applied to designing a recognizer having a liftering-based feature extraction module in a five-class, Japanese vowel recognition task. Each input (to the feature extractor) pattern is represented as a fixed-dimensional cepstrum vector, which corresponds to a center fragment of a vowel sound.

7.3.1 Recognizer structure

Again, liftering is done by multiplying an input cepstral value by a weight value. Taking this into account, the recognizer illustrated in Fig. 7.2 was used.

The recognizer consists of a liftering feature extractor and a three-layer (one hidden layer) perceptron neural network classifier. The lifter is implemented with a set of weights, each associated with a straight connection at one of the 128 (0th to 127th) quefrency positions. The classifier is a usual, fully-connected multi-layer perceptron network. The input layer of this classifier has 128 nodes, corresponding to the lifter structure. The top output layer has nodes in accordance with the number of classes. The output functions, one at each node, are nonlinear (sigmoidal) only at the classifier hidden layer, while being linear at the other layers. A cepstrum vector pattern, i.e., a system input (corresponding to $\mathbf{c}$), is first liftered by an inner-product computation with the lifter weight vector at the feature extraction module, and then the resulting weighted pattern

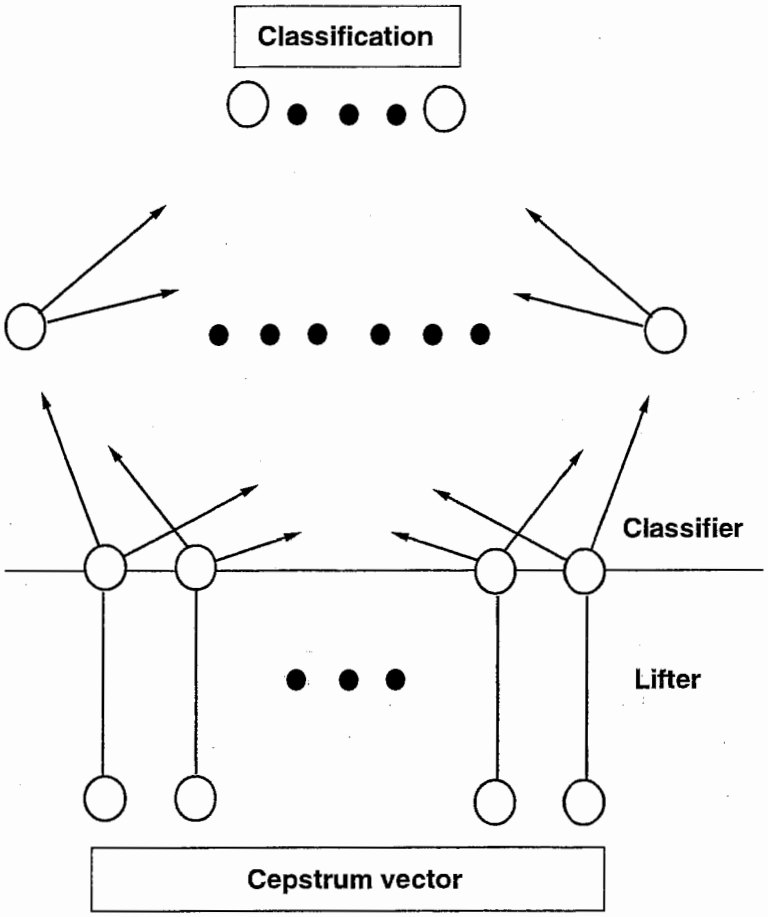


Figure 7.2: Recognizer structure consisting of a three-layer perceptron classifier with a liftering module.

(corresponding to $\boldsymbol{x}$) is classified based on the standard network computation at the classification module. The node of the top classifier layer outputs the discriminant function value.

Apparently, the recognizer seems to be a standard four-layer network. However, as its lowest straight connection is purposely selected for implementing liftering, the recognizer is obviously a modular system having differences in nature between the front- and post-end modules.

This modularity assumption, which is a fundamental concept of the DFE definition, is used in the system initialization for training.

7.3.2 Training process

Let $o_i(\mathbf{c})$ be the output corresponding to the class C_i due to the cepstrum vector $\mathbf{c}$ (belonging to category C_k), the misclassification measure is defined as:

$$d_k(\mathbf{c};\Phi) = -o_k(\mathbf{c};\Phi) + \left[\frac{1}{M-1}\sum_{j\neq i}^{M} o_j(\mathbf{c};\Phi)^{\eta}\right]^{\frac{1}{\eta}} \tag{7.15}$$

$$= -o_k(\mathbf{c};\Phi) + o_{\overline{k}}(\mathbf{c};\Phi), \tag{7.16}$$

where η is a positive number controlling the degree of misclassification to be considered and M is the number of output nodes, which corresponds to the number of class. Φ represents the overall recognizer parameter. The above definition of the misclassification measure is aimed at taking into account both the recognized class as well as the misclassified classes. Again, a positive value of $d_k(\mathbf{c})$ corresponds to a misclassification and a negative value is a good recognition. So, through the choice of η and the error function $\mathbf{c} \rightarrow \ell(\mathbf{c};\Phi)$, one can control the loss over which the optimization process is made. The choice of $\eta = 3$ was done, meaning that of a "soft" MCE scheme was carried out.

The chosen smooth derivable function as error function is simply the sigmoid, defined as

$$\ell(d_k(\mathbf{c};\Phi)) = \frac{1}{1 + e^{\alpha d_k(\mathbf{c};\Phi)}} \tag{7.17}$$

which means approximating the 0-1 step function by a sigmoid whose smoothness is controlled by the parameter α. Training is done through usual back-propagation of the error. Thus, if w is a weight within the network, we have,

$$w[\tau + 1] = w[\tau] - \epsilon_\tau \ell'(d_k(\mathbf{c};\Phi)) \frac{\partial d_k(\mathbf{c})}{\partial w} . \tag{7.18}$$

By making use of the chain rule, we have

$$\frac{\partial d_k(\mathbf{c};\Phi)}{\partial w} = \sum_{j=1}^{M} \frac{\partial d_k(\mathbf{c};\Phi)}{\partial o_j(\mathbf{c};\Phi)} \frac{\partial o_j(\mathbf{c};\Phi)}{\partial w} \tag{7.19}$$

with

$$\frac{\partial d_k(\mathbf{c};\Phi)}{\partial o_j(\mathbf{c};\Phi)} = \begin{cases} -1 & \text{for } j = k \\ \dfrac{1}{M-1} \left(\dfrac{o_j(\mathbf{c};\Phi)}{o_{\overline{k}}(\mathbf{c};\Phi)} \right)^{\eta-1} & \text{for } j \neq k. \end{cases} \tag{7.20}$$

The computation of $\dfrac{\partial o_j(\mathbf{c};\Phi)}{\partial w}$ is carried by the back-propagation algorithm (see appendix B).

7.4 Speech Data

The input patterns were generated by segmenting vowel center fragments, with a 21 msec Hamming time window, from 500 phonetically-balanced sentences uttered by 5 speakers (3 males and 2 females, 100 sentences/speaker) under noise-free conditions and using 256-point FFT. Speech data was digitized at a 12 kHz sampling frequency and stored at 16 bits. The total number of generated input samples was 3500; half of them (70 samples per speaker and per vowel) were used for design, the other half (70 samples per speaker and per vowel) for testing.

7.5 Results and Discussion

Considering the nature of the GPD's adaption, several pairs of training and testing for each system selection, such as the selection of the classifier hidden nodes and the selection of training initialization, was ran, changing the training conditions, such as the setting of the learning factor ϵ and the order of design sample presentation. Since the variation among these conditions was minor, only the best accuracies (lowest recognition error rate) for each system selection are presented in this chapter.

It is advisable that, similar to general cases of gradient-based optimization algorithms, the DFE-trained, or GPD-trained, recognizer is initialized in some reasonable way. However, it is rather unclear how one can reasonably initialize the back-end network classifier. Therefore, the neural network classifier was initialized randomly as is usually done in neural network applications. As for the feature extractor, the following three initialization methods were investigated:

1. A random initialization that sets the lifter weights to small random values. In this framework, the modularity process is not considered, except in the node connections. Thus, the weights in the lifter are treated in manner similar to other weights without specificity.

2. A uniform initialization that sets the lifter weights to the identical value 1.0. This is equivalent to a non-liftering situation before training. The DFE algorithm is given a "free" starting point in finding the appropriate lifter.

3. A rectangular initialization that sets the low (quefrency)-pass rectangular lifter with some preset duration. This is equivalent to the use of a priori knowledge in choosing an initial configuration of the feature extractor. The DFE algorithm is expected to find the "best" lifter, from this initial configuration.

For comparison purposes, baseline system in which the feature extractor module was fixed and only the classifier module was trained with MCE/GPD was investigated. The baseline system uses a priori knowledge. In the baseline system, the lifter was fixed to the rectangular shape, as described in the third initialization case above. Because of the lack of exact information on the advisable length of the low-pass lifter, which would lead to accurate vowel classification, four lifter duration settings were examined, namely 8, 16, 32, and 128 as seen in Fig. 7.3. Note that the rectangular lifter with the duration of 128 works as an all-pass lifter, corresponding to the situation of no liftering.

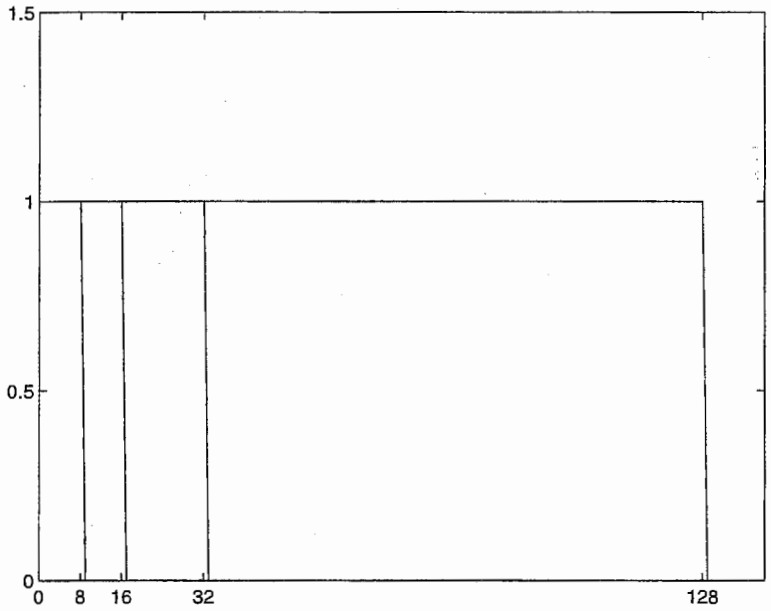


Figure 7.3: Baseline rectangular lifters.

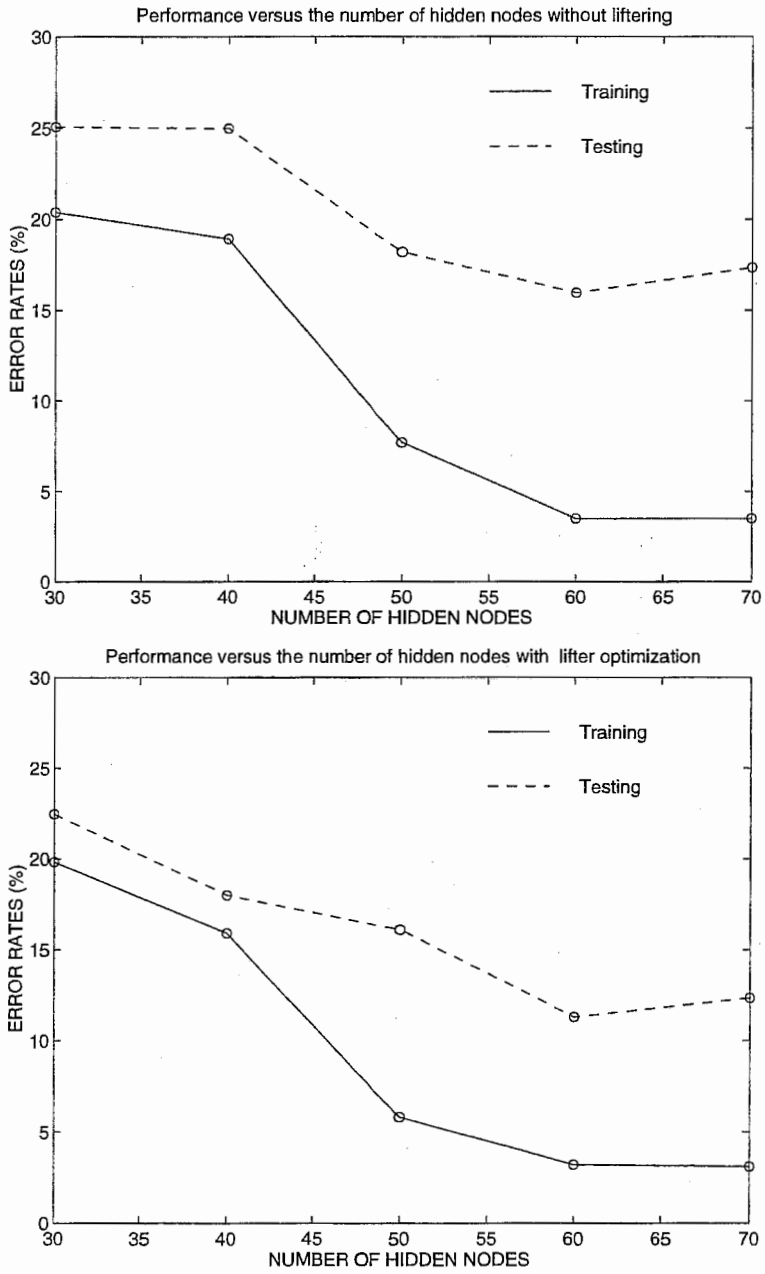


Figure 7.4: Error rates versus the number of hidden nodes. **Upper:** baseline system without liftering (with all pass lifter). **Bottom:** uniformly unitialized DFE-trained system.

Generally, the number of trainable parameters affects the achievable accuracy of the classifier. To investigate this point in this framework, preliminary experiments was carried out, setting the number of the classifier hidden nodes to 30, 40, 50, 60, and 70, for the MCE/GPD-trained baseline system having the 128-length all-pass lifter and the uniformly-initialized, DFE-trained system.

Fig. 7.4 shows the achieved error rates, one for each number of hidden nodes: the upper figure is for the baseline system, and the bottom is for the DFE-trained system. The results show that using 60 hidden nodes is the most suitable for the experimental framework. In light of the above results, a classifier having 60 hidden nodes was used in the subsequent experiments.

7.5.1 Baseline system

First, let us compare the four low-pass lifter lengths which correspond to the baseline systems. As summarized in Table 7.1, the length of 16 produced the best performance over the testing data. The no-liftering situation achieved quite high accuracy on the training set but resulted in lower accuracy on the testing set. The results may suggest that the training in the no-liftering case used the information that was specific to the training samples but irrelevant to finding the true class boundary. Thus, this result shows that an appropriate liftering is indispensable for alleviating this problem.

7.5.2 Effect of initial lifter in DFE training

Let us compare the three initialization methods in the context of DFE training. Based on the above results, the length of the initial lifter was set to 16 in the rectangular initialization. The results for these three initialization cases are also shown in Table 7.1. DFE successfully achieved 11.3% in the uniform initialization case, which the best result across all cases. Importantly, the DFE training provided this improvement over the testing data, while on the training data, it showed a similar accuracy to that of the no-liftering baseline system. This may allow us to argue that the DFE-trained lifter successfully extracted features that are general and more useful for the classification of vowels. The random initialization performed poorly. This maybe explained by the fact that the random initialized provide a "bad" starting point the gradient descent algorithm. This result can be contrasted with the uniform initialization case, which takes into account the modularity of the recognizer and does not impose an initial lifter, thus providing a "good" starting point for DFE optimization.

Between the random initialization and the no-initial lifter situation, the rectangular initialization scheme (16-length lifter) imposes an initial lifter based on expertise. Indeed, this training scheme has produced the second best result in the testing set while showing similar performance on the training set when the lifter is un-trained. Consequently, a priori knowledge may help the DFE training, but at the same time it may inhibit the full capability of the DFE training.

Table 7.1: Error rates for the baseline systems and the DFE-trained systems.

Liftering process	Training	Testing
baseline system (no liftering (128-length fixed lifter))	3.5%	16.0 %
baseline system (32-length fixed lifter)	13.4 %	15.3%
baseline system (16-length fixed lifter)	8.4 %	14.5%
baseline system (8-length fixed lifter)	9.7 %	14.9%
DFE-trained system (16-length rectangle initialization)	8.4 %	14.2%
DFE-trained system (random initialization)	7.0 %	16.8 %
DFE-trained system (uniform initialization)	3.2 %	11.3 %

7.5.3 Lifter analysis

Fig. 7.5 shows a typical shape of the trained lifter in the uniform initialization case and for the rectangular initialization case. Both lifters de-emphasizes two quefrency regions: 1) the high quefrency region that corresponds to pitch harmonics and spectral minute structure, and is irrelevant to vowel discrimination, 2) the lower quefrency region (0-2 quefrency region) that is dominated by the bias and slant of the overall spectrum added to speaker characteristics, while enhancing the region of 3-20 quefrency that mainly corresponds to the spectral formant structure. Thus, the cut-off quefrency can be said to be around 20. Note that rectangular lifter of 16 gave the best result among the baseline methods. This reasonable result might be a good support for the argument cited above and shows that the DFE-trained lifter provide a *meaningful* shape in terms of scientific knowledge.

Investigation of the significance of the 3-20 quefrency regions was carried out by analyzing the pitch frequencies of the design speech samples. It was found that female pitches ranged from 40 quefrency to 60 quefrency and male pitched was concentrated in the 80 quefrency region. Thus, the trained lifter in Fig. 7.5 successfully suppresses these pitch components, though few ripples remain in the corresponding regions. The pitch components are better suppressed when using the 16 length lifter.

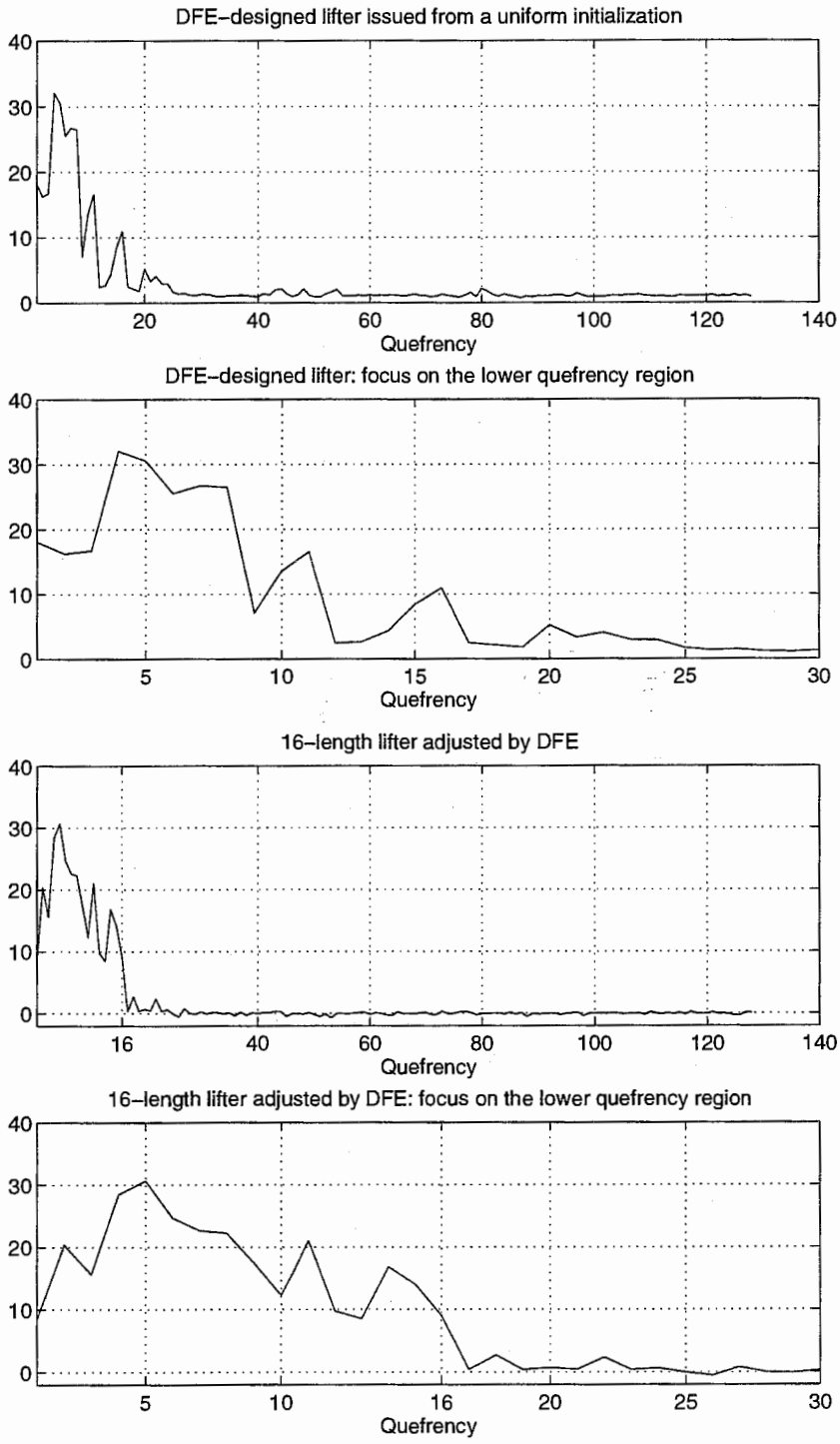


Figure 7.5: **Top figure:** a typical DFE-designed lifter issued from the uniform initialization. **Upper-top:** the entire view. **Bottom-top:** the same lifter with a focus on the lower quefrency region. **Bottom Figure:** a typical DFE-designed lifter issued from the 16-length rectangular initialization. **Upper:** the entire view. **Bottom:** the same lifter with a focus on the lower quefrency region.

7.5.4 Comparison with statistically based lifter

It is obvious that these trained lifters are different from conventional ones based on *a priori* knowledge (e.g., see [Ohyama *et al.*, 1981b]). Comparison of the resulting lifters with those obtained from the data-driven approach proposed in [Tohkura, 1987] was made.

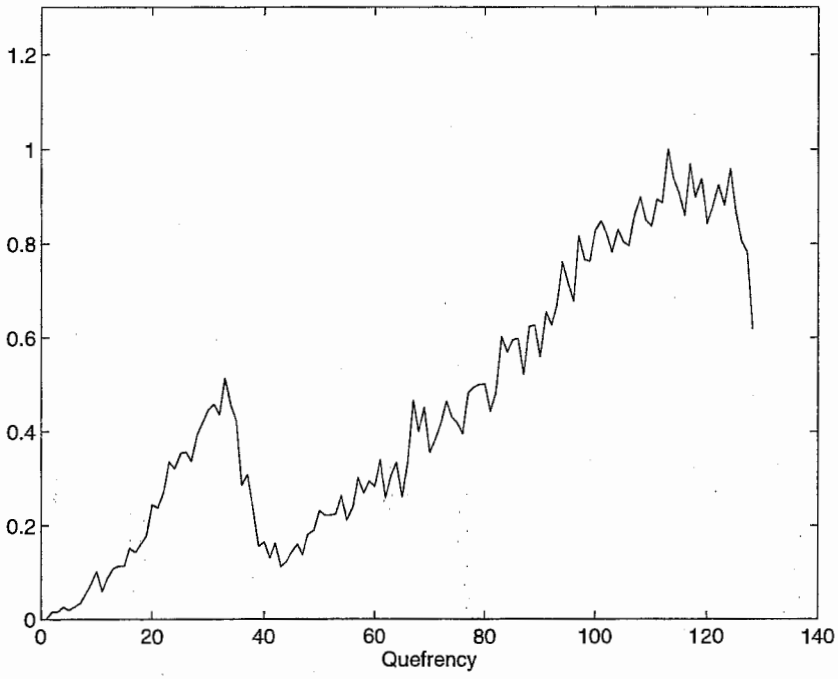


Figure 7.6: Normalized inverse variance over the training set. Normalization is performed such that the maximum value is equal to 1.

Fig. 7.6 shows the inverse function curve of the average intraclass variance, computed according to the method in [Tohkura, 1987]. One should note that the DFT-based cepstrum was here, while [Tohkura, 1987] used an LPC-based one; therefore, it is difficult to compare both directly. However, it is clear that the curve successfully suppresses the lower-quefrency components but fails to suppress the higher-quefrency components. Again, the comparison clearly shows that DFE provides an important departure from the conventional design approach.

7.5.5 Comparing MCE and MSE

The same study can be made using the MSE criterion, instead of the MCE. An experiment was performed where a uniform lifter (that is, no initial liftering) and random lifter were adjusted by the MSE criterion, using the same architecture describe above. For reminding, the MSE criterion is defined as

$$\ell_{MSE}(\mathbf{c}) = \sum_{k=1}^{M} (t_k(\mathbf{c}) - o_k(\mathbf{c}))^2 \tag{7.21}$$

where $t_k(x)$ signifies the k-th coefficient of the target vector t_k for input vector x.

Note that only one difference between DFE/MSE and DFE/MCE-based design is the selection of the learning objective. Several training runs were performed, changing the order of data presentation.

Table 7.2: Recognition rates for the Japanese vowels task using the MCE criterion and the MSE criterion.

Liftering initialization values	MCE		MSE	
	train	test	train	test
1.0	96.8%	88.7%	96.8%	86.2%
random	93%	83.2%	89.8%	80.8%

Table 7.2 summarizes the best recognition rates using MCE/GPD and the standard minimum-square error criterion (MSE) aimed at lifter design. Similar to the MCE 's case, the random initialization performs poorly, when using the MSE criterion.

The comparison between MCE and the MSE training shows that MCE provided more accurate recognition rates than the MSE training in the training set, while showing similar result in the training set.

7.5.6 MCE-based lifter and MSE-based lifter

Fig. 7.7 shows the resulting lifter trained with the MSE loss.

Similar to the MCE-based lifter, the MSE-based lifter also suppresses the high quefrency region and enhance the low quefrency values which corresponds to phoneme characteristics. Comparing MSE-based and MCE-based lifter illustrations, one notes that the MCE-based lifter is smoother, which is consistent with the higher generalization power in recognition accuracy of the MCE-trained lifter. Both show high variability occurring in the low quefrency region certainly due to the high variance of the data in that region.

Spectral analysis

To deepen the understanding of the proposed approach, investigation of power spectra were carried out, each corresponding to the trained lifter.

Fig. 7.8 shows the logarithmic power spectra of a single frame spectrum; one corresponding to the raw cepstrum vector and the other to the lifter-processed cepstrum vector. The liftered spectra clearly brings out the formant structure of the frame, thus confirming its importance in vowel classification shown in psychology. The MCE-based spectrum is smoother than the MSE-based one and also show more similarities in shape to the spectrum envelope. Significant notches still remain on the smoothed spectrum for both MCE and MSE, which seems to be due to an inability in suppressing high quefrency elements and the ripple-like variation in lifter shape.

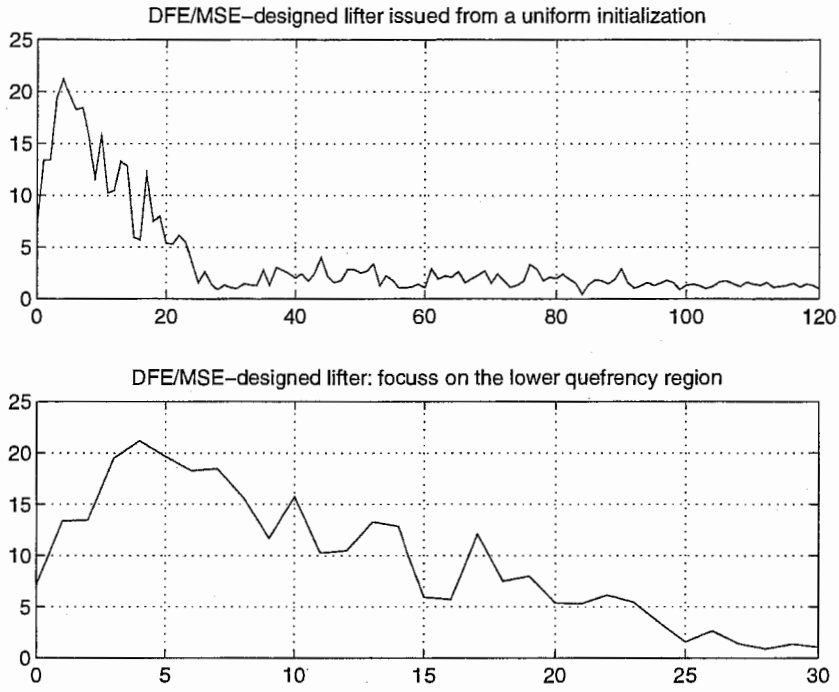
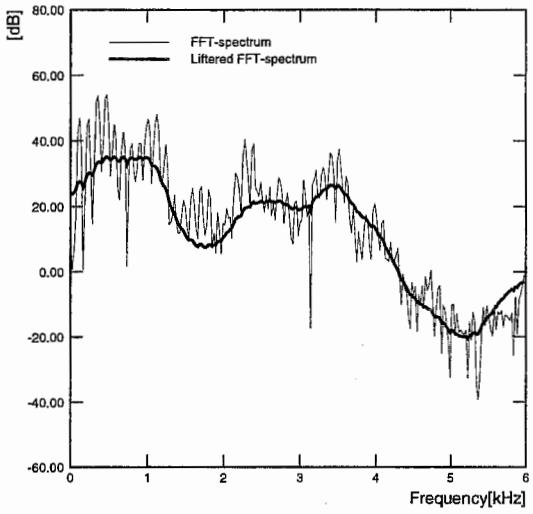


Figure 7.7: MSE-based lifter.

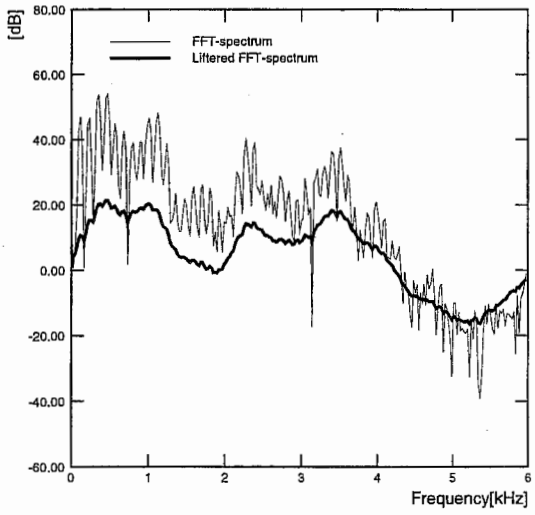
7.6 Discussion

The difference between DFE and classical use of MCE/GPD has a large significance, although a rather simple one. That is, the design scope extended by DFE will achieve more accurate recognition, given system resources (e.g., trainable parameters), by extracting more suitable features for classification; this property can conversely lead to system size reduction and faster recognition computation, relieving the burden on the classifier. Furthermore, by controlling the training conditions, such as the learning factor, the wider scope will realize a more flexible inter-module interaction. In the extreme case, this interaction can be a practical way to easily retrain the pre-designed status of either the feature extractor or the classifier, though it somewhat contradicts the DFE concept.

DFE substantially increases the number of adjustable system parameters. This fact would enlarge the ratio of the number of parameters to that of given training samples, hence probably increasing the statistical bias of training results. However, in contrast to the usual case of increasing the parameters of a classifier which operates in a fixed feature space, the nature of such bias in the DFE training framework is unclear. This point is an important future research issue.



MCE-based liftered spectrum of a frame of /o/



MSE-based liftered spectrum of a frame of /o/

Figure 7.8: MCE-based liftered power spectrum and MSE-based liftered power spectrum of a vowel frame.

7.7 Conclusion

In this chapter, Discriminative Feature Extraction application to a classical feature representation of speech, namely an optimal lifter design, was described.

For evaluation, a vowel fragment recognition experiment was conducted. It was shown that a DFE-designed lifter successfully achieved fewer classification errors than the baseline MCE/GPD-trained classifier represented by a rectangular lifter shape, when adjusting a uniformly initialized lifter (i.e., no use of a priori knowledge in initializing the feature extractor parameter). Briefly, the following points can be stated:

- Imposing a random lifter shape at the beginning of the process gives poor performance. In others words, the modularity of the recognizer should be taken into consideration on the DFE training process.
- A uniform lifter at the beginning of the process gives the best result. A uniform lifter is equivalent to non-liftering at the beginning of the DFE process. Thus, DFE optimality is achieved when an initial feature extractor is not imposed.
- Using a priori knowledge in the form of an initial rectangular lifter gives the second best performance. That is, expertise can be a reasonable starting point for an initial feature extractor design. However, this initial feature extractor may inhibit the realization of a general optimal recognizer.

The lifter was designed using a Feed-Forward Neural Network classifier. Its shape clearly departs from classical lifter shapes while showing an emphasis of the low-terms cepstral coefficients, believed to carry the discriminant informations, and provides higher recognition compared to classical rectangular lifter. The DFE process has rather enhanced the region of interest, than suppressed higher quefrency terms in contrast to classical methods. Investigation of the liftering effect on the spectrum of speech shows that the formant structures are kept, thus confirming its importance in vowel classification.

Comparison of MSE and MCE, targeting lifter design, was also made. MCE-based criterion achieves better performance and display a smoother lifter than MSE.

Chapter 8

Discriminative Feature Extraction Applied to Filter Bank Design

No knowledge is so easily found as when needed

-Robert Henri-

In speech recognition, filter bank modeling has been the basis for the production of various speech features. Given a task, there are still question marks concerning the choice of a particular filter bank model, the proper number of filters to be used, the appropriate frequency scale (perceptual or linear) and the spacing of the filters. Although the answer is obviously task-dependent, there is no certainty about the optimality of the adopted model. Thus, an optimal tuning of the filter bank parameters, given a task and a classifier structure is investigated within the DFE framework.

8.1 Introduction

Filter bank modeling is one of the most the popular approach to speech parameterization. In speech recognition, a short fragment of an input speech is often converted to an output of a bank-of-filters model, and then the overall input is represented as a sequence of these output vectors. Various parameters, such as center frequency, bandwidth or gain should be taken into consideration when designing a filter bank front-end feature extractor. This chapter focuses its attention on designing a filter bank aiming at realizing highly-accurate speech recognition, in the minimum error sense [Biem and Katagiri, 1993a; Biem and Katagiri, 1994] Again, design of the filter bank is linked with various crucial decisions. The set of filters must be chosen so as to extract the important part of the speech, i.e. relevant to the back-end classification. That is, an appropriate number of filters must carefully investigated for accurate resolution of the speech spectrum. An important parameter in filter bank design is the spacing interval between two adjacent filters. The number of filters, their type as well the spacing method (overlapping or non-overlapping) depends on the task and the processing environment (noisy environment, signal from dial-up phone, etc..).

Here, the design of a minimum classification-based filter bank through the DFE method is shown. A bank-of-filter feature extractor module is comprehensively optimized with the classifier's parameters for minimization of the errors occurring at the back-end classifier [Biem and Katagiri, 1993a]. Evaluation is done in two experimental tasks: one recognizing static vowel fragments, each being a filter bank output vector, and one recognizing *dynamic* (variable-durational) words, each being a sequence of filter bank output vectors, within the ATR directory assistance task framework [Woudenberg *et al.*, 1995]. The first, relatively simple task enables us to carefully investigate the nature of DFE training, e.g., plausibility of designed filter bank shapes; the second task enables us to study utility of DFE in a more realistic environment using dynamic and distorted telephone speech utterances. In both tasks, DFE-designed recognizers are compared with conventionally-designed, baseline systems, in which the filter bank is defined based on psychoacoustic findings and the classifier is trained by using Maximum Likelihood Estimation and MCE/GPD. MLE, in the context of the prototype-based recognizer used in chapter takes the form of the k-means clustering [Juang and Rabiner, 1991].

