

**A GENERALIZATION OF THE MINIMUM  
CLASSIFICATION ERROR (MCE) TRAINING METHOD  
FOR SPEECH RECOGNITION AND DETECTION**

A Thesis  
Presented to  
The Academic Faculty

by

Qiang Fu

In Partial Fulfillment  
of the Requirements for the Degree  
Doctor of Philosophy in the  
School of Electrical and Computer Engineering

Georgia Institute of Technology  
April 2008

**A GENERALIZATION OF THE MINIMUM  
CLASSIFICATION ERROR (MCE) TRAINING METHOD  
FOR SPEECH RECOGNITION AND DETECTION**

Approved by:

Professor Biing-Hwang Juang, Advisor  
School of Electrical and Computer  
Engineering  
*Georgia Institute of Technology*

Professor Erik I. Verriest  
School of Electrical and Computer  
Engineering  
*Georgia Institute of Technology*

Professor Mark Clements  
School of Electrical and Computer  
Engineering  
*Georgia Institute of Technology*

Professor Ming Yuan  
School of Industrial and Systems  
Engineering  
*Georgia Institute of Technology*

Professor Chin-Hui Lee  
School of Electrical and Computer  
Engineering  
*Georgia Institute of Technology*

Date Approved: 4 December 2007

*To my beautiful wife and my family*

## ACKNOWLEDGEMENTS

I would like to express my appreciation to everyone who has supported and helped me during my Ph.D. career.

I would like to thank my advisor, Prof. Biing-Hwang Juang, for his invaluable instruction and guidance. He introduced me to the wonderful speech recognition world and always supported and educated me as a perfect teacher and an ideal colleague. I learned so much from him and could not expect to meet a better advisor.

I would like to thank Prof. Chin-Hui Lee and Prof. Clements for their helpful classes and instructions.

I would like to thank Prof. Erik I. Verriest and Prof. Ming Yuan for kindly taking over the task of being my committee members.

I would like to thank Keith May for his support of the computer facilities, and all the staff in the School of Electrical and Computer Engineering for their supporting work.

I would like to thank Li Deng, Xiaodong He, Jinyu Li, Peng Liu, Chengyuan Ma, Dwi Sianto Mansjur, Antonio Moreno-Daniel, Frank Soong, Jian-Lai Zhou and everyone in my research group for their thoughtful discussions and constructive suggestions.

I would like to thank Ate He, Rongqiang Hu, Zhensheng Jia, Gang Huang, Brett Matthew, Bo Pan, Hua Qian, Wei Zhang, Yong Zhao and everyone else for the time we spent together.

And finally, without the support of my beautiful and smart wife Jia, I would not have achieved this work. I would like to thank my parents and my grandfather. Their encouragement will always accompany me in my future career.

# TABLE OF CONTENTS

DEDICATION . . . . .	iii
ACKNOWLEDGEMENTS . . . . .	iv
LIST OF TABLES . . . . .	ix
LIST OF FIGURES . . . . .	xi
SUMMARY . . . . .	xiii
I INTRODUCTION . . . . .	1
II THE CONVENTIONAL PATTERN RECOGNITION PARADIGM . . . . .	5
2.1 Bayes Decision Theory for Pattern Recognition . . . . .	5
2.2 Conventional Speech Recognition Framework with Hidden Markov Modeling . . . . .	8
2.2.1 Acoustic Features . . . . .	9
2.2.2 Acoustic Modeling . . . . .	9
2.2.3 Language Modeling . . . . .	12
2.2.4 Recognition and Decoding . . . . .	13
2.3 Chapter Summary . . . . .	13
III DISCRIMINATIVE TRAINING METHODS AND THE MINIMUM CLASSIFICATION ERROR (MCE) METHOD . . . . .	14
3.1 Discriminative Training Methods for Speech Recognition . . . . .	15
3.2 The MCE Training Method . . . . .	17
3.2.1 Theory of the MCE Method . . . . .	18
3.2.2 Examples of Using the MCE Method on Speech Recognition . . . . .	19
3.3 Chapter Summary . . . . .	21
IV NON-UNIFORM ERROR COST CRITERIA FOR PATTERN RECOGNITION . . . . .	22
4.1 Motivation . . . . .	22
4.2 Bayes Decision Theory vs. Support Vector Machine . . . . .	24

4.3	Decision Policy with Non-Uniform Error Cost . . . . .	25
4.4	A Generalization of the MCE Method . . . . .	26
4.4.1	MCE with Non-Uniform Error Cost . . . . .	26
4.4.2	Optimization Method for the System Cost Function . . . . .	31
4.5	Gaussian Mixture Classifiers – An Example . . . . .	33
4.5.1	Modeling $g(X; \Lambda)$ as a function of the Non-Uniform Error Cost and $P(X C_i)$ as GMM . . . . .	34
4.5.2	Modeling $g(X; \Lambda)$ as GMM . . . . .	37
4.6	Experimental Results for General Pattern Recognition . . . . .	38
4.6.1	True Models and Simulation Settings . . . . .	39
4.6.2	Non-Uniform Error Cost Training Based on Correct Classifier Models . . . . .	42
4.6.3	Non-Uniform Error Cost Training for Mismatched Models . . . . .	47
4.6.4	Discussions . . . . .	50
4.7	Chapter Summary . . . . .	52
V	NON-UNIFORM ERROR COST CRITERIA FOR SPEECH RECOGNITION . . . . .	53
5.1	Revised Non-Uniform Error Cost Framework for ASR Tasks . . . . .	54
5.1.1	Smoothing of the Empirical Cost for Parameter Optimization . . . . .	54
5.1.2	Updating Equations for Weighted MCE Training Using GPD Optimization . . . . .	56
5.2	Applying the Non-Uniform Error Criterion to ASR Applications . . . . .	57
5.2.1	An Example for Non-Uniform Error Cost in ASR . . . . .	57
5.2.2	Training Scenarios and Weighting Strategies in ASR . . . . .	59
5.2.3	Weighted MCE and MPE/MWE Method . . . . .	62
5.3	Experimental Results for Speech Recognition . . . . .	64
5.3.1	Small Vocabulary Task – TIDIGITS . . . . .	65
5.3.2	Large Vocabulary Task – WSJ . . . . .	71
5.4	Chapter Summary . . . . .	76