The use of the prototype-based distance classifier would be considered rather traditional nowadays. However, the classifier is mathematically equivalent, conditioned by the selection of distance measure, to a simplified version of a classifier based on hidden Markov model (HMM) (See [McDermott and Katagiri, 1994; McDermott, 1997] for details) and has sucessfully been used for filter bank optimization [Biem *et al.*, 1995].

8.2 Filter Bank Modeling of Speech

In filter bank modeling of speech, a speech waveform is passed through a set of I bandpass filters. The filter bank process can be implemented either in the time domain or in the frequency

domain of speech: a) executing filtering computation through finite impulse response filters (FIR) or recursive filters (IIR) b) simulating filter outputs using a DFT-based power spectrum. Filter bank implementation in the time domain permits a better control of time and frequency resolution, which may result in better feature extraction. If IIR filters are less computationally costly, FIR filters enable the realization of precise linear phase filter with well-defined frequency characteristics. When a precise frequency resolution is not required, the filter process can be simulated directly on the power spectrum. Given the simplicity and easier implementation through FFT, most filter bank implementation in the speech recognition field have been carried out using DFT-estimated power spectrum. For the filter bank parameter adjustment, which is the goal of the DFE training, this can be directly implemented through use of the FFT-based power spectrum. Consequently, throughout this report, the filter bank process will be simulated by weighting of the FFT-generated power spectrum.

8.2.1 Filter bank-based feature extractor

Again, for simplicity, emulation of filtering is done by using FFT computation. An input speech signal is first converted to a sequence of FFT-based power spectrum vectors, $\boldsymbol{s}_1^{\mathcal{T}} = \{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_t, \ldots, \boldsymbol{s}_{\mathcal{T}}\}$ where $\boldsymbol{s}_t$ is an FFT-based, F-dimensional power spectrum vector at time t (time window position) and $\mathcal{T}$ is the (sequence) length of $\boldsymbol{s}_1^{\mathcal{T}}$; $\boldsymbol{s}_t = [s_{t,1}, \ldots, s_{t,f}, \ldots, s_{t,F}]^{\mathrm{T}}$ where $s_{t,f}$ is the f^{th} element of $\boldsymbol{s}_t$ (the element of $\boldsymbol{s}_t$ at the f^{th} frequency index[1].). Assuming that a filtering function is $\mathcal{F}(\cdot;\Theta)$ where Θ is a set of trainable control parameters of the filter bank, the filter bank converts $\boldsymbol{s}_1^{\mathcal{T}}$ to its corresponding filter bank output pattern $\boldsymbol{x}_1^{\mathcal{T}}$ in a vector-by-vector mode; $\boldsymbol{x}_1^{\mathcal{T}} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t, \ldots, \boldsymbol{x}_{\mathcal{T}}\}$ where $\boldsymbol{x}_t$ is the I (usually $< F$)-dimensional filter bank output vector that corresponds to $\boldsymbol{s}_t$, i.e., $\boldsymbol{x}_t = [x_{t,1}, \ldots, x_{t,i}, \ldots, x_{t,I}]^{\mathrm{T}} = \mathcal{F}(\boldsymbol{s}_t;\Theta)$ for all possible t's; $\boldsymbol{x}_1^{\mathcal{T}} = \mathcal{F}(\boldsymbol{s}_1^{\mathcal{T}};\Theta)$.

The detailed process of filter bank modeling using FFT is as follows. Filtering is done by weighting of the input power spectrum vector. For a power spectrum vector $\boldsymbol{s}_t$ (short time window position), the I-channel filter bank model transforms each $\boldsymbol{s}_t$ into a lower dimensional vector $\boldsymbol{x}_t = [x_{t,i}]^{\mathrm{T}}$ for $i \in \{1, \ldots I\}$ such that an output feature $x_{t,i}$ is the log energy in the channel:

$$x_{t,i} = \log_{10}\left(\sum_{f \in B_i} w_{i,f} s_{t,f}\right) \quad \text{for } i = 1, \ldots, I, \tag{8.1}$$

where B_i represents the channel interval and $w_{i,f}$ the weighting at frequency f within the i^{th} channel. This basic formulation of filtering can be re-written as,

$$\boldsymbol{x}_t = \mathcal{F}(\boldsymbol{s}_t;\Theta) = \left[\log_{10}(\boldsymbol{w}_1{}^{\mathrm{T}}\boldsymbol{s}_t), \ldots, \log_{10}(\boldsymbol{w}_i{}^{\mathrm{T}}\boldsymbol{s}_t), \ldots, \log_{10}(\boldsymbol{w}_I{}^{\mathrm{T}}\boldsymbol{s}_t)\right]^{\mathrm{T}}, \tag{8.2}$$

where I, which is the dimension of the filter bank output vector, corresponds to the number of filters of the filter bank; $\boldsymbol{w}_i$ is the F-dimensional weight vector of the i^{th} channel filter, i.e., $\boldsymbol{w}_i = [w_{i,1}, \ldots, w_{i,f}, \ldots, w_{i,F}]^{\mathrm{T}}$ where $w_{i,f}$ is the weight coefficient of the i^{th} channel filter at the

[1]Concretely, f is an integer describing the sequence of DFT bins

f^{th} frequency index. This conversion is repeated in the vector-by-vector mode, i.e., for every t, thus transforming $\boldsymbol{s}_1^{\mathcal{T}}$ into $\boldsymbol{x}_1^{\mathcal{T}}$.

Usually, the filters of the model are spaced in a perceptual frequency scale. These perceptually-based spacing methods have been shown to be useful to increase speech recognition accuracy. Also, to make the output vectors $\boldsymbol{x}_t$ smoother, the filters are placed so that they overlap each other, and the set of weight vectors of the filter bank, i.e., $\mathcal{W} = \{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_i, \ldots, \boldsymbol{w}_I\}$, are determined by using some (small number of) continuous functions. However, in principle, each element of the vector, $w_{i,f}$, itself can be individually treated as a trainable parameter. This latter approach seems attractive, because it permits the design of a more general filter bank. However, this flexible setting also increases sensitivity to training data because of the increase in the number of parameters, which may result in a less robust design for unknown input patterns.

8.2.2 Parameter selection for filter bank design

Again, there are several possibilities for implementing FFT-based filter banks. Here, two types of implementation were selected, i.e., 1) each filter's frequency response has a Gaussian form determined by three kinds of trainable parameters, center frequency, bandwidth, and gain factor (*G-type*) , and 2) each filter is a set of independent weight coefficients (*I-type*). The Gaussian form was selected based on its smoothness and tractability.

DFE-optimization of the filter bank is illustrated in Fig. 8.1 and is as follows. The i-th filter response within the G-type implementation is defined as:

$$w_{i,f} = \alpha_i \exp\left[-\beta_i \left\{p(\gamma_i) - p(f)\right\}^2\right], \quad \text{for } i = 1, \cdots, I, \tag{8.3}$$

where α_i simulates the gain factor, β_i (> 0) determines the bandwidth, γ_i determines the center frequency, respectively, at the i^{th} channel; $p(\cdot)$ is a frequency mapping function that determines frequency scaling. Thus, $\Theta = \{\{\theta_i\}\}_{i=1}^{I} = \{\{\alpha_i, \beta_i, \gamma_i\}\}_{i=1}^{I}$, and this definition leads to the following four cases of DFE training for the G-type filter implementation, based on the selection of the trainable parameters:

1. **selective training I (Gc-training)**: Gc-training trains only the center frequencies γ_i's, while keeping the remaining filter parameters fixed. Note that modifying the spacing of the filters, with a fixed bandwidth and gain, results in a different coverage of the available frequency range.

2. **selective training II (Gb-training)**: Gb-training only adjusts the bandwidths β_i's. A larger value of β_i signifies a narrower filter and vice-versa.

3. **selective training III (Gg-training)**: Gg-training only adjusts the gain factors α_i's. A large value of gain factor puts emphasis on the filter output energy.

4. **simultaneous training (GS-training)**: In GS-training, the three types of adjustable parameters, i.e., center frequencies, bandwidths, and gain factors, are simultaneously trained.

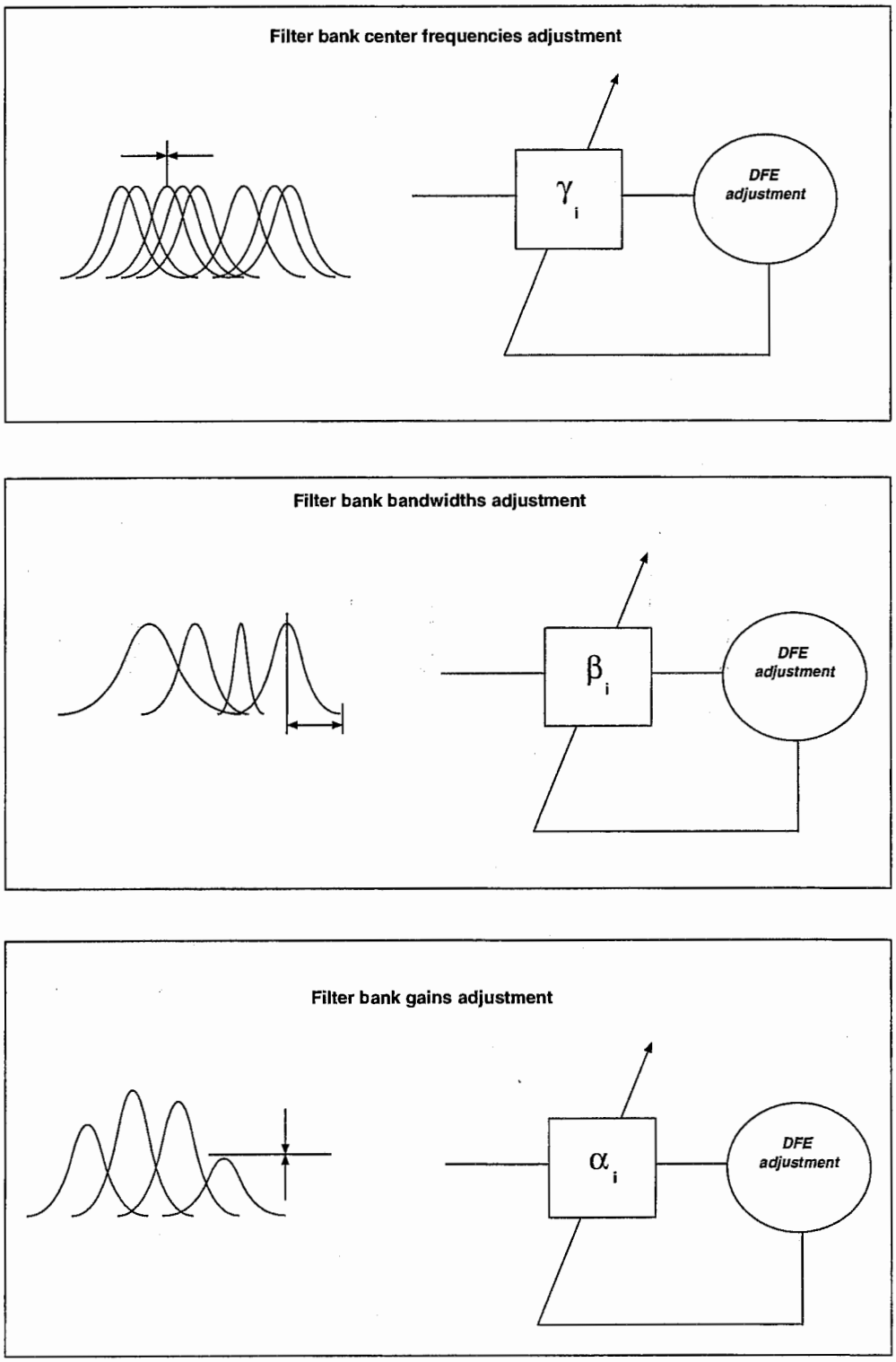


Figure 8.1: *G-type* training. The center frequencies, bandwidth and gain of the i-th filter can be optimized by adjusting the parameters γ_i, β_i and α_i

It should be noted here that the above training cases keep the Gaussian shape of individual filters. And, unlike the T-type case, optimizing filter centers, bandwidths or gains of the filter bank is a less tangible phenomenon than adjusting a filter weight.

In the framework of I-type filter implementation, the weight functions are considered independent and are not constrained to be of a pre-chosen filter shape, thus $\Theta = \left\{\{w_{i,f}\}_{f=1}^{F}\right\}_{i=1}^{I}$. This then leads to only one training case, namely, **I-type** training where one treats each filter weight $w_{i,f}$ as an independent trainable parameter. The DFE adjustment rule is applied to $\hat{\vartheta} = \log(\vartheta)$ instead of ϑ (ϑ corresponds to either α_i, β_i, γ_i, or $w_{i,f}$). An altertive is to use the squared root as in [Biem and Katagiri, 1994]. This formalism has also been applied to speaker normalization [Woundenberg *et al.*, 1997].

8.3 Recognizer Structure

For generality of description, the task of recognizing dynamic speech patterns of M classes such as phonemes and words; $\{C_1, \cdots, C_j, \cdots, C_M\}$. The speech recognizer is a modular system, consisting of a front-end filter bank-based feature extractor and a back-end prototype-based distance classifier. Fig. 8.2 illustrates this modular recognizer.

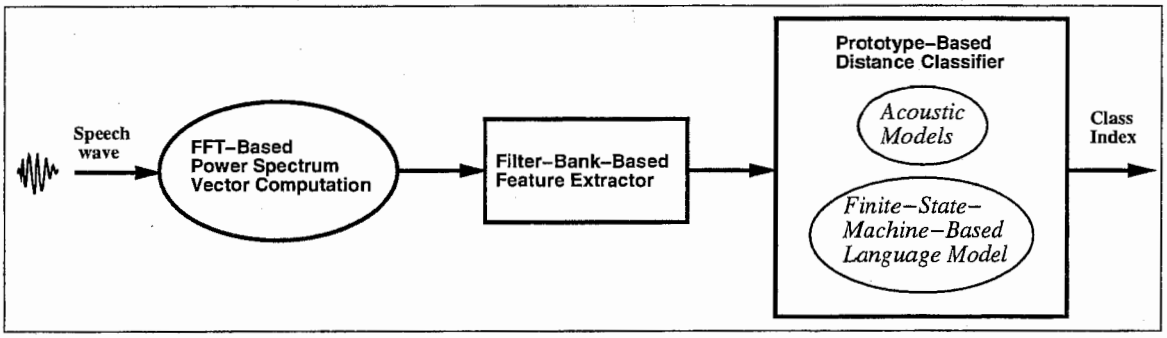


Figure 8.2: Block diagram of a modular speech recognizer consisting of a front-end filter bank-based feature extractor and a back-end multi-prototype distance classifier.

8.3.1 Classifier structure

The classifier is based on the one in [McDermott and Katagiri, 1994]. It consists of an acoustic model and a language model. The acoustic model uses a state transition structure, similar to an HMM; the language model is a finite state machine structure. The decision rule in this stage is based on the most fundamental rule in the Bayes decision theory; i.e.,

$$a(\boldsymbol{x}_1^T) = C_i \qquad \text{if } i = \arg\min_j \left\{g_j(\boldsymbol{x}_1^T; \Lambda)\right\}, \tag{8.4}$$

where $C(\cdot)$ is classification operation, Λ is a trainable classifier parameter set, i.e., a set of prototype vectors, and $g_j(\boldsymbol{x}_1^T; \Lambda)$ is a discriminant function, defined in terms of Λ, which indicates the degree

to which $\boldsymbol{x}_1^{\mathcal{T}}$ belongs to C_j. Note here that a smaller value of the discriminant function indicates a higher possibility that an input to the classifier belongs to the corresponding class.

Prototype-based distance classifier

In this classifier, each class (phoneme/word/phrase) is modeled as a string of sub-phonetic states, each being assigned prototypes in the filter bank output vector space. The model resembles an HMM in structure but uses distance computation in place of probability estimation.

Concretely, we are given a finite set of P phonetic models,

$$\Lambda = \{\lambda_1, \ldots, \lambda_i, \ldots, \lambda_P\}, \quad 1 \leq i \leq P, \tag{8.5}$$

where λ_i is composed of a set of prototypes distributed among the states of the model. Each phonetic model λ_i has S_i states. And the total number of states is S.

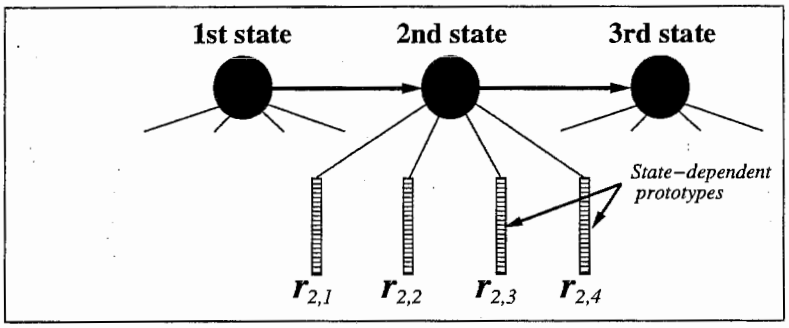


Figure 8.3: Prototype-based minimum error classifier structure.

Fig. 8.3 illustrates a model with 3 states and 4 prototypes per state. According to [McDermott and Katagiri, 1994], the discriminant function $g_j(\boldsymbol{x}_1^{\mathcal{T}}; \Lambda)$ is defined in the following way.

State-distance

Given a state ψ, the state-distance indicates the average similarity between the filter bank output vector $\boldsymbol{x}_t$ and prototypes in the state. The state-distance is defined as

$$D_\psi(\boldsymbol{x}_t) = \left\{ \sum_{n=1}^{N_\psi} \sigma(\boldsymbol{x}_t, \mathbf{r}_{\psi,n})^{-\nu} \right\}^{-\frac{1}{\nu}}, \tag{8.6}$$

where ν is a positive constant. $\mathbf{r}_{\psi,n}$ is the n-th prototype in state ψ and N_ψ is the number of prototypes in the state. $\sigma(\boldsymbol{x}_t, \mathbf{r}_{\psi,n})$ measures the *local distance* between a filter bank output vector and a prototype as follows:

$$\sigma(\boldsymbol{x}_t, \mathbf{r}_{\psi,n}) = (\boldsymbol{x}_t - \mathbf{r}_{\psi,n})^{\mathrm{T}} (\boldsymbol{x} - \mathbf{r}_{\psi,n}). \tag{8.7}$$

This distance is simply the squared Euclidean norm of the two corresponding vectors in the filter bank vector space. The state-distance is thus an L_p norm of local distances. The local distance

can take the general form of a Mahalanobis distance. However, for reducing the complexity of the system, the local distance measure has been restricted to an Euclidean distance.

Discriminant function

Given the sequence of filter bank output vectors $\boldsymbol{x}_1^{\mathcal{T}}$, the discrimination measure of the category C_j, is found by computing an aggregate state distance over possible state sequences, as follows:

$$g_j(\boldsymbol{x}_1^{\mathcal{T}};\Lambda) = \left[\sum_{\varpi_j}\left\{V_{\varpi_j}(\mathcal{D}_{\mathcal{T},\mathrm{S}})\right\}^{-\zeta}\right]^{-1/\zeta}, \tag{8.8}$$

where $\mathcal{D}_{\mathcal{T},\mathrm{S}}$ is a matrix of state distances, where each element (t,ψ) contains $D_\psi(\boldsymbol{x}_t)$; $V_{\varpi_j}(\mathcal{D}_{\mathcal{T},\mathrm{S}})$ is the *path distance* representing an accumulated sum of state distances along the path ϖ_j, one of possible paths within a region of $\mathcal{D}_{\mathcal{T},\mathrm{S}}$ permitted by a Dynamic Programming procedure for C_j; ζ is a positive constant. Accordingly, the classifier recognizes the filter bank output $\boldsymbol{x}_1^{\mathcal{T}}$ by applying the discriminant function (8.8) to the decision rule (8.4).

The distance classifier works in a quite similar manner to HMM-based classifiers. The technical merit here is that, without any serious defect in modeling capability, one can use any reasonable distance measure in place of the probability density function, which is used in HMM and whose estimation is usually more complex than the computation of distance measures because of the strict condition that the integral of the density function over its corresponding sample space must be equal to one. Many distance measures such as the Mahalanobis distance and the Euclidean distance are directly derived from the common Normal probability distribution. This emphasizes the close relationship between the distance-based classifier and HMM-based classifiers. This relationship accordingly allows us to consider that the discussion in this chapter should fundamentally hold true in the general case of using HMM classifiers. However, the structure used here allows one to use any metric (for instance, Itakura-Saito distance [Itakura and Saito, 1970]) in place of (8.7).

8.4 DFE implementation

As said before, a traditional design framework simply determines the filter bank $\mathcal{F}(\cdot;\Theta)$ by means which are independent of the design of the classifier, or in other words, the design of the discriminant function set $\{g_j(\cdot;\Lambda)\}$. Clearly, this approach is not optimal and can be improved by DFE training. For the system structure described above, the DFE training is implemented as follows.

Assume that a design sample $\boldsymbol{s}_1^{\mathcal{T}}$ $(\in C_k)$ is given at the training time index τ. Again, the filter bank $\mathcal{F}(\cdot;\Theta)$, converts $\boldsymbol{s}_1^{\mathcal{T}}$ to $\boldsymbol{x}_1^{\mathcal{T}}$. Let $\Psi^j = \{\psi_1^j, \psi_2^j, ..., \psi_{\mathcal{T}}^j\}$, be the *best* sequence of states found by the DP procedure, corresponding to category C_j. ψ_t^j corresponds to the state occupied by the feature-frame $\boldsymbol{x}_t$ along this path. Thus, $D_{\psi_t^j}(\boldsymbol{x}_t)$ is the state-distance, corresponding to the feature-vector $\boldsymbol{x}_t$. For DFE implementation, the discriminant function only considers the accumulated sum

of state-distance, along the best path. That is,

$$g_j(\boldsymbol{x}_1^{\mathcal{T}};\Lambda)=\sum_{t=1}^{\mathcal{T}} D_{\psi_t^j}(\boldsymbol{x}_t), \tag{8.9}$$

which is equivalent to letting $\zeta \to \infty$, in the definition given in (8.8). During training, the correct path $g_k(\boldsymbol{x}_1^{\mathcal{T}};\Lambda)$ is simply found by performing the DP procedure constrained by the word/phoneme transcription.

Normalized misclassification measure

The straightforward way to implement MCE on recognizer is to make use of the classical misclassification measure, which is defined, in this context as,

$$d_k\left(\mathcal{F}\left(\boldsymbol{s}_1^{\mathcal{T}};\Theta\right);\Lambda\right)=g_k\left(\boldsymbol{x}_1^{\mathcal{T}};\Lambda\right)-\left\{\frac{1}{M-1}\sum_{j\neq k}^{M} g_j\left(\boldsymbol{x}_1^{\mathcal{T}};\Lambda\right)^{-\xi}\right\}^{-\frac{1}{\xi}}. \tag{8.10}$$

when using a PBMEC structure. However, using a simple difference as misclassification measure, result in a wide range of values spanned by the misclassification measure. An alternative is to make use of a normalized misclassification measure. For $\boldsymbol{s}_1^{\mathcal{T}}$ belonging to category C_k, the normalized misclassification measure is defined as

$$d_k\left(\mathcal{F}\left(\boldsymbol{s}_1^{\mathcal{T}};\Theta\right);\Lambda\right)=d_k\left(\boldsymbol{x}_1^{\mathcal{T}};\Lambda\right)=1-\frac{\left\{\frac{1}{M-1}\sum_{j\neq k}^{M} g_j\left(\boldsymbol{x}_1^{\mathcal{T}};\Lambda\right)^{-\xi}\right\}^{-\frac{1}{\xi}}}{g_k\left(\boldsymbol{x}_1^{\mathcal{T}};\Lambda\right)}, \tag{8.11}$$

where ξ is a positive number which controls the relative contribution of the classes considered. Again, the above misclassification is an attempt to normalize the general form of the misclassification measure given in Chapter 4. Its value range spans a narrower interval and thus permits the use of classical loss function. The normalized misclassification measure can be written in the following more compact form

$$d_k\left(\boldsymbol{x}_1^{\mathcal{T}};\Lambda\right)=1-\frac{g_{\overline{k}}\left(\boldsymbol{x}_1^{\mathcal{T}};\Lambda\right)}{g_k\left(\boldsymbol{x}_1^{\mathcal{T}};\Lambda\right)}. \tag{8.12}$$

Again, the anti-discriminant function $g_{\overline{k}}\left(\boldsymbol{x}_1^{\mathcal{T}};\Lambda\right)$ is defined as

$$g_{\overline{k}}\left(\boldsymbol{x}_1^{\mathcal{T}};\Lambda\right)=\left\{\frac{1}{M-1}\sum_{j\neq k}^{M} g_j\left(\boldsymbol{x}_1^{\mathcal{T}};\Lambda\right)^{-\xi}\right\}^{-\frac{1}{\xi}} \tag{8.13}$$

In accordance with the MCE concept, the sigmoid was used as smooth binary (0-1 step) loss. That is,

$$\ell_k\left(\boldsymbol{s}_1^{\mathcal{T}}[\tau];\Phi\right)=\ell\left(d_k(\boldsymbol{x}_1^{\mathcal{T}}[\tau];\Lambda)\right)=\frac{1}{1+\exp\left\{-ad_k(\mathcal{F}_\Theta(\boldsymbol{s}_1^{\mathcal{T}}[\tau]);\Lambda)\right\}}, \tag{8.14}$$

where a is a positive constant. The misclassification measure and the loss function are each a function of both Θ and Λ. Clearly, the interaction between the filter bank and the classifier is embodied by this definition of functionals.

8.4.1 Classifier parameter adjustment

The set of $\mathbf{r}_{\psi_t^j,n}$ defines the classifiers parameters. Again the adjustment rule is obtained as follows:

$$\mathbf{r}_{\psi_t^j,n}[\tau] = \mathbf{r}_{\psi_t^j,n}[\tau] - \epsilon_\tau \delta \mathbf{r}_{\psi_t^j,n} \tag{8.15}$$

where ϵ_τ is a small, monotonically-decreasing, positive number at τ (the training rate), and $\mathbf{r}_{\psi_t^j,n}[\tau]$ denotes the status of $\mathbf{r}_{\psi_t^j,n}$ at τ. By making use of the chain rule of differential calculus, the increment $\delta \mathbf{r}_{\psi_t^j,n}$, is given by

$$\delta \mathbf{r}_{\psi_t^j,n} = -2\ell'\left(d_k\left(\boldsymbol{x}_1^{\mathcal{T}};\Lambda\right)\right)\Upsilon_{kj}\left\{\frac{D_{\psi_t^j}(\boldsymbol{x}_t)}{\sigma(\boldsymbol{x}_t,\mathbf{r}_{\psi_t^j,n})}\right\}^{-(1+\nu)}\left(\boldsymbol{x}_t - \mathbf{r}_{\psi_t^j,n}\right) \tag{8.16}$$

Υ_{kj} is determined by the form of the misclassification measure. When using a normalized misclassification measure, we have

$$\Upsilon_{kj} = \begin{cases} \dfrac{g_{\overline{k}}\left(\boldsymbol{x}_1^{\mathcal{T}};\Lambda\right)}{g_k^2\left(\boldsymbol{x}_1^{\mathcal{T}};\Lambda\right)} & \text{if } j = k \\ -\dfrac{1}{(M-1)g_k\left(\boldsymbol{x}_1^{\mathcal{T}};\Lambda\right)}\left\{\dfrac{g_{\overline{k}}\left(\boldsymbol{x}_1^{\mathcal{T}};\Lambda\right)}{g_j\left(\boldsymbol{x}_1^{\mathcal{T}};\Lambda\right)}\right\}^{1+\xi} & \text{if } j \neq k. \end{cases} \tag{8.17}$$

For a classical misclassification measure as defined in (8.10),

$$\Upsilon_{kj} = \begin{cases} 1 & \text{if } j = k \\ -\dfrac{1}{(M-1)}\left\{\dfrac{g_{\overline{k}}\left(\boldsymbol{x}_1^{\mathcal{T}};\Lambda\right)}{g_j\left(\boldsymbol{x}_1^{\mathcal{T}};\Lambda\right)}\right\}^{1+\xi} & \text{if } j \neq k. \end{cases} \tag{8.18}$$

It can be observed that the adjustment rule of (8.16) is extremely close to an LVQ-like training. That is, reference vectors that belong to the right category are pulled closer and the reference vectors that belong the wrong category are pushed away.

8.4.2 Filter Bank optimization scheme

Let ϕ be any adjustable filter bank parameter. To keep the physical meaning of the parameters, there is the need to ensure that the values of the parameters remain positive during training. This is done by using the transformation $\phi = \exp(\overline{\phi})$. The chain rule of differential calculus is used for adjusting the filter bank parameters. DFE process performs the following adjustment:

$$\overline{\phi}[\tau+1] = \overline{\phi}[\tau] - \rho_\tau \mathbf{U2}\delta\overline{\phi}. \tag{8.19}$$

where

$$\delta\overline{\phi} = \frac{\partial \ell\left(d_k\left(\boldsymbol{s}_1^{\mathcal{T}};\Phi\right)\right)}{\partial\overline{\phi}}. \tag{8.20}$$

Since, the filter bank is shared across all states and paths, the following chain rule of differential calculus is applied,

$$\frac{\partial \ell\left(d_k\left(\boldsymbol{s}_1^{\mathcal{T}};\Phi\right)\right)}{\partial \overline{\phi}} = \sum_{t=1}^{\mathcal{T}}\sum_{i=1}^{I} \frac{\partial \ell\left(d_k\left(\boldsymbol{s}_1^{\mathcal{T}};\Phi\right)\right)}{\partial x_{t,i}} \frac{\partial x_{t,i}}{\partial \overline{\phi}}, \tag{8.21}$$

The term $\frac{\partial \ell\left(d_k\left(\boldsymbol{s}_1^{\mathcal{T}};\Phi\right)\right)}{\partial x_{t,i}}$ which depends on the classifier structure is independent on the filter bank parameter type. This term is referred to as "classifier's input-derivatives" at time t, because this term characterizes the gradient of the loss versus the input to the classifier. The second term $\frac{\partial x_{t,i}}{\partial \overline{\phi}}$ is determined by the filter bank process and depends on the filter bank parameter type.

Classifier's input derivatives

For a sequence of spectral frames $\boldsymbol{s}_1^{\mathcal{T}}$ of category C_k, each input frame is compared against a set of a reference vectors within a state of a certain phonetic model as decided by the DP procedure along a the path of category C_k. Having this in mind and applying the chain rule of differential calculus to $\frac{\partial \ell\left(d_k\left(\boldsymbol{s}_1^{\mathcal{T}};\Phi\right)\right)}{\partial x_{t,i}}$, we have

$$\frac{\partial \ell\left(d_k\left(\boldsymbol{s}_1^{\mathcal{T}};\Phi\right)\right)}{\partial x_{t,i}} = \sum_{j=1}^{M} \frac{\partial \ell\left(d_k\left(\boldsymbol{s}_1^{\mathcal{T}};\Phi\right)\right)}{\partial D_{\psi_t^j}(\boldsymbol{x}_t)} \sum_{n=1}^{N_{\psi_t^j}} \frac{\partial D_{\psi_t^j}(\boldsymbol{x}_t)}{\partial \sigma(\boldsymbol{x}_t, \mathbf{r}_{\psi_t^j,n})} \frac{\partial \sigma(\boldsymbol{x}_t, \mathbf{r}_{\psi_t^j,n})}{\partial x_{t,i}}. \tag{8.22}$$

From the definition of $\sigma(\boldsymbol{x}_t, \mathbf{r}_{\psi_t^j,n})$ given in (8.7), we have the following properties

$$\frac{\partial \sigma(\boldsymbol{x}_t, \mathbf{r}_{\psi_t^j,n})}{\partial x_{t,i}} = -\frac{\partial \sigma(\boldsymbol{x}_t, \mathbf{r}_{\psi_t^j,n})}{\partial r_{\psi_t^j,n,i}} \tag{8.23}$$

where $\delta r_{\psi_t^j,n,i}$ is the i^{th} component of $\delta \mathbf{r}_{\psi_t^j,n}$. From (8.23), it follows that

$$\frac{\partial \ell\left(d_k\left(\boldsymbol{s}_1^{\mathcal{T}};\Phi\right)\right)}{\partial x_{t,i}} = \sum_{j=1}^{M}\sum_{n=1}^{N_{\psi_t^j}} -\frac{\partial \ell\left(d_k\left(\boldsymbol{s}_1^{\mathcal{T}};\Phi\right)\right)}{\partial r_{\psi_t^j,n,i}} \tag{8.24}$$

$$= -\sum_{j=1}^{M}\sum_{n=1}^{N_{\psi_t^j}} \delta r_{\psi_t^j,n,i}. \tag{8.25}$$

Thus, the classifier's input derivatives are simply the negative sum over the increment of all reference vectors along all DP paths. The above relation establishes the close link between the classifier parameters and feature extraction at the local level. This relation is still valid if the distance σ is any symmetrical similarity measure (for instance a Gaussian mixtures, in case of HMM).

In light of the above, the increment $\delta\overline{\phi}$ is given by

$$\delta\overline{\phi} = \sum_{t=1}^{\mathcal{T}}\sum_{i=1}^{I}\sum_{j=1}^{M}\sum_{n=1}^{N_{\psi_t^j}} \delta r_{\psi_t^j,n,i} \frac{\partial x_{t,i}}{\partial \overline{\phi}} \tag{8.26}$$

$$= \sum_{t=1}^{\mathcal{T}} \sum_{j=1}^{M} \delta \mathbf{r}_{\psi_i^j,n}{}^{\mathrm{T}} \frac{\partial \boldsymbol{x}_t}{\partial \overline{\phi}} . \tag{8.27}$$

Filter bank parameter derivatives

Here, the focus is on expanding the filter bank's parameter derivatives $\frac{\partial x_{t,i}}{\partial \overline{\phi}}$. These derivatives do not depend on the back-end classifier, but rather on the structure of the filter bank. Consequently, the formula displayed in below could be used on any recognizer which uses the filter bank model described in this chapter. The general chain rule is

$$\frac{\partial x_{t,i}}{\partial \overline{\phi}} = \sum_{f=1}^{F} \frac{\partial x_{t,i}}{\partial w_{i,f}} \frac{\partial w_{i,f}}{\partial \overline{\phi}} . \tag{8.28}$$

$\frac{\partial x_{t,i}}{\partial w_{i,f}}$ can be computed in a straightforward manner. Departing from the definition in (8.1), we have

$$\frac{\partial x_{t,i}}{\partial w_{i,f}} = \frac{s_{t,f}}{\log(10) \left(\sum_{f' \in B_i} w_{i,f'} s_{t,f'} \right)} \tag{8.29}$$

$$= \frac{s_{t,f}}{\log(10) \exp_{10}(x_{t,f})}, \tag{8.30}$$

which is the derivative of features $x_{t,i}$ versus a weight $w_{i,f}$. $\frac{\partial w_{i,f}}{\partial \overline{\phi}}$ depends on the parameter type. Its computation is described below on a case by case basis.