VI	A NEW APPROACH TO SELECTING AND ORGANIZING RECOGNITION HYPOTHESES IN THE MCE METHOD . . . . .	78
6.1	Motivation . . . . .	78
6.2	The Phone-Discriminating MCE (P-MCE) Method . . . . .	80
6.2.1	Target and Competing Events . . . . .	80
6.2.2	Objective Function and Optimization Algorithm . . . . .	81
6.2.3	N-best List vs. Graph in Selecting Competing Tokens . . . . .	83
6.3	Experimental Results . . . . .	84
6.3.1	Baseline system . . . . .	84
6.3.2	P-MCE and Other MCE Methods . . . . .	85
6.3.3	MMI and MPE Methods . . . . .	85
6.4	Chapter Summary . . . . .	87
VII	DETECTION-BASED SPEECH RECOGNITION AND THE MINIMUM VERIFICATION ERROR (MVE) METHOD . . . . .	89
7.1	Detection-Based ASR and Minimum Verification Error (MVE) Method . . . . .	89
7.1.1	Introduction of Detection-Based ASR . . . . .	89
7.1.2	Minimum Verification Error (MVE) Method . . . . .	91
7.1.3	Phoneme Detector Training . . . . .	94
7.2	Rescoring Using MVE-Trained Detectors . . . . .	98
7.2.1	Variants of the MVE Training . . . . .	101
7.2.2	Rescoring Algorithms . . . . .	102
7.2.3	Experimental Results . . . . .	105
7.3	Chapter Summary . . . . .	110
VIII	CONCLUSION, CONTRIBUTIONS, AND FUTURE WORK . . . . .	112
8.1	Conclusion and Contributions . . . . .	112
8.1.1	The Non-Uniform Error Criterion . . . . .	112
8.1.2	A New Approach to Selecting and Organizing the Recognition Decisions in the MCE Method . . . . .	113
8.1.3	Discriminative Detector Design for Detection-Based Speech Recognition . . . . .	114

8.2 Future Work . . . . .	114
REFERENCES . . . . .	116
VITA . . . . .	123

## LIST OF TABLES

1	Phoneme classification accuracy (%) by the two-layer classifier . . . . .	21
2	The normalized error cost comparison between different modeling technologies for the scoring function $g(X; \Lambda)$ at the operating point of using 256 training data. . . . .	52
3	Performance comparison between the conventional MCE training method and the weighted MCE training method using the non-uniform error cost matrix as defined in Eq.(109) . . . . .	67
4	The confusion matrix of the baseline . . . . .	67
5	The confusion matrix of the weighted MCE models . . . . .	68
6	Performance comparison between the conventional MCE training method and the weighted MCE training method using the non-uniform error cost matrix for phone models . . . . .	72
7	Word Error Rate (WER) and Sentence Error Rate (SER) for WSJ0-eval (Nov-92) using the MCE method and the W-MCE method . . . . .	76
8	Phoneme recognition accuracy (%) for the conventional string-based MCE, graph-based MCE and P-MCE . . . . .	85
9	Values of the objective functions of the MMI and MPE methods . . . . .	86
10	Phoneme recognition accuracy for the MMI method, the MPE method, and the P-MCE method . . . . .	86
11	Comparison between different problem descriptions in speech recognition	91
12	Mapping rule for the 6-class category. . . . .	95
13	Mapping rule for the 14-class category. . . . .	96
14	Minimum total error rate (%) for the 6-class category based on grid search on threshold . . . . .	97
15	Minimum total error rate (%) for the 14-class category . . . . .	99
16	Minimum total error rate (%) for the 48-class category . . . . .	100
17	Baseline phone accuracy and upper bounds for intra-linguistic level rescoring. . . . .	107
18	Intra-linguistic level rescoring performance for different rescoring algorithms (The first row is the rescored accuracy and the second row is the relative improvement). . . . .	108

19	Intra-linguistic level rescoring performance for different detector training methods (The first row is the rescored accuracy and the second row is the relative improvement). . . . .	109
20	Baseline word accuracy and upper bounds with bigram for inter-linguistic level rescoring . . . . .	109
21	Inter-linguistic level rescoring performance using the best score-fusion method (RCM) in our experiments . . . . .	110
22	Inter-linguistic level rescoring performance using decision-fusion methods. . . . .	110

## LIST OF FIGURES

1	A simple example of the isolated phone classification using the MCE method . . . . .	20
2	GMM models and data samples. . . . .	39
3	Error cost and empirical Bayes risk computed using the true models and the EM-trained models. . . . .	40
4	Some performance metrics with uniform error cost matrix on the training and test set. . . . .	42
5	Uniform error cost for training and test set when modeling $g(X; \Lambda)$ as a function of the non-uniform error cost . . . . .	43
6	Non-uniform error cost for training and test set when modeling $g(X; \Lambda)$ as a function of the non-uniform error cost . . . . .	44
7	Performance over training iterations with 256 training data when modeling $g(X; \Lambda)$ as a function of the non-uniform error cost. . . . .	45
8	Uniform error cost for training and test set when modeling $g(X; \Lambda)$ as a 4-mixture GMM . . . . .	45
9	Non-uniform error cost for training and test set when modeling $g(X; \Lambda)$ as a 4-mixture GMM . . . . .	46
10	Performance over training iterations with 256 training data when modeling $g(X; \Lambda)$ as a 4-mixture GMM. . . . .	46
11	Uniform error cost for training and test set when modeling $g(X; \Lambda)$ as a 2-mixture GMM . . . . .	48
12	Non-uniform error cost for training and test set when modeling $g(X; \Lambda)$ as a 2-mixture GMM . . . . .	49
13	Performance over training iterations with 256 training data when modeling $g(X; \Lambda)$ as a 2-mixture GMM. . . . .	49
14	Uniform error cost for training and test set when modeling $g(X; \Lambda)$ as a 6-mixture GMM . . . . .	50
15	Non-uniform error cost for training and test set when modeling $g(X; \Lambda)$ as a 6-mixture GMM . . . . .	50
16	Performance over training iterations with 256 training data when modeling $g(X; \Lambda)$ as a 6-mixture GMM. . . . .	51
17	Illustration of the strict-boundary confidence measure and the GCM. . . . .	75

18	Illustration of creating competing training tokens for the P-MCE method.	81
19	ROC curves for the 6-class category. . . . .	97
20	ROC curves for 4 representative classes from the 14-class category. . .	98
21	The rescoring diagram using HMM-based detectors. . . . .	99

## SUMMARY

The model training algorithm is a critical component in the statistical pattern recognition approaches which are based on the Bayes decision theory. Conventional applications of the Bayes decision theory usually assume uniform error cost and result in a ubiquitous use of the maximum *a posteriori* (MAP) decision policy and the paradigm of distribution estimation as practice in the design of a statistical pattern recognition system. The minimum classification error (MCE) training method is proposed to overcome some substantial limitations for the conventional distribution estimation methods. In this thesis, three aspects of the MCE method are generalized. First, an optimal classifier/recognizer design framework is constructed, aiming at minimizing non-uniform error cost. A generalized training criterion named weighted MCE is proposed for pattern and speech recognition tasks with non-uniform error cost. Second, the MCE method for speech recognition tasks requires appropriate management of multiple recognition hypotheses for each data segment. A modified version of the MCE method with a new approach to selecting and organizing recognition hypotheses is proposed for continuous phoneme recognition. Third, the minimum verification error (MVE) method for detection-based automatic speech recognition (ASR) is studied. The MVE method can be viewed as a special version of the MCE method which aims at minimizing detection/verification errors. We present many experiments on pattern recognition and speech recognition tasks to justify the effectiveness of our generalizations.

# CHAPTER I

## INTRODUCTION

This thesis presents generalizations for the minimum classification error (MCE) training method [37][36][7][58][59][56][72][55] for pattern recognition problems. The generalizations address some fundamental issues as well as implementation techniques based on the classical Bayes decision theory [17]. In particular, generalizations for speech recognition and detection applications are extensively investigated.