Filter bank weight adjustment

Let us consider the case in which ϕ is the weight of a given channel $\hat{\imath}$ at frequency $\hat{f}$. That is, $\phi = w_{\hat{\imath},\hat{f}}$. This is the I-type training case where the filter bank weight are adjusted without keeping the Gaussian constraint. Let $\overline{w}_{\hat{\imath},\hat{f}} = \exp(w_{\hat{\imath},\hat{f}})$ From (8.3), it is straightforward that

$$\frac{\partial w_{i,f}}{\partial \overline{w}_{\hat{\imath},\hat{f}}} = \boldsymbol{\chi}(i,\hat{\imath})\, \boldsymbol{\chi}(f,\hat{f}) w_{\hat{\imath},\hat{f}} \tag{8.31}$$

where

$$\boldsymbol{\chi}(a,b) = \begin{cases} 0 & \text{if } a \neq b \\ 1 & \text{otherwise.} \end{cases} \tag{8.32}$$

Center frequencies adjustment

Let ϕ represents the center frequency of a channel $\hat{\imath}$. The perceptual mapping function being monotonic, the adjustment can be done done directly in the perceptual domain. Thus, let $\phi = \Gamma_{\hat{\imath}} = p(\gamma_{\hat{\imath}})$. Again, $p(\cdot)$ maps the linear scale to the perceptual frequency scale. For $\overline{\Gamma}_{\hat{\imath}} = \exp(\Gamma_{\hat{\imath}})$. it follows that

$$\frac{\partial w_{i,f}}{\partial \overline{\Gamma}_{\hat{\imath}}} = -2\beta_{\hat{\imath}}\, \Gamma_{\hat{\imath}}\, (\Gamma_{\hat{\imath}} - p(f))\, w_{i,f}\, \boldsymbol{\chi}(i,\hat{\imath}). \tag{8.33}$$

Bandwidth adjustment

Here ϕ is the parameter $\beta_{\hat{\imath}}$ of channel $\hat{\imath}$. Again, let $\overline{\beta}_{\hat{\imath}} = \exp(\beta_{\hat{\imath}})$. From (8.3), we have

$$\frac{\partial w_{i,f}}{\partial \overline{\beta}_{\hat{\imath}}} = -\beta_{\hat{\imath}} \left(p(\gamma_{\hat{\imath}}) - p(f)\right)^2 w_{i,f}\, \chi(i,\hat{\imath}). \tag{8.34}$$

Gain adjustment computation

ϕ is the parameter $\alpha_{\hat{\imath}}$ of channel $\hat{\imath}$. For $\overline{\alpha}_{\hat{\imath}} = \exp(\alpha_{\hat{\imath}})$,

$$\frac{\partial w_{i,f}}{\partial \overline{\alpha}_{\hat{\imath}}} = w_{i,f}\, \chi(i,\hat{\imath}). \tag{8.35}$$

The derivative of the weight versus the log of the gain within a channel is equal to the weight.

8.5 System initialization

DFE training is based on gradient search optimization. Consequently, a good initialization of the trainable parameters is advisable. That is, a reasonable initial setting is essential for the shaping and positioning of all of the filters as well as the classifier prototypes. Concerning the filter shape, each filter was initially set to a Gaussian function even in the I-type case where the filter weights were treated as independent coefficients. Concerning the filter positioning, two initial settings of frequency scaling were considered: the typical psychoacoustics-based scale called the Mel scale (*Mel-scaling*) and the linear scale (*Linear-scaling*). To embody the Mel-scaling, the following approximation equation, as provided by [Zwicker and Terhardt, 1980], was used

$$p(f) = 2595 \log_{10}\left(1 + \frac{f}{700}\right). \tag{8.36}$$

The Linear-scaling was simply implemented as

$$p(f) = f. \tag{8.37}$$

In both the Mel-scaling and the Linear-scaling cases, the filter center frequencies were uniformly spaced on the corresponding frequency scale so as to cover a full range of possible frequency band, which is limited by the Nyquist sampling frequency. Moreover, to avoid discontinuity, the bandwidth of each filter was selected so that adjacent filters crossed at the middle position between the corresponding centers. Gain factors were initialized at value one (1). Note that the same approach have been applied using the bark scale [Biem and Katagiri, 1994].

Fig. 8.4 depicts two example shapes, each with 16 channels, of the filter bank defined above: the top shape for the Mel-scaling and the bottom for the Linear-scaling.

DFE training generates a new filter shape, even though the filter is initially set to the Gaussian function. Clearly, the number of trainable parameters within the G-type is much smaller than that of I-type case. For generalization capabilities, the number of training parameters must be small compared to the number of available training samples. However, the independent weighting case

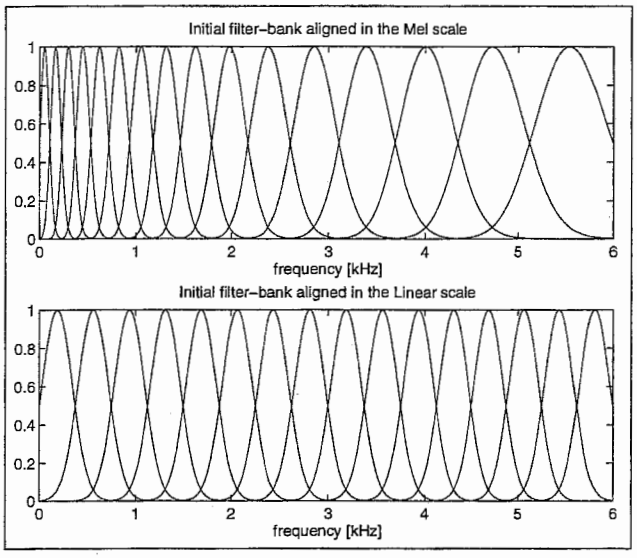


Figure 8.4: Initial filter bank aligned in the Mel scale and the Linear scale.

possesses much larger capability to design different filter shapes, which may lead to a better feature representation. This apparent conflict is a point to be studied in the later comparative experiments, though it may be rather difficult to find a general theoretically-proven solution.

8.6 Vowel Fragment Recognition Task

8.6.1 Task

This section describes the application of the DFE-trained filter bank-based recognizer to the task of recognizing static vowel fragments, each extracted from a Japanese vowel segment by a short Hamming time window; $\mathcal{T} = 1$ for all of the possible fragment patterns. This simple application was chosen mainly because it provided a tractable framework for the analysis of DFE behavior: the spectral characteristics of vowel sounds are rather well known.

The number of vowel classes was five; i.e., /a/, /i/, /u/, /e/, /o/; $M = 5$. A database consisted of 500 phonetically balanced sentences spoken by 5 speakers (3 male and 2 female speakers) in a sound-proof room in order to extract 3,500 vowel fragment samples in total. Thus, the vowel patterns came from various spoken contexts with variability such as speaking rate, speaker variability and coarticulation. Half of the samples were used for training (design), and the other half for testing; i.e., 1,750 patterns for training and 1,750 patterns for testing. Training and testing samples were statistically balanced in terms of speaker and vowel class. That is, 70 samples per speaker and per vowel were used in the training stage as well as the testing stage.

Digital representation of the speech waves was made at 12 kHz sampling rate and at 16 bits. Each fragment pattern was extracted by applying a 21 ms Hamming time window to the center position of a vowel segment, labeled by hand. The fragment was then converted to a 128-dimensional,

FFT-based power spectrum as input to a filter bank feature extractor (FFT was performed with 256 points); $\mathcal{F} = 128$.

8.6.2 Experimental settings

System

Similar to other studies [McDermott and Katagiri, 1994], the number of channels of the filter bank was set to 16 ($I = 16$). Each filter was initially set to the Gaussian function form. The filter bank used in the experiments of this chapter are therefore the same as those illustrated in Fig. 8.4.

Since the input pattern was static, a simplified version of the classifier structure comprising only one sub-phonetic state model for every class was used. However, since the number of prototypes within each state often determines the recognition accuracy and since it is rather difficult to determine the optimal number theoretically, four different settings of the prototype numbers per class was studied; i.e., $1, 3, 5,$ and 7. The prototypes were initialized by using k-means [Duda and Hart, 1973] clustering techniques. The k-means procedure is an iterative clustering techniques, which selects a set of prototypes (each representing a cluster), minimizing the average distortion measure across the whole data set. In the case of an Euclidean distance, it can be viewed as a Maximum Likelihood estimation technique, which uses single Gaussian model with unit variance.

Training

For every size of recognizer (every prescribed number of prototypes per class), the five types of DFE training was conducted, namely, Gc-training, Gb-training, Gg-training, GS-training, and I-type training. It should be noted that in all of these cases, DFE training adjusts the classifier prototypes, as in [McDermott and Katagiri, 1994], as well as the filter bank parameters. Moreover, for comparison purposes, evaluation of the performance of two types of baseline training was performed: *k-means clustering training* that corresponds to the initialization cited above and *MCE/GPD training* that adjusts only the classifier prototypes with MCE/GPD after the initialization.

The gradient-based loss minimization of the DFE training and the MCE/GPD training includes several factors/parameters, such as the training rate and the order of presentation of the design samples. These factors must be experimentally or empirically selected and could result in somewhat different recognition performances. For every training case, several runs of DFE training were carried out by controlling the training factors. However, for presentation clarity, only the best recognition accuracy (lowest recognition error rate) for every training case, are presented.

8.6.3 Results: general comments

A summary of main experimental results are presented in Fig. 8.5 and Fig. 8.6 according to the selection of the frequency scale on which the filter bank is initially designed.

Fig. 8.5 shows recognition error rates for the Mel-scale-based initialization and Fig. 8.6 for the Linear-scale. The error rates are plotted as a function of the number of prototypes per class. Graphs in the figures clearly illustrate the results of the k-means clustering-based training, the MCE/GPD training and the DFE-based training methods.

MCE training and k-means

The k-means clustering training, which is based on the Minimum Distortion criterion, dramatically reduces the error rate as the number of prototypes increases. Careful observation also shows that the accuracy of the k-means clustering training method is affected by the selection of the initial frequency scale: the error rates of the Mel-scaling case are consistently about 5% lower than those of the Linear-scaling.

The difference in error rate is rather clear between the k-means clustering training and the discriminative training, i.e., the MCE/GPD training or the DFE training. That is, with regards to important results over testing data, both the MCE/GPD training and the DFE training achieved about 13% (in the MCE/GPD training of the largest size classifier using the Linear-scaling initialization and in the DFE's I-type training of the smallest size classifier using the Linear-scaling), while the k-means clustering training produced about 19% (in the case of the largest size classifier using the Mel-scaling initialization). These results clearly demonstrate the effect of discriminative training.

A particularly interesting finding over the discriminative training results is that the error rates on testing data are not clearly linked to either the number of prototypes or the selection of frequency scaling (initialization in the DFE cases). This is probably due to the discriminative nature of MCE/GPD or DFE that made the best use of the available prototypes so as to set class boundaries as accurately as possible in the feature vector spaces, each determined by the initial filter bank (for the MCE/GPD case) or the DFE-designed filter bank.

DFE and classical MCE

The differences between classical MCE/GPD and DFE training are not so large, i.e., at most about 2% increase in recognition rate. One possible explanation is that both training methods did not completely reached the desirable (global) minima of the empirical loss surface over the corresponding training data, in light of their adjustment principle, i.e., the gradient-based search optimization. In particular, different from the MCE/GPD training that adjusts homogeneous parameters, i.e., the prototypes, the DFE training adjusts heterogeneous parameters, i.e., the filter bank parameters and the prototypes, which can be difficult to handle in terms of parameter sensitivity control in the adjustment procedure. Another possible explanation is that both method has achieved near-to-the lowest possible error rate on the task.

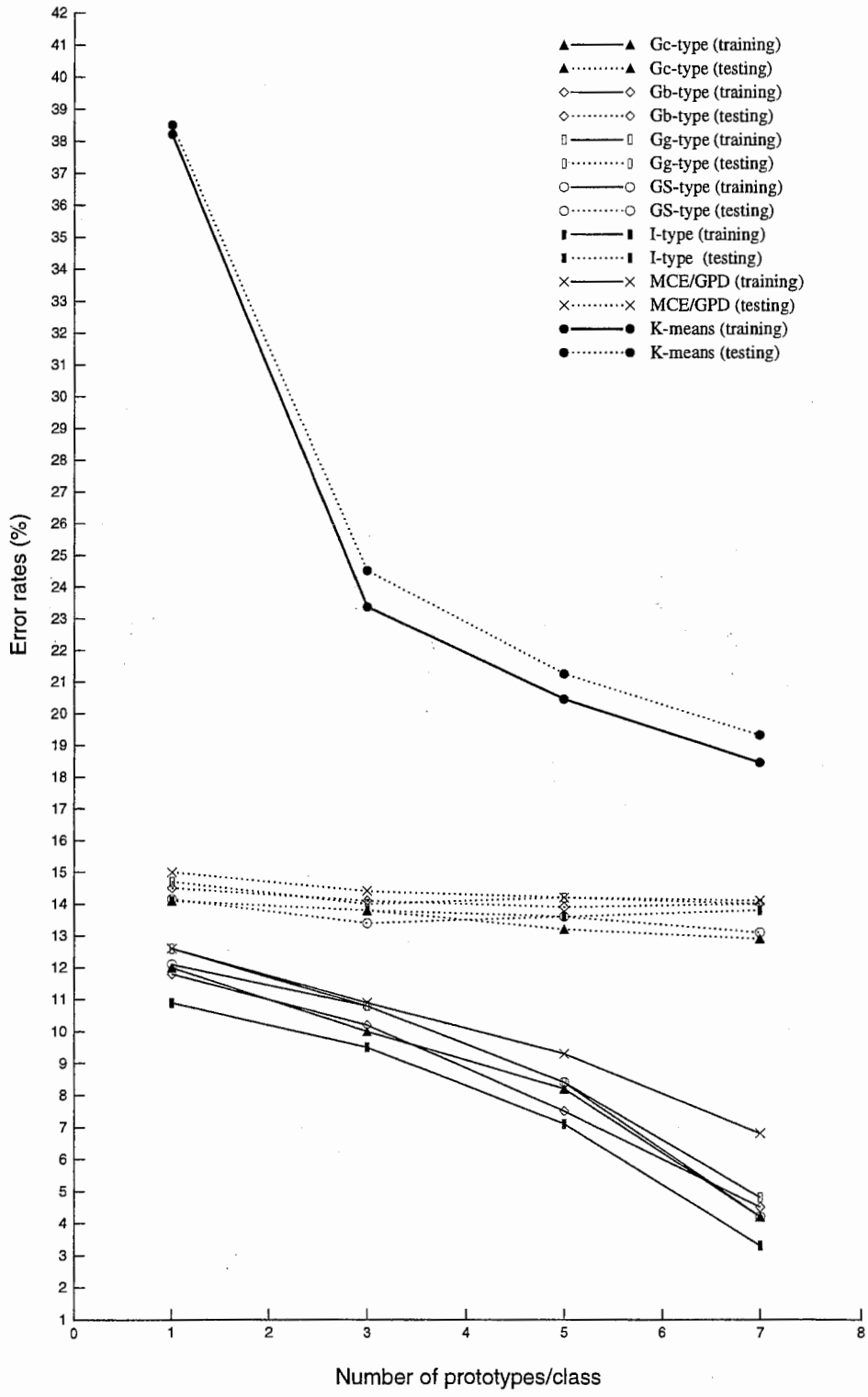


Figure 8.5: Error rates versus classifier size for various configurations when using an initial filter bank aligned on the Mel scale in the vowel recognition task.

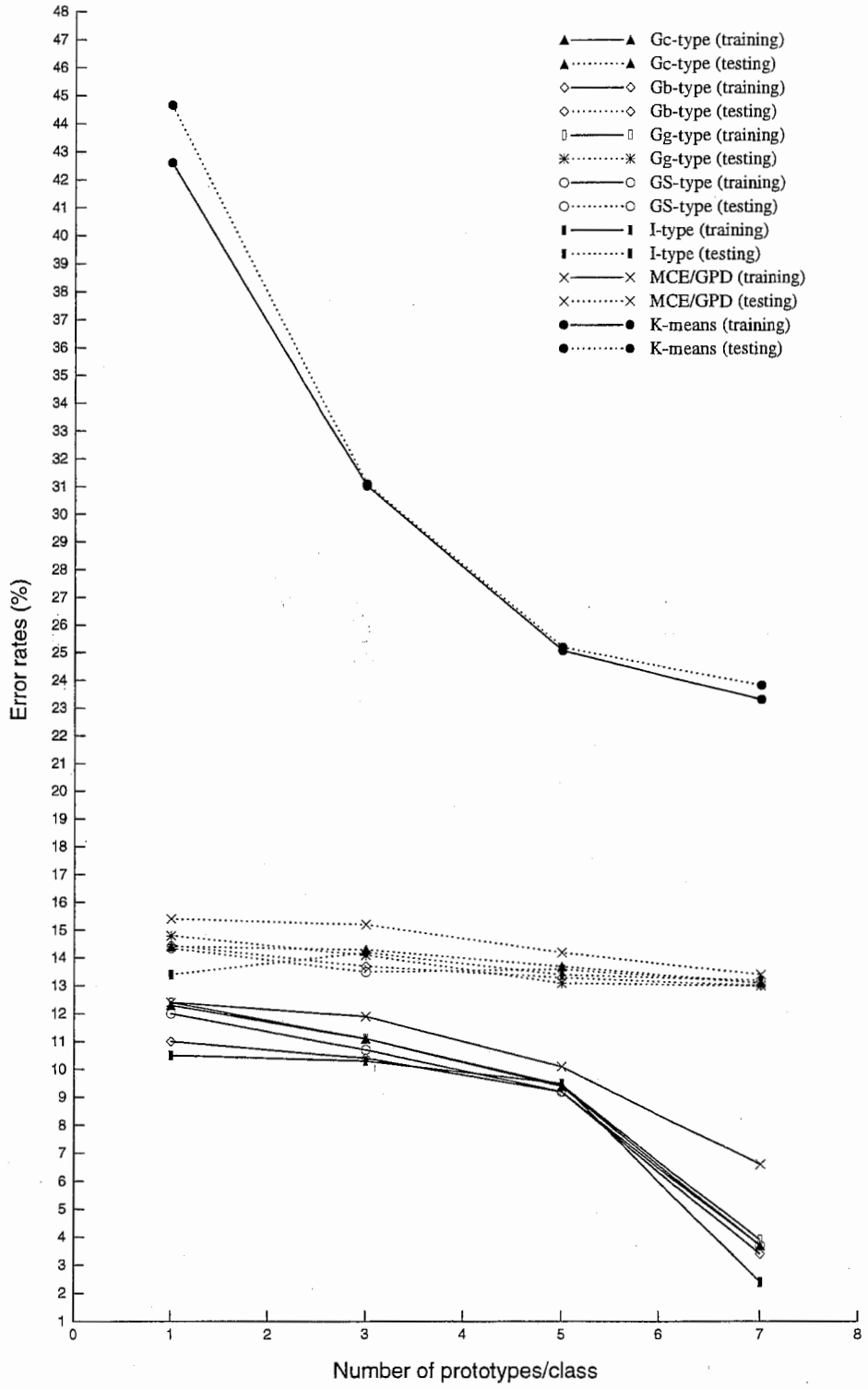


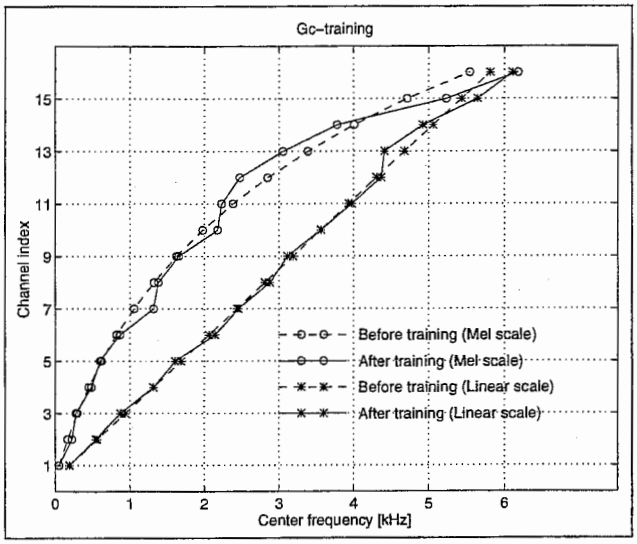
Figure 8.6: Error rates versus classifier size for various configurations when using an initial filter bank aligned on the Linear-frequency scale in the vowel recognition task.

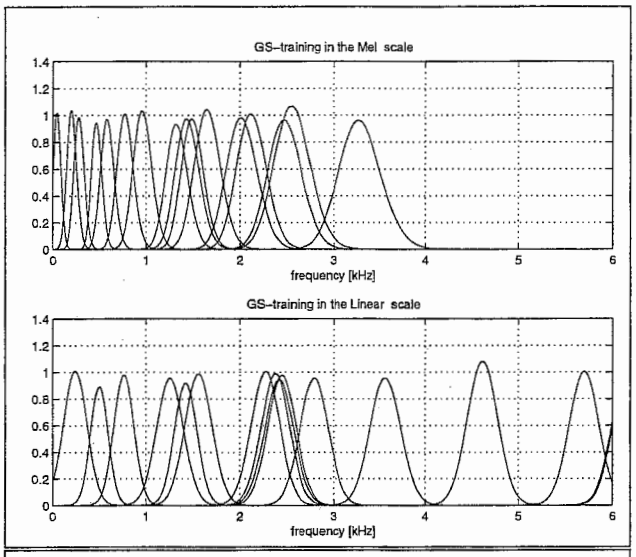
Nevertheless, the results show that DFE is clearly superior to the k-means clustering and even to MCE/GPD training in terms of *training efficiency*, which means making good use of available system parameters in order to achieve as high recognition accuracy as possible. For example, let us compare the smallest Gc-trained (DFE-trained) recognizer case and the largest MCE/GPD-trained recognizer case (Fig. 8.5). The smallest recognizer using 96 trainable parameters (16 filter center frequency parameters and 80 classifier parameters (16×5 prototypes)) achieved an error rate of about 14%, which is slightly better than the rate of the largest MCE/GPD-trained system using 560 trainable parameters ($16 \times 7 \times 5$ prototypes). This efficiency can be easily observed for most of the plots on the dotted (testing data) error rate curves. Indeed, DFE succeeded in reducing the system resource to 1/6, while keeping the high recognition accuracy achieved by the MCE/GPD training. Even if recent progress of hardware technology are alleviating the size limitation in system design, a small size system, added to the limited number of available resources, is an inevitable requirement in system development, which may addressed the DFE approach.

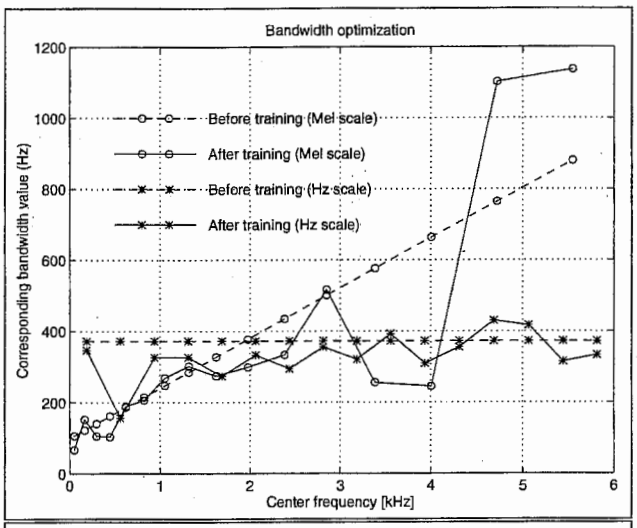
Filter bank analysis

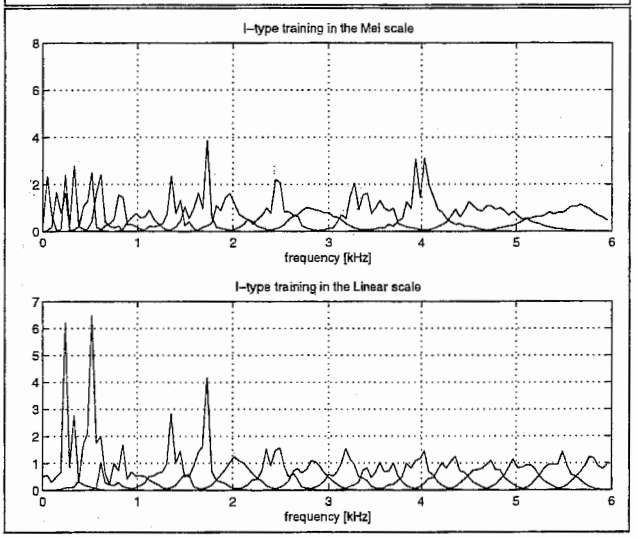
As shown above, DFE is effective in terms of accuracy and training efficiency than the other two training methods in this particular context. One may expect here that this effectiveness is either due to the DFE's data-driven feature representation or to the use of classification information in feature extraction design. In other words, the DFE-trained filter bank is expected to find new reasonable features, which may be different from conventional ones. Therefore, analysis of the relation between the trained filter bank shapes and the input spectrum vectors was carried out. Fig. 8.7 illustrates the DFE-trained filter banks in the case of the smallest classifier of one prototype per class.

This figure is composed of five parts, with each part showing results of both the Mel-scaling and the Linear-scaling initialization cases. In the figure, the top left part presents the results of Gc-training, drawn as a function of filter center frequencies; the bottom left part presents results of Gb-training, focusing on the bandwidths of individual filters. The upper right part depicts the filter banks of which all center frequencies, bandwidths, and gains were simultaneously updated (GS-training); the bottom right part depicts the results of I-type training. All these results do not allow an easy analysis. In fact, due to the high dimension of the data, a simple observation is clearly insufficient for analyzing the details of the training mechanism, even if, as can be observed, the resulting filter banks are affected by their initial configuration. Furthermore, due to the local optimality of training, the results are not guaranteed to be truly optimal (in the sense of global optimality over training data). Nevertheless, through careful observations, it can found that the DFE training successfully updated the filter banks so as to use the class identity information of the spectrum vectors more effectively.









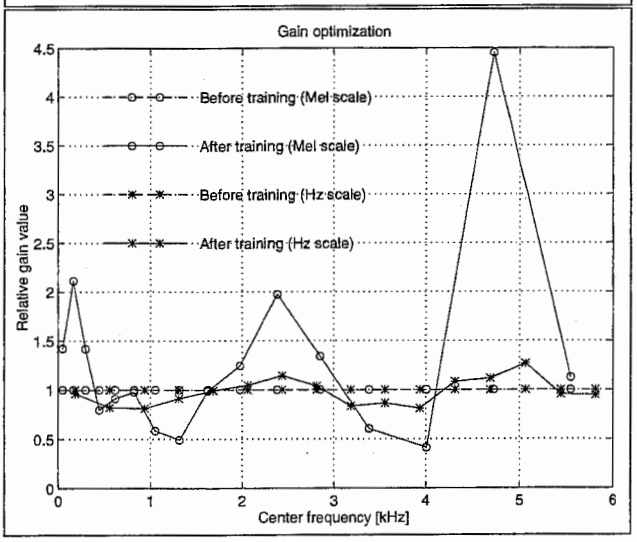


Figure 8.7: Resulting filter banks after DFE optimization of various filter bank parameters in the context of 1 prototype/per vowel.

Let us focus on the results of GS-training using the Mel-scaling initialization (upper-right part of the figure). Most of the filters shift to the region lower than 3 kHz, and the filters that were originally wide are sharpened, increasing the frequency resolution of filter bank outputs. These changes must probably have appeared so that the filter bank could effectively extract features useful for classification, which originally existed in the region. Concerning the error rate over testing data, the GS-training using the Mel-scaling, achieved the lowest error rate of about 14%, within the one prototype per class system structure.

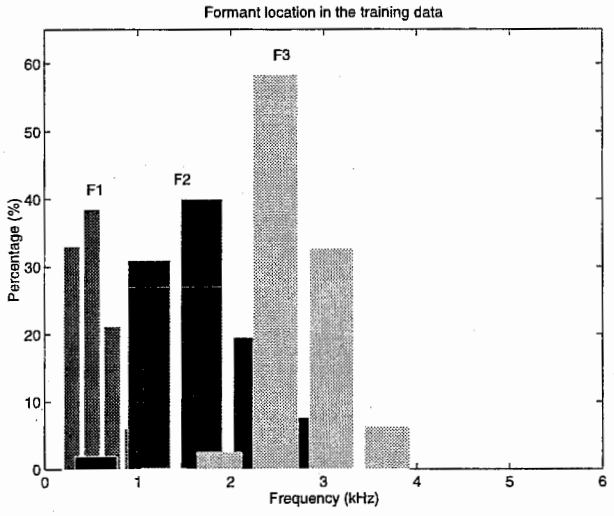


Figure 8.8: Formant location in the training data aiming at vowel recognition.

To verify the physical validity of the GS-trained filter bank, investigation of the formant location of vowel fragments in training data was done. Formants are known to be important for vowel classification. Fig. 8.8 shows a histogram of the lower 3 formants as represented in training data. The formants were estimated by using an LPC-based root finding method, followed by human verification. The values of formants span the 0-4 kHz region, and in particular, most of them exist in the region below 3 kHz. It could be argued that there is a strong correlation between the GS-trained filter bank shapes and the formant distribution. On the other hand, the filter bank in the Gc-training using the Linear-scaling initialization showed only minor changes. Also, in the Mel-scaling case, the Gg-training provided the large gain in the frequency region higher than 4 kHz, where no important formants exist usually. It seems that, in order to represent the class information included in the input data, DFE updated the filter bank feature extractor (together with the classifier), by using the 16 filters as effectively as possible, according to the degree of freedom available within each channel-filter. However, the DFE training may have suffered from sensitivity to training parameters due to the heterogeneous selection of parameters in the filter bank feature extractor and the prototype-based classifier, thus not leading to the best status of the filter bank. Nevertheless, in general, it may be able to argue from Fig. 8.8 that DFE successfully adjusted the filter banks so that they could selectively extract the useful information.

8.6.4 Detailed analysis of the filter bank

Center frequency optimization

The center frequency adjustment tries to find the optimal spacing method, given a fixed bandwidth and a fixed gain. Obviously, the process may produce a non-uniformly spaced filter bank model showing an accumulation of filters in those regions relevant to discrimination. However, a fixed bandwidth may result in non-covered region in spectrum.

The results for an original spacing in Mel and linear frequency domain is shown in Fig. 8.7. Concerning the Mel domain, one can see that main changes occur in the F2 domain where the center frequency of the 7^{th} bandpass filter has moved to the location of the 8^{th} filter, resulting into an emphasis of the F2 region. Most filters in the F3 domain has also moved closer to emphasize the F3 region. Center frequencies of the region above 4 kHz have moved to higher value which means a de-emphasis of this region. No changes in the F1 region. This is consistent with the performance: 12.0%/14.1% error rate on the training/testing set compared to 12.6%/15.0% achieved by the baseline recognizer.

The linear frequency scale did not show significant changes as can be seen in Fig 8.7. The frequency has been sorted in ascending values after optimization. The filter bank has remained rather linear with small shift of frequency in the 1.5-2 kHz region and the small emphasis of the 4.5 kHz region by the 13^{th} filter getting closer the 12^{th} filter. This is consistent with the performance obtained: similar to the baseline in the training set with rather better generalization in the testing set.

Bandwidth optimization

Results for bandwidth optimization are shown in Fig 8.7. A uniformly spaced bandwidth model in the Mel domain resulted in a model showing larger bandwidths in the higher frequency regions. While, most filters in the region below 1.5 kHz remained relatively stable, the region above 3 kHz displays the most noticeable changes: the last 2 filters have seen an increase in their bandwidths and the 12^{th} and 14^{th} filter have seen a decrease in their bandwidth values. The model has shown a rather poor generalization 14.4% on the testing set in contrast to the rather good performance on the training set: 11.9%. This may be explained by the fact that a narrower filter bank signifies a selection of single frequencies which may no lead to accurate generalization.

In the linear-scale case, when looking at the bandwidth value versus the center frequency of the channel in Fig 8.7, one can notice that most bandwidth values have decreased to lower values. As said before, the process may result in an emphasis of center frequency in detriment to regions. This may explain the relatively high recognition in rate in the training set (11.0 %) while exhibiting a poor generalization (14.4%). The same phenomena was noticed in Mel-scale based bandwidth optimization task.

Gain factor optimization

Looking at Fig 8.7, in the Mel-scale domain, the gain optimization has resulted in an significant t emphasis of the energies of the filter in the region above 4 kHz as well a rather noticeable emphasis of the 2-3 kHz region. However, the performance are rather close to the baseline method. This could be explained by the use of logarithm, which compresses the output of the filter bank. Consequently, only higher gain changes have any noticable effect.

The linear-frequency-gain optimization has resulted into an emphasis of 2-3 kHz region and 4-5 kHz region. Although, as in the Mel scale, the modification are relatively small: a value of 1.27 for the 14^{th} filter is the maximum. As in the Mel-scale, this may explain the fact the model show similar result in the training set with the baseline system (12.4%) while proving a slightly better generalization (14.8%).

Weighting optimization

The "weighting" optimization (I-type training) task resulted into sharper filters which tend to select single optimum frequencies instead of spectrally meaningful regions for both frequency scale. In particular, the harmonic structure of the spectrum is exposed.

In the Mel-scale domain, the 9^{th} filter which did not show any significant changes in the previous optimization cases, seems to single out its center frequency. The 13^{th} filter which has shown a decrease in its bandwidth as well its gain in the previous optimization cases, now exhibits an emphasis on two single frequencies. Thus, the process has resulted into a relatively poor generalization: similar result was obtained on the testing set with the center frequency optimization case while showing a good performance on the training set: 10.9%.

Using linear scale, one can observe that the Gaussian form of the frequency response of the filters have been modified into wider filters with emphasis on frequencies in the 0-7 kHz and 1-2 kHz region (F1 region). This new set of filters seems to be more efficient filter bank model according to its performance: 10.5% error rates on the training set and 13.4%error rate on the testing set. This results contrast with the Mel-scale task, in which weighting optimization have shown a relatively poor generalization.

Center, bandwidth and gain simultaneous optimization

Optimizing center, bandwidth, gain at the same time may help provide a globally efficient model. Again, this is the GS-training scheme.

Fig. 8.7 shows the adjusted filter bank model using Mel-Scale. The frequency spacing has been modified to produce a model which spans a smaller frequency range : 0-4 kHz. Moreover, the model clearly puts emphasis on the F2 and F3 regions by appropriate shift of center frequencies while showing a decrease in the bandwidth and a rather stable gain values. This may explain the relative good performance achieved by the model: 12.1%/14.1% on the training/testing set.

In the linear-scale, the spacing is rather different from the spacing obtained when optimizing the independent center frequency optimization case. One can notice the accumulation of center frequencies in 1.5 kHz region (F2 domain) and 2-3 kHz region (F3 domain). Furthermore, most filters have become narrower. The center frequencies of the last two filters have move to higher value. This signifies that the two filters do not cover the frequency range of interest. It could be said that the new filter bank model has a lower number of filters: 14 precisely.

GS-training of filters bank parameters, significantly affects the spacing of the filter model as well the spanned frequency range. On average, GS-optimization appear to perform better than when optimizing each parameter independently.

8.6.5 Classifier and filter bank interaction analysis

The nature of the interaction between the filter bank extractor and the classifier and the way both modules collaborates in order to reduce the errors at the output of the recognizer may be grasped by studying the modification of the filter bank according to the number of parameters of the classifier. A richer classifier may be sufficient in achieving satisfying performance. This signifies less need to alter the original filter bank model. An estimate of the filter bank modification can be given by calculating for each parameter of the filter bank model $(\alpha_i, \beta_i, \gamma_i)$, the gap between the modified value and the original value of the parameter and averaging these gaps across the number of parameters in the model. This estimate is referred to as the *degree of filter bank modification* (FBMOD)).