The classical Bayes decision theory [17] is the foundation of statistical pattern recognition. Bayes' analysis of the pattern recognition problem is built upon the notion of an *expected* system performance, rather than the evaluation of any particular instances of recognition decisions. The theory assumes knowledge of the joint distribution of the pattern observation and the class identity. Conventional applications of the Bayes decision theory result in a ubiquitous use of the maximum *a posteriori* (MAP) decision policy and the paradigm of distribution estimation as a practice in the design of a statistical pattern recognition system. Usually, distribution estimation approaches follow the maximum likelihood (ML) estimation criterion [41][17] and employ the expectation-maximization (EM) algorithm [3][14][17] to optimize model parameters.

However, the optimality of the distribution estimation methods is often taken for granted. One well-known argument is that the lack of the functional form of real data distribution would hinder the accuracy of the distribution estimation. Furthermore, a “perfect” estimation of the probabilistic distribution can rarely be achieved in real applications due to insufficient training data [36]. Therefore, pursuing the best distribution estimation doesn't necessarily lead to the optimal classifier/recognizer design

in terms of commonly used system evaluation measures, such as an error rate.

To overcome these limitations, we need an optimal classifier design approach with the following features: First, the MAP rule cannot be correctly implemented if the *a posteriori* probability cannot be accurately estimated. As a result, no performance optimality can be ascertained. A better method is one that, given any class-dependent function for discrimination, is able to conduct effective model training even without any prior knowledge of the data distributions. Second, the method is constructed upon a direct relation between the system performance measure and the model parameters. One needs to be able to optimize the model parameters according to the system objective. Finally, the method should be able to asymptotically converge to the optimal classifier/recognizer in terms of the system evaluation objective as the size of the training data grows.

In the last two decades, the MCE method has become the dominant method as it has shown remarkable success in pattern recognition and speech recognition tasks. As a well-constructed paradigm which relates the system performance directly to the model parameters, the MCE framework points out an important direction to optimal classifier/recognizer design thus inspiring many potential research topics in recent years. The MCE training routine can be summarized as a four-step process [36][24], which is listed as follows:

1. Define the performance objective, the corresponding task evaluation measure, and the decision rule for a specific task;
2. Specify the target event (i.e., the true identity of the data), competing events (i.e., the incorrect hypotheses resulting from the recognizer), and the corresponding models (as well as the organization of the training events);
3. Construct the objective function;
4. Choose a suitable optimization method to estimate model parameters.

The algorithmic issues of the MCE method, particularly, the third and the fourth step in the above list, have been extensively studied [39][58][56][72][55][29][82]. The generalizations in this thesis mainly focus on the first two steps, which are the basis of the last two steps. The first step (i.e., *evaluation measure, training error definition and decision rule*) determines the formulation of the objective function and is the foundation of the MCE framework. The second step (i.e., *target and competing events selection and organization*) evaluates and organizes the data according to the MCE paradigm and is critically important in the execution of the system optimization procedure.

This thesis consists of three major generalizations for the MCE method. First, the conventional MAP decision rule is derived based on the assumption that the error cost function is *uniform* (i.e., the cost for each error is identical). However, the uniform error cost function is not always the best choice for evaluating a pattern recognition system. When the cost function is not uniform, the best decision policy is not necessarily the one that achieves the maximum *a posteriori* probability. We propose a new framework to design the optimal classifier with non-uniform error cost functions. Second, a new scheme is proposed to manage the recognition hypotheses in the MCE method (i.e., *target and competing events*) for continuous phoneme recognition. Third, the minimum verification error (MVE) method [45][69][71][19] is studied as a special version of the MCE method which aims at minimizing detection/verification errors. It provides a viable path to train accurate speech attribute detectors [51][54][31][19], which are the basis of reliable detection-based ASR systems [47].

This thesis is organized as follows: Chapter II introduces the background of the conventional pattern recognition paradigm based on the Bayes decision theory, the MAP rule and the distribution estimation methods. As an important application in

this approach, the conventional speech recognition method with hidden Markov modeling (HMM) [67][68] is also reviewed. Chapter III presents the development of other discriminative training (DT) methods for speech recognition and the conventional MCE method. Chapters IV and V introduce the most important work in this thesis - the non-uniform error cost training criteria and the corresponding weighted MCE method for various training scenarios. In particular, Chapter IV presents the fundamental framework and demonstrative experiments on general pattern recognition problems. Chapter V introduces the applications of employing the non-uniform criterion on speech recognition tasks. Chapter VI presents an efficient method to manage the recognition hypotheses in the MCE method for continuous phoneme recognition tasks. We present a study of the MVE method and its applications on the detection-based ASR in Chapter VII, showing that the methodology of the MCE method is also productive to solve detection/verification problems. Finally, the summary and the contributions of the entire thesis are presented in Chapter VIII.

## CHAPTER II

### THE CONVENTIONAL PATTERN RECOGNITION PARADIGM

#### *2.1 Bayes Decision Theory for Pattern Recognition*

The classical Bayes decision theory [17] is the foundation of the statistical approach to the problem of pattern recognition. Bayes' analysis of the pattern recognition problem is built upon the notion of an *expected* system performance, as opposed to the evaluation of any particular instances of recognition decisions. Consider a pattern recognition task involving  $M$  classes of events or patterns (e.g., the task of recognizing a handwritten digit with  $M = 10$ ). An unknown pattern, say  $X$ , is observed and recognized as belonging to one of the  $M$  classes. Thus, a recognizer is a function  $C$  that maps  $X$  to a class identity denoted by  $C_i$ , where  $i \in I_M = \{i, i = 1, 2, \dots, M\}$ . We denote this function as a decision function  $C(X)$ . Obviously, some decisions are likely to be correct while others are wrong, and correct decisions are preferred over wrong decisions. In other words, every decision is associated with a cost which can be expressed as an entry  $\epsilon_{ij}$  in an  $M \times M$  matrix where  $i, j \in I_M$ , signifying the cost in identifying a pattern from the  $j^{\text{th}}$  class as one of the  $i^{\text{th}}$  class. Suppose at our disposal we have the knowledge of the *a posteriori* probabilities  $P(C_i|X), \forall i \in I_M$ . Then, following the teaching of Bayes, given  $X$ , the conditional cost of making a decision of  $C(X) = C_i$  can be defined as [17]

$$R(C_i|X) = \sum_{j=1}^M \epsilon_{ij} P(C_j|X) \quad (1)$$

and the system performance in terms of the *expected* loss is

$$\mathcal{L} = E\{R(C(X)|X)\} = \int R(C(X)|X)p(X)dX \quad (2)$$

Traditionally, a simple error count is used as the cost of recognition decision with

$$\epsilon_{ij} = \begin{cases} 1, & i \neq j \\ 0, & i = j \end{cases} \quad (3)$$

which is a typical error cost function. The cost function of (3) is the most intuitive and prevalent performance measure in pattern recognition as it is related to the probability of error in simple terms. With the cost function of (3), (1) becomes

$$R(C_i|X) = \sum_{j \neq i}^M P(C_j|X) = 1 - P(C_i|X) \quad (4)$$

and if we institute the decision policy as

$$C(X) = \arg \min_i R(C_i|X) = \arg \min_i [1 - P(C_i|X)] = \arg \max_i P(C_i|X) \quad (5)$$

The expected loss of (2) will be minimized due to the fact that  $p(X)$  is non-negative. This is the ubiquitous maximum *a posteriori* (MAP) decision rule that guarantees minimum system cost, or Bayes risk [17] under the assumption that  $P(C_i|X)$  is exactly known.