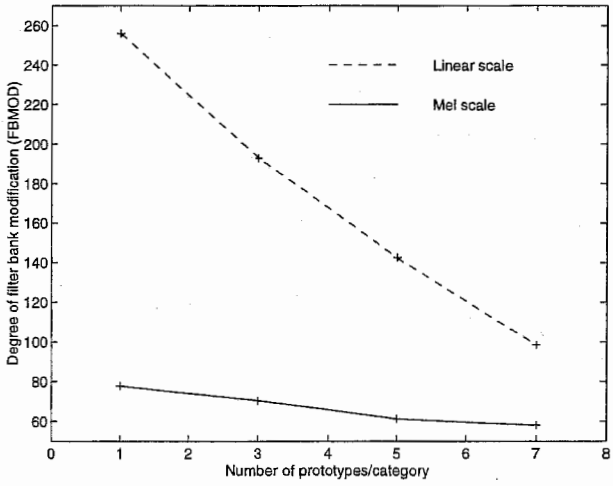


Figure 8.9: Filter bank modification versus the number of prototypes in the classifier. The degree of modification (FBMOD) is a equal to the average distance across all parameter of the model to the original model.

Fig. 8.9 shows how the FBMOD varies as function of the number of prototypes per category in the classifier when optimizing all parameters at the same time, in the Mel scale and the linear

scale. As expected, the FBMOD decreases when the number of prototypes increases. The Mel scale-based filter bank model has required less modifications than the linear scale based one. These results show how sensitive DFE is to the original design of the filter bank model.

8.7 Directory Assistance Task

8.7.1 Task

The framework is the design of a system which recognizes Japanese persons' names and forwards calls to staff members within the ATR laboratory. Thus, the system is intended to act as an automatic switching operator in charge of accepting spoken names and making connection to the recognized persons. The point of interest is to improve the speech recognition accuracy of the system by applying the DFE training and test the DFE ability on a more challenging and practical task.

Data were automatically collected in an office environment by a system that periodically called staff members to repeat 5 randomly selected names. Each name utterance was therefore spoken in the isolated word mode. The process produced 684 utterances in total from 47 speakers (3/4 of whom are male). 570 utterances were used for training and 114 for testing. The utterances were phoneme-labeled using their transcriptions by a segmental k-means clustering procedure [Juang and Rabiner, 1991]. The labels created by this process were then verified through human correction. The segmental k-means procedure is an iterative procedure which realizes an optimal segmentation, in minimum distortion sense, of a given body of training utterance. The segmentation is achieved by finding the optimal state by the Dynamic Programming and, after clustering the speech frames along this optimal path, estimate new models by performing clustering within each state. It is, thus Maximum Likelihood Estimation procedure (see Chapter 9 for further details on the segmental k-means procedure).

The speech signal, coming from the telephone transmitter, was digitized at 8 kHz sampling rate and at 16 bits. To compute the acoustic vectors (inputs to the filter bank feature extractor), a Hamming window of 21 ms was shifted over an input speech utterance every 5 ms. At each window position, a segmented utterance was converted to its corresponding 128-dimensional FFT-based power spectrum-vector ($\mathcal{F} = 128$).

8.7.2 Experimental settings

system

In this experiment too, 16 channel, Gaussian-shaped filter bank; $I = 16$ was used. The filters were placed so as to cover the 0-4 kHz frequency range, and their gain values each were set to one (1).

For classification, 26 context-independent phoneme models was considered. This phonemic set corresponds to the 5 Japanese vowels, 20 consonants (/b/, /ch/, /d/, /f/, /g/, /h/, /j/, /k/, /m/, /n/, /N/, /p/, /r/, /s/, /sh/, /t/, /ts/, /w/, /y/, /z/) and silence. Since the input utterances

are essentially dynamic, the length of input vector sequence is changeable; $\mathcal{T}$ depends on each input word. Unlike the work in the vowel task, which was selected for a detailed comparison study, pre-selected settings for the number of states and the number of prototypes was used; That is, each phoneme model consisted of 3 states, each being associated with 2 prototypes.

Training

As in the previous task, 7 types of training was conducted. 1) baseline training using the segmental k-means clustering, which is the Minimum Distortion Estimation, 2) baseline training using the MCE/GPD training, 3) Gc-training, 4) Gb-training, 5) Gg-training, 6) GS-training, and 7) I-type training. However, for simplicity, only the Mel-scaling initialization was used in this task. In all of the training cases, the filters were therefore initially aligned as shown in the top illustration of Fig. 8.4, but covering the 0-4kHz bandwidth. In the baseline MCE/GPD training, the filter bank was fixed and only the classifier prototypes were re-trained. In the 5 DFE-based training cases, both the filter bank parameters and the prototypes were adjusted jointly.

8.7.3 Results

Controlling several training factors such as the order of training sample presentation, several training/testing runs were carried out. All training cases, except the GS-training, quite successfully converged to the almost perfect accuracy over training data.

Fig. 8.10 shows the lowest error rates for all training cases, except the GS-training. The GS-training that jointly adjusted the three kinds of parameters of the Gaussian filters failed to converge appropriately, and therefore, its results is not shown in the figure. This non-convergence may be related to the difficulty of selecting right training parameters within gradient-based optimization.

In the figure, except the rather limited result of I-type training, which may be due to the problem of the high number of system parameters (over-learning problem), the results clearly showed the utility of DFE as a whole. All DFE training cases using the G-type filters, i.e., the Gc-training, the Gg-training, and the Gb-training, improved the result of MCE/GPD training, i.e., 7.9 % over testing data, and in particular, the Gg-training achieved 5.3 %. Also, a comparison between the training and testing data results suggests the validity of the DFE-trained filter banks. Over training data, the achieved error rates of DFE were almost the same with that of the MCE/GPD training. However, there is a clearer differences in error rates over testing data between the MCE/GPD and DFE training cases. Then, it may be argued that this phenomenon was due to the effect of the classification-oriented filter bank that was realized by the DFE training. DFE succeeded in designing the feature space that was more relevant (essential) to classification, and thus achieved the lower error rates over the unknown testing data.

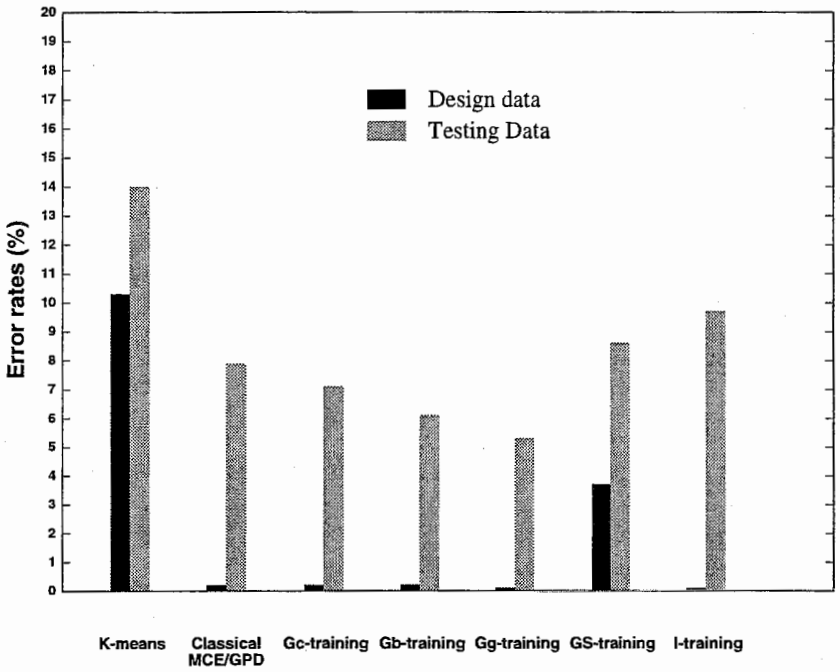


Figure 8.10: Experimental results of the directory assistance task for various adjusted parameters. With a 95% confidence interval the the histogram corresponds to the following error rates. Baseline MLE: 7.9%+− 0.0495. Gc-training: 7.1% +− 0.0471. Gb-training: 6.1% +− 0.0439. Gg-training: 5.3% +− 0.0411. I-type training: 9.7% +− 0.05431.

Filter bank analysis

Fig. 8.11 displays the resulting filter bank model after optimization. One can notice that the optimized model differs from the vowel optimization case. In the first three optimization cases (center, bandwidth and gain), the filter bank model has put an emphasis on higher frequency region. In particular, in the center frequency optimization case, re-spacing occurred only in the 2.5-3.5 kHz region and Gain optimization tends to decrease the energies of the 1-2 kHz region. However, unlike the vowels recognition case, it is quite clear what are the important features here.

It can be observed that, the "weighting optimization" did not out-performed the baseline method. This could be explained by the fact that the use of a richer classifier structure adds to the complexity of the system : antagonist influences are likely to occur when using various type of parameters. For instance, the "weighting" optimization has almost kept the Gaussian form of the frequency-response with slight modifications in the lower frequency region. It could be seen that, the resulting model has changed the spacing of filter: filters have moved to the lower frequency region. This is in contrast with the individual parameter optimization case which rather emphasize the higher frequency region.

These results are not similar to the vowel recognition case and confirm the fact that the relevant

parameter is task-dependent and classifier structure dependent. Also, the initial set of reference vectors, here based on 8 kHz sampling rate, may have an influence in the resulting model.

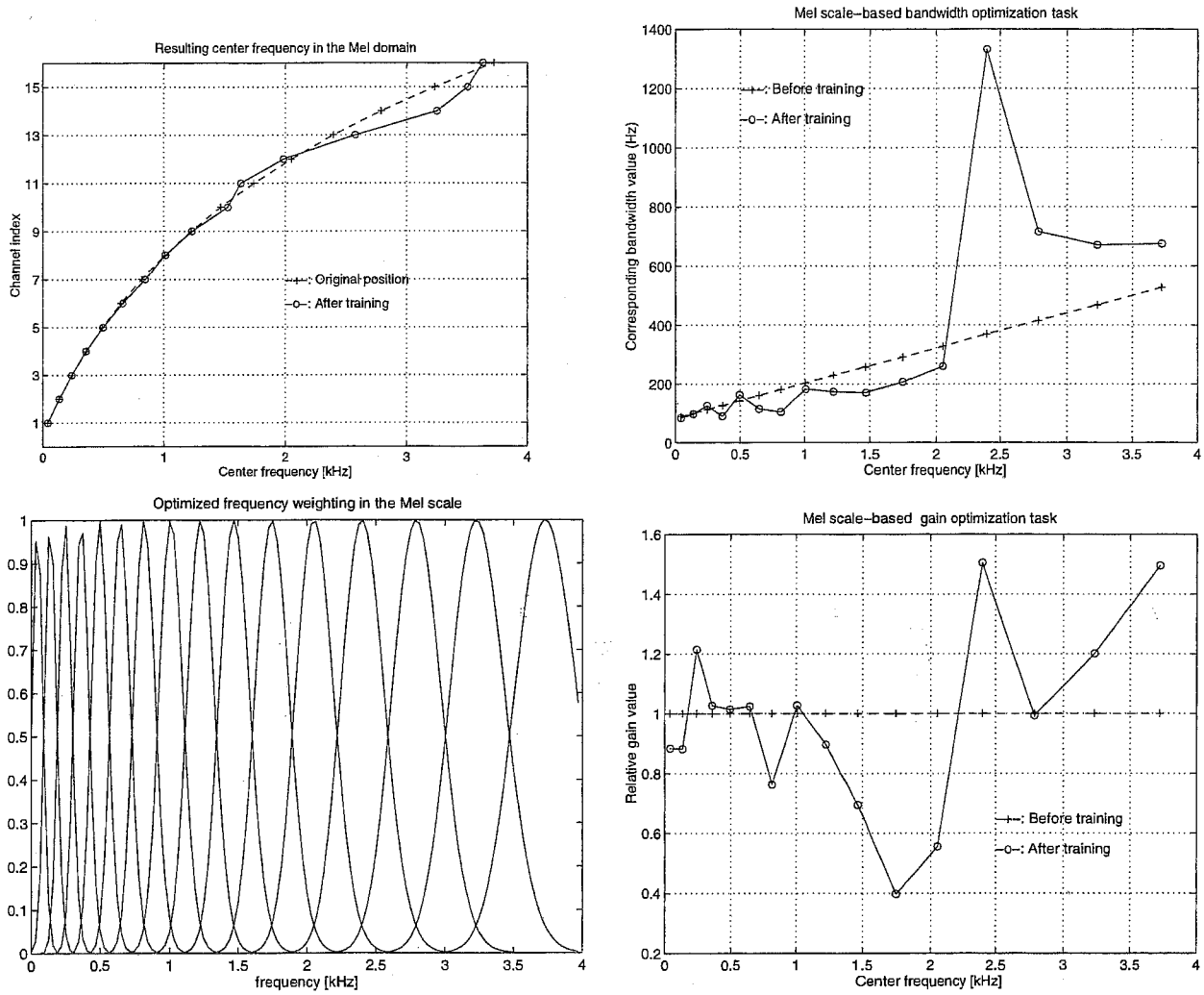


Figure 8.11: Experimental results of Mel-scale based filter bank optimization task in the directory assistance task.

8.8 Conclusion

In this chapter, experimental evaluations of the Discriminative Feature Extraction design method in speech pattern recognition tasks using a particular system structure, which consists of the filter bank feature extractor and a multi-prototype distance classifier with state-transitions, was described. DFE determines all of trainable parameters of the overall recognizer using a single design criterion of reducing a smoothed recognition error counts. First, evaluation of DFE for single vowel frame pattern recognition task was done and detailed analysis of the DFE-trained filter bank within the context of this simple task was carried out. Investigation was performed by controlling experimental settings such as the number of prototypes and different frequency scale initializations, namely the linear and the Mel scale. Observation of the DFE-designed filter banks showed that, in certain cases in which the spacing of the filter bank is a free parameter, the training appropriately focuses on the formant region, which is considered relevant to vowel categorization. Second, the capability of DFE was tested in the more realistic and difficult task, called the ATR directory assistance task, which consisted of recognizing names over the telephone. DFE was compared to the segmental k-means clustering and the MCE/GPD training, and it successfully achieved the lowest word error rate of 5.3 %. However, analysis of the filter is rather difficult in this situation. The results in this chapter enlighten the utility of DFE in the common selection of a filter bank.

Chapter 9

Discriminative Feature Extraction Applied to FFT-based Cepstral Parameters

One must pursue the tree to all its root fibers

-Carl Friedrich Gauss-

In previous chapter described the application of Discriminative Feature Extraction to designing various filter bank parameters aiming at minimum classification error. Here, an extension of this approach is presented for readjusting Mel frequency cepstral coefficients (MFCC), which is the most widespread speech parameterization method in the speech research community. The use of cepstrum is motivated by the fact that it provides a good compactness of information by appropriate decorrelation of features while representing local spectral properties.

9.1 Introduction

Cepstrum coefficients, either based on filter bank or LPC model of speech, constitute the most widely used speech parameterization method. Cepstrum parameterization derives from homomorphic signal processing techniques, which provides a convenient way to separate the influence of the source from the vocal tract in the source/vocal tract model of the speech production. Thus, making use of cepstrum coefficients ensures a good compactness of information by representing with few parameters the general aspect of the estimated speech spectrum. Also, cepstra produce decorrelated features without specific use of data statistics. This is particularly useful in the FFT-based estimation of the speech spectrum, where the original spectrum is first smoothed through a set of overlapping filters, which leads to a high degree of correlation among the components of the filter bank output energies. Thus, the experiments described in the previous chapter bear this shortcoming.

The matrix performing the transformation of the filter bank output energies into cepstral parameters is chosen *a priori*. Consequently, for filter bank-based cepstrum, performance depends on appropriate design of the filter bank. Most filter bank-based cepstrum applications have relied on the perception-based Mel scale (e.g. MFCCs). However, the relation between perceptually-motivated feature extraction and statistical pattern recognition remains unclear. Perceptually-motivated cepstral parameters may not be the optimal features within the framework of statistical speech-pattern recognition.

In this chapter, the study of the previous chapter is extended by designing a filter bank model using a cepstral distance measure at the frame-level [Biem and Katagiri, 1997b; Biem and Katagiri, 1997a]. Again, in a manner similar to the previous chapter, the study was done in three steps. First, a vowel fragment recognition task was carried out with the motivation of analyzing the inherent characteristics of DFE-optimized cepstrum. Second, the target is word recognition using telephone-speech. In this latter study, the focus is on a particular aspect of the DFE training, that is, the level at which training should be performed. For a word-recognition target, the system usually uses a loss, which reflects the error at the string level. However, it was seen in the previous chapter that DFE acts sequentially in the frame by frame mode. Thus, for a DFE-based feature extractor to be efficient, there is the need to consider local-frame errors in the optimization scheme. The second part of this chapter is devoted to this study.

The third part studies various DFE implantation in the context of word-modeling by continuous HMM. Here, the phonetic-based approach is replaced by word-level models. First, a joint MLE/DFE technique is investigated, in which MLE is performed on DFE-based features. Second, an implementation of DFE which embeds dynamic features, in the form of regression coefficients is carried out.

9.2 Filter Bank-based Cepstrum

In filter bank modeling of the speech spectrum, the cepstrum is computed at the output of the filter bank. In a manner similar to the previous chapter, the filter bank model simulated on the DFT domain by weighting of the DFT bins with the magnitude frequency response of the filter. Thus, for a sequence of speech spectral vectors $\boldsymbol{s}_1^{\mathcal{T}} = \{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_t, \ldots, \boldsymbol{s}_{\mathcal{T}}\}$ in which $\boldsymbol{s}_t = [s_{t,1}, \ldots, s_{t,f}, \ldots, s_{t,F}]^{\mathrm{T}}$ is the magnitude spectrum of the frame (short time window position); $s_{t,f}$ represents the f-th element of frame-vector. F is the maximum frequency. As seen in the previous chapter, an I-channel filter bank model transforms each $\boldsymbol{s}_t$ into a lower dimensional vector $\boldsymbol{x}_t = [x_{t,1}, \ldots, x_{t,i}, \ldots, x_{t,I}]^{\mathrm{T}}$ such that an output feature $x_{t,i}$ is the windowed log energy of the i-th channel:

$$x_{t,i} = \log_{10}\left(\sum_{f \in B_i} w_{i,f} s_{t,f}\right) \quad \text{for } i = 1, \ldots, I, \tag{9.1}$$

where B_i represents the channel interval and $w_{i,f}$ the weighting at frequency f provided the i^{th} filter.

From the vector of log energies, the cepstrum vector $\mathbf{c}_t = [c_{t,1}, \ldots, c_{t,q}, \ldots, c_{t,Q}]^{\mathrm{T}}$ is computed via an inverse discrete cosine transform (IDCT). The DCT has its origin in signal interpolation. Various implementations of the DCT are available in the literature. the following implementation is used, as proposed in Mermelstein and Davis [Mermelstein and Davis, 1980]:

$$c_{t,q} = \sum_{i=1}^{I} x_{t,i}\left(\frac{q\pi}{I}\left(i - \frac{1}{2}\right)\right), \tag{9.2}$$

for $q = 1, \ldots, Q$, where Q is the number of cepstral coefficients. In this chapter, the cepstrum vector $\mathbf{c}_t$ is the input feature to the classifier. In cepstral terminology, q is referred to as the quefrency (see Chapter 7). The inverse DCT implementation of (9.2) can be viewed as a linear transformation of filter bank output energies, in which the transformation matrix is fixed. The above definition of cepstrum is somewhat different from the standard definition of the cepstrum as described in Chapter 7 (see also [Rabiner and Schafer, 1978] for detail). That is, the cepstrum is computed by performing the inverse Fourier Transform on the log spectrum. However, implementation of the cepstrum as displayed in (9.2) is the standard approach in speech recognition, and is believed to convey the significant information needed in speech application task. The cepstral production process is labeled as $\mathcal{F}(\cdot;\Theta)$ where Θ is a set of trainable control parameters. Thus, for a sequence of spectral frames, $\boldsymbol{s}_1^{\mathcal{T}}$, the feature extractor output is a sequence of cepstral vectors $\mathbf{c}_1^{\mathcal{T}} = \mathcal{F}(\boldsymbol{s}_1^{\mathcal{T}};\Theta)$.

The type of cepstrum depends on the underlying filter bank since the DCT is chosen *a priori*. Various standard cepstral parameterization methods are derived by appropriate choice of the frequency scale which determines the particular filter bank setting. For instance, using a linear frequency scale, gives birth to linear filter bank cepstral coefficients. A Bark scale produces Bark filter bank cepstrum coefficients (BFCC) and a Mel scale mapping generates Mel cepstral coefficients (MFCC) [Mermelstein and Davis, 1980]. All these cepstral parameters enjoy a widespread use used in speech recognition.

In contrast to filter bank output energies, cepstral coefficients displays less correlation. Fig. 9.1 displays the correlation matrix of the Mel-based filter bank output energies and MFCC taken from a database of vowel sounds. It can be seen that MFCC are less correlated than filter bank output energies and thus carries in a compact form the signal properties. This property of the cepstrum may explain its widespread use in HMM-based speech recognition system. The cepstral coefficient can be seen as approximating a principal component analysis transform [Pols, 1966], without the need to compute the covariance matrix.

Furthermore, particular quefrency terms carry different type of information. For instance, the low quefrency terms correspond to short term correlation in the signal (vocal tract shape, spectrum envelope) and maxima of higher quefrency term correspond to long-term correlation of the signal, such as pitch. In speech recognition, usually, only lower quefrency terms are taken into consideration. That is, $Q < I$, in the definition given in (9.2). A trade-off takes place concerning the optimal values of Q and I. Usually, higher values of I are chosen so as to have a good frequency resolution of the spectrum. Another property of the cepstrum is that, due to use of a non-linear operator (logarithm function), the cepstrum is believed to be robust to certain type of noise and signal distortion [Rabiner and Schafer, 1978].

9.3 DFE-based Design

As said before, the type of cepstral parameter depends on the setting of the filter bank. In particular, filter bank designed on the Mel-scale, that is, MFCC have become the standard cepstral-based feature parameterization in most speech recognition systems, due to the good results obtained by Mermelstein and Davis [Mermelstein and Davis, 1980] in a word recognition task. Other studies have shown conflicting results. Leung [Leung *et al.*, 1993] reported rather good results for BFCC in a phonetic classification task for clean and telephone speech. However, the fact remains that MFCC seem to be the standard speech parameterization method.

Improvement of the cepstral representation can be achieved by designing the filter bank using the DFE approach, in a manner similar to Chapter 8. Again, for practical requirement and commodity of gradient-based optimization, as was done in Chapter 8, the magnitude response of the filter $w_{i,f}$ in i^{th} channel is constrained to a Gaussian-form:

$$w_{i,f} = \alpha_i \exp\left[-\beta_i \left\{p(\gamma_i) - p(f)\right\}^2\right], \quad \text{for } i = 1, \cdots, I, \tag{9.3}$$

for $i = 1, \ldots, I$, where the trainable parameters $\beta_i > 0$ and γ_i determine bandwidth and center frequency, and α_i is the trainable "gain" parameter in the i-th channel. $p(f)$ maps the linear frequency f onto the perceptual representation. Usually, MFCC are implemented using triangular filter uniformly spaced on the Mel frequency scale. However, the use of Gaussian filters does not show a fundamental difference in the cepstral representation. The block diagram of cepstrum optimization by DFE is shown in Fig. 9.2.

DFE-based optimized filter bank cepstrum coefficients (DFCC) are designed by appropriate

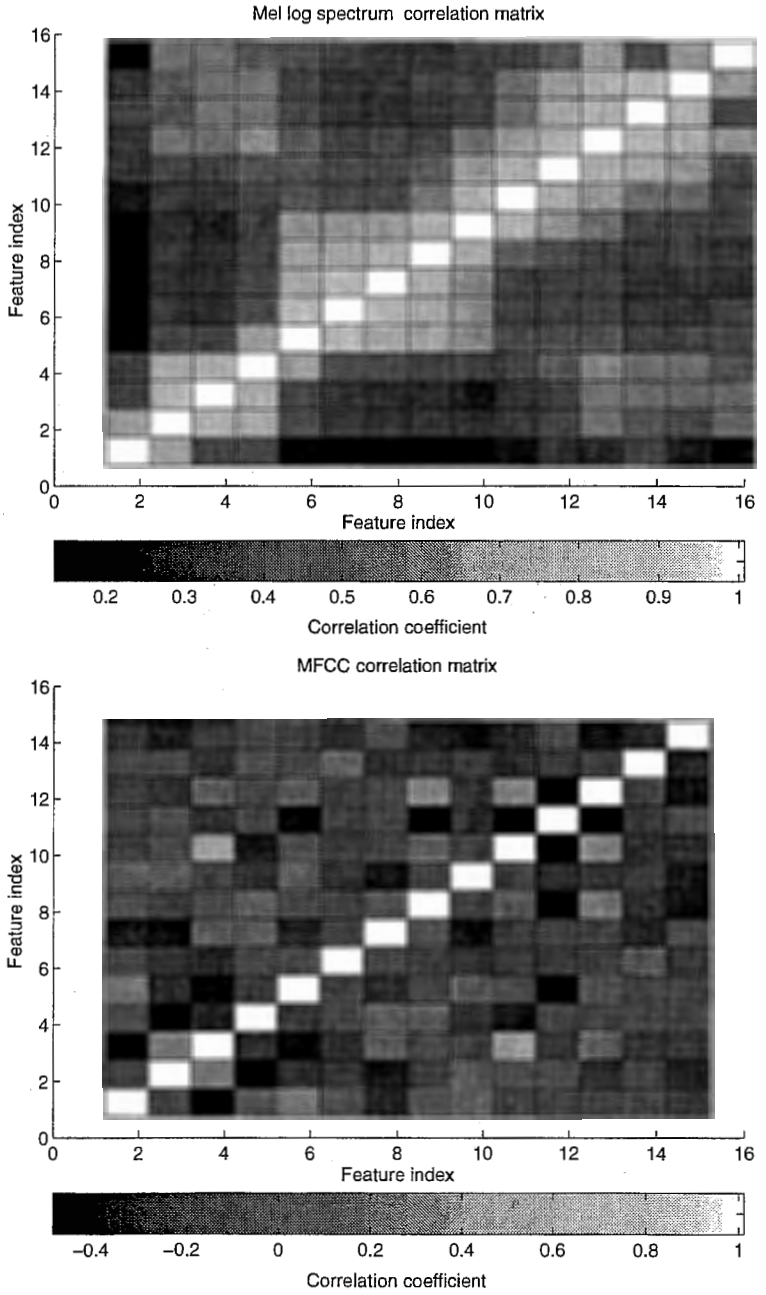


Figure 9.1: Top: Mel log spectrum correlation matrix. Bottom: MFCC correlation matrix.

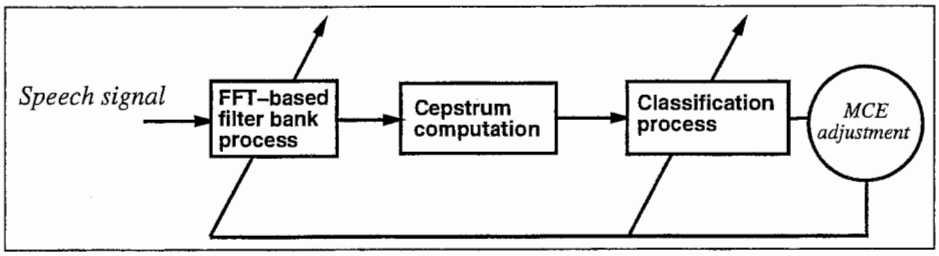


Figure 9.2: Block diagram of DFE-based cepstrum optimization.

optimization of center frequency, bandwidth and gain. To find the relevant parameter and reduce the complexity thereof, each parameter could be adjusted independently while others are fixed. Thus, center frequency-optimized cepstrum coefficients (C-DFCC), bandwidth-optimized DFCC (B-DFCC), gain-optimized DFCC (G-DFCC) and weighting-optimized DFCC (W-DFCC) could be designed by such a method. Here the term "weighting" refers to optimizing each frequency weight without keeping the Gaussian constraint. For generating a globally efficient model, a simultaneous optimization of center frequency, bandwidth and gain (S-DFCC) could be carried out.

9.4 Classifier Structure

In the experiment described in this chapter, two types of classifier were used: the PBMEC, which was described in Chapter 8 and continuous hidden Markov models (HMMs). Here, the HMM structure is described as applied within the DFE framework. See (8.3.1) for more detailed explanations concerning PBMEC. Again, the main difference, between HMM and PBMEC is the use of the probability density function as "similarity measure" to a state of the system added to a probabilistic transition from one state to another.

A category (phoneme/word/sentence) is represented as a string of phonetic models, in which each phonetic model consists of a concatenation of sub-phonemic states. However, in the letter recognition experiment later described, an HMM is assigned to each letter.

Let $\mathbf{c}_1^{\mathcal{T}} = \{\mathbf{c}_1, \ldots, \mathbf{c}_{\mathcal{T}}\}$ be a sequence of cepstral vectors belonging to class C_k. Given a cepstral vector $\mathbf{c}_t$, its "distance", in the HMM framework to state ψ is

$$D_\psi(\mathbf{c}_t) = b_\psi(\mathbf{c}_t) = \sum_{n=1}^{N_\psi} m_{\psi,n} \mathcal{N}(\mathbf{c}_t, \boldsymbol{\mu}_{\psi,n}, \boldsymbol{\Sigma}_{\psi,n}), \tag{9.4}$$

where $\boldsymbol{\mu}_{\psi,n}$ is the n-th means in state ψ and N_ψ is the number of mixture components in the state. $\boldsymbol{\Sigma}_{\psi,n}$ is the covariance matrix associate with the n-th mixture within the state ψ. The set of $m_{\psi,n}$ are the mixing weights satisfying the constraint

$$\sum_{n=1}^{N_\psi} m_{\psi,n} = 1. \tag{9.5}$$

$\mathcal{N}(\mathbf{c}_t, \boldsymbol{\mu}_{\psi,n}, \boldsymbol{\Sigma}_{\psi,n})$ is the well-known multi-variate Gaussian density defined as,

$$\mathcal{N}(\mathbf{c}_t, \boldsymbol{\mu}_{\psi,n}, \boldsymbol{\Sigma}_{\psi,n}) = \frac{1}{(2\pi)^{\frac{q}{2}}|\boldsymbol{\Sigma}_{\psi,n}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{c}_t - \boldsymbol{\mu}_{\psi,n})^{\mathrm{T}} \boldsymbol{\Sigma}_{\psi,n}{}^{-1}(\mathbf{c}_t - \boldsymbol{\mu}_{\psi,n})\right). \tag{9.6}$$

Let $\Psi^j = \{\psi_1^j, \psi_2^j, ..., \psi_{\mathcal{T}}^j\}$, be the *best* sequence of states found by the Viterbi procedure for category C_j and ψ_t^j corresponds to the state occupied by the cepstrum-vector $\mathbf{c}_t$ along this path. The probability of the model generating this sequence, given the path Ψ^j, is defined in terms of the observation probabilities $b_\psi(\mathbf{c}_t)$ and the state-to-state transition probabilities a_{ij}. That is,

$$\Pr(\mathbf{c}_1^{\mathcal{T}}; \Lambda|\Psi^j) = \prod_{t=1}^{\mathcal{T}} a_{\psi_{t-1}^j \psi_t^j} b_{\psi_t^j}(\mathbf{c}_t). \tag{9.7}$$

The discrimination measure corresponding to a category C_j is simply the log of the likelihood along the path, which gives,

$$\begin{aligned} g_j(\mathbf{c}_1^{\mathcal{T}}; \Lambda) &= \log(\Pr(\mathbf{c}_1^{\mathcal{T}}; \Lambda|\Psi^j)) & (9.8)\\ &= \sum_{t=1}^{\mathcal{T}} \log a_{\psi_{t-1}^j \psi_t^j} + \sum_{t=1}^{T} \log b_{\psi_t^j}(\mathbf{c}_t). & (9.9) \end{aligned}$$

The discrimination function $g_k(\mathbf{c}_1^{\mathcal{T}}; \Lambda)$ of the correct category is simply the log likelihood along the Viterbi path, given the transcription.

9.4.1 Segmental k-means for ML estimation

It can be seen that the above formalism does not consider the likelihood of the model but rather the likelihood along the best path as found by the Viterbi algorithm. This is a simplification scheme which may be justified by the observation that, in practice, computing the likelihood along the best path approximates well the overall likelihood of the model.

Within this Chapter, the discriminant function of (9.9) was also used for Maximum Likelihood Estimation, instead of computing the Likelihood over all path of the HMM. That is, MLE is implemented through the segmental k-means algorithm [Juang and Rabiner, 1991].

The usual EM (or Baum-Welch) procedure estimates Λ at each iteration so as to maximize

$$\log(\Pr(\mathbf{c}_1^{\mathcal{T}}; \Lambda)).$$

This maximization is a hill-climbing process which leads to a local optimum. Let Ψ be any state sequence along the HMM. The segmental k-means focuses on maximizing

$$\max_{\Psi} \log(\Pr(\mathbf{c}_1^{\mathcal{T}}; \Lambda|\Psi))$$

relatively to Λ. Thus, the segmental k-means includes two steps: a segmentation step for finding the optimal state sequence and an optimization step for optimizing the likelihood along the optimal state sequence. The segmental k-means can be viewed as an extension of Viterbi training [Bahl *et al.*, 1983].

The EM algorithm, through the Baum-well instantiation and the segmental k-means are both optimal in the sense there are proof of convergence for both [Rabiner and Juang, 1993]. The segmental k-means, however, is more simpler with lesser computational difficulties. Consequently, it is the method used here ML estimation.

9.4.2 Structure-dependent misclassification measure

In the framework of PBMEC, the misclassification measure is defined as the simple difference,

$$d_k\left(\mathcal{F}\left(\mathbf{s}_1^{\mathcal{T}};\Theta\right);\Lambda\right) = d_k\left(\mathbf{c}_1^{\mathcal{T}};\Lambda\right) = g_k\left(\mathbf{c}_1^{\mathcal{T}};\Lambda\right) - \left\{ \frac{1}{M-1}\sum_{j\neq k}^{M} g_j\left(\mathbf{c}_1^{\mathcal{T}};\Lambda\right)^{-\xi} \right\}^{-\frac{1}{\xi}}, \tag{9.10}$$

Again, ξ is a positive number which controls the relative contribution of the classes considered. In this chapter, $\xi \to \infty$, which is equivalent to implementing a strict MCE scheme. In the case of HMM, as suggested by [Chou *et al.*, 1992], the misclassification measure takes the form

$$d_k\left(\mathcal{F}\left(\mathbf{s}_1^{\mathcal{T}};\Theta\right);\Lambda\right) = -g_k\left(\mathbf{c}_1^{\mathcal{T}};\Lambda\right) + \log\left\{ \frac{1}{M-1}\sum_{j\neq k}^{M} e^{g_j\left(\mathbf{c}_1^{\mathcal{T}};\Lambda\right)\xi} \right\}^{\frac{1}{\xi}}. \tag{9.11}$$

In this later case, the anti-discrimination function is defined as

$$g_{\overline{k}}(\mathbf{c}_1^{\mathcal{T}};\Phi) = \log\left\{ \frac{1}{M-1}\sum_{j\neq k}^{M} e^{g_j\left(\mathbf{c}_1^{\mathcal{T}};\Lambda\right)\xi} \right\}^{\frac{1}{\xi}}. \tag{9.12}$$

9.5 DFE Optimization

An important issue in phonetic-based word or sentence recognition is the level at which training should be performed (string-level, phoneme-level or frame-level), especially when labeling information is not available. A string-level training will ensure that the correct string (i.e., a list of phonemes) displays the optimum discriminant function, regardless of the error that may occurs for individual phonemes composing the string. String-level training is the natural level of optimization since it is closely related to the target task (word/sentence recognition).

Thus, in the classical MCE application to word recognition [Rainton and Sagayama, 1992], the error is defined for unrecognized string, even if some frames are mis-aligned by the DP process. However, DFE acts in a frame-by-frame mode, thus processing sequentially the incoming frames of an utterance. This means that local frame-errors may be of importance within the DFE process. Those frame-errors are not provided in a direct manner by the string-level optimization scheme, which focuses on the final score of the string.

Clearly, the filter bank should produce cepstrum values that are close to the corresponding model state, given the current frame. Thus, a frame-level optimization of the overall recognizer, was here investigated, along the line given in [Chen and Soong, 1994]. That is, the filter bank was optimized for each frame that causes a deviation from the correct path.

9.5.1 String-level optimization

For a pattern $\mathbf{c}_1^{\mathcal{T}}$ of category C_k, the normalized misclassification measure $d_k(\mathbf{c}_1^{\mathcal{T}};\Phi)$ which reflects the overall string error, for instance using PBMEC, is defined as

$$d_k(\mathbf{c}_1^{\mathcal{T}};\Phi) = 1 - \frac{\left\{ \frac{1}{M-1} \sum_{j \neq k}^{M} g_j(\mathbf{c}_1^{\mathcal{T}};\Phi)^{-\xi} \right\}^{-\frac{1}{\xi}}}{g_k(\mathbf{c}_1^{\mathcal{T}};\Phi)}, \tag{9.13}$$

where ξ is a positive number which controls the relative contribution of the classes considered. Theoretically, M denotes the total number of categories (the vocabulary size for a word recognition task or the number of potential sentences in a sentence recognition framework). Usually, for large task, M refers to the best competing categories are found by the N-best Viterbi decoding [Chou *et al.*, 1993].

The reader should remember that for word or sentence recognition, an utterance is modeled as a concatenation of phonemic models. Thus, the misclassification measure above considers the error at the string-level since the discrimination function is the result of a summation of local scores along the best Viterbi path.

The misclassification measure can be written in the more general form

$$d_C(\mathbf{c}_1^{\mathcal{T}};\Phi) = 1 - \frac{g_{\overline{k}}(\mathbf{c}_1^{\mathcal{T}};\Phi)}{g_k(\mathbf{c}_1^{\mathcal{T}};\Phi)}, \tag{9.14}$$

where $g_{\overline{k}}(\mathbf{c}_1^{\mathcal{T}};\Phi)$ is defined in (8.13) for PBMEC and in (9.12) for HMM. Again, $\xi \rightarrow \infty$, which, are said previously, is equivalent to implementing a strict MCE scheme. In this case, the gradient of the string-level loss is given by

$$\frac{\partial \ell(\mathbf{c}_1^{\mathcal{T}};\Phi)}{\partial \phi} = \ell'(d_k(\mathbf{c}_1^{\mathcal{T}};\Phi)) \sum_{j=k, j=\overline{k}} \sum_{t=1}^{\mathcal{T}} \frac{\partial d_k(\mathbf{c}_1^{\mathcal{T}};\Phi)}{\partial D_{\psi_t^j}(\mathbf{c}_t)} \frac{\partial D_{\psi_t^j}(\mathbf{c}_t)}{\partial \phi}. \tag{9.15}$$

The above adjustment only considers two competing categories: the correct category and the recognized category.

As already stated, the advantage of using a normalized misclassification measure is that the interval spanned by the misclassification measure value is much more narrower, thus, enabling efficient learning by a classical sigmoid loss. The main drawback is that the normalized misclassification measure introduces an asymmetry in the learning process, thus gradients of the recognizer parameters that belong to both the correct and the incorrect path do not cancel out. From (9.14) and as illustrated in Fig. 9.3, it can be seen that all phonetic models belonging to the correct path and the ones belonging to the incorrect path are updated at each data presentation. Consequently, acoustic models that belong to both paths are updated twice. i.e, within the correct path and within the incorrect path. This is likely to complicate the filter bank optimization scheme since a frame is supposed to belong to a specific acoustic model and may lead to non-convergence. Another observation is that, when using the strict MCE, the best path changes in an uncontinuous manner, which may result in a less smooth optimization scheme by the DFE process.

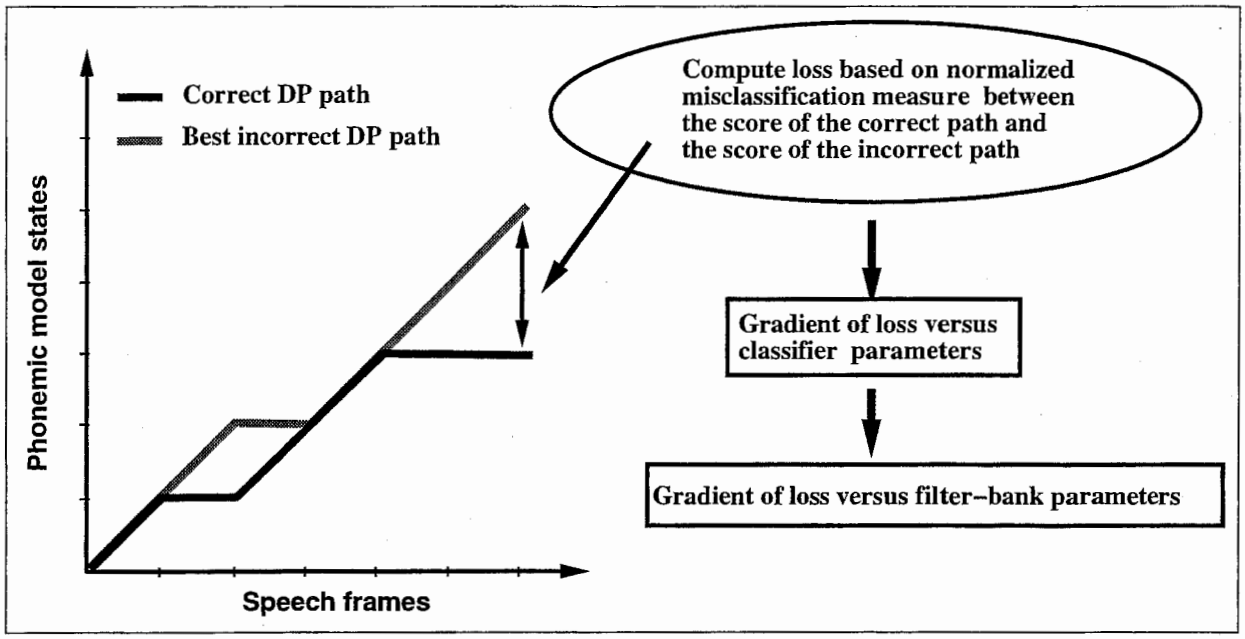


Figure 9.3: String-level optimization using a normalized misclassification measure. When using a normalized misclassification the gradient of the loss for models that belong to both paths do not cancel out

Note that the problems mentioned above only occur in the context of a normalized misclassification measure. For a classic misclassification measure, the gradients of the loss cancel out for models belonging to both the correct path and the incorrect path, given a frame, thus producing a balanced learning for the overall recognizer.

9.5.2 Frame-level optimization

The frame level is an attempt to redress the asymmetry effect encountered in the string-level optimization when using a normalized misclassification measure. The idea is to use a loss, which reflects local mismatches along the correct path and the incorrect path at a given frame and only update the models for those mismatches. For the frame level optimization [Chen and Soong, 1994], the loss is the sum of a local frame-based losses:

$$\ell(d_k(\mathbf{c}_1^{\mathcal{T}};\Phi)) = \sum_{t=1}^{\mathcal{T}} \ell\left(\delta_k^{\overline{k}}(\mathbf{c}_t;\Phi)\right) \tag{9.16}$$

where

$$\delta_k^{\overline{k}}(\mathbf{c}_t;\Phi) = 1 - \frac{D_{\psi_t^{\overline{k}}}(\mathbf{c}_t)}{D_{\psi_t^k}(\mathbf{c}_t)} \tag{9.17}$$

represents the local misclassification of the t-th frame. The local misclassification measure takes into consideration the mismatch at the frame level between to the paths, which result into a smoother transition of DP path between iterations.

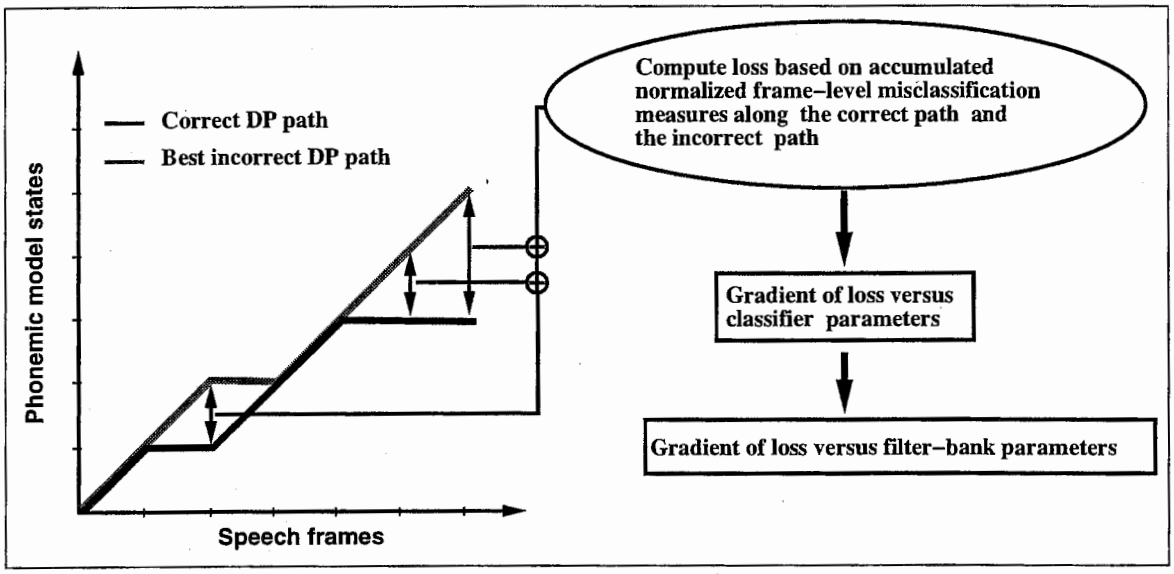


Figure 9.4: Frame-level optimization using frame-based normalized misclassification measure. Only models that do not belong to both path are updated

The frame-level optimization is illustrated in Fig. 9.4. It can be seen from this figure and from (9.16) that only phonetic models belonging to different paths, given a frame, are updated (the loss value is chosen to be zero for negative misclassification measures). Consequently, the filter bank parameters are updated according to those frames that have produced the mismatch between the two paths. This is believed to give consistent phonetic information to the filter bank optimization process. The frame-level optimization, as presented here, is quite similar to the optimization scheme introduced in [Mellouk and Gallinari, 1994] in the context of a hybrid NN and DP matching.

The obvious drawback of the frame-level optimization is that the error is not directly linked to the string-error, which is the prime target. However, based on the assumption that lesser frame-errors imply lesser string-errors, the frame-level optimization may result in more robustness.

9.5.3 Classifier's parameter adjustment

A similar optimization as shown in Chapter 8 was carried out. For PBMEC, the detailed adjustment of classifier parameters is described in section 8.4.1. Here, the focus is on describing the HMM case. The HMM is characterized by various parameters such as means, variances and mixtures. In this report, only the means were updated while the variances and mixtures remained unchanged. The adjustment rule for the means is as follows:

$$\boldsymbol{\mu}_{\psi_t^j,n}[\tau] = \boldsymbol{\mu}_{\psi_t^j,n}[\tau] - \epsilon_\tau \delta \boldsymbol{\mu}_{\psi_t^j,n}. \tag{9.18}$$

Again, ϵ_τ is a small, monotonically-decreasing, positive number at τ (the training rate), and $\boldsymbol{\mu}_{\psi_t^j,n}[\tau]$ denotes the status of $\boldsymbol{\mu}_{\psi_t^j,n}$ at τ. Using the chain rule of differential calculus,

$$\delta\boldsymbol{\mu}_{\psi_t^j,n} = -2\ell'\left(d_k\left(\mathbf{c}_1^{\mathcal{T}};\Lambda\right)\right)\Upsilon_{kj}\frac{m_{\psi,n}}{b_{\psi_t^j}(\mathbf{c}_t)}\mathcal{N}(\mathbf{c}_t,\boldsymbol{\mu}_{\psi,n},\boldsymbol{\Sigma}_{\psi,n})\boldsymbol{\Sigma}_{\psi,n}{}^{-1}(\mathbf{c}_t-\boldsymbol{\mu}_{\psi_t^j,n}) \tag{9.19}$$

Again, Υ_{kj} is determined by the form of the misclassification measure. For the classification measure defined in (9.11),

$$\Upsilon_{kj}=\begin{cases}-1 & \text{if } j=k\\ \dfrac{1}{(M-1)}\left\{\dfrac{e^{g_j(\mathbf{c}_1^{\mathcal{T}};\Lambda)}}{e^{g_{\overline{k}}(\mathbf{c}_1^{\mathcal{T}};\Lambda)}}\right\} & \text{if } j\neq k.\end{cases} \tag{9.20}$$

9.5.4 Filter bank adjustment

For filter bank parameter adjustment, the formula displayed from each parameter case have been described in the Chapter 8. The slight difference concerns the use the cepstrum transformation. Let ϕ be any adjustable filter bank parameter. Again, the transformation $\phi=\exp(\overline{\phi})$ is used to constrain the filter bank parameter to stay positive. The update rule is

$$\overline{\phi}[\tau+1] = \overline{\phi}[\tau]-\rho_\tau\mathbf{U2}\delta\overline{\phi}. \tag{9.21}$$

where

$$\delta\overline{\phi} = \frac{\partial\ell\left(d_k\left(\boldsymbol{s}_1^{\mathcal{T}};\Phi\right)\right)}{\partial\overline{\phi}}. \tag{9.22}$$

The chain rule of differential calculus gives

$$\frac{\partial\ell\left(d_k\left(\boldsymbol{s}_1^{\mathcal{T}};\Phi\right)\right)}{\partial\overline{\phi}} = \sum_{t=1}^{\mathcal{T}}\sum_{q=1}^{Q}\frac{\partial\ell\left(d_k\left(\boldsymbol{s}_1^{\mathcal{T}};\Phi\right)\right)}{\partial c_{t,q}}\frac{\partial c_{t,q}}{\partial\overline{\phi}} \tag{9.23}$$

$$= \sum_{t=1}^{\mathcal{T}}\sum_{q=1}^{Q}\frac{\partial\ell\left(d_k\left(\boldsymbol{s}_1^{\mathcal{T}};\Phi\right)\right)}{\partial c_{t,q}}\sum_{i=1}^{I}\frac{\partial c_{t,q}}{\partial x_{t,i}}\frac{\partial x_{t,i}}{\partial\overline{\phi}} \tag{9.24}$$

$$= \sum_{t=1}^{\mathcal{T}}\sum_{q=1}^{Q}\sum_{i=1}^{I}\frac{\partial\ell\left(d_k\left(\boldsymbol{s}_1^{\mathcal{T}};\Phi\right)\right)}{\partial c_{t,q}}\cos\left(\frac{q\pi}{I}\left(i-\frac{1}{2}\right)\right)\frac{\partial x_{t,i}}{\partial\overline{\phi}}. \tag{9.25}$$

The computation of $\dfrac{\partial\ell\left(d_k\left(\boldsymbol{s}_1^{\mathcal{T}};\Phi\right)\right)}{\partial c_{t,q}}$ is given in 8.4.2, by replacing $x_{t,i}$ with $c_{t,q}$ and I with Q. $\dfrac{\partial x_{t,i}}{\partial\overline{\phi}}$ is detailed in 8.4.2 for each filter bank parameters.

9.5.5 Embedding dynamic features

Most speech recognition systems nowadays make use of dynamic features to capture local spectral movements. Regression coefficients as introduced by Furui [Furui, 1986] are the most widely used. Other approaches to capture spectral dynamic include dynamic cepstrum as proposed by Aikawa

[Aikawa *et al.*, 1993]. Here, the concern is on the use of dynamical feature in the DFE framework, in particular on the use of regression coefficients.

The inclusion of dynamic features in the DFE framework can be done by two ways. . The first and straightforward method is to perform the regression on the DFE-optimized static features. The second method embeds the regression within the DFE training. In this later case, the DFE optimization is as follows.

Let $\Delta \mathbf{c}_t = [\Delta c_{t,1}, \ldots, \Delta c_{t,q}, \ldots, \Delta c_{t,Q}]^{\mathrm{T}}$, where $\Delta c_{t,q}$ is the regression coefficients along the feature index q. The polynomial regression coefficients is defined as

$$\Delta c_{t,q} = \frac{\sum_{\rho=1}^{R} \rho(c_{t+\rho,q} - c_{t-\rho,q})}{2\sum_{\rho=1}^{R} \rho^2} \tag{9.26}$$

$$= \frac{\sum_{\rho=-R}^{R} \rho c_{t+\rho,q}}{2\sum_{\rho=1}^{R} \rho^2} \tag{9.27}$$

$$\approx \frac{\partial c_{t,q}}{\partial t}, \tag{9.28}$$

where R is the number of forward and backward frames used for calculating the regression coefficients. (9.26) shows that $\Delta \mathbf{c}_t$ tries to approximate the temporal cepstral derivatives $\frac{\partial \mathbf{c}_t}{\partial t}$ by the derivative of a polynomial that fits the cepstral trajectory. $\Delta \mathbf{c}_t$ is usually referred to as delta cepstrum. If the feature vector contains delta parameters computed by (9.26), optimization of the a filter bank parameter $\overline{\phi}$ should take this fact into account by adding the term

$$\frac{\partial \ell\left(d_k\left(\boldsymbol{s}_1^{\mathcal{T}};\Phi\right)\right)}{\partial \Delta c_{t,q}} \frac{\partial \Delta c_{t,q}}{\partial \overline{\phi}} = \frac{\sum_{\rho=-R}^{R} \frac{\partial \ell\left(d_k\left(\boldsymbol{s}_1^{\mathcal{T}};\Phi\right)\right)}{\partial c_{t+\rho,q}} \frac{\partial \Delta c_{t+\rho,q}}{\partial \overline{\phi}}}{2\sum_{\rho=-R}^{R} \rho^2} \tag{9.29}$$

to (9.25). The overall chain is thus

$$\frac{\partial \ell\left(d_k\left(\boldsymbol{s}_1^{\mathcal{T}};\Phi\right)\right)}{\partial \overline{\phi}} = \sum_{t=1}^{\mathcal{T}}\sum_{q=1}^{Q}\sum_{i=1}^{I} \frac{\partial \ell\left(d_k\left(\boldsymbol{s}_1^{\mathcal{T}};\Phi\right)\right)}{\partial c_{t,q}} \cos\left(\frac{q\pi}{I}\left(i-\frac{1}{2}\right)\right) \tag{9.30}$$

$$+\sum_{t=1}^{\mathcal{T}}\sum_{q=1}^{Q}\sum_{i=1}^{I} \frac{\partial \ell\left(d_k\left(\boldsymbol{s}_1^{\mathcal{T}};\Phi\right)\right)}{\partial \Delta c_{t,q}} \cos\left(\frac{q\pi}{I}\left(i-\frac{1}{2}\right)\right) \left(\frac{\sum_{\rho=-R}^{R} \rho \frac{\partial x_{t+\rho,i}}{\partial \overline{\phi}}}{2\sum_{\rho=1}^{R} \rho^2}\right)$$

which finally gives

$$\frac{\partial \ell\left(d_k\left(\boldsymbol{s}_1^{\mathcal{T}};\Phi\right)\right)}{\partial \bar{\phi}} = \sum_{t=1}^{\mathcal{T}}\sum_{q=1}^{Q}\sum_{i=1}^{I}\cos\left(\frac{q\pi}{I}\left(i-\frac{1}{2}\right)\right) \left\{\frac{\partial \ell\left(d_k\left(\boldsymbol{s}_1^{\mathcal{T}};\Phi\right)\right)}{\partial c_{t,q}} + \frac{\partial \ell\left(d_k\left(\boldsymbol{s}_1^{\mathcal{T}};\Phi\right)\right)}{\partial \Delta c_{t,q}}\left(\frac{\sum_{\rho=-R}^{R}\rho\frac{\partial x_{t+\rho,i}}{\partial \bar{\phi}}}{2\sum_{\rho=1}^{R}\rho^2}\right)\right\} \quad (9.31)$$

Note that the computation of $\frac{\partial \ell\left(d_k\left(\boldsymbol{s}_1^{\mathcal{T}};\Phi\right)\right)}{\partial \Delta c_{t,q}}$ is similar to $\frac{\partial \ell\left(d_k\left(\boldsymbol{s}_1^{\mathcal{T}};\Phi\right)\right)}{\partial c_{t,q}}$ and is given by the classifier structure.

9.6 Vowel Segment Recognition

As in the previous chapter, the process was first analyzed by performing a simple vowel classification experiment.

9.6.1 Task and classifier structure

The task was to recognize the 5-class Japanese vowel uttered under various phonemic contexts by several speakers. A database of 500 sentences spoken by 5 speakers (3 males and 2 females) was used to extract 1750 tokens for training and 1750 token for testing from the middle position of the vowel onset. The training body and the testing body was balanced among the speakers and the vowel-category.

The speech signal was digitized at 12kHz and stored at 16 bits. A Hamming window of 21 ms was used for the production of the spectral frames, prior to filter bank analysis and cepstrum computation.

9.6.2 Experimental conditions and results

In the following experiment, the initial filter, before training, was based on the Mel scale. The Mel scale maps the perceived frequency of a pure tone onto a linear scale. Hence, various analytical approximations of the original Mel scale has been derived from psychoacoustic experiments, all of them being approximatively linear below 1 kHz and logarithmic above 1kHz. Again, the following approximation, as provided by [Zwicker and Terhardt, 1980] was used:

$$p(f) = 2595\log_{10}(1+\frac{f}{700}). \quad (9.32)$$

It is common practice to choose a lower number of cepstrum coefficients as compared to the number of filters. This is motivated by the fact that a higher number of filters permits a better resolution of spectral characteristics and a lower number of cepstral coefficients reduces the dimensionality of the

feature. However, reducing the feature vector dimensionality also means that few information are given to the recognition process. For investigation, two kind of feature representations were tested. The first representation (representationI) uses the same number of filters and cepstrum (I = 16, Q = 15). 15 cepstrum coefficients is due to the fact the 16-th cepstral component is zero. The second representation (representation II) uses 20 filters with 10 cepstral coefficients (I = 20, Q = 10). In both cases, the filters were initially aligned in the Mel scale. A k-means algorithm was ran to select the initial set of prototypes, prior to MCE/GPD (no training of feature extraction module) or DFE training (simultaneous training of classifier and feature extractor). Note that when the filter bank is kept constant, the resulting features are simply MFCC. Thus, the MCE/GPD training was done using MFCC. MCE/GPD training was carried out as baseline for MFCC testing. DFE training produced the various DFCC in a segment classification basis. Only two architectures were tested: using only one state with 1 and 3 prototypes per category in the first representation case and 1 prototype per category in the second representation case.

9.6.3 General results

The best results achieved across the optimization methods are summarized in Table 9.1 for representation I and in table 9.2 for representation II.

Contrasting representation I and representation II using MFCC shows that representation II achieves better performance in the context of k-means training. Both representations have rather close results when using classical MCE or DFE.

In representation I, C-DFCC, W-DFCC and S-DFCC show the best results so far in the testing set. In particular, the performance of C-DFCC are rather close to the performance of S-DFCC. The meaning of these results is that center frequency optimization (spacing) contributes the most in the quality of cepstral features. W-DFCC put an emphasis on single optimum frequency which has resulted into the best of the DFCC in the training set.

In representation II, the best performance is achieved by C-DFCC in the testing set, although W-DFCC and S-DFCC show the best results in the training set. Thus, when looking across the two types of representations, it can be argued that the center frequency optimization contributes the most to the performance.

For both representation, it could be noticed that optimizing any parameter of the filter bank model showed an improvement of the system performance. The best improvement are achieved when optimizing the center, the "weighting" and the main three parameters at the same time. Optimizing the gain showed less improvement of the original model. The obvious conclusion is that DFCC make it possible to be more efficient in average than classical MFCC.

9.6.4 Optimized filter bank analysis

Formants are essential feature in recognizing vowels. For a deeper analysis of DFCC, an analysis on how the resulting filter bank model performs formants resolutions was done. Again, Fig. 8.8 shows

Table 9.1: Experimental results using various filter bank cepstral parameters in terms of error rates (%) in the vowel recognition task for I=16 and Q = 15 (representation I).

types of features	1 prototype/class		3 prototypes/class	
	train	test	train	test
MFCC(k-means)	29.7	27.1	25.3	23.7
MFCC (MCE/GPD)	13.1	15.5	11.8	14.0
C-DFCC	12.4	14.3	11.1	13.3
B-DFCC	12.0	14.9	10.9	13.6
G-DFCC	13.0	14.9	11.6	13.7
W-DFCC	10.4	14.3	10.7	12.9
S-DFCC	12.4	14.2	11.0	13.5

Table 9.2: Experimental results using Mel-based DFCC in terms of error rates in the vowel recognition task for Q=20 and L= 10 (representation II).

types of features	train	test
k-means	21.0	22.3
MFCC	13.0	15.5
C-DFCC	12.1	14.5
B-DFCC	12.0	14.6
G-DFCC	13.3	15.2
W-DFCC	11.9	14.9
S-DFCC	11.9	14.9

an histograms of formants location in the train database as found by an LPC-based root finding method. It can be seen that F1 formants are mostly located in the region below 1 kHz, whereas most F2 formants are contained in the region between 1 and 3 kHz. 90 % F3 formants spans the region between 2.5 and 3.5 kHz.

Center frequency optimization

Figure 9.5 illustrates the modified filter bank after center frequency optimization by DFE for the two representations.

It could be seen that for both representations, modifications of the filter bank spacing, have occurred in the 1.5 kHz region (F2 domain), where some filters have moved closer for an emphasis of this region. The same observation stands for the filters in the 2.5 kHz region. More noticeable is the shift toward lower value of the center frequency to accentuate the 4 kHz region, where F3 formants are located.

It seems that the spacing of the filters has been modified for a better resolution of the formants independently of the number of cepstral coefficients, which may explain the better performance achieved by this model for both representations.

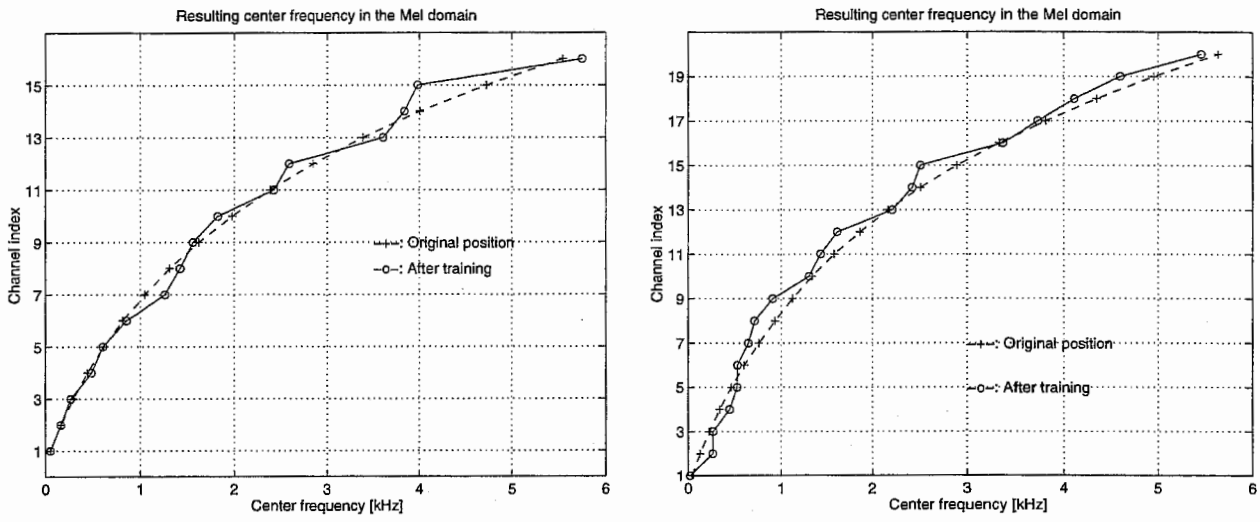


Figure 9.5: **Left**: Resulting Mel-scale based filter bank center optimization task for I = 16 and Q = 16 (representation I). **Right**: Resulting Mel-scale based filter bank center optimization task for I = 20 and Q = 10 (representation II). Each channel is plotted versus its modified center.

Bandwidth optimization task

Optimization of the filter bank through the parameter β_i maximizes or minimizes the contribution of those regions which may be important for minimum error. Note that, the center frequency are *fixed*, thus optimization is done on pre-chosen perceptually based regions. For better visualization of the bandwidth, the notion of "corresponding bandwidth value" (CBW) of a channel is introduced. The CBW of a channel is the interval in the *Hertz* domain between the two frequencies whose weighting is half of the weighting at the center frequency of the channel.

Figure 9.6 shows the CBW versus the center frequencies of the filter bank model for the two representations. In general, most filters above the 3 kHz region have seen a significant change of their bandwidth value. In contrast to the original model, the obtained bandwidth values are not a monotonic function of the center frequencies. Note that a decrease in CBW value signifies that the filters tend to select fewer frequencies in the channel.

In representation I, the 13th and the 14th filters have recorded a decrease of the CBW of more than 200 Hz. However, this has been compensated by the 15th and 16th filters whose bandwidths have recorded an increase of their CBW of more than 300Hz.

In representation II, the 15-th filter has increased its CBW value of almost 2 kHz, fully covering the 3 kHz region, to the detriment of the surrounding filters.

Both representations show similar behavior in certain regions of the spectrum. For instance, both show a decrease in bandwidth around the 1.5 kHz region and 3.5 kHz region, and an increase in bandwidth in 4.5 kHz region. This observation may lead to the conclusion that both models attempt to put emphasis or de-emphasis on meaningful part of the spectrum. Both representations display a good performance on the training set (12.0%) while showing a relatively poor generalization on

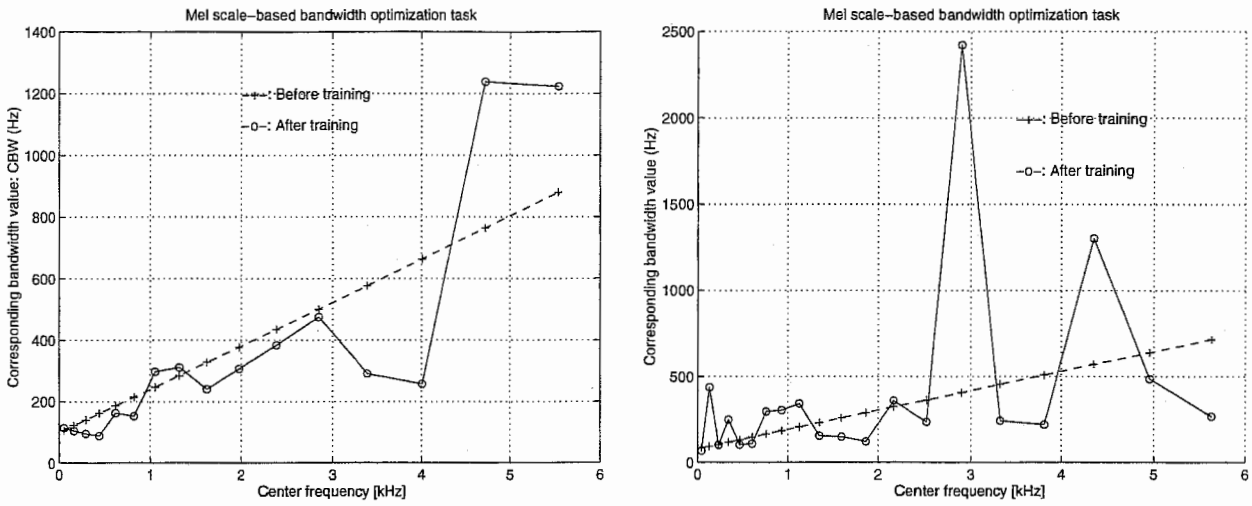


Figure 9.6: **Left**: Resulting Mel-scale based filter bank bandwidth optimization task for I = 16 and Q = 15 (representation I). **Right**: Resulting Mel-scale based filter bank bandwidth optimization task for I = 20 and Q = 10 (representation II). The CBW is plotted versus the center frequency in each channel.

the testing set: 14.9% for representation I and 14.6% for representation II.

Gain optimization task

The gain optimization puts an emphasis on those outputs relevant to minimum error.

Figure 9.7 shows the value of the gain (the parameter α_i) versus the center frequency in each channel for both representations. It can be seen that both representations show a similar shape although the value range is different: both accentuate and de-accentuate similar regions of the spectrum.

Filters in the region below the 1.5 kHz frequency region have shown a small decrease of the gain parameter. While the filters above the 4 kHz have shown significant increase of their gain value. Note that only high value of the gain should have a noticeable effect on the performance of the system because of the use of the logarithm in calculating the log energies which tends to attenuate the small modifications of the gain value. The sharp increase of the gain in the region above 4 kHz can be compared to a pre-emphasis process widely used a to accentuates higher frequency energies.

Weighting optimization task

As said before, each parameter $w_{i,f}$ can be optimized independently without keeping the Gaussian constraint.

As can be seen from Fig. 9.8, the resulting model did not keep the Gaussian form of the frequency response. One can notice that within certain channel, several optimum frequency have been emphasized. The fewer number of channel in representation I has resulted in filters having

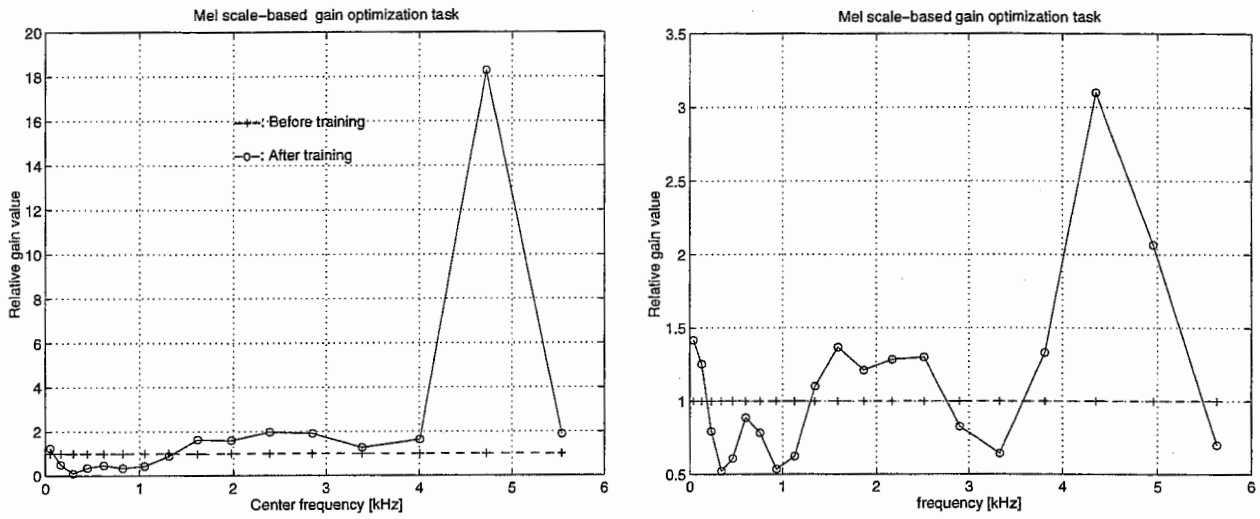


Figure 9.7: **Left**: Resulting Mel-scale based filter bank gain optimization task for I = 16, Q = 15 (representation I). **Right**: Resulting Mel-scale based filter bank gain optimization task for I = 20, Q = 10 (representation II). The values of α_i are plotted against the center frequencies in the channels.

higher energies than filters in representation II. In representation I, filters in the region below 1.5 kHz have not shown any significant increase of the weighting.

The selection of multiple optimum frequencies within a channel may, in a sense, be compared to the search for the *the best* within-channel optimum frequency as in center optimization case. However, the higher number of free parameters may inhibit generalization capabilities.