Note that Eq.(5) can be re-written as

$$C(X) = \arg \max_i P(C_i|X) = \arg \max_i P(X|C_i)P(C_i)/P(X) \quad (6)$$

Since  $P(X)$  is not a function of the class index and, thus, has no effect of the MAP rule, Eq.(6) shows that the direct estimate of the posterior probability  $P(C_i|X)$  can be decomposed into two parts - the parameter estimation of the conditional probability  $P(X|C_i)$  and the prior probability  $P(C_i)$ .

The parameter estimation can be approached by many methods. Two estimation criteria are commonly used: the maximum likelihood (ML) criterion [41][17] and the Bayesian estimation criterion [41][17][46]. They represent two different views of estimating data distributions. The ML estimation criterion assumes that the parameters of data distributions are unknown but *fixed*, and the best estimate of the model parameters is the value to maximize the likelihood of the training data. In other words,

given a set of training data  $\Omega = \{X_1, X_2, \dots, X_N\}$  that is drawn from the distribution  $P(\Omega|\Lambda)$ , the ML estimation is to estimate the parameter  $\Lambda$  that maximizes the likelihood of the data set  $P(\Omega|\Lambda)$ , i.e.,

$$\hat{\Lambda}_{ML} = \arg \max_{\Lambda} P(\Omega|\Lambda) \quad (7)$$

On the other hand, the Bayesian estimation criterion views the model parameters as random variables with prior distributions, i.e.,

$$\hat{\Lambda}_{Bayesian} = \arg \max_{\Lambda} P(\Omega|\Lambda)g(\Lambda) \quad (8)$$

where  $g(\Lambda)$  is the prior probability of parameter  $\Lambda$ . In practice, the estimation methods based on the ML criterion are more popular because of two main reasons. First, they usually have good convergence properties when the number of training data increases. Second, the implementation of these methods is simpler than the methods based on the Bayesian estimation.

The expectation-maximization (EM) method [3][14][17] is a popular algorithm to obtain the ML estimate iteratively when the training data is incomplete or has missing values. The EM algorithm consists of two steps, the *expectation* step (E-step) and the *maximization* step (M-step). Assume a complete data set  $\Omega = \{X, Y\}$ , where  $X$  is the observed (or *incomplete*) data and  $Y$  is the *missing* data. The E-step computes the expectation of the complete data log-likelihood  $\log p(X, Y|\Lambda)$  given the incomplete data  $X$  and the current parameter estimate  $\Lambda'$ . By defining the auxiliary function as

$$Q(\Lambda, \Lambda') = E[\log p(X, Y|\Lambda)|X, \Lambda'] \quad (9)$$

we have the property that when  $Q(\Lambda, \Lambda') > Q(\Lambda', \Lambda')$ ,  $p(X|\Lambda) > p(X|\Lambda')$  [14][46], where  $\Lambda'$  represents the existing parameters and  $\Lambda$  represents the updated parameters. Hence, by iteratively maximizing the auxiliary function, we can reach a stationary point of the likelihood function of the observed data. The step applied to maximize the auxiliary function is called the M-step. The E-step and the M-step are repeated

as necessary until convergence. The converged solution is a stationary point solution which attains, at least, a local maximum, that is

$$\hat{\Lambda} = \arg \max_{\Lambda} Q(\Lambda, \Lambda') \quad (10)$$

## 2.2 *Conventional Speech Recognition Framework with Hidden Markov Modeling*

As an important pattern recognition application, conventional speech recognition is based on the Bayes decision theory and employs the decision rule in Eq.(6). The conditional probability,  $P(X|C_i)$ , is usually called the *acoustic model*, which models the distribution of observations (i.e., feature vectors extracted from speech signals) given correct class labels. The prior probability,  $P(C_i)$ , is usually approximated by the so-called *language model* which indicates the probability of the occurrence of the underlying training token (i.e., the training utterance).

In general, there are four major components in the conventional speech recognition paradigm. First, appropriate acoustic features need to be extracted from raw speech signals. Second, a suitable probabilistic model needs to be formulated to describe the distribution of the acoustic features, which is referred to as *acoustic modeling*. Third, we need to formulate a statistical model to represent the syntax information (and occurrence frequency) of speech units in the speech corpus. This procedure is called *language modeling*. Finally, the recognition procedure (often referred to as *decoding*) gives us the recognized word/phone sequences as required.

In speech recognition, because the recognized results are a sequence of speech units, we can rewrite the acoustic model as  $P(X|W)$  and the language model as  $P(W)$  for simplicity. The symbol  $W = (w^1, \dots, w^N)$  is the label of the corresponding observation  $X$ , which is usually a word or phone sequence.

### 2.2.1 Acoustic Features

To construct a good speech recognition system, raw speech signals need to be transformed into appropriate acoustic feature vectors for training and recognition purposes. Usually, high performance acoustic features which carry most linguistic information are computed using linear prediction or spectrum/cepstrum analysis. In particular, one kind of feature called mel-frequency cepstral coefficients (MFCC) [68] is widely used in speech recognition systems for reading speech corpus. Another one named perceptual linear prediction (PLP) [30] is popular in conversational telephony speech recognition tasks. In this thesis, our work is based on the MFCC feature vectors.

There exist methods to further improve the speech recognition performance by manipulating acoustic features. One commonly used method is to append the dynamic features (which usually refer to the first and second derivatives) after the original feature vectors. In addition, standard pattern classification techniques such as the traditional linear discriminant analysis (LDA) [28] and heteroscedastic linear discriminant analysis (HLDA) [44] can be used to create better features. Recently, Povey et.al [65] achieved great success in designing a mechanism so that acoustic feature vectors can be improved iteratively according to an objective which is highly related to the final system performance measure.

### 2.2.2 Acoustic Modeling

Conventionally, the objective of acoustic modeling is to accurately estimate conditional probabilities  $P(X|W)$ , where  $W$  is the label corresponding to a sequence of speech units. The formulation of the acoustic model is expected to fulfill several conditions to construct a good speech recognition system. First, speech is a time-varying signal with the property of short-time stationarity. Furthermore, certain dependency or memory exists in specific sound pairs due to articulatory or phonotactic and phonological constraints. Therefore, the acoustic model is expected to be

capable of modeling such a sequential structure as well as the short-time stationarity. Second, to balance computational complexity and system performance in speech recognition tasks, a practical strategy for acoustic modeling is to train statistical models for elementary speech units. For example, the speech unit could be “word” for small vocabulary speech recognition and “phoneme” for large vocabulary tasks. Therefore, a flexible formulation of the acoustic model is necessary so that models of speech units on higher linguistic levels can be easily formed by combining models on lower levels. In practice, the hidden Markov model (HMM) [67][68] satisfies these requirements and is shown to be an effective statistical model in characterizing the conditional probability  $P(X|W)$  for speech recognition tasks.