Simultaneous optimization of center frequency, bandwidth and gain

A global optimization of the filter bank maybe achieved by optimizing at the same time, the center, the bandwidth, the gain. This enables an efficient interaction between the three parameters while reducing the degree of liberty which may lead to the over training dilemma.

Fig. 9.9 shows the resulting filter bank for S-DFCC cases. Hence, in representation I, the best result on the testing set has been achieved so far by this model: 14.2% on the testing set while displaying a similar result in the training set with C-DFCC.

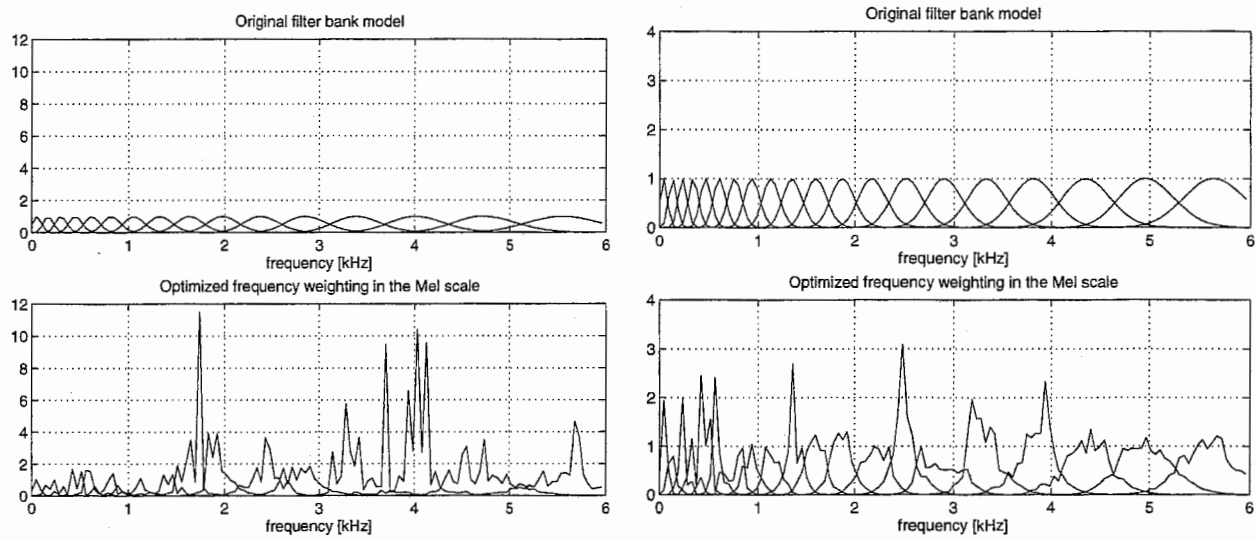


Figure 9.8: Resulting Mel-scale based filter bank weighting optimization. **Left**: Resulting Mel-scale based filter bank weighting optimization task for I= 16, Q = 15 (representation I). **Right**: Resulting Mel-scale based filter bank weighting optimization task for I = 20, Q = 10 (representation II).

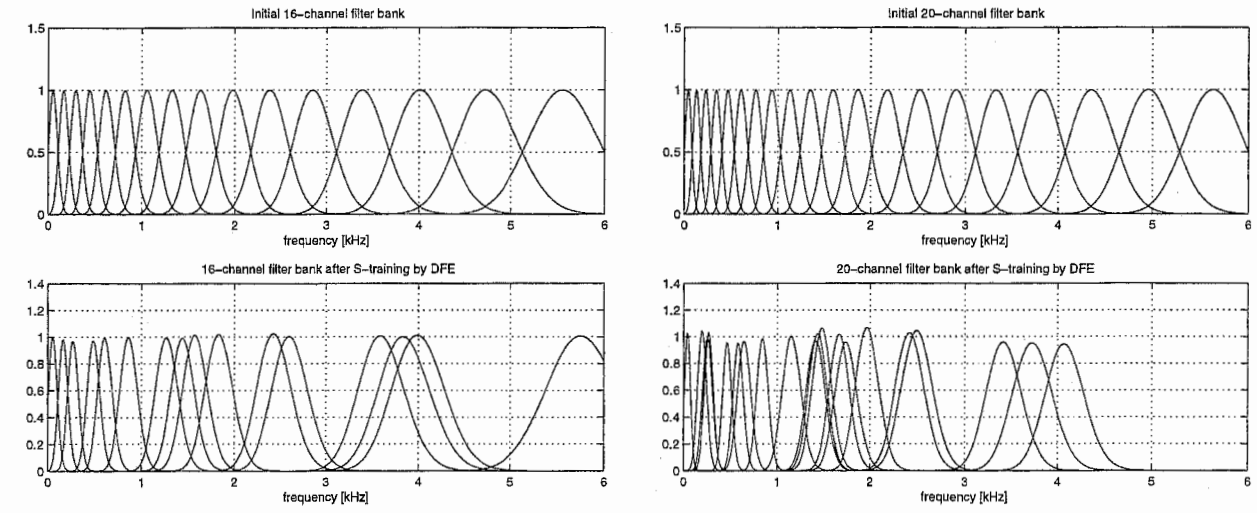


Figure 9.9: Resulting Mel-scale based filter bank CBG optimization task. **Left**: Resulting Mel-scale based filter bank center-bandwidth-gain optimization task for I = 16, Q = 15 (representation I). **Right**: Resulting Mel-scale based filter bank center-bandwidth-gain optimization task for I = 20, Q = 10 (representation II).

Looking at the DFE-adjusted models of Fig. 9.9, one can see that center frequency modification contributes the most in the adjustment of the filter bank. Thus, the spacing of the filter clearly puts an emphasis on well-defined regions: the 1-2 kHz region (F2 domain), the 2.5 kHz region covered by 11th filter (F2 region) and 12th filter and the 3.5-4 kHz region (F3 domain). It can be noted that, a similar filter spacing was achieved on the C-DFCC task. Furthermore, the spacing of the filters in representation II has been modified so as to covers the 0-5 kHz region, which is acknowledged to carry much of the spectral information. The bandwidth and the gain have remained relatively stable.

It seems that when optimizing the main parameter of the filter bank at the same time, DFE focuses on finding the optimal spacing for minimum error by center frequency modifications and puts less emphasis on bandwidth and gain modification in the context of the Mel scale.

9.7 Directory Assistance Task

This section presents application of DFCC to a directory assistance task.

9.7.1 Task

The purpose is the design of a system which recognizes Japanese's names and forwards calls to staff members within the ATR laboratory. Thus, the system is intended to act as an automatic switching operator in charge of accepting spoken names and dial up the phone call to recognized persons. The point of interest is to improve the speech recognition accuracy of the system by applying DFE-optimized cepstral parameters.

Data were automatically collected in office environment by a system that periodically called staff members to repeat 5 randomly selected names. Each name utterance was therefore spoken in isolated word recognition mode. The process resulted into 684 utterances in total from 47 speakers (3/4 of them are male) with the target of recognizing 64 names;

Closed testing set

570 utterances were used as training and 114 as a closed testing set (closed-test). The closed-test features the same speakers as in the training data as well as the same vocabulary (64 names) from 47 speakers.

Open testing

Also, 234 utterances were collected during a demonstration of the system, which involved 34 new names (open-test). The open-test contains names that were not in the training vocabulary, uttered by new speakers in a more noisy atmosphere (which was different from the training conditions). Thus, the total vocabulary for the open-test is 98 names for total number of utterance equal to 234. .

The closed-test features the same speakers as in the training data as well the same vocabulary, while the open-test set contains names that were not in the training vocabulary, uttered by new speakers in a more noisy environment.

The utterances were phoneme-labeled using their transcriptions by the segmental k-means clustering procedure; the labels created this process were then verified through a human correction work.

9.7.2 Experimental settings

The speech signal was digitized at 8 kHz sampling rate and at 16 bits. A Hamming window of 21 ms was shifted every 5 ms over an input speech utterance, thus producing 128 FFT-based power spectrum ($F = 128$), as input to a 20-channel-filter bank whose output was converted to 10 cepstral parameters. For acoustic modeling, 26 context-independent phoneme models were used, which correspond to 5 Japanese vowels (/a/, /i/,/u/,/e/, /o/), 20 consonants (/b/, /ch/, /d/, /f/, /g/, /h/, /j/, /k/, /m/, /n/, /N/, /p/, /r/, /s/, /sh/, /t/, /ts/, /w/, /y/, /z/) and silence. A finite state grammar was used to constrain the search.

Each phoneme model was a PBMEC structure which consisted of 3 states, each being associated with 1 prototype.

9.7.3 System initialization

The initial prototypes of the classifier module produced by Maximum Likelihood (ML)-based segmental k-means provided an estimated segmentation of each utterance. This ML-produced baseline system was further trained by one of the five types of DFE training. For comparison purposes, classical MCE/GPD training using MFCC was also carried out.

9.7.4 Results

The results for string-level and frame level training are shown in Table 9.3.

The results (in terms of recognition rates) show that all MCE-based training outperform ML training in both testing sets. The frame-based optimization shows better performance in average than string-level training for both MCE (only classifier adjustment) and DFE (joint optimization), on both testing sets. This result is explained by the fact that the adjustment rate of the filter bank parameters is higher within the frame-level optimization than string-level optimization. Consequently, given a few number of training data, such as in the current task, the frame-level training provides better performance.

Open and Closed test set

Comparison of performance on the open-test and closed-test is a crucial test for robustness. The gap in results between the open-test and the closed-test is due to the large difference of the experimental conditions between the two sets. W-DFCC seem the most robust features for string-level training

Table 9.3: Experimental results of ATR names recognition task using string-level training and frame-level training for various cepstral features.

features	String-level			Frame-level		
	train	closed-test	open-test	train	closed-test	open-test
MFCC (ML)	67.0	54.3	35.9	-	-	-
MFCC (MCE)	97.3	91.2	57.9	97.3	94.7	59.9
C-DFCC	96.9	92.9	60.0	98.2	92.9	59.5
B-DFCC	97.0	92.9	57.8	98.5	94.7	56.8
G-DFCC	97.5	94.7	59.0	99.1	94.7	60.3
S-DFCC	96.4	92.9	55.3	98.7	95.6	64.5
W-DFCC	97.8	92.9	61.6	98.8	93.8	56.9

while S-DFCC give the best results for frame-level training. MFCC appear to be relatively robust within the frame-level optimization.

String-level and Frame-level

For string-level training, the best performance on the closed test is realized by G-DFCC and on the open test set by W-DFCC (61.6% compared to 57.8% for MFCC). Also, C-DFCC appear to be relatively robust considering its relative performance across the two testing sets.

For frame-level optimization, the best result is achieved by S-DFCC on both testing sets. In particular, the result in the open test set is far ahead of other cepstral features (64.5%, compared to 59.9% for MFCC). This is in contrast to its rather limited performance achieved within the string-level optimization. Its seems that frame-level enables the gathering of short-time salient information in a more efficient way than the string-level training. This is consistent with the frame by frame way to process speech signals in conventional speech recognizers.

MFCC and DFCC

For the closed test set, MFCC and DFCC provide relatively close performance for both levels of training, although in most cases, DFCC display better recognition rates. For the open speaker/vocabulary test set, DFCC appear to be more robust than MFCC for both level of training. However, the best type of DFCC depends on both the task and the level of training.

9.8 Isolated Letter Recognition

This section presents application of DFCC to the ISOLET database. The ISOLET database, collected and distributed by the Oregon Graduate Institute consists of two examples of each letter of the English alphabet uttered by 150 American English speakers, 75 males and 75 females. The

database is divided into 5 portions of 30 speakers. It is common to use the first 4 portions as training (120 speakers, 6240 utterances) and the last portion as testing, that is, 30 speakers (1560 utterances. Among the letter of the English alphabet, the E-set subset is of particular interest. The E-SET is the set of /e/ sounding letter, That is {"B", "C", "D", "E", "G", "P", "T" and "V", "Z"}. All this letter shares the /e/ sound and are particularly difficult to discriminate. The E-SET is comprised of 2160 training utterances and 540 testing utterances.

The speech signal is digitized at 8 kHz. For all experiment, a frame rate of 5 ms and Hamming window shift of 10ms was used.

9.8.1 Recognizer structure

Speech parameterization

Throughout this section, the features were 12 cepstral coefficients computed from a 24-order filter bank. Again, the initial configuration simply emulates MFCC, with Gaussian filters. Here, only S-DFCC was performed. That is, simultaneous optimization of center-frequency, bandwidth and gain.

System structure

Within this section, a word is modeled by an HMM, which is quite different from the previous study of DFE application which was based on phonemic models. Each letter is thus modeled by a left-to-right continuous HMM consisting of 5 states. Each state is assigned 5 mixture probability density functions. Throughout this section, the variances and mixtures are fixed, and only mean vectors are considered free parameters for acoustic-modeling.

9.8.2 ML-estimation using DFCC

In the vowel and the directory assistance task, the starting point for DFE optimization was based on MFCC. That is, the choice of the initial PBMEC prototypes was estimated by the segmental k-means , which optimizes the MLE criterion, using MFCC.

This approach may be unoptimal for a number of reasons. First, the optimization technique is based on the gradient search, and MFCC as the starting point for DFE may not be the optimal solution. Second, it is desirable that the DFE method should produce features that may be of use even in a classical framework of optimization using MLE. This may be particularly useful in extremely large vocabulary recognition task, where the EM algorithm enables a straightforward re-estimation paradigm. The interesting point of this approach is that, similar to the experiments described in Chapter 3, the MLE-estimation is based on discriminative features, that is features that bear knowledge of the classification scheme.

The simple algorithm described below allows one to iteratively improves the performance of the MLE-based model using the DFE technique. The algorithm is as follows, in the context of filter bank cepstral parameters.

1. Initialize cepstral parameters to MFCC configuration.

2. Initialize acoustic models using MLE training on MFCC.

3. Estimate MCE loss function using set of misclassification measures from the training data.

4. Optimize features using DFE with the MLE-based classifier (no classifier training).

5. Estimate new acoustic models by MLE using optimized features and last overall recognizer status as a starting point for optimization.

6. if the error rate on training data does not increase when compared to the previous value, stop and use current feature and *un-trained model* as starting for acoustic model MCE training. otherwise go to step 3).

In this study, using a mixture of error-functions as MCE loss enables a straightforward estimation of the loss parameters at step 3, according to the set of misclassification measures over the training data set.

Optimization of features in step 4 can be done using any optimization method available. In particular, since the acoustic models are not trained at this stage, GPD, MGDP or IGPD are not necessary. For instance, for faster optimization, the quick-prop algorithm proposed by Fahlman [Fahlman, 1988] can be applied. The quick-prop algorithm is a compromise between the off-line gradient descent and the Newton method, which uses an approximation of the Hessian matrix. For an untrained classifier, only a (well-selected) subset of the training set can be used at this step since only few parameters need to be adjusted.

The rationale between this optimization at step 4 considers the fact that the errors provided by the MLE-based acoustic models are an estimate of the "quality" of features at this point in time. This enables us to further optimize these features according to this estimate. Once, the error has been sufficiently reduced by the MCE optimization, the MLE estimates new acoustic models at step 5 using optimized features. Since features are optimized based on the performance of MLE-derived models, this procedure permits to improve the performance of MLE by performing MLE on "improved" features.

The philosophy behind this approach is as follows. The new models bear the characteristics of the discriminative features. The error estimate using these new models can be further reduced, thus producing a better feature set at each estimation. When MLE performance can not be further improved, this is interpreted as having achieved the best possible features possible for MLE , given the acoustic-modeling structure. Furthermore, these features may provide a better starting point for DFE optimization of the overall recognizer.

Fig. 9.10 shows the resulting filter bank after the joint MLE/DFCC was carried out on the ISOLET database. Clearly, this filter bank is quite different shown in the same figure.

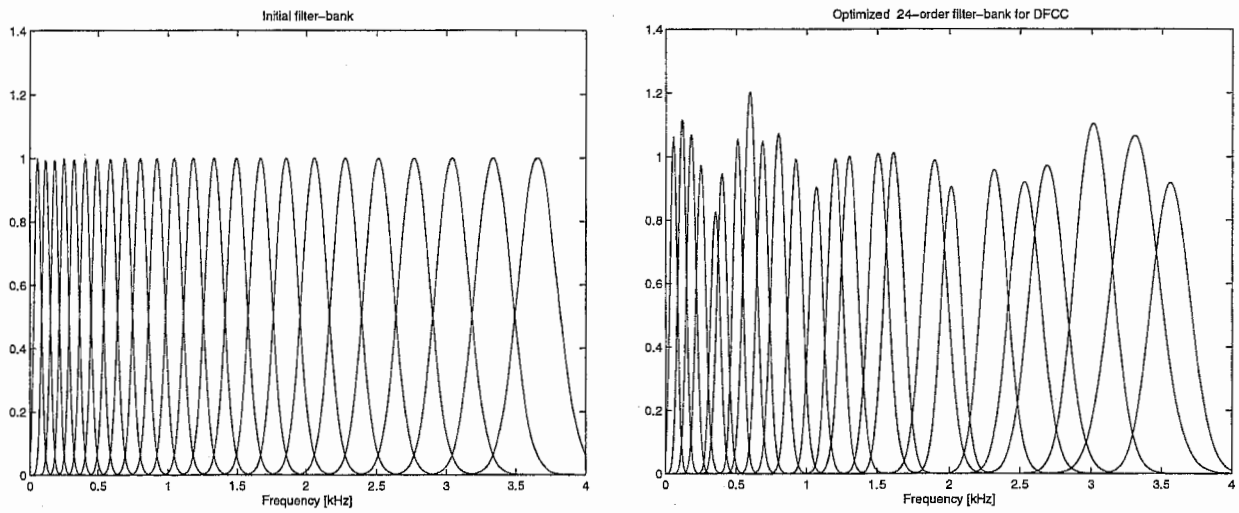


Figure 9.10: Initial and Optimized filter bank using the joint MLE/DFE optimization scheme in the ISOLET task.

Experimental evaluation

Fig. 9.11 shows the performance of the DFCC features versus the number of trials. A trial is defined as the period covering two MLE optimization. Again, acoustic-models are not trained, but re-estimated after each trial by MLE. This figure also displays the performance of DFCC at each trial *before* ML estimation of DFCC.

It can be observed that the performance of DFCC and DFCC+MLE estimation increases at each trial, except for the last trial which displays a decrease in the performance of DFCC+MLE. The decrease of the MLE performance after the third trial signifies that the "quality" of features, in term of MLE, has reached its optimum, given the preset classifier structure, and do not longer need to be optimized. Note that DFCC, prior to MLE, still display an increase in performance as the number of trial increases.

MLE-based acoustic models realized 77.82% recognition rate using MFCC and 78.78% recognition using the DFCC at the second trial. However, the best performance at the last trial is realized by DFCC, prior to MLE estimation of acoustic models. At this point, the DFCC are already optimized and MLE appears to be an error contrasting criteria.

Dependency on the classifier structure

In this section, it is examined whether DFCC coefficients obtained by the method presented above can be used in a different classifier structure. DFCC were obtained using a preset classifier configuration. That is, a 5-states left-to-right HMM with 5 Gaussian mixtures per state. Clearly, these DFCC has been optimized for this particular structure. By matter of curiosity, its performance using a different configuration of the classifier is investigated. That is, various recognizer structures

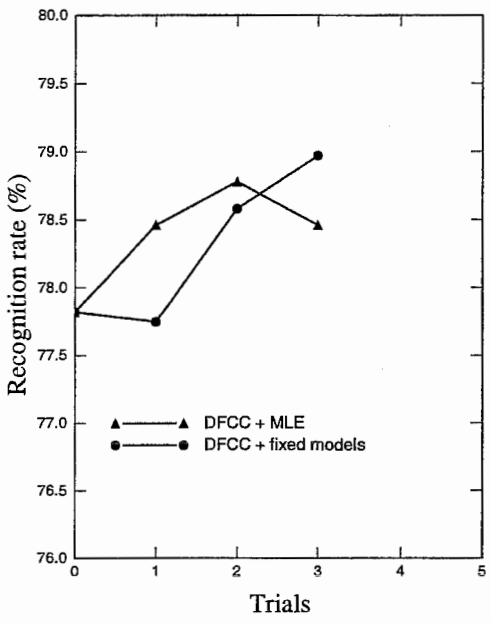


Figure 9.11: DFCC performance on ISOLET. The classifier is untrained by MCE but re-estimated at each trial by MLE using DFE-estimated features

are optimized by MLE using MFCC and the optimum DFCC obtained by the HMM configuration presented above.i.e,. 5 Gaussian mixtures within a 5-states HMM.

Various configurations can be obtained by changing various HMM parameters such as the state transition probabilities, or emission probabilities, or the number of states. Here, the configurations were determined by changing the number of mixture-components.

Table 9.4 shows the results as function of the number of mixtures. As expected, DFCC only outperform MFCC for the configuration for which it is derived. This results confirm the fact that DFE-features are matched to the classifier for which they are derived.

9.8.3 DFE-training

In the vowel and name recognition task, DFE was applied to optimizing jointly the overall recognition systsem. Here, as was stressed in Chapter 6, another alternative is presented. It consists of gradually updating the modules of the recognizer using MCE.

After MLE/DFE-based DFCC-extracted features through the method presented above, two methods are offered to us for DFE-training. The first method is the classical global optimization of the overall recognizer as was done previously. The second method relies on the fact that since DFCC are already derived from an MCE criterion, one only need to optimize the acoustic models

Table 9.4: Recognition rate on ISOLET and E-SET using MLE with various mixture components for MFCC and preset DFCC. The preset DFCC has been obtained with using a 5-mixtures-Gaussian-pdf-HMM.

# of mixtures	ISOLET		E-SET	
	MFCC	DFCC	MFCC	DFCC
3	77.05	76.08	53.88	50.37
5	77.82	78.78	52.59	55.00
8	79.16	78.26	55.74	54.62
12	78.20	76.98	55.37	53.88

using MCE. Note that the latter approach does not contradict the DFE concept, since its uses the same criterion for both the classification stage and the feature extraction stage, which is the basic philosophy of the DFE approach. If the latter approach is taken, it is advisable to perform a last tuning of the feature extractor using (fixed) MCE-acoustic models to re-ensure that the overall recognizer is optimized. This DFE implementation may be well-suited for large-sized system where tuning of the learning parameter is time-consuming as was suggested in Chapter 6.

To summarize, the overall procedure is as follows:

1. Extract DFCC by the joint MLE/DFE method.

2. Train the overall recognizer or train acoustic models by MCE using DFCC. In the latter case, re-tune features to untrained model using the MCE criterion.

Let DFE-J refers to training the overall recognizer and DFE-S for training classifier and the feature extraction separately. Again, DFE-S is simply training the classification with MCE on MCE-optimized filter bank-based cepstral features. Table 9.5 shows the result of various optimization methodologies for the ISOLET task and its E-SET sub-task. Note that the recognizer was not trained to perform the E-SET discrimination but discrimination of all letters. For clarity of comparison, the results of the MLE optimization is also shown. From this table is is quite obvious that all MCE-based approaches outperform MLE-based ones across the feature sets.

MLE performance improves when estimation is based on DFCC. This is clearly shown on the E-SET task, when the use of DFCC has resulted into an improvement over MFCC: from 52.59 % to 55.00 %. However, the improvement over the entire database is less: 77.82 % for MFCC and 78.78 % for DFCC. It seems that DFCC have focussed on discriminating the most acoustically similar categories as expected. Similar to the results obtained in Chapter 3, this experiment shows that discriminative features are able to improve MLE-based models.

MCE/MFCC is using classical MCE on MFCC and DFE-S is using classical MCE on pre-trained DFCC. In the first approach, the models are discriminant but the features are not. In the second approach, the features and the models are both discriminant but not embedded. Classical MCE gives the same result independently of the feature set in the ISOLET task. This confirms the MCE power regardless of the feature space in use. That is unlike MLE, MCE is less sensitive to the

Table 9.5: Recognition rate on ISOLET and E-SET using MLE, classical MCE, DFE-S (refers to MCE optimization of previously derived DFCC) and DFE-J (refers to joint optimization) with static filter bank cepstral coefficients.

Method	Task	
	ISOLET	E-SET
MLE/MFCC	77.82	52.59
MLE/DFCC	78.78	55.00
MCE/MFCC	83.84	66.85
DFE-S	83.84	68.14
DFE-J	84.35	68.70

feature space on which estimation is made. This also make any comparison of feature "quality" using the MCE criterion, difficult. It seems that DFE learning focus more on separating the more similar categories, which is what is actually expected. In this task. The importance of DFCC as compared to MCE/MFCC can still be grasped when observing the performances on the difficult E-SET task (that is, a task which requires higher discriminative capabilities). On the E-SET task, MCE performs better using DFCC (68.14%) than when using MFCC (66.85%). Again, this is in sharp contrast with the result obtained on ISOLET.

The joint DFE optimization scheme, referred here as DFE-J, gives the best result for both task. This confirms the fact that joint optimization permits a better match between the feature extractor and the classifier in a more efficient way than separate optimization. The performance of the joint optimization scheme is more striking in the difficult E-SET task than the overall ISOLET database, confirming the inherent property of DFE to focus its learning on less separable categories.

9.8.4 Using Dynamic Features

As already indicated, two ways are offered for making use of delta parameters in DFE optimization. The first and easy way is to compute delta DFCC using the classical regression formulas on previously derived static DFCC. The second method is embedding delta DFCC in the DFE process using the optimization scheme described in section 9.5.5. The two methods were implemented and compared with the use of delta MFCC. Delta DFCC computed by classical regression equations is referred to as ΔDFCC-R and delta DFCC computed by embedding the regression process into the DFE is referred to as ΔDFCC-E.

Table 9.6 displays the performance of various optimization methodologies when using dynamic parameters. ΔDFCC-R were calculated using previously optimized static DFCC by the MLE/DFE method described in section 9.8.2. Similar to ΔDFCC-R, ΔDFCC-E were calculated in a similar way with the difference that, instead of performing the regression after DFCC derivation, the regression is embedded in the DFE optimization. It is quite clear, from this table that all MCE-based frameworks outperform MLE-based ones across all feature sets.

Table 9.6: Recognition rate on ISOLET and E-SET for DFE-S, DFE-J and classical MCE using delta parameters. ΔDFCC-R refers to calculating delta parameters using the regression formula on previously optimized static DFCC. ΔDFCC-R refers to delta parameters, which are derived by embedding the regression formula within the DFE optimization process.

Method	Task	
	ISOLET	E-SET
MLE/(MFCC+ΔMFCC)	85.32	68.88
MLE/(DFCC + ΔDFCC-R)	86.31	68.70
MLE/(DFCC + ΔDFCC-E)	85.57	67.77
MCE/(MFCC+ΔMFCC)	90.25	80.37
MCE/(DFCC+ΔDFCC-R)	87.69	84.25
DFE-S	89.55	81.66
DFE-J	90.44	81.85

MLE training using DFCC appended with ΔDFCC-R and, DFCC appended with ΔDFCC-E, perform slightly better than MFCC+ΔMFCC on the ISOLET task. However, MFCC give slightly better result in the E-SET task. If the result on the E-SET task are contrasted with those obtained when no delta parameters are used, it seems reasonable to conclude that, for MLE estimation, the regression coefficients introduce a discrepancy. This is particularly true for ΔDFCC-E. In this case, the feature extractor parameters have been optimized for dealing explicitly with the regression. The way the MLE deals with feature derivatives in the HMM framework has been the object of vast literature. During decoding, the HMM structure assigns a frame to a given state, without a clear model of the surroundings of the frame. The way this affects the embodiment of feature derivatives in the DFE optimization is worth of investigation.

MCE was investigated using ΔMFCC and ΔDFCC-R, calculated as shown above. Using ΔMFCC has resulted into a quite better performance than using ΔDFCC-R. This result may be explained by the fact performing the regression formula on pre-optimized DFCC may have produced a less optimized model, since the regression process was not explicitly taken into account during the optimization process. Nevertheless, when targeting acoustically close categories, that is the E-SET task, ΔDFCC-R outperforms ΔMFCC: 80.37 % for MFCC+ΔMFCC compared to 84.25% for DFCC+ΔDFCC-R.

This explanation is supported by the results obtained using DFE-S, which is equivalent to optimizing the classifier with MCE using DFCC + ΔDFCC-E. Indeed, DFE-S performs better than MCE/(DFCC+ΔDFCC-R), but lesser than MFCC+ΔMFCC on ISOLET. DFE-S and DFE-J display a slight improvement on the E-SET task and ISOLET.

It could be concluded that, even if DFCC+ΔDFCC-R have the effect of introducing some discrepancy in the DFE optimization, it may be satisfactory to some extent to use them in place of ΔDFCC-E when one would like to avoid the burden of the overall optimization process. However, further investigation is required before having a precise conclusion.

9.9 Conclusion

A method to derive discriminant filter bank cepstral parameters was described in this chapter and shown to be more robust than classical Mel-based filter bank cepstral parameters. The method consisted of optimizing the filter bank parameters under a cepstral distance when adjusting the classifier parameters with DFE process. First, analysis of the method was done using a simple vowel recognition task. In this framework, discriminative filter bank-based cepstral parameters (DFCC) were analyzed.

Second, to test the performance of DFCC, experiments using telephone speech for a word recognition task was carried out. Two level of training was investigated. A frame-level training and a string-level. DFE and MCE training seem more efficient in average when training at the frame-level for test data close to design data. Using an open data test set, the best result was achieved by simultaneously adjusting center frequency, bandwidth and gain using the frame-level optimization.

Third, the method was applied to isolated letter recognition (American english alphabet), which includes the difficult E-set task. This has provided the opportunity to study the embodiment of delta parameters within the DFE process and implement a variant of the DFE process for use in the MLE framework.

Chapter 10

Conclusion

What is to be closed should first be opened

-Lao Tzeu-

10.1 Summary of the Report

The report has focussed on following issues.

In the introduction, speech recognition was viewed as a particular case of the statistical pattern recognition framework. It was stressed that the pattern recognizer is a modular system, which consists of a feature extraction module and a classification (recognition) module. The feature extraction falls into two main categories: Expertise-based Feature Extraction (EXFE), which implements feature extraction by use of an algorithm, based on expertise or data statistics, and Embedded Feature Extraction (EMFE), in which the feature extraction is parameterized and optimized jointly with the classifier. The EXFE, which enjoys the most widespread use in speech recognition, highly depends on the current available knowledge. This knowledge is usually not complete. More alarming in the EXFE scheme is the fact that features are designed based on a criterion which is not directly linked with the classification criterion. This shortcoming of the EXFE technique is not encountered in the EMFE framework since the feature extractor and the classifier are optimized under the same criterion. The goal of this report was the re-designing of standard feature extractor by EMFE where the criteria to be optimized is simply the error rate of the recognizer in the speech recognition framework. This can be viewed as a new framework for feature analysis in light of their discriminative capabilities.

The usefulness of discriminant features was shown in Chapter 3, where MSE-derived features outperformed traditional speech parameterization based on cepstral analysis. In this experiment, a Neural Network was used as pre-processor for a Maximum-Likelihood-based HMM. Consequently, this experiment provided a test on the usefulness of discriminant features in conventional recognizer design (in which the feature extractor and the classifier are designed separately). The Neural Network used in this experiment was the Large-phonetic Time Delay Neural Network (LTDNN) [Sawai *et al.*, 1989]. The LTDNN was trained to discriminate phonemes using the MSE criterion and its output was treated as feature vector for HMM. The LTDNN-features were compared to classical features, which consisted of LPC cepstral coefficients, delta cepstral coefficients and energy for recognizing Japanese phonemes uttered by one male professional announcer. The 24-dimensional discriminant feature-vector achieved 96.1% recognition rate in average whereas the 37-dimensional cepstral features realized 89.0%. These results emphasize the importance of discriminant features, even in the context of Maximum Likelihood Modeling.

This was a preliminary test on the usefulness of discriminant features as compared to expertise-based features in the conventional framework of design. Still, it was a limited approach, since the feature extraction was optimized by a criterion different from the recognition target. As reviewed in Chapter 2, the best criterion for optimal recognizer design is to minimize the probability of error (error rate). Discriminant criteria such as the MSE or MMI are not directly related to the error minimization of the system. That is, MSE and MMI are not a monotonic function of the classification error rate of the system.

Among the various discriminative training methodologies, the framework of the Minimum Classification Error/Generalized Probabilistic Descent method (MCE/GPD) provides the basis for implementing discriminative training aiming at reducing a smooth error count measure, which is closely related to the error rate of the system. That is, the 0-1 classification cost (which counts the error of the system) is approximated by a smooth function, at least first-order differentiable such as a sigmoid, subject to optimization. This smooth error is a function of the recognizer discriminant functions, which signifies that the classifier decision is embedded in the optimization criterion [Katagiri *et al.*, 1990; Katagiri *et al.*, 1991b].

The GPD framework also provides an optimization method that can be carried out to minimize an expected loss, which as shown in Chapter 4, is related to the probability of error. The optimization technique belongs to the family of stochastic gradient descent methods and is asymptotically efficient. In practical application, the empirical loss function, defined as the average of the MCE loss across the data set, approximates the empirical error rate.

MCE/GPD, introduced as an alternative to the widespread use of Maximum Likelihood estimation of acoustic models, has been proven to be quite useful in various speech recognition tasks. All comparative studies have shown that MCE/GPD yields a clear gain in performance if contrasted with the MLE criterion. As was shown in Chapter 4, when compared to other discriminative training approaches, such as MSE or MMI, the MCE advantage is that it directly attempts to reduce the empirical error rate, given a training data set, thus leading to more efficiency in practical application tasks.

In Chapter 4, the Discriminative Feature Extraction (DFE) [Katagiri *et al.*, 1993; Biem *et al.*, 1997] approach was consequently formalized as an extension of the MCE criterion applied to the design of the feature extraction and the classifier, aiming at the the single purpose of minimizing system mistakes. Thus, in the DFE framework, the feature extractor is matched the classification stage for minimum error achievement.

Again, the MCE criterion aims at minimizing the expected loss function. However, in practice, only the empirical loss defined on the set of data is available. It was shown, within Chapter 5 that using a recognizer of finite VC-dimension guarantees uniform convergence in probability of a minimum of the empirical loss to the minimum of the expected loss when the number of data increases.

Chapter 6 discusses optimization methodologies in the DFE context. In particular, the Modular Generalized Probabilistic Descent Method (MGPD) and the Incremental Generalized Probabilistic Descent Method (IGPD) were proposed as optimization methodology in the DFE context. In speech recognition task, a straightforward use of the GPD algorithm may not lead to convergence due the different nature of the parameters involved (which leads to a complex error surface). The MGPD algorithm assigns different learning rate to each module and thus permits more flexibility during training phase. However, one is still faced with choosing the optimal learning rate for each module. The IGPD learning scheme assigns a private learning rate to each parameter of the recognizer and each learning rate is adjusted during the learning process, according to its contribution to the

overall learning progress.

After underlying the theoretical framework of DFE, assuming that the empirical cost function is a reasonable approximation of the expected cost function, the nest task was the re-design of selected speech recognition-oriented parameterization methods.

In Chapter 7, the DFE approach is applied to lifter design. Liftering consists of applying a window over cepstral parameters to extract useful information and remove useless ones. It is a standard feature extraction which was studied extensively during the 80's. Cepstrum coefficients contains various information such as the phonemic-identity of the sound and speaker characteristics. It is believed that lower cepstral terms are more concerned with phonemic-identity. Thus, applying a low-pass lifter is usually believed to be the appropriate solution. However, issues such the shape of the lifter, the cut-off cepstral terms are still under investigation. By designing a lifter through the DFE approach, one would realize, at least locally, an optimal lifter for the task. Thus, a lifter was designed by DFE using a Feed-Forward Neural Network architecture aiming at vowel recognition. Contrasting the DFE-based lifter with those based on models of speech production, shows how the DFE approach manages to extract relevant information as well as achieving better performance in contrast to standard lifters.

In Chapter 8, the DFE approach was applied to another speech parameterization technique, which is filter bank modeling. A set of Gaussian filters is iteratively optimized according to the back-end recognition errors. Various parameters, such as center-frequency, bandwidth, gain were adjusted leading to an optimal filter-bank model given the recognizer structure. A preliminary study made on a vowel recognition task shows how the filter bank performs feature extraction according to the degree of freedom allowed within the filter bank during the optimization process. When allowing the center frequency of each filter to be adjusted, the filters tend to move to formant-centric regions, which are acknowledged to be important features for vowels. This approach was then applied to an automatic switching operator system. The system is intended to recognize names over the telephone and forward calls to the recognized persons. In this task, DFE-optimized filter-bank achieved the lowest word error rate when adjusting the gain of each filters (around 5% error rate whereas a fixed filter bank using classical MCE achieved around 8% error rate).

Finally, DFE was applied in Chapter 9 to readjusting Mel frequency Cepstrum coefficients (MFCC). MFCC can be said to enjoy the most widely used feature extraction technique in the speech community. MFCC is based on perceptual findings: the Mel scale is based on human perception of pitch, which has no clear link with statistically-designed classifier. However, in various studies, MFCC has been shown to perform better than other speech parameterization techniques. In this Chapter, optimization of MFCC was done by readjusting the underlying filter bank in a manner similar to Chapter 8. Again, a preliminary study is made on vowel recognition, which shows that in certain regions of the spectrum, the DFE-based cepstrum coefficients (DFCC) departed from the standard ones and moved the filters to meaningful region of the spectrum. Here also, the degree of modification of the filter bank depends on the degree of freedom allowed in the filter bank.

DFCC was then applied to the directory assistance task. When applying the DFE approach to the recognition of words, two optimization criteria can be performed: a string-level optimization and a frame-level optimization. Both were tested by the DFE approach. Testing was done through a close test set (the testing data obeys similar recording condition with the training data) and an open test set (testing data comes from different recording condition in contrast with training data). At the string-level training, in the close test set, the best performance was achieved by DFCC (94.7% as compared to 91.2% for MFCC). In the open test set, DFCC yielded 61.6% whereas MFCC achieved 57.9%. At the frame-level optimization, using the close test set, DFCC realized 95.7% and MFCC, 94.7%. Frame-level training on the open test set gave 64.5% for DFCC and 59.9% for MFCC.

Lastly, the method was applied to isolated letter recognition, which contains the difficult E-set task. Here, DFCC refers the simultaneously optimizing the center frequencies, bandwidths and gains. Various strategies of DFE was investigated. For instance, application of DFCC to MLE estimation shows that DFCC can improve MLE performance. This is particularly the case on the E-set task: 52.59% for MFCC, in the context of 5 states and 5 mixtures word-HMM, and 55.00% for DFCC. Using the MCE criteria, classical MCE on MFCC realized 66.85% while DFE achieved 68.70% on the E-set task. However result on the overall database are rather close.

It was also examined how to integrate delta parameters within the DFE process. This can be done within two strategies: calculating delta parameters using pre-optimized static DFCC or embedding the delta derivation process within DFE. Although the first method displayed better performance on the E-set task, in general there is no conclusive affirmation and the experiment requires further investigation.

10.2 Further extensions

The work presented in this report can be extended by one of the following suggestions.

10.2.1 Application to large vocabulary

In a large vocabulary speech recognition task, a method to integrate the language constraint into the learning scheme, down to the parameters of the feature extraction is provided by DFE. However, a cheap way to apply DFE for large task is required. That is, for large database, performing the DFE optimization can be extremely tedious. One direction for reducing the computational cost is to perform DFE on a carefully chosen subset of the database, which can be said to be representative of the task.

Another method is to optimize the feature extraction on a particular phonemic class (for instance, considering only broad-class categories such vowels, semi-vowels, fricatives, stops, ...) and, in the the next stage, use the derived feature extractor for acoustic model estimation through MLE or MCE. This scheme can be used as an approximated DFE approach, designed particularly to handle the computational cost which characterizes large vocabulary speech recognition.

10.2.2 Accurate optimization

It is worth investigating other optimization methods other than gradient -descent. Even if gradient-based optimization is rather fast and straightforward, an optimization scheme that guarantees convergence to a global minima is of interest. Simulated annealing and genetic algorithms are obvious candidates.

10.3 Few Potential Applications

In this report, application of the DFE method was shown in few selected tasks. The focus was particularly on the re-estimation of lifters and a filter-banks which are the basis of most speech parameterization techniques. However, the same approach can be applied using other feature extraction techniques. Here, some potential applications are briefly discussed.

10.3.1 DFE application to time-frequency resolution

All experiments described in this report was done within a preset time-frequency resolution provided by the short-time Fourier transform. Time-frequency resolution can be improved by an appropriate choice of the sliding window or by making use a wavelet transform.

DFE application to the short-time window selection

Given a speech signal $s(m)$, a short short-time window $w(n)$ transforms the signal into

$$s(m,n) = s(m+n)w(n) \quad 0 \leq n \leq N-1, \tag{10.1}$$

where $s(n,m)$ denotes the n-th new speech sample at time m N is the preset length of the window.

In this report, a Hamming window was used as a sliding window. However, the choice of the Hamming window is not guaranteed to be the best. It has been shown in various studies that the resolution power of short-time Fourier transform depends on the choice of the sliding window [Rabiner and Schafer, 1978]. Since the effect of this weighting in the time -domain is equivalent to a convolution process in the frequency domain, the choice of window function affects both the frequency resolution of the selected speech segment (the main lope of the window spectrum must be narrow and sharp) and introduces spurious distortion outside the main lobe due to the convolution process. These two requirements are mutually exclusive. However, the DFE approach can be used to optimally select the window which provides the best performance for the recognizer structure at hand. This can be done in the following way.

Most window are a special case of a more generalized window form, known as the Generalized Cosine Window, which is defined as

$$w(n) = A - B\cos\left(2\pi\frac{n}{N-1}\right) + C\cos\left(2\pi\frac{2n}{N-1}\right) \tag{10.2}$$

where A, B and C are constant that determine a particular window type and n refers to the digitized time lag.

A Hamming window is defined by choosing $A = 0.54, B = 0.46$ and $C = 0$. A Hanning is implemented with $A = 0.5, B = 0.5$ and $C = 0$ and for a Blackman window, we have $A = 0.42, B = 0.5$ and $C = 0.08$. DFE can thus be used to find the optimal configuration of A, B and C.

DFE application to the wavelet transform

The frame based approach, which imposes a fixed sliding window, does not provide appropriate temporal resolution of the waveform. An alternative to the use of a fixed-length window frame is the wavelet transform. For decades, the wavelet transform has been introduced by mathematicians as alternative to the use of the Fourier transform, well-suited to non-stationary signals. Recently, the wavelet transform has found wide application in speech processing, sonar estimation and image analysis [Grossman and Kronland-Martinet, 1988].

The basic idea of the wavelet transform is to use an invertible linear transform as a time-frequency analyzer, which integrates a time-scaling process. For a signal $s(t)$, the wavelet transform is defined as

$$S(t,a) = \int s(\tau)\frac{1}{\sqrt{|a|}}\psi\left(\frac{\tau - t}{a}\right)d\tau. \tag{10.3}$$

The transformation above can be viewed as a decomposition of the signal along the basis functions

$$\frac{1}{\sqrt{|a|}}\psi\left(\frac{t-b}{a}\right)$$

called "wavelet". For $|a| > 0$, there is a time expansion and for $|a| < 0$, there is a time contraction. A classical choice of the wavelet is the Morlet wavelet, defined as

$$\frac{1}{\sqrt{|a|}}\exp\left(\alpha\left(\frac{t-b}{a}\right)^2\right)e^{jwt} \tag{10.4}$$

where α is a positive constant.

In can be seen that the success of the wavelet transform depends on the value a and, in the case of the Morlet wavelet on α as well. The DFE approach can use in the selection of these parameters.

10.3.2 DFE for noise adaptation

In the presence of noise, specific adaptation techniques have been shown to improve performance since, in this situation, there is a clear mismatch between training and testing conditions. Among the many suggestions for reducing this mismatch, the technique of spectral transformation as proposed by [Mokbel and Chollet, 1991; Mokbel, 1992], performs a linear transformation of the clean speech unto noisy speech, using a least-mean squared error criteria. This transformation, however, is done on the cepstral domain (e.g. MFCC). One alternative is to use a method similar to the one presented in Chapter 9 or in [Woundenberg *et al.*, 1997]. That is optimizing the filter-bank itself for spectral adaptation.

10.3.3 DFE within LPC-based model

Most of the work presented in this report relies on an FFT estimation of the spectrum. It goes without saying that the same method can be applied within the context of the LPC estimation of the speech spectrum.

Without going into detailed explanations, the DFE approach can be used to optimize the spectrum by adjusting the poles of the all-pole model $H(z)$ aiming at minimum error or adjusting the frequency scale by means a bilinear transformation.

10.3.4 DFE within digital filter design

DFE Application to LPC model can be viewed as a particular case of using the DFE approach to in designing a digital filter, in a manner similar to adaptive filter design [Widrow and Stearns, 1985].

The idea here is to replace the usual least-mean-squared criterion with an estimate of the error as provided by the acoustic models. The main problem is keeping with the stability of the filter. That is, keeping the pole inside the unity circle. This can be achieved by a specific parameter transformation, which maps the zero into an area of the unit circle.

10.4 Final Discussion

DFE is particularly useful when design resources are limited, such as in a realistic environment. A limited number of design samples often makes a classifier less accurate for unknown samples, resulting in a less robust system design. A conventional solution to this problem is to increase the statistical stability of (discriminant function) estimation by increasing the ratio of the number of design samples to the number of adjustable parameters the classifier has. However, as shown in statistics [Duda and Hart, 1973; Fukunaga, 1972] and artificial neural networks studies [Bishop, 1995], the issue has yet to be solved.

MCE/GPD has partly contributed to alleviating the problem; its innovation was the introduction of smoothness in estimating the error rate. The smoothness makes classification operations smoother and is equivalent to increasing the number of design samples, bringing higher robustness [Juang and Katagiri, 1992a]. However, increasing the smoothness also means increasing the parameters-to-data ratio, and therefore even MCE/GPD cannot be a sufficient solution.

The use of classification-oriented features clearly makes the classification decision easier and more accurate, and it can also achieve a more robust decision: The feature representation that is suitable for classification should appropriately suppress pattern variations irrelevant to the classification. Therefore, a desirable method of recognizer design needs to incorporate the design of the feature extractor in the classifier design. However, as shown in the literature, there has conventionally been no direct interaction, except for subspace-method (SM)-related cases, and artificial Neural Networks, between the the feature extractor and the classifier aiming at the same criteria. Thus,

a novel mathematical framework such the DFE, is clearly needed to realize for such interaction in practical application. Within, this report, DFE was simply applied to readjust few state-of-art speech feature extraction method. It goes without saying that better performance can be obtained by using a more complex feature extractor such as neural network.

Note that the gradient-based optimization is not essential for the formalization concept of DFE. An important point is to formalize the entire recognition process consistently and directly as a training procedure. However, the gradient method is still useful from the practical viewpoint of computation load. As well known, the method does not guarantee to achieving the global minimum status of the error surface. However, irrespective of this defect, experimental results demonstrated that the DFE implementation which aims at achieving at least a local optimal design with the gradient optimization, can successfully be carried out.

In the design of pattern recognizers, conventionally, recognition accuracy improvement is attempted by increasing the size of trainable parameters used for classification. However, as often criticized and as demonstrated in this report, such an approach cannot be an acceptable solution. A larger size classifier is time-consuming as well as resource-consuming. Moreover, increasing the classification capability often causes the over-learning problem. On the other hand, if samples are represented in a highly separable feature space, the process of classifying them can be rather simple and easy. A small size classifier would be sufficient for handling such patterns correctly. The experimental results in the report show that DFE realizes this efficient classifier by finding salient pattern characteristics, as manifested in the training data, thus alleviating the over-learning problem.

DFE can not circumvent, however, the over-learning problem completely: DFE is a data-driven training and its design result relies on the nature of finite samples available in the training stage. Thus, naturally, a careful design manner that entirely covers the task including training data preparation is severely required even in the use of DFE.

Since the problem setting of classification usually assumes that patterns to classify are represented *a priori* in the fixed feature space, over-learning in classifier design is often studied in the conventional approach where one attempts to alleviate it by controlling the representation capability of classifier. In contrast, since DFE designs the feature space itself in the data-driven manner, such a conventional approach may be inadequate for further analyses of the characteristics of DFE, such as its over-learning mechanism. A new mathematical framework is needed for analyzing the feature representation and the classification decision, both determined jointly by DFE. This can be an important future research issue.

One main purpose of the DFE design is to improve classification accuracy of a preset recognizer by appropriate selection of features through a computational training. However, this design framework should also be recognized for a another contribution: DFE can feed-back the knowledge acquired by machine-decision-oriented feature representation to traditional expertise regarding characteristics of input pattern and/or human capability.

Appendix A

Fundamentals of speech recognition

A.1 What is Speech

The speech signal is the main mean of communication between human beings. Building machines that can understand human speech is therefore a natural investigation. However, despite having been the subject of intensive research for the last 4 decades, speech recognition by machine still has long way to go. Speech is highly variable in a number of ways. One person voice is different from another's voice and even the same speaker can not utter the same word in exactly the same way on two occasions. Another problem is that the duration of a word change usually at utterance depending on the manner of pronunciation. Also, ambiguity is a problem: there is no acoustic difference between the words "to", "two" and "too". Furthermore, speech signals are continuous and in fluent conversation, there are no pauses between the words, which requires a segmentation to be taken place before speech analysis and decoding.

In general, the idea is to divide the problem into smaller problems. So, only some particular aspects of speech recognition are considered. Usually, the task is divided according to the back-end application. For instance, for a voice-controlled remote control system, one only needs to recognize few digit numbers. This problem falls into the context of isolated word recognition. That is, trying to recognized single word or a sequence of words separated by a period of silence (connected word recognition). Continuous speech recognition, however, does not assume predefined boundaries between words.

Restriction can also be made on the number of speakers using the system or the vocabulary in use. For instance, a speaker independent recognition requires that any speaker should have his voice recognized by the machine. If only selected speakers can get their voice recognized, the problem is said to be speaker dependent recognition. The number of words recognized by the system is obviously a decreasing function of the number of speakers.

A.2 Various Views of Speech

The way the speech signal shall be regarded before processing, is inherently linked to a specific interpretation regarding the nature of the speech signal. This gives rise to various acoustic-modeling methods.

A.2.1 Speech production

In this framework, the speech signal is regarded as the output of the speech production system, which carries the characteristic of the vocal apparatus such as vocal tract resonance, vibration of the vocal cords, or nasal cavities. The speech recognizer tries to separate influence of each part of the vocal apparatus and determine which one constitutes the invariant parameter able to be use for decoding.

A.2.2 Speech perception

Human speech perception is certainly the most astounding speech recognizer. Hence speech perception research can provide the basis for building accurate speech recognizers. Such a system tries to simulate the human auditory reception process by extracting speech parameters and classifying them as it is done in the ears, the auditory nerves and the brain. The difficulties of such an approach is that it relies on psychoacoustics findings, which itself depends on sophisticated instrumentation.

A.2.3 Acoustic modeling

The speech is simply considered as a random mathematical signal The speech signal is passed through the general signal analysis techniques (Fourier frequency spectrum analysis, principal component analysis, statistical decision procedure, and other mathematical schemes) to establish the identity of the input speech. Decoding relies on statistical pattern recognition decision theory. This constitutes the most widely used decoding method in the speech field. The most widely powerful methods nowadays are based on hidden Markov models, which provide a powerful to model dynamic pattern.

A.3 Speech Recognition by Machine

The main processes underlying a speech recognition nowadays include the capture of speech utterance (data acquisition), the analysis of the raw speech into some suitable set of parameters (feature extraction), the comparison of these features with some previously stored knowledge (pattern classification) and the making of the decision.

Most systems rely on digital data representation, which signifies sampling the speech signal and quantizing it before processing. Special precautions must be made in choosing the sampling rate. Typical sampling rate range from 8 to 16 kHz with a bit storage of at least 12 bit/samples.

A anti-aliasing filter may be required to avoid under-sampling effect which may occurs when the signal bandwidth is unknown.

The data acquisition process is made, for example, by a microphone. Then a preprocessor transforms the speech signal into some useful representation (feature extraction). After isolating the word from the surrounding silence (segmentation), the unknown speech unit is compared with some pre-stored knowledge and the result is produced (classification/recognition).

All these processes are intertwined and inter-dependent and do not show a clear barrier between them. For instance, data acquisition also performed some form of signal parameterization thus, overlapping with the feature extraction process. However, it is always, possible, to represent each process as a "black box system", with clear a input-output subject to analysis.

A.3.1 Speech representation

Speech representation constitutes the most important process in the speech recognition since it affect the overall recognizer. Various constraints underly the speech parameterization process. The resulting features must carry the relevant information, given the task at hand. the first difficulty is encountered in "what" constitutes those relevant parameters given the high variability if the signal. An accurate feature extractor must also discard the irrelevant parameters. For instance, in multi-speaker word recognition task, information concerning speakers, manner of speaking, background noise, channel carriers, must be removed and only word-specific information be kept. This feature extraction is a difficult task and a great deal of research have been done in the last 50 years to find the invariant of the speech signals for an exhaustive search in speech signal processing techniques.

The representation of speech signals can be divided into two parts : the time-domain features and the frequency-domain features. Time-domain based speech representation treats the signal in the time domain to find regularities. Early speech recognizers relied on these features for recognition. Time-domain features could be:

- Energy or Amplitude of the signal.
- Autocorrelation function of the speech waveform.
- Linear predictive coding coefficients (LPC).
- Voiced/unvoiced, pitch value measured in the waveform.
- zero crossing rate.

Frequency domain representations decompose the signal along a set of basis functions. Spectrally-based features constitutes the most widely method of speech parameterization nowadays.

Fourier transforms-based features are the basis for most speech recognizers. Short-time Fourier transforms provides a straightforward method of estimating the spectral dynamic of the speech signal. Discrete Fourier Transform (DFT) and Linear Predictive coding (LPC) are the main methods of short time Fourier spectrum estimation. The speech signal is divided into overlapping frames

by a sliding window (especially chosen to smooth out the spectral resolution), and the spectral analysis is run at each window position producing a set of spectral vectors which represents the spectrogram of the speech signal. The spectral frames may be further processed to enhance some spectral region of the spectrum or derive new features. Examples of frequency-domain features are:

- Filter bank output.
- FFT-based filter bank.
- LPC cepstral parameter.
- FFT-based cepstral parameters (MFCC, BFCC, LFCC).

A.4 Reduction Techniques

The speech wave produces a great amount of data. Due to limited computer memory. It may be difficult to handle it all. There are data reduction techniques are proposed.

A.4.1 Vector quantization

This technique codes each frame of the speech data into a prestored reference vector. The set of this reference vectors is called a codebook. In other words, all the speech frames are represented by this codebook. For computational ease, the number of codes in the codebook is, in many cases, a power of $2(2^b)$ such that b is the number of bits in each vectors. It seems obvious to say that speech recognition performance depends on the production of this codebook. Generally this codebook is generated by a splitting procedure that minimizes the average distortion.

A.4.2 Time-normalization

In the field of word recognition, one of the most important problem is that as uttered a word rarely has the same duration in different realizations. A solution should be to normalize all each speech pattern in order to make them the same size. This is called " Linear time normalization". There is also a "Non-linear time normalization" proposed for larger vocabularies. DTW is one of such non-linear normalization technique, for example.

A.5 Speech decoding

After data acquisition and parameterization of the speech waveform, the next step is to recognize the pattern composed of speech features. In general, the recognition processed is passed through of classification of part the pattern into smaller unit such as phonemes and then recognition is done based on this classification. Sophisticated approaches exist for speech classification and/or decoding. The most widely used methods based on Hidden Markov Model and Neural Network.

The simplest way to build a speech recognizer might be to store speech knowledge through the form a pre-chosen templates, representing the words/phonemes/sentences and compare the incoming patterns with prestored pattern (called templates), using an appropriate distance measure. This approach constitutes early speech recognition system and is described below

A.5.1 Pattern matching

The basic principle of the pattern matching approach is to compare the unknown speech $\boldsymbol{x}$ with a set of reference patterns ($\boldsymbol{P}_i$). Several ways of comparison are given and each depends on the choice of the distance and the pattern alignment. Below is a description of the most common system of comparison, named Dynamic Time Warping (DTW).

A.5.2 Dynamic time warping (DTW)

Let's consider an unknown speech signal $\boldsymbol{x} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_j, ..., \boldsymbol{x}_T\}$ to be compared with a set of reference patterns $\boldsymbol{P}_i$, where each $\boldsymbol{P}_i$ is itself a N dimensional vector:

$$\boldsymbol{P}_i = [P_{i,1}, P_{i,2}, ..., P_{i,N}]^{\mathrm{T}}$$

The DTW algorithm aligns the sequence of vectors in a two-dimensional field and process the distance through an optimal path. This optimal path is found using the principle of the dynamic programming.

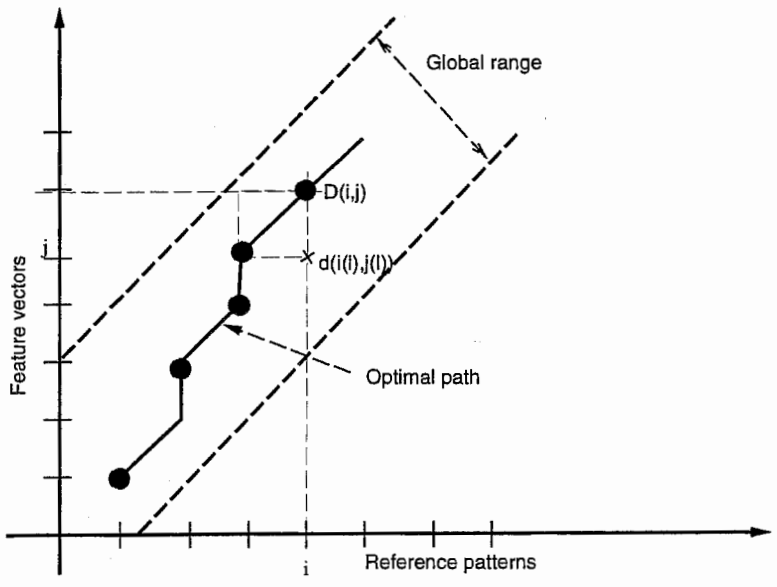


Figure A.1: Principle of DTW

The cumulated distance C at a given point (i, j) of a path is :

$$C(i,j) = \sum_{l=1}^{L} d(i(l), j(l))w(l)$$

where $(i(l), j(l))$ for $l = 1, ..., .L$ describe the different point of the path and $w(l)$ is a weighted function. So, the distance between the speech S and one reference pattern P_k is defined as :

$D(X, \boldsymbol{P}_k) = \min C(M, N)$ over all paths.

There are various constrains such as limiting the search of the path slope and many search algorithm (for example the Beam search) that provide less computationally expensive suboptimal path. The recursive dynamic programming algorithm that yields the optimal path is as follows:

$$D(i, j) = d(i, j) + [D(i-1, j-1), D(i, j-1), D(i-1, j)]$$

where $d(i, j)$ describe the local distance between the two frames i and j and $D(i, j)$ is the cumulated distance. Thus, dynamic programming involves local optimization which leads to a global solution. DTW is also used in connected words recognition. In this case, we must find the path through the boundaries separating the words. The principle however is the same.

A.5.3 Statistical decision rule

The pattern matching approach uses a metric as a similarity measures with the set of reference patterns. However, the speech signal is highly variable and the use of a simple distance is far from being sufficient to provides the required robustness in most applications. The alternative nowadays is to represent a given category C_j (a word/phone) by a probability densities functions $P(\boldsymbol{x}|C_j)$. Stochastic modeling approaches in speech recognition is the most widely used method pattern classification in particular under the Hidden Markov framework. Their best interest is due to the fact that there is no need for time-normalization and statistically-based framework makes room for robustness. The principle is as follows.

$$\boldsymbol{x} \text{ belongs to } C_i \text{ if } \Pr(C_i|\boldsymbol{x}) = \max_j \Pr(C_j|\boldsymbol{x}). \tag{A.1}$$

$\Pr(C_j|\boldsymbol{x})$ denotes the a posteriori probability to have category C_j, given the speech signal $\boldsymbol{x}$. However, these probabilities are not available directly and should estimated. This is usually a Maximum Likelihood Estimation is carried out to estimate the class-conditional probability density functions $\mathrm{p}(\boldsymbol{x}|C_j)$ using training data (Neural Network can also be used to estimate the a posteriori probabilities of a class given the data). Thus, given $\mathrm{p}(\boldsymbol{x}|C_j)$ and using the Bayes'rule

$$\Pr(C_j|\boldsymbol{x}) = \frac{\mathrm{p}(\boldsymbol{x}|C_j)\Pr(C_j)}{\Pr(\boldsymbol{x})} \tag{A.2}$$

and the computation of $\Pr(C_j)$ leads to the rule of "the Maximum Likelihood Classifier":

$$\Pr(\boldsymbol{x}|C_i) = \max_j \Pr(\boldsymbol{x}|C_j)$$

for equi-probable categories.

Hidden Markov Modeling (HMM) is the most popular stochastic method used nowadays on speech recognition. It popularity mainly originated from the fact that it is well suited to the dynamic nature of the speech signal since it divides the speech signal into states thus enabling local properties modeling. A set of algorithm such Expectation-Maximization (EM), with convergence proofs are available which make possible an iterative estimation of the model parameters, given available data.

A.6 Language Modeling

The output of the speech recognizer is often passed on to a Natural Language Processor system, which attempts to model the structure of the language and the conceptual relationships of words according to the application. The language model comprehends syntax (the language structure), semantics (the meaning of encoded language concepts) and a lexicon (the vocabulary). Modeling the syntax can be done in various ways. The most popular approaches uses parsers, formal grammar, finite-state language models and statistical models such as n-gram.

Appendix B

Back-propagation for classification

The back-propagation training algorithm is an iterative gradient algorithm designed to miminize an error criterion of a feed-forward network. Each category is assigned an output node. The largest output is chosen as decoded category of the recognizer. We suppose M categories.

1. Initialize the weights: set the weights to small values
2. Present an input vector $[x_1, x_2, ..., x_n]^{\mathrm{T}}$ knowing its category C_k. For MSE, get teacher output $[t_1, t_2, ..., t_M]^{\mathrm{T}}$; $t_j = \delta_{jk}$ for δ_{jk} kronecker notation.
3. Get corresponding output values $[o_1, o_2, ..., o_M]^{\mathrm{T}}$. For MCE, compute misclassification measure.

   $$d_k = -o_k + \overline{o}_k, \quad \overline{o}_k = \left[\frac{1}{M-1}\sum_{j\neq i}^{M} o_j^{\eta}\right]^{\frac{1}{\eta}}, \quad \eta > 0$$

4. Compute error E

   $$E = \begin{cases} \frac{1}{2}\sum_{j=1}^{M}(o_j - \delta_{jk})^2 & \text{for MSE} \\ \text{sigmoid}(d_k) & \text{for MCE} \end{cases} \tag{B.1}$$

5. Adjust weight, recursively starting from the output nodes back to the hidden nodes. The adjustment is as follows

   $$w_{ij}[\tau + 1] = w_{ij}[\tau] + \epsilon_\tau \dot{x}_i \nabla_j$$

   w_{ij} : weight from node i to node j at iteration τ; $\dot{x}_i$: output of node i; ϵ_τ : learning at iteration τ; ∇_j : error term (gradient) of node j;

   - if node j is an *output* node then
     For MSE:

     $$\nabla_j = o_j(1 - o_j)(t_j - o_j)$$

     For MCE

     $$\nabla_j = \begin{cases} E(1-E) & \text{for j = k} \\ -\frac{E(1-E)}{M-1}\left(\frac{o_j}{\overline{o}_k}\right)^{\eta-1} & \text{for } j \neq k \end{cases}$$

- if node j is an *internal node* node

$$\nabla_j = \dot{x}_j(1 - \dot{x}_j) \sum_l \nabla_l w_{jl}$$

where l spann all nodes linked with node j in the layer above node j.

6. go to step 2

Appendix C

Filter bank optimization by DFE

For a pattern $s_1^{\mathcal{T}} = \{s_1, \ldots, s_t, \ldots, s_{\mathcal{T}}\}$ where each $s_t = [s_{t,1}, \ldots, s_{t,f}, \ldots, s_{t,F}]^{\mathrm{T}}$ is a power-spectrum frame and $x_t = [x_{t,1}, \ldots, x_{t,i}, \ldots, x_{t,I}]^{\mathrm{T}}$, is the corresponding vector of log energies. f represents the FFT bins. We have

$$x_{t,i} = \log_{10}\left(\sum_{f \in B_i} w_{i,f} s_{t,f}\right),$$

for $i = 1, \ldots, I$, and

$$w_{i,f} = \alpha_i \exp\left[-\beta_i \left\{p(\gamma_i) - p(f)\right\}^2\right]$$

for $i = 1, \ldots, I$ where $w_{i,f}$ is the weight at frequency f provided by the i-th filter.

If ϕ is any filter bank parameter, the adaptation rule is

$$\phi[\tau + 1] \;=\; \exp\left(\log(\phi[\tau]) - \rho_\tau \mathbf{U2}\delta\overline{\phi}\right) \tag{C.1}$$

with

$$\delta\overline{\phi} = \begin{cases} \sum_{t=1}^{\mathcal{T}} \sum_{q=1}^{Q} \sum_{i=1}^{I} \cos\left(\frac{q\pi}{I}(i - \frac{1}{2})\right) \mathcal{I}_{t,q} \mathcal{O}_{t,i} & \text{if cepstrum transformation of size } Q. \\ \sum_{t=1}^{\mathcal{T}} \sum_{i=1}^{I} \mathcal{I}_{t,i} \mathcal{O}_{t,i} & \text{if no cepstrum transformation.} \end{cases} \tag{C.2}$$

- $\mathcal{I}_{t,i}$ is the classifier's i-th input derivative of the t-th feature-vector. $\mathcal{I}_{t,i} = \dfrac{\partial \ell\left(d_k\left(s_1^{\mathcal{T}}; \Phi\right)\right)}{\partial x_{t,i}}$ if no cepstrum transformation. $\mathcal{I}_{t,q} = \dfrac{\partial \ell\left(d_k\left(s_1^{\mathcal{T}}; \Phi\right)\right)}{\partial c_{t,q}}$ if there is cepstral tranformation. For Prototype-based or Gaussian Mixtures HMMs,

$$\mathcal{I}_{t,i} = -\sum_{j=1}^{M} \sum_{n=1}^{N_{\psi_t^j}} \delta r_{\psi_t^j, n, i}.$$

where $\delta r_{\psi_t^j, n, i}$ is the increment of the n-th prototype vector of state ψ_t^j occupied at the time t along the path of categoty C_j.

- $\mathcal{O}_{t,i}$ is the feature extractor i-th output derivative of the t-th feature-vector (the derivative of the filter bank output in channel i versus the filter bank parameter $\overline{\phi}$). $\mathcal{O}_{t,i}$ *does not depend* on the classifier.

$$\mathcal{O}_{t,i} = \frac{\partial x_{t,i}}{\partial \overline{\phi}} = \sum_{f=1}^{F} v_{t,i,f} \xi_{i,f} \tag{C.3}$$

with

$$v_{t,i,f} = \frac{s_{t,f}}{\log(10) exp_{10}(x_{t,i})} \tag{C.4}$$

and

$$\xi_{i,f} = \begin{cases} \boldsymbol{\chi}(i,\hat{\imath})\, \boldsymbol{\chi}(f,\hat{f}) w_{\hat{\imath},\hat{f}} & \text{if } \phi \text{ is the weight } w_{\hat{\imath},\hat{f}}. \\ 2\beta_{\hat{\imath}}\, \Gamma_{\hat{\imath}}\, (\Gamma_{\hat{\imath}} - p(f))\, w_{i,f}\, \boldsymbol{\chi}(i,\hat{\imath}) & \text{if } \phi \text{ is the center frequency } \Gamma_{\hat{\imath}} = p(\gamma_{\hat{\imath}}) \text{ of channel } \hat{\imath} \\ & \text{as expressed in the perceptual domain.} \\ -\beta_{\hat{\imath}}\, (p(\gamma_{\hat{\imath}}) - p(f))^2\, w_{i,f}\, \boldsymbol{\chi}(i,\hat{\imath}) & \text{if } \phi \text{ is the is the parameter } \beta_{\hat{\imath}} \\ & \text{controlling the bandwidth of channel } \hat{\imath}. \\ w_{i,f}\, \boldsymbol{\chi}(i,\hat{\imath}) & \text{if } \phi \text{ is parameter } \alpha_{\hat{\imath}} \text{of channel } \hat{\imath}\text{-th, which controls the gain.} \end{cases} \tag{C.5}$$

where

$$\boldsymbol{\chi}(a,b) = \begin{cases} 0 & \text{if } a \neq b \\ 1 & \text{otherwise} \end{cases} \tag{C.6}$$

Bibliography

[Aikawa *et al.*, 1993] K. Aikawa, H. Singer, H. Kawahara, and Y. Tohkura. A Dynamic Cepstrum Incorporrating Time-Frequency Masking and its Application to Continous Speech Recognition. In *Proc. IEEE ICASSP*, volume 2, pages 668–671, Apr. 1993.

[Amari, 1967] S. Amari. A Theory of Adaptive Pattern Classifiers. *IEEE Trans. on Electronic Computers*, EC-16(3):299–307, 1967.

[Andrews, 1968] D. R. Andrews. The IBM 1975, optical page reader: Part III. Recognition logic development. *IBM J. Res. Develop.*, 12, 1968.

[Asoh and Otsu, 1989] H. Asoh and N. Otsu. Nonlinear Data Analysis and Multilayer Perceptrons. In *Proc. of IJCNN-89*, volume II, pages 411–415, 1989.

[Ayer *et al.*, 1993] C.M Ayer, M. J. Hunt, and D.M Brookes. A Discriminatively Derived Linear Transformation for Improved Speech Recognition. In *Proc. EUROSPEECH'93*, Sep. 1993.

[Bachiani and Aikawa, 1994] M. Bachiani and K. Aikawa. Optimization of Time-Frequency Masking Filters Using Minimum Classification Error Criterion. In *Proc. IEEE ICASSP*, volume 2, pages 197–200, 1994.

[Bahl *et al.*, 1983] L. R. Bahl, P. F. Brown, P. V. de Souza, and R. L. Mercer. A Maximum Likelihood Approach to Continuous Speech Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5:179–190, 1983.

[Bahl *et al.*, 1986] Lalit R. Bahl, Peter F. Brown, Peter V. de Souza, and Robert L. Mercer. Maximum Mutual Information Estimation of Hidden Markov Model Parameters for Speech Recognition. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 49–52, 1986.

[Bahl *et al.*, 1988] L. R. Bahl, P. F. Brown, P. V. de Souza, and R. L. Mercer. A New Algorithm for the Estimation of Hidden Markov Models Parameters. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 493–496, 1988.

[Bartlett, 1993] P. L. Bartlett. Vapnik-Chervonenkis Dimension Bounds for Two-and Three-Layer Networks. *Neural Computation*, 5(3):371–373, 1993.

[Battiti, 1992] T. Battiti. First- and Second-Order Methods for Learning: Between Steepest Descent and Newton's Method. *Neural Computation*, 4(2):141–166, 1992.

[Baum and Haussler, 1989] E. B. Baum and D. Haussler. What Size Net Gives Valid Generalization? *Neural Computation*, 1(1):151–160, 1989.

[Baum, 1972] L. E. Baum. An Inequality and Associated Maximization Technique in Statistical Estimation for Probabilistic Functions of Markov Processes. *Inequalities*, 3:1–8, 1972.