Consider a first-order  $N$ -state Markov chain with the initial probability  $\pi_i$  and a transition probability matrix,  $A = [a_{ij}]$ , where  $a_{ij}$  is the probability of making a transition from state  $i$  to state  $j$ . In speech recognition, we normally use a simplified version of the HMM model, which only allows a state to transit to itself or the next state (i.e.,  $a_{ij} \neq 0$  if  $i = j$  or  $i = j - 1$ ). Hence, there exist a starting state and an ending state, which simplify the expression of the initial probability into  $\pi$ . The skipping transition is only allowed for some special cases (e.g., for the model of silence).

In an HMM model, each state contains a statistical model  $b_j, j = 1, 2, \dots, N$ , which models the acoustic characteristics of a certain part of the corresponding speech unit. Let  $\mathbf{X} = (X_1, X_2, \dots, X_T)$  be a speech observation sequence where  $X_t$  is a feature vector computed within a specific time window at time frame  $t$ . The term  $b_{q_t}(X_t) = P(X_t|q_t, W)$  is the probability of the observation  $X_t$  given state  $q_t$  at time  $t$ . By defining a *state sequence* as  $\mathbf{q} = (q_0, q_1, \dots, q_T)$ , the probability of an HMM

model is evaluated as

$$\begin{aligned}
 P(\mathbf{X}|\pi, A, \{b_j\}_{j=1}^N) &= P(\mathbf{X}|\Lambda) = \sum_{\mathbf{q}} P(\mathbf{X}|\mathbf{q}, \Lambda)P(\mathbf{q}|\Lambda) \\
 &= \sum_{\mathbf{q}} \pi_{q_0} \prod_{t=1}^T a_{q_{t-1}q_t} b_{q_t}(X_t)
 \end{aligned} \tag{11}$$

where  $\Lambda = (\pi, A, \{b_j\}_{j=1}^N)$  is the parameter set of the model. Note that the observation distribution  $b_j(X)$  can be modeled in different ways (e.g., discrete probabilities [33], semi-continuous probability distributions [32], or continuous probability distributions [68]) depending on different tasks. For state-of-the-art speech recognition systems with widely used spectral features such as MFCC or PLP, continuous distributions such as Gaussian mixture density [17] are usually employed to model the observation distribution  $b_j(X)$ .

Hidden Markov modeling includes three basic problems: the evaluation problem in which we want to compute the probability given a model and a sequence of observations; the decoding problem in which we intend to find an optimal state sequence; and the estimation problem in which we attempt to optimize the model parameters to best represent a given observation sequence [67][68]. The estimation problem is the key of the research in acoustic modeling. Based on the Bayes decision theory and the MAP decision rule, a distribution estimation method for model training is easy to construct. As discussed before, the HMM model allows models on higher linguistic levels to be combined using lower level models. Therefore, the model for each training utterance  $P(\mathbf{X}|W)$  can be formed by simply concatenating appropriate word/phoneme models based on the label and the lexicon. The Baum-Welch algorithm [3][68] is a popular variant of the EM algorithm for training HMM parameters under the ML criterion; its auxiliary function can be written as:

$$Q(\Lambda, \Lambda') = \sum_{\mathbf{q}} \log P(\mathbf{X}, \mathbf{q}|\Lambda)P(\mathbf{X}, \mathbf{q}|\Lambda') \tag{12}$$

The training of the HMM parameters can be repeated iteratively following the routine

of the general EM algorithm.

### 2.2.3 Language Modeling

The language model  $P(W)$  is a statistical model which represents the syntax information and occurrence frequency of every speech unit in a speech corpus. A good language model is very critical in large vocabulary speech recognition systems. Though  $W$  can be a sequence of any kind of speech units, here we assume  $W = w^1, \dots, w^N$  to be a word sequence with  $N$  words without loss of generality.

The most popular language model in state-of-the-art speech recognition systems is the  $n$ -gram language model [68], which provides the occurrence probability of the underlying word given  $n - 1$  words prior to it. Assume  $w^m$  is the  $m^{\text{th}}$  word in word sequence  $W$ , and based on the Bayes' theorem,  $P(W)$  can be computed by

$$P(W) = P(w^1 \dots w^N) = P(w^1)P(w^2|w^1) \dots P(w^N|w^{N-1} \dots w^1) \quad (13)$$

However, it is very expensive to reliably estimate all conditional probabilities in the above equation. The prior knowledge of  $w^m$  can be approximated by

$$P(w^m) = P(w^m|w^{m-1}, \dots, w^1) \approx P(w^m|w^{m-1}, \dots, w^{m-n+1}) \quad (14)$$

For simplicity, we can let

$$P(w^m|w^{m-1}, \dots, w^{m-n+1}) = P(w^m|w^1, \dots, w^{m-1}) \text{ if } m < n. \quad (15)$$

Naturally, the simplest situation in  $n$ -gram language modeling is the case in which the occurrence of every speech unit obeys a uniform distribution. In other words, every word is equally possible to follow any arbitrary word. This technique is also referred to as 0-gram or a simple word loop.

In this thesis, we focus on acoustic modeling and use the standard 0-gram, unigram and bi-gram language models (i.e.,  $n = 0, 1, 2$ , respectively) created by the CMU statistical language model toolkit ([http://www.speech.cs.cmu.edu/SLM\\_info.html](http://www.speech.cs.cmu.edu/SLM_info.html)).

### 2.2.4 Recognition and Decoding

If we replace the conditional probability in Eq. (6) by the acoustic model described as of (11) , and replace the prior knowledge in (6) using the standard  $n$ -gram language model as (13), the recognition decision for a speech utterance  $W$  with word  $w^1$  to  $w^N$  based on the MAP rule becomes

$$W = \arg \max_W \left[ \sum_{\mathbf{q}} \pi_{q_0} \prod_{t=1}^T a_{q_{t-1}q_t} b_{q_t}(X_t) \right] \left[ \prod_{m=1}^N P(w^m | w^1, \dots, w^{m-n+1}) \right] \quad (16)$$

However, the computational complexity is very high because of the large volume of the state sequence  $\mathbf{q}$ . A reasonable simplification of (16) can be implemented based on the principle of the Viterbi decoding through the well-known dynamic programming procedure [68], where the total probability of the HMM model is approximated by the probability of the single “best” state sequence. Therefore, a more practical recognition rule can be written as:

$$W = \arg \max_W \left[ \max_{\mathbf{q}} \pi_{q_0} \prod_{t=1}^T a_{q_{t-1}q_t} b_{q_t}(X_t) \right] \left[ \prod_{m=1}^N P(w^m | w^1, \dots, w^{m-n+1}) \right] \quad (17)$$

The expense of computation in (17) can be further reduced by pruning searching pathes during Viterbi decoding.

## 2.3 Chapter Summary

In this chapter, we reviewed the conventional pattern recognition framework. We introduced the classical Bayes decision theory and the MAP decision rule in the context of the uniform error cost function. The ML and Bayesian estimation criteria as well as the EM algorithm were also revisited. In addition, the conventional speech recognition paradigm using HMM modeling were introduced.

## CHAPTER III

### DISCRIMINATIVE TRAINING METHODS AND THE MINIMUM CLASSIFICATION ERROR (MCE) METHOD

As we mentioned before, distribution estimation methods usually cannot lead to the optimal classifier design for two main reasons. First and most important, the precise implementation of the Bayes decision theory requires that the posterior probability,  $P(C_i|X)$  be available to the system. In practice, the knowledge of *a posteriori* distribution needs to be “learned” from labeled data, so as to embed the knowledge of the “ground truth” in the parameters for the recognition system to use. The MAP decision rule in Eq.(5) thus has led to the conventional paradigm of distribution estimation as a fundamental step towards the design of a pattern recognition system. However, an important distinction has to be made here about learned statistical knowledge and true distribution. Statistical learning normally involves optimization of the parameter values based on the given labeled data, but the *functional form* of the distribution is chosen or determined independently. For example, the true data distribution may be Rayleigh but a log-normal distribution function may be prescribed. This means that the learned knowledge embedded in the optimized parameter values still may not represent the right information about the distribution. Second, because the training data is normally inadequate, the accuracy of the distribution estimation cannot be assured even with the correct formulation of the distribution form.

To overcome these limitations of the distribution estimation approach to pattern recognition, discriminative training (DT) methods were proposed aiming at linking the optimization of the model parameters to a specific objective function, which is usually related to a prescribed system performance measure. The MCE method, by

formulating the objective function as an approximation of the system performance measure, provides a rigorous guideline of designing a classifier that optimizes the classification performance over the given set of labeled training data. In this chapter, we review the development of the DT methods and introduce the discipline of the MCE method.

### ***3.1 Discriminative Training Methods for Speech Recognition***

In contrast to the traditional estimation criteria which aims at maximizing the data likelihood (e.g., the ML criterion), discriminative training is a family of training criteria which usually aims at optimizing model parameters by minimizing/maximizing specific objective functions which are related to the system performance measure. As an alternative to the conventional distribution estimation methods, discriminative training methods arose and led to successful results in various automatic speech recognition (ASR) tasks. In general, there are three popular DT methods: the maximum mutual information (MMI), the MCE, and the minimum phone/word error (MPE/MWE) method. The MMI method aims at maximizing the mutual information between the acoustic observation  $X$  and its correct lexical symbol  $W$  [7]. Bahl et al. [2] applied the MMI method to speech recognition. Chow [11] integrated a decoded N-best list with MMI training for continuous speech recognition. Normandin [60] generalized the extended Baum-Welch (EBW) algorithm proposed by Gopalakrishnan et al. [27] to MMI parameter estimation. Kapadia et al. [38] exploited the MMI method on the TIMIT database [48] for continuous phone recognition. Valtchev et al. [76] initiated the application for large vocabulary continuous speech recognition (LVCSR) using the MMI approach. Povey presented many experimental results for using the MMI method on LVCSR tasks in his Ph.D. dissertation [64].

One important drawback of the MMI method is that there is no direct connection between system performance (which is usually defined by the error cost/risk)

and the optimization criterion. Juang et al. [37] first proposed the MCE method, which creates a direct link between the system performance measure and the given training data. Also, Juang et al. [37] employed the generalized probabilistic descent (GPD) method as an optimization method, which can be shown to converge to a local minimum under some mild conditions [1][39]. Furthermore, Chou et al. [9][10] proposed the segmental GPD algorithm and implemented the MCE training method on string-based speech recognition. Chou also summarized the development of the MCE method in [7] and compared the MCE criterion with the MMI criterion. Rosenberg et al. [71] proposed a special version of the MCE method for optimal detector design and applied it to speaker verification. McDermott et al. [58] demonstrated that the MCE criterion function commonly used for the discriminative design of pattern recognition systems is equivalent to a Parzen-window based estimate [17] of the theoretical classification risk. In [56], some other gradient methods to optimize the MCE objective function were investigated. By using the MCE method and gradient descent based optimization algorithms, McDermott et al. [59] achieved remarkable success in large vocabulary continuous speech recognition (LVCSR) tasks. Schluter et al. [72] and Machery et al. [55] applied the EBW algorithm to MCE training and also achieved impressive results in LVCSR tasks. Recently, He et al. [29] proposed a modified EBW algorithm for the MCE method that achieved better performance than conventional GPD optimization.

The MPE/MWE method, proposed by Povey et al. [66][64], uses a weighted recognition accuracy as the objective function. The weighting function is a term called “raw accuracy” that roughly measures the accuracy of the recognition hypotheses of the underlying training token. Povey [64] also used the EBW method as the optimization method. He conducted many experiments and showed that MPE/MWE consistently outperformed MMI. Also, the MPE/MWE method outperformed the MCE method in most of his LVCSR experiments [64]. Recent study showed that the

MCE method could achieve comparable performance with the MPE/MWE method in LVCSR tasks with appropriate parameter tuning [55].

We have witnessed great effort in the whole speech recognition community to compare, unify, and generalize these criteria [7][72]. Recent research indicated that it is possible to formulate many discriminative training methods under a unified framework [72].

In addition to the DT methods mentioned above, some new discriminative criteria such as large margin estimation (LME) [53], soft margin estimation (SME) [52], and minimum divergence (MD) [16] have been proposed. The LME method maximizes the minimum multi-class “margin” defined using correctly recognized data samples. In a generalization, Li et al. [52] tried to maximize the “soft margin” that is computed using both correctly and incorrectly recognized data samples. The objective function of the SME method has two targets for optimization. One is to minimize the empirical risk, and the other is to maximize the “margin”, which is related to classifier generalization. These two objectives are combined into one function for minimization. In the MD method, the Kullback-Leibler distance [17] between two HMM models is used to measure the distance between the corresponding speech units. It formulates a similar objective function as MPE/MWE but replaces the weighting function (i.e., the “raw accuracy”) by the distance between recognition hypotheses and data labels (which is represented by the KL distance between corresponding HMM models). All these methods achieved very good performance on the TIDIGITS database [49].

### ***3.2 The MCE Training Method***

The method of minimum classification error for pattern recognition system training is built upon a fundamental concept in which the system’s recognition decision on a given pattern/token (in the training set) is evaluated as part of the estimate of the overall system performance, which is defined as a smooth function of the system

parameters that can be optimized.