[Becker and Le Cun, 1989] S. Becker and Y. Le Cun. Improving the Convergence of Back-Propagation Learning with Second Order Methods. In D. Touretzky, G. Hinton, and T. Sejnowski, editors, *Proceedings of the 1988 Connectionist Models Summer School*, pages 29–37, Pittsburg 1988, 1989. Morgan Kaufmann, San Mateo.

[Bellman, 1961a] R. Bellman. *Adaptive Control Processes: A Guided Tour.* Princeton University Press, 1961.

[Bellman, 1961b] R. Bellman. *Dynamic Programming.* Princeton University Press, 1961.

[Bengio *et al.*, 1990a] Y. Bengio, R. Cardin, and R. De Mori. Speaker Independent Speech Recognition with Neural Networks and Speech Knowledge. In D.S. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2, pages 218–225, Denver 1989, 1990. Morgan Kaufmann, San Mateo.

[Bengio *et al.*, 1990b] Yoshua Bengio, Renato De Mori, Giovanni Flammia, and Ralf Kompe. Global Optimization of a Neural Network - Hidden Markov Model Hybrid. Technical Report TR-SOCS-90.22, McGill University, School of Computer Science, 3480 University street, H3A2A7, Montreal, Qc., Canada, December 1990.

[Bengio *et al.*, 1995] Yoshua Bengio, Yann LeCun, Craig Nohl, and Chris Burges. LeRec: A NN/HMM Hybrid for On-Line Handwriting Recognition. *Neural Computation*, 7(6):1289–1303, 1995.

[Biem and Katagiri, 1992a] A. Biem and S. Katagiri. Cepstrum Liftering Based on Minimum Classification Error. In *Technical Meeting of Institute of Electrical Information Communcation Engineering of Japan (IEICE)*, volume 92-126, pages 17–24, Jul. 1992.

[Biem and Katagiri, 1992b] A. Biem and S. Katagiri. Integrating Feature Extraction in the Classification Process: Lifter Design Using Minimum Classification Error. In *Proc. Acoustical Society of Japan*, Fall conference, pages 209–2109, Oct. 1992.

[Biem and Katagiri, 1993a] A. Biem and S. Katagiri. Designing Filter Bank by the Discriminative Feature Extraction Method. In *Proc. of International Worshop on Speech Processing*, pages 35–40, 1993.

[Biem and Katagiri, 1993b] A. Biem and S. Katagiri. Feature Extraction Based on Minimum Classification Error/Generalized Probabilistic Descent Method. In *Proc. IEEE ICASSP*, volume 2, pages 275–278, Apr. 1993.

[Biem and Katagiri, 1994] A. Biem and S. Katagiri. Filter Bank Design based on Discriminative Feature Extraction. In *Proc. IEEE ICASSP*, volume 1, pages 485–489, Apr. 1994.

[Biem and Katagiri, 1997a] A. Biem and S. Katagiri. Cepstrum-Based Filter-Bank Design Using Discriminative Feature Extraction Training at Various Levels. In *Proc. IEEE ICASSP*, volume 2, pages 1503–1506, Apr. 1997.

[Biem and Katagiri, 1997b] A. Biem and S. Katagiri. String- and Frame-level optimization of speech Recognizer Using Discriminative Feature Extraction. In *Proc. Acoustical Society of Japan*, Spring conference, pages 105–106, Mar. 1997.

[Biem and Sugiyama, 1991] A. Biem and M. Sugiyama. Study on Combining Hidden Markov Model and Neural Networks. Technical Report TR-I-174, ATR Interpreting Telephony Research Laboratories, 1991.

[Biem and Sugiyama, 1992] A. Biem and M. Sugiyama. A Hybrid Stochastic Connectionist Approach to Automatic Speech Recognition. In *Actes du 1er Colloque Africain en Informatique*, pages 605–613, Oct. 1992.

[Biem *et al.*, 1993] A. Biem, S. Katagiri, and B.-H. Juang. Discriminative Feature Extraction For Speech Recognition. In *Proc. IEEE Worshop on Neural Networks for Signal Processing*, pages 392–401, Sep. 1993.

[Biem *et al.*, 1995] A. Biem, E. McDermott, and S. Katagiri. A Discriminative Filter Bank Model for Speech Recognition. In *Proc. EUROSPEECH' 95*, volume 1, pages 545–548, Oct 1995.

[Biem *et al.*, 1997] A. Biem, S. Katagiri, and B.-H. Juang. Pattern Recognition based on Discriminative Feature Extraction. *IEEE Transactions on Signal Processing*, 45(02):500–504, 1997.

[Bimbot *et al.*, 1990] F. Bimbot, G. Chollet, and J.-P. Tubach. Phonetic Features Extraction Using Time-Delay Neural Networks. In *International Conference on Spoken Language Processing*, 1990.

[Bishop, 1995] C. M. Bishop. *Neural Network for Pattern Recognition.* Oxford University Press, 1995.

[Bocchieri and Wilpon, 1993] E. L. Bocchieri and J. G. Wilpon. Discriminative Feature Selection For Speech Recognition. *Computer Speech and Language*, 1993.

[Bottou and Gallinari, 1991] Léon Bottou and Patrick Gallinari. A Framework for the Cooperation of Learning Algorithms. In Richard P. Lippmann, John E. Moody, and David S. Touretzky, editors, *Advances in Neural Information Processing Systems*, volume 3, pages 781–788. Morgan Kaufmann Publishers, Inc., 1991.

[Bottou, 1991] L. Bottou. *Une Approche théorique de l'apprentissage connectionniste: application à la Reconnaissane de la Parole.* PhD thesis, Université de Paris Sud, 1991.

[Bourlard and Kamp, 1988] H. Bourlard and Y. Kamp. Auto-association by Multilayer Perceptrons and Singular Values Decomposition. *Biological Cybernetics*, 59:291–294, 1988.

[Bridle and Dodd, 1991] John S. Bridle and L. Dodd. An Alpha-Net Approach to Optimising Input Transformations for Continuous Speech Recognition. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 277–280, Toronto, Canada, May 1991.

[Bridle, 1990] John S. Bridle. Alpha-Nets: A Recurrent "Neural" Network Architecture with a Hidden Markov Model Interpretation. *Speech Communication*, Special Neurospeech issue 1990.

[Brown, 1987] P. F. Brown. *The Acoustic-Modeling Problem in Automatic Speech Recognition.* PhD thesis, CMU, 1987.

[Chang and Juang, 1992] P.-C. Chang and B.-H. Juang. Discriminative Template Training for Dynamic Programming Speech recognition. In *Proc. IEEE ICASSP*, volume 1, pages 493–496, 1992.

[Chen and Soong, 1994] J.-K. Chen and F. Soong. An N-best Candidates-based Discriminative Training for Speech Recognition Applications. *IEEE Transactions on Speech and Audio Processing*, Jan 1994.

[Chou *et al.*, 1992] W. Chou, B.-H. Juang, and C.-H. Lee. Segmental GPD training of HMM Based Speech Recognizer. In *Proc. IEEE ICASSP*, volume 1, pages 473–476, mar 1992.

[Chou *et al.*, 1993] W. Chou, C.-H. Lee, and B.-H. Juang. Minimum Error Rate Training based on N-best String Models. In *Proc. IEEE ICASSP*, volume 2, pages 652–655, 1993.

[Comon, 1994] P. Comon. Independant Component Analysis-A New Concept ? *IEEE Transactions on Signal Processing*, 36:287–314, 1994.

[Cover and Van Campenhout, 1977] T. M. Cover and J. M. Van Campenhout. On the possible orderings in the measurement selection problem. *IEEE Transactions on Systems, Man, and Cybernetics*, pages 657–661, 1977.

[Culioli, 1994] J. C. Culioli. *Introduction à l'optimisation.* Ellipses, 1994.

[Darken *et al.*, 1992] C. Darken, J. Chang, and J. Moody. Learning Rates Schedule for Faster Stochastic Gradient Search. In *Proc. IEEE Worshop on Neural Networks for Signal Processing*, volume II, pages 3–12, 1992.

[De la Torre *et al.*, 1996] A. De la Torre, A. M. Peinado, A. J. Rubio, and V. Sanchez. An Application of Minimum Classification Error to Feature Space Transformation for Speech Recognition. *Speech Communication*, 20:273–290, 1996.

[De la Torre *et al.*, 1997a] A. De la Torre, A. M. Peinado, A. J. Rubio, and P. Garcia. Discriminative Feature Extraction For Speech Recognition in Noise. In *Proc. Eurospeech'97*, volume 1, pages 291–294, Sep. 1997.

[De la Torre *et al.*, 1997b] A. De la Torre, A. M. Peinado, A. J. Rubio, and V. Sanchez. A DFE-Based Algorithm For Feature Selection in Speech Recognition. In *Proc. IEEE ICASSP*, volume 2, pages 1519–1522, Apr. 1997.

[Dempster *et al.*, 1977] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Royal Statistical Society Ser. B*, 39:1–38, 1977.

[Devijvier and Kittler, 1982] P. A. Devijvier and J. Kittler. *Pattern Rcognition: A Statistical Approach.* Prentice-Hall, 1982.

[Doddington, 1989] G. Doddington. Phonetically Sensitive Discriminants for Improved Speech Recognition. In *Proc. IEEE ICASSP*, pages 556–559, 1989.

[Doob, 1953] J. L. Doob. *Stochastic Processes.* New York: Wiley, 1953.

[Duda and Hart, 1973] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* Wiley Interscience Publications, 1973.

[Euler, 1995] S. Euler. Integrated Optimization of Feature Transformation for Speech Recognition. In *Proc. EUROSPEECH' 95*, volume 1, pages 109–112, Oct 1995.

[Fahlman, 1988] S. E. Fahlman. An Empirical Study of Learning Speech in Back-Propagation Networks. Technical report, Canergie Mellon University, 1988.

[Fant, 1973] G. Fant. *Speech Sound and Features.* MIT Press, 1973.

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

[Fu, 1968] K. S. Fu. *Sequential Method in Pattern Recognition and Machine Learning.* Academic Press, 1968.

[Fukunaga, 1972] K. Fukunaga. *Introduction to Statistical Pattern Recognition.* Academic Press, 1972.

[Furui, 1986] S. Furui. Speaker-Independant Isolated Word Recognition Using Dynamic Features of Speech Spectrum. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 34(1):52–59, Feb 1986.

[Gallinari *et al.*, 1991] P. Gallinari, S. Thiria, F. Badran, and F. F. Soulie. On the Relation between Discriminant Analysis and Multi-layer Perceptrons. *Neural Networks*, 4(3):349–360, 1991.

[Gauvain and Lee, 1991] J.-L. Gauvain and C.-H. Lee. Bayesian Learning of Gaussian Mixture Densities. In *Proc. ARPA Speech and Natural Language Workshop*, pages 272–277, Feb. 1991.

[Geman *et al.*, 1992] S. Geman, E. Bienenstock, and R. Doursat. Neural Networks and the Bias/Variance Dilemma. *Neural Computation*, 4(1):1–58, 1992.

[Ghitza, 1991] O. Ghitza. Auditory Nerve Representation as a Basis for Speech Processing. In S. Furui and M Sondhi, editors, *Advances in Speech Signal Processing*. Marcel Dekker, 1991.

[Girosi and Poggio, 1989] F. Girosi and T. Poggio. Representation Properties of Networks: Kolmogorov's Theorem Is Irrelevant. *Neural Computation*, 1(4):465–469, 1989.

[Gopalakrishnan *et al.*, 1988] P. S. Gopalakrishnan, S. P. Kanevsky, D. Nadas, D. Nahamoo, and M.A Picheny. Decoder Selection Based on Cross-Entropy. In *Proc. IEEE ICASSP*, volume I, pages 20–23, 1988.

[Grossman and Kronland-Martinet, 1988] A. Grossman and R. Kronland-Martinet. Time and Scale Representation Obtained Through Continuous Wavelet Transforms. In *Proc. Int. Conf, EUSIPCO'88*, Signal Processing IV: Theories and Applications, pages 475–482. Elsevier Science Publishers, 1988.

[Gupta *et al.*, 1987] V.N Gupta, M. Lennig, and P. Mermelstein. Integration of Acoustic Information in a Large Vocabulary Word Recognizer. In *Proc. IEEE ICASSP*, pages 697–700, 1987.

[Haffner *et al.*, 1989] P. Haffner, A. Waibel, H. Sawai, and K. Shikano. Fast Back Propagation Learning Methods for Large Phonemic Neural Networks. In Proc. IEEE ICASSP, editor, *European Conference on Speech Communication and Technology*, pages 553–556, Sp. 1989.

[Haffner *et al.*, 1991] P. Haffner, M. Franzini, and A. Waibel. Integrating Time Alignment and Neural Networks for High Performance Continuous Speech Recognition. In Proc. IEEE ICASSP, editor, *Proc. IEEE ICASSP*, pages 105–108, 1991.

[Haffner, 1994] P. Haffner. A New Probabilistic Framework for Connectionist Time Alignment. In *International Conference on Spoken Language Processing*, volume 3, pages 1559–1562, Yokohama, Japan, 1994.

[Hampshire and Waibel, 1990] J. Hampshire and A. Waibel. A Novel Objective Function for Improved Phoneme Recognition using Time-Dealy Neural Networks. *IEEE transactions on Neural Networks*, 1(2):216–228, 1990.

[Hanson and Wakita, 1986] B. Hanson and H. Wakita. Spectral Slope Based Distorsion Measure for All Poles Models of Speech. In *Proc. IEEE ICASSP*, pages 757–780, 1986.

[Hernando *et al.*, 1995] J. Hernando, J. Ayarte, and E. Monte. Optimization of Speech Parameter Weghting for CDHMM words recognition. In *Proc. EUROSPEECH' 95*, volume 1, pages 105–108, Oct 1995.

[Hinton *et al.*, 1986] G. E. Hinton, J. L. McClelland, and D. E. Rumelhart. Distributed Representations. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. I: Foundations*, chapter 3. Bradford Books/MIT Press, Cambridge, MA, 1986.

[Hoeffding, 1963] W. Hoeffding. Probabilities Inequalities for Some of Bounded Random Variable. *J. American Statist. Asoc.*, 58, 1963.

[Huang *et al.*, 1990] X. D. Huang, Y. Ariki, and M. A. Jack. *Hidden Markov Models For Speech Recognition.* Edinburgh University Press, 1990.

[Hunt and Lefèbvre, 1986] M. J. Hunt and C. Lefèbvre. Speech Recognition using a Cochlea Model. In *Proc. ICASSP'86*, pages 1979–1982, 1986.

[Hunt and Lefèbvre, 1989] M. J. Hunt and C. Lefèbvre. A Comparison of Several Acoustic Representations for Speech Recognition with Degraded and Undegraded Speech. In *Proc. ICASSP'89*, volume 2, pages 262–265, 1989.

[Huo and Lee, 1997] Q. Huo and C.-H. Lee. On-Line Adaptive Learning of the Continuous Density Hidden Markov Model Based on Approximate Recursive Bayes Estimate. *IEEE Transactions on Speech and Audio Processing*, 5(2):161–172, Mar. 1997.

[Iijima, 1989] Iijima. *Pattern Recognition Theory.* Morikita, 1989. In Japanese.

[Imai and Ando, 1992] T. Imai and A. Ando. An HMM Learning Algorithm for Minimizing an Error Function on all Training Data. *Proc. Acoustical Society of Japan*, 13(6), 1992.

[Intrator, 1990] N. Intrator. A Neural Network For Feature Extraction. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems*, pages 719–728. Morgan Kaufmann Publishers, Inc., CA: San Mateo, 1990.

[Itakura and Saito, 1970] F. Itakura and S. Saito. A Statistical Method for Estimation of Speech Spectral Density and Formant Frequency. *Electronics and Communications in Japan*, 53A:36–43, 1970.

[Itakura, 1975] F. I. Itakura. Minimum Prediction Residual Principle Applied to Speech Recognition. In *Proc. IEEE ICASSP*, volume 1, pages 67–72, feb 1975.

[Jacobs, 1988] R.A. Jacobs. Increased Rates of Convergence Through Learning Rate Adaptation. *Neural Networks*, 1:295–307, 1988.

[Jacobson *et al.*, 1961] R. Jacobson, G. Fant, and M. Halle. *Preliminaries to Speech Analysis: The Distinctive Features and their Correlates.* MIT Press, 1961.

[Jelinek, 1972] F. Jelinek. Continuous Speech Recognition by Statistical Method. *Proc. of the IEEE*, 64:3043–3054, 1972.

[Johansen and Johnsen, 1994] F. T. Johansen and M. H. Johnsen. Non-linear Input Transformation for Discriminative HMMs. In *Proc. IEEE ICASSP*, volume 1, pages 225–228, 1994.

[Juang and Katagiri, 1992a] B.-H. Juang and S. Katagiri. Discriminative Learning for Minimum Error Classification. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 40(12):3043–3054, Dec. 1992.

[Juang and Katagiri, 1992b] B.-H. Juang and S. Katagiri. Discriminative Training. *Proc. Acoustical Society of Japan*, 13(6):341–349, Nov. 1992.

[Juang and Rabiner, 1991] B.-H. Juang and L. Rabiner. The Segmental k-means Algorithm for Estimating Parameter of Hidden Markov Models. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 38(9):1639–1641, 1991.

[Juang *et al.*, 1986] B. H. Juang, L. R. Rabiner, and J. G. Wilpon. On the use of Bandpass Liftering in Speech Recognition. In *Proc. IEEE ICASSP*, volume 1, pages 765–768, Apr. 1986.

[Junqua and Wakita, 1993] J. C. Junqua and H. Wakita. Evaluation and Optimization of Perceptually-based Front-End. *IEEE Transactions on Speech and Audio Processing*, 1(1):476–479, 1993.

[Karhunen *et al.*, 1997] J. Karhunen, E. Oja, L. Wang, R. Vigario, and J. Joutsensalo. A Class of Neural Network for Independent Components Analysis. *IEEE Transactions on Neural Networks*, 8(3):486–504, May 1997.

[Katagiri and Lee, 1990] S. Katagiri and C. Lee. A New HMM/LVQ Hybrid Algorithm For Speech Recognition. *Proc. of IEEE Communications Society*, Dec. 1990.

[Katagiri *et al.*, 1990] S. Katagiri, C.-H. Lee, and B.-H. Juang. A Generalized Probabilistic Descent Method. In *Proc. Acoustical Society of Japan*, Fall conference, pages 141–142, Sep. 1990.

[Katagiri *et al.*, 1991a] S. Katagiri, C-H. Lee, and B-H. Juang. Discriminative Multilayer Feed-Forward Networks. In *Proc. IEEE Worshop on Neural Networks for Signal Processing*, pages 309–318, 1991.

[Katagiri *et al.*, 1991b] S. Katagiri, C-H. Lee, and B.-H. Juang. New Discriminative Training Algorithms Based on the Generalized Descent Method. In *Proc. IEEE Worshop on Neural Networks for Signal Processing*, pages 309–318, 1991.

[Katagiri *et al.*, 1993] S. Katagiri, B.-H. Juang, and A. Biem. Discriminative Feature Extraction. In R. J. Mammone, editor, *Artificial Neural Networks For Speech and Vision*. Chapman and Hall, 1993.

[Kohonen *et al.*, 1979] T. Kohonen, G. Németh, K-J. Bry, M. Jalanko, and H. Riittinen. Spectral Classification of Phonemes by Learning Subspaces. In *Proc. IEEE ICASSP*, pages 97–100, Apr. 1979.

[Komori and Katagiri, 1992] T. Komori and S. Katagiri. Application of a Generalized Probabilistic Descent Method To Dynamic Time Warping-based Speech Recognition. In *Proc. IEEE ICASSP*, volume 1, pages 497–500, mar 1992.

[Komori and Katagiri, 1995] T. Komori and S. Katagiri. A Novel Spotting-Based Approach to Continuous Speech Recognition: Minimum Error Classification of Keyword-Sequences. *Proc. Acoustical Society of Japan*, 16(3):147–157, June 1995.

[Kůrková, 1991] V. Kůrková. Kolmogorov's Theorem Is Relevant. *Neural Computation*, 3(4):617–622, 1991.

[Lee, 1989] K.-F Lee. *Automatic Speech Recognition - The Development of the Sphinx System.* Kluwer Academic Publisher, 1989.

[Leung *et al.*, 1993] C. Leung, B. Chigier, and R. Glass. A Comparative Study on Signal Representation and Classification Techniques for Speech Recognition. In *Proc. IEEE ICASSP*, volume 2, pages 680–683, 1993.

[Lin and Unbehauen, 1993] J. Lin and R. Unbehauen. On the Realization of a Kolmogorov Network. *Neural Computation*, 5(1):18–20, 1993.

[Linde *et al.*, 1980] Yoseph Linde, Andres Buzo, and Robert M. Gray. An Algorithm for Vector Quantizer Design. *IEEE Transactions on Communications*, pages 84–95, 1980.

[Lipmann, 1987] R. Lipmann. An Introduction to Computing with Neural Nets. *IEEE Magazine*, Apr. 1987.

[Liu *et al.*, 1995] C.-S Liu, C.-H Lee, W. Choi, B.-H Juang, and A. Rosenberg. A Study on Minimum Error Discriminative Training for Speaker Recognition. *Journal of the Acoustical Society of America*, 97(1):637–648, Jan 1995.

[Lowe and Webb, 1991] D. Lowe and A. Webb. Optimized Feature Extraction and the Bayes Decision in Feed-Forward Classifier Networks. *IEEE Transactions on PAMI*, 13(4):355–364, Apr. 1991.

[Makhoul, 1975] J. Makhoul. Linear Prediction: A Tutorial Review. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 63:561–580, 1975.

[Matsui and Furui, 1995] T. Matsui and S. Furui. A Study of Speaker Adaptation Based on Minimum Classification Error Training. In *Proc. EUROSPEECH'95*, pages 81–84, 1995.

[McDermott and Katagiri, 1991] E. McDermott and S. Katagiri. LVQ-based Shift-Tolerant Phoneme Recognition. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 39:1398–1411, 1991.

[McDermott and Katagiri, 1992] E. McDermott and S. Katagiri. Prototype-Based Discriminative Training For Various Speech Units. In *Proc. IEEE ICASSP*, volume 1, pages 417–420, Mar 1992.

[McDermott and Katagiri, 1994] E. McDermott and S. Katagiri. Prototype-based Minimum Classification Error/Generalized Probabilistic Descent for Various Speech Units. *Computer Speech and Language*, 8(8):351–368, Oct. 1994.

[McDermott, 1997] E. McDermott. *Discriminative Training for Speech Recognition.* PhD thesis, Waseda University, 1997.

[Mellouk and Gallinari, 1994] A. Mellouk and P. Gallinari. Discriminative Training For Improved Neural Prediction Systems. In *Proc. IEEE ICASSP*, volume I, pages 233–236, 1994.

[Mermelstein and Davis, 1980] P. Mermelstein and S. Davis. Comparaison of Parametric Representation for Monosyllabic Word Recognition in Continuously Spoken Sentences. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 28(4):357–366, Aug. 1980.

[Miyatake *et al.*, 1990] M. Miyatake, H. Sawai, Y. Minami, and K. Shikano. Integrated Training for Spotting Japanese Phonemes Using Large Phonemic Time-Delay Neural Networks. In *Proc. IEEE ICASSP*, volume 1, pages 449–452, 1990.

[Mokbel and Chollet, 1991] C. Mokbel and G. Chollet. Word Recognition in Car. Speech Enhancement/Spectral Transformation. In *Proc. IEEE ICASSP*, pages 925–928, 1991.

[Mokbel, 1992] C. Mokbel. *Reconnaissance de la parole dans le ruit: bruitage/debruitage.* PhD thesis, Ecole National Supérieure des Télécommunications, 1992.

[More, 1986] B. C. More. *Frequency Selectivity in Hearing.* Academic Press, 1986.

[Morgan and Bourlard, 1990] N. Morgan and H. Bourlard. Continuous Speech Recognition using Multilayer Perceptrons with Hidden Markov Models. In *Proc. IEEE ICASSP*, pages 413–416, 1990.

[Nadas *et al.*, 1988] A. Nadas, D. Nahamoo, and A. Picheny. On a Model-Robust Training Method for Speech Recognition. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 36(9):1432–1435, 1988.

[Nadas, 1983] A. Nadas. A Decision Theoretic Formulation of a Training Problem in Speech Recognition and a Comparison of Training by Unconditional Versus Conditional Maximum Likelihood. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 31(4):814–817, 1983.

[Neyman and Pearson, 1933] J. Neyman and E.S. Pearson. On the Problem of the Most Efficient Tests of Statistical Hypothesis. *Phil. Trans. Royal Soc. London*, pages 289–337, 1933.

[Niles and Silverman, 1990] L. T. Niles and H. F. Silverman. Combining Hidden Markov Model and Neural Network Classifiers. In *Proc. IEEE ICASSP*, pages 417–420, 1990.

[Noll, 1964] A. Noll. Short-Time Spectrum and Cepstrum Techniques for Vocal Pitch Detection. *Journal of the Acoustical Society of America*, 36:296–302, 1964.

[Normandin, 1991] Y. Normandin. *Hidden Markov Models, Maximum Mutual Information Estimation, and the Speech Recognition Problem.* PhD thesis, McGill University, Montreal, Department of Electrical Engineering, 1991.

[Ohyama *et al.*, 1981a] G. Ohyama, S. Katagiri, and K. Kido. A New Method of Cepstrum Analysis Using Comb Type Quefrency Window. *Journal of the Acoustical Society of Japan*, pages 135–139, 1981.

[Ohyama *et al.*, 1981b] G. Ohyama, S. Katagiri, and K. Kido. A New method of cepstrum analysis using comb type quefrency window. *Proc. Acoustical Society of Japan*, pages 135–139, 1981.

[Oja, 1978] E. Oja. *Subspace Method of Pattern Recognition.* Research Studies Press, England, 1978.

[Paliwal *et al.*, 1995] K. K. Paliwal, M. Bachiani, and Y. Sagisaka. Minimum Classification Error Training Algorithm for Feature Extractor and Pattern Classifier in Speech Recognition. In *Proc. EUROSPEECH'95*, volume 1, pages 541–544, 1995.

[Paliwal, 1982] K. K. Paliwal. On the Performance of Quefrency-Weighted Cepstral Coefficients in Vowels Recognition. *Speech Communication*, 1:151–154, May 1982.

[Paliwal, 1992] K.K. Paliwal. Dimensionality Reduction for the Enhanced Feature set for HMM Speech Recognizer. *Digital Signal Processing*, 2:882–885, 1992.

[Parsons, 1987] T. W. Parsons. *Voice and Speech Processing.* McGraw-Hill Book Company, 1987.

[Pathasarathy and Cooker, 1992] S. Pathasarathy and C. H. Cooker. On Automatic Estimation of Articulatory Parameters in Text-to-Speech System. *Computer Speech and Language*, 6:37–75, 1992.

[Pols, 1966] L. C. W. Pols. *Spectral Analysis and Indentification of Dutch Vowels in Monosyllabic Words.* PhD thesis, Free University, Amsterdam, 1966.

[R. de Mori and Laface, 1984] R. de Mori and P. Laface. On the Use of Phonetic Knowledge for Automatic Speech Recognition. In R. de Mori and C. Y. Suen, editors, *New Systems and Architectures for Automatic Speech Recognition and Synthesis*, volume 16 of *NATO ASI Series.* Springer-Verlag, 1984.

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

[Rabiner and Schafer, 1978] L. R. Rabiner and R. W. Schafer. *Digital Processing of Speech Signal.* Prentice-Hall Processing Series, 1978.

[Rabiner, 1989] L. R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. In Alex Waibel and Kai-Fu Lee, editors, *Readings in Speech Recognition*, pages 267–296. Morgan Kaufmann, 1989.

[Rahim and Lee, 1996] M. G. Rahim and C. H. Lee. Simultaneous ANN Feature and HMM Recognizer Design Using String-Based Minimum Classification Error (MCE) Training. In *International Conference on Spoken Language Processing*, pages 1824–1827, 1996.

[Rainton and Sagayama, 1992] D. Rainton and S. Sagayama. Word Level Minimum Error Training of Phoneme HMM. In *Proc. Acoustical Society of Japan*, pages 3–4, Mar 1992.

[Rajaseharan and Doddington, 1985] P. Rajaseharan and G. Doddington. Speech Recognition in the F16 Cockpit Using the Principal Spectral Components. *Proc. IEEE ICASSP*, pages 882–885, 1985.

[Rathinavelu and Deng, 1996] C. Rathinavelu and L. Deng. HMM-based Speech Recognition Using State-Dependent, Linear Transforms on Mel-Warped DFT Features. In *Proc. IEEE ICASSP*, volume 1, pages 9–12, May 1996.

[Richard and Lippmann, 1991] M. D. Richard and R. P. Lippmann. Neural Network Classifiers Estimate Bayesian *a posteriori* Probabilities. *Neural Computation*, 3(4):461–483, 1991.

[Robinson and Fallside, 1990] Tony Robinson and Frank Fallside. Phoneme Recognition from the TIMIT Database using Recurrent Error Propagation Networks. Technical Report CUED/F-INFENG/TR.42, Cambridge University Engineering Department, March 1990.

[Rumelhart *et al.*, 1986] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning Internal Representations by Error Propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. I: Foundations*, chapter 8. Bradford Books/MIT Press, Cambridge, MA, 1986.

[Sawai *et al.*, 1989] Hidefumi Sawai, Alex Waibel, Masanori Miyatake, and Kiyohiro Shikano. Spotting Japanese CV-Syllables and Phonemes using Time-Delay Neural Networks. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 25–28, 1989.

[Sawai, 1991] H. Sawai. TDNN-LR Continuous Speech Recognition System Using Adaptive Incremental TDNN Training. In *Proc. IEEE ICASSP*, volume 1, pages 53–55, 1991.

[Seneff, 1986] S. Seneff. A Computational Model for the Peripheral Auditory System. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 1983–1986, 1986.

[Shikano *et al.*, 1986] K. Shikano, K. Lee, and D. R. Reedy. Speaker Adaptation Through Vector Quantization. In *Proc. IEEE ICASSP*, Apr. 1986.

[Shroeter and Sondhi, 1994] J. Shroeter and M. Sondhi. Techniques for Estimating Vocal-Tract Shapes from Speech Signal. *IEEE Transactions on Speech and Audio*, 2(1), jan 1994.

[Sugiyama and Biem, 1991] M. Sugiyama and A. Biem. TDNN-HMM Approach to Phoneme Recognition. In *Proc. Acoustical Society of Japan*, pages 149–150, Oct. 1991. In Japanese.

[Sugiyama and Kurinami, 1992] M. Sugiyama and K. Kurinami. Minimum Classification Error Optimization For a Speaker Mapping Neural Network. In *Proc. IEEE Worshop on Neural Networks for Signal Processing*, volume II, pages 233–242, 1992.

[Sugiyama, 1981] M. Sugiyama. LPC Peak Weighted Spectral Matching Measures. *Electron. Commun. Jap*, 64, Mai 1981.

[Sutton, 1992] R. S. Sutton. Adapting Bias by Gradient-Descent: An Incremental Version of Delta-Bar-Delta. In *Proc. of AAAI-92*, 1992.

[Tohkura, 1987] Y. Tohkura. A Weighted Cepstral Distance Measure for Speech Recognition. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 35(10):301–309, Nov. 1987.

[Tseng *et al.*, 1987] H. P. Tseng, M. Sabin, and E. Lee. Fuzzy Vector Quantization Applied to Hidden Markov Modeling. In *Proc. IEEE ICASSP*, pages 641–644, 1987.

[Vapnik and Bottou, 1993] V. Vapnik and L. Bottou. Local Algorithms for Pattern Recognition and Dependencies. *Neural Computation*, 5(6):893–909, 1993.

[Vapnik and Chervonenski, 1971] V. N. Vapnik and A. Chervonenski. On the Uniform Convergence of Relative Frequencies of Events to their Probabilities. *Theory of Probabilities and Their Applications*, 16, 1971.

[Vapnik, 1982] V. Vapnik. *Estimation of Dependences Based on Empirical Data.* Springer Series in Statistics, 1982.

[Viterbi, 1967] A. J. Viterbi. Error Bound for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm. *IEEE Transactions on Information Theory*, 13(IT-13):260–269, 1967.

[Waibel *et al.*, 1987] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang. Phoneme Recognition Using Time–Delay Neural Networks. Technical report, ATR Interpreting Telephony Research Laboratories, October 1987.

[Wald, 1950] R. O. Wald. *Statistical Decision Function.* John Wiley, 1950.

[Watanabe and Katagiri, 1995] H. Watanabe and S. Katagiri. Discriminative Subspace Method for Minimum Error Pattern Recognition. In Proc. IEEE Worshop on Neural Networks for Signal Processing, editor, *Proc. IEEE Worshop on Neural Networks for Signal Processing*, pages 77–86, Sept. 1995.

[Watanabe and Katagiri, 1997] H. Watanabe and S. Katagiri. HMM Speech Recognizer Based on Discriminative Metric Design. In *Proc. IEEE ICASSP*, volume 4, pages 3237–3240, 1997.

[Watanabe *et al.*, 1967] S. Watanabe, P.F. Lambert, C.A. Kulikowski, and J.L. Buxton. Evaluation and Selection of Variables in Pattern Recognition. In J.T. Tou, editor, *Computer and Information Sciences II.* Academic Press, 1967.

[Watanabe *et al.*, 1995] H. Watanabe, T. Yamaguchi, and S. Katagiri. A Novel Approach to Pattern Recognition based on Discriminative Metric Design. In *Proc. IEEE Worshop on Neural Networks for Signal Processing*, pages 48–57, Sep. 1995.

[Watanabe *et al.*, 1996] H. Watanabe, A. Biem, and S. Katagiri. Toward Unified Design of Pattern Recognizer. In S. Usui, Y. Tohkura, S. Katagiri, and E. Wilson, editors, *Proc. IEEE Worshop on Neural Networks for Signal Processing*, pages 283–292, 1996.

[Watrous, 1987] R.L. Watrous. Learning Algorithms for Connectionist Networks: Applied Gradient Methods of Nonlinear Optimization. In M. Caudill and C. Butler, editors, *IEEE International Conference on Neural Networks*, volume 2, pages 619–627, San Diego 1987, 1987. IEEE, New York.

[Weiye and Compornelle, 1990] Weiye and Compornelle. TDNN Labeling for HMM Recognizer. *IEEE Magazine*, 1990.

[White, 1989] H. White. Learning in Artificial Neural Networks: A Statistical Perspective. *Neural Computation*, 1(4):425–464, 1989.

[Widrow and Stearns, 1985] B. Widrow and S. D. Stearns. *Adaptive Signal Processing.* Prentice-Hall, Inc, 1985.

[Woudenberg *et al.*, 1995] E. A. Woudenberg, E. McDermott, and S. Katagiri. A Telephone-based Recognition System Adaptively Trained Using Minimum Classification Error/Generalized Probabilistic Descent (MCE/GPD). In *Proc. Acoustical Society of Japan*, Spring conference, pages 87–88, Mar. 1995.

[Woundenberg *et al.*, 1997] E. Woundenberg, A. Biem, E. McDermott, and S. Katagiri. Efficient Normalization based Upon GPD. In *Proc. IEEE ICASSP*, volume 4, pages 3245–3248, Apr. 1997.

[Young, 1990] Steve J. Young. Competitive Training: A connectionist Approach to the Discriminative Training of Hidden Markov Models. Technical Report CUED/F-INFENG/TR.41, Cambridge University Engineering Department, Trumpington Street, Cambridge CB2 1PZ, UK, March 1990.

[Zwicker and Terhardt, 1980] E. Zwicker and E. Terhardt. Analytical Expression for Critical Band Rate and Critical Bandwidth as a Function of Frequency. *Journal of the Acoustical Society of America*, 68:1523–1525, Dec. 1980.