### 3.2.1 Theory of the MCE Method

Let  $g_j(X; \Lambda) \geq 0$  be the discriminant function for the  $j^{\text{th}}$  class,  $j = 1, 2, \dots, M$  where  $\Lambda$  is the parameter set that defines the function. If the *a posteriori* probability (as a function of  $X$ ) is available,  $g_j(X; \Lambda)$  can be chosen as  $P(C_j|X)$ . Assume that the usual (uniform or unweighted) error count of (3) is used as the cost function. The recognition decision is reached according to

$$C(X) = \arg \max_i g_i(X; \Lambda) \quad (18)$$

That is, the recognizer chooses the class that leads to the largest value among all discriminants evaluated on  $X$ . This decision rule has to be embedded in a performance function for parameter optimization over a given training set of data.

Suppose  $X \in C_j$ , and  $C(X) = \arg \max_{i, i \neq j} g_i(X; \Lambda)$ ; i.e., class  $i$  is the most competitive class, given the training token  $X$  with a label for class  $j$ . If  $g_j(X; \Lambda) \geq g_i(X; \Lambda)$ , no decision error is made; otherwise, an error has occurred. Define a misclassification measure for an  $j^{\text{th}}$  class token:

$$d_j(X; \Lambda) = -g_j(X; \Lambda) + G_j(X; \Lambda) \quad (19)$$

where

$$G_j(X; \Lambda) = \left\{ \frac{1}{M-1} \sum_{i \neq j} \exp\{g_i(X; \Lambda)\eta\} \right\}^{1/\eta} \quad (20)$$

represents a “smoothed” combination of the discriminants other than that of the label class  $j$ . Note that when  $\eta \rightarrow \infty$ ,

$$G_j(X; \Lambda) \rightarrow \max_{i, i \neq j} g_i(X; \Lambda) \quad (21)$$

With this definition,  $d_j(X; \Lambda) > 0$  implies a misclassification or misrecognition and  $d_j(X; \Lambda) \leq 0$  means a correct decision. When the misclassification measure is cast in

a sigmoid function, a token-based performance evaluation in the form of a smoothed error count is obtained:

$$l_j(X; \Lambda) = \frac{1}{1 + \exp(-\gamma d_j(X; \Lambda) + \theta)} \quad (22)$$

with the smoothing parameter  $\gamma$  and the threshold parameter  $\theta$ . Finally, the system performance is defined as the expected cost of error

$$\mathcal{L} = E\{l(X; \Lambda)\} \quad (23)$$

which is estimated empirically as

$$\mathcal{L} \leftarrow L = \frac{1}{N} \sum_{n=1}^N l_j(X_n; \Lambda), \quad X_n \in C_j \quad (24)$$

In the above expression,  $n$  denotes the token index in the training set and the identity label has been made implicit. The empirical loss  $L$  is a smooth function in  $\Lambda$  which can be optimized over the training set using gradient descent methods such as the generalized probabilistic descent algorithm (GPD) or its variants [39][36][57][59]. This method has been used extensively, with good results, in automatic speech recognition applications where the data distribution is considered difficult to model.

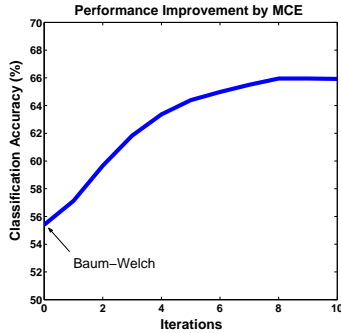
The method of minimum classification error directly evaluates the recognition system's decision on each presented token, while the conventional distribution estimation approach to pattern recognition attempts to learn the probability distribution function from the given data, without specifically knowing if a token will be misrecognized or not, not to mention which class the token may be incorrectly assigned to.

### 3.2.2 Examples of Using the MCE Method on Speech Recognition

In this section, the effectiveness of the MCE method is manifested by two simple experiments.

The first demonstration experiment is an isolated phoneme classification task on the TIMIT database [48]. The task is to classify 48 monophones, each of which

is modeled by a three-state left-to-right HMM model. The model is initiated by the Baum-Welch method using 39 dimensional mel-frequency cepstral coefficient (MFCC) vectors [68]. The MCE method is then used to improve performance. Though quite simple, it demonstrates the effect of applying the MCE on the isolated phone classification. Figure 1 shows that the performance rises from around 55% to 66% in 10 iterations.



**Figure 1:** A simple example of the isolated phone classification using the MCE method

The second application is to use the MCE on the category-dependent phone classification described in [35]. The comparison between the performance of the ML models and MCE-trained models is listed in Table 1. A *category*  $C_n$  is defined as the union of two or more classes  $w_1, \dots, w_i$ . In this case, the MCE method is applied to a two-layer phone classifier. The first layer of this two-layer classifier is a category classifier, which differentiates the broad phonetic categories. The second layer conducts the classification within categories. A category model is assumed to be the linear combination of the class models within that category, i.e.,

$$g_{C_n}(X; \Lambda_{C_n}) = \sum_{i, w_i \in C_n} \alpha_i g_{w_i}(X; \Lambda_{w_i}) \quad (25)$$

where  $g_{w_i}(X; \Lambda_{w_i})$  is an HMM model. The parameter  $\alpha_i$  is the linear weights for class  $w_i$  and needs to be updated as well as the normal HMM model parameters. The experimental configurations are identical to those in the last example.

**Table 1:** Phoneme classification accuracy (%) by the two-layer classifier

Category classification rate using ML models	79.25
Category classification rate after MCE training	80.06

### ***3.3 Chapter Summary***

In this chapter, we reviewed several popular discriminative training methods. Among these methods, the MCE method directly links the system performance measure and the model parameters and, thus, provides a rigorous guideline of designing a classifier that optimizes the classification performance over the given set of labeled training data. We reviewed the theory of the MCE method and presented two simple speech recognition experiments.

## CHAPTER IV

# NON-UNIFORM ERROR COST CRITERIA FOR PATTERN RECOGNITION

In this chapter, a new training criterion for the design of the optimal classifier for non-uniform error cost is proposed. The term “non-uniform cost” means that the risks between committing different errors are different. In this situation, the MAP rule does not hold and the distribution estimation methods are not applicable at all. We derive a new decision rule and develop viable implementation strategies based on the conventional MCE method for different scenarios (which we name it the *weighted MCE* method). The demonstration of the non-uniform error cost criteria for general pattern recognition problems is provided here, while we will introduce the applications of this paradigm on speech recognition in the next chapter.

### ***4.1 Motivation***

It has been well stated [36] that the lack of knowledge of real data distribution seriously degrades the accuracy of implementing the MAP rule. The other critical assumption in the MAP rule, however, is often taken for granted. The MAP rule is derived based on the *uniform* error cost function defined in Eq.(3). Nevertheless, the cost function of (3) is not the only choice for evaluating a pattern recognition system. When the cost function is not uniform, the best decision policy is not necessarily the one that achieves maximum *a posteriori* probability. For example, consider a 3-class case with the following cost matrix, *a posteriori* probability and conditional

risk vectors:

$$[\epsilon_{ij}] = \begin{bmatrix} 0 & 0.1 & 0.5 \\ 0.1 & 0 & 0.5 \\ 0.1 & 0.1 & 0 \end{bmatrix} \quad [P(C_j|X)] = \begin{bmatrix} 0.35 \\ 0.4 \\ 0.25 \end{bmatrix} \quad \text{and} \quad [R(C_i|X)] = \begin{bmatrix} 0.165 \\ 0.16 \\ 0.075 \end{bmatrix}$$

The MAP rule would have resulted in  $C(X) = C_2$  but

$$C_3 = \arg \min_{i=1,2,3} R(C_i|X) = \arg \min_{i=1,2,3} \sum_{j=1}^M \epsilon_{ij} P(C_j|X)$$

Clearly, from the cost matrix, one can see that a mistake on a third class pattern costs five times that of any other errors that leads to a discrepancy between a MAP decision and a decision that attempts to minimize the risk. This non-uniform or asymmetric error cost function is quite common in real world applications. For instance, a digit 1 mistaken as a digit 7 may cause more concern than 7 as 1 in financial transactions.

Thus, it is necessary to revisit the Bayes decision theory and discuss the validity of the conventional MAP policy when non-uniform error criteria are employed. The purpose of our work is therefore to reformulate a decision-theoretic framework for such applications in pattern recognition.

Our work here also takes into accounts the limitation associated with the conventional paradigm of pattern recognition system design via probability distribution estimation. We have alluded to the fact that Bayes's optimal decision assumes knowledge of the true *a posteriori* distribution, over classes and for any given observation  $X$ . The form of such a distribution function (or probability density function) for many real world data may not be known and the chosen function may not approximate the true distribution well, even with an infinite amount of data, leaving no assurance of any optimality in the pattern recognition task. This fact compounds the problem of recognition risk minimization. We thus attempt to provide what may be considered a reasonable system design methodology to follow under such circumstances.

## 4.2 *Bayes Decision Theory vs. Support Vector Machine*

Note that the Bayes decision theory is considered the foundation of statistical pattern recognition, which aims at solving the pattern classification problem by relating the probabilistic distributions to the expected performance of the recognition policy. One method recently popularized in the pattern recognition and machine learning community, however, is the support vector machine (SVM) [77][73], which tries to interpret the pattern recognition problem from the perspective of boundary optimization and margin maximization. The essence of the idea behind the SVM method is the so-called “transductive learning”, which conducts classification by directly computing the boundary with the largest “margin” and bypasses the step of training parameterized data distributions. The other contribution of the SVM method is to link the generalization ability of a classifier to the “capacity” of the classification model, which is represented by a quantity named VC dimension [77]. In the last decade, the SVM has been successfully applied to many pattern recognition applications [6][4][12].

Though the SVM method has achieved some success and presents a fresh view of the pattern recognition problem, we believe that it by no means replaces the classical Bayes decision theory in many situations. There are still several open issues in the SVM method. First, while some (loose) performance bounds exist for the SVM method, the asymptotic optimality of the classifier design (and the convergence of design) is not immediately ascertained. Second, though it is straightforward to apply SVM to two-class pattern recognition problems, the construction of the “best” margin in SVM for multi-class problems is relatively vague, especially when the margin has to be in the context of optimizing the system performance measure. Further more, if a constructed non-uniform error cost function is imposed on different errors, the appropriate weighting mechanism for different support vectors is complex and a clear guideline is lacking. Third, it is not easy to extend the SVM method to recognition problems with feature vectors presented to the classifier in sequential forms (e.g.,

continuous speech recognition) because of its rigid and expensive computational routine. With each new training token, the support vectors have to be re-calculated. Hence, flexible token-by-token sequential training, which is one of the advantages of our method, is hard to implement in the SVM methodology.

### 4.3 *Decision Policy with Non-Uniform Error Cost*

The conditional risk of (1) and the expected loss of (2) are general expressions of system performance without any particular conditions imposed on the error cost function  $\epsilon_{ij}$ . Again, since  $p(X)$  is non-negative,

$$\min_C \mathcal{L} = \min_C E\{R(C(X)|X)\} = \int \min_C R(C(X)|X)p(X)dX \quad (26)$$

and to achieve the minimum risk, the recognizer function must implement the following policy,

$$C(X) = \arg \min_{C_i} R(C_i|X) = \arg \min_{C_i} \sum_{j=1}^M \epsilon_{ij}P(C_j|X) \quad (27)$$

We call this the minimum risk (MR) or minimum cost (MC) rule. In practice, we generally require that  $\epsilon_{ij} = 0$  for  $i = j$  and  $\epsilon_{ij} \geq 0$  for  $i \neq j$ .

The MR rule of (27) does not lead to the MAP policy of (5) even if the knowledge of the true distribution (*a posteriori* probabilities) is available to the recognizer. Implementation of the MR rule requires multiplication of the cost matrix and the posterior probability vector, a direct result of the non-uniformity of the cost function.

In real applications, the *a posteriori* distribution needs to be learned through class identity labels as part of the conventional design paradigm for a recognition system. The estimated posterior distribution (for all classes and over the entire space of  $X$ ) may have to be substantially more accurate in the non-uniform cost case than in the uniform case, because one may argue that the rank order of posterior probabilities (as required in uniform cost situations) is likely to be less sensitive to small deviations than the quantities themselves (which is required in the non-uniform case). The

applicability of the distribution estimation paradigm for recognition system design thus warrants serious re-examinations. Note that the MR rule can be re-written as

$$C(X) = \arg \min_{C_i} \sum_{j=1}^M \epsilon_{ij} P(C_j|X) = \arg \min_{C_i} \sum_{j=1}^M \epsilon_{ij} P(X|C_j)P(C_j)/P(X) \quad (28)$$

for ease in applying some empirical knowledge in choosing the form of distribution.

#### 4.4 A Generalization of the MCE Method

We have reviewed the method of minimum classification error [37][36] as an alternative to the distribution estimation paradigm for pattern recognition system design in Chapter II. Clearly, the method of minimum classification error directly evaluates the recognition system's decision on each presented token, while the conventional distribution estimation approach to pattern recognition attempts to learn the probability distribution function from the given data, without specifically knowing if a token will be misrecognized or not, not to mention which class the token may be incorrectly assigned to. Therefore, the conventional distribution estimation approach will find it impossible to incorporate a non-uniform class-dependent error cost into the design of a pattern recognition system. The MCE method nonetheless offers a possible solution to the problem of pattern recognizer design with non-uniform cost as discussed below.

##### 4.4.1 MCE with Non-Uniform Error Cost

The incorporation of a class-dependent non-uniform error cost function incurs two factors that require careful consideration. First, as discussed in Chapter 4.3, the system needs to implement the MR rule defined in (27). Following the development of a smoothed error function in MCE, we need to embed this *decision rule* (or decision operation) in a functional form so that optimization can be performed to obtain the values of the system parameter set. Second, as the overall system performance is defined over a non-uniform, weighted error cost, the particular decision for each of the training tokens becomes an integral part of the performance measure and

has to be included in the objective function for optimization. The second factor is unique because once a decision is rendered by the recognizer, what matters is not only whether the decision is right or wrong but how much error cost the decision actually incurs. We shall see how these factors are taken into account in the proposed schemes for non-uniform error cost minimization.

#### 4.4.1.1 Construction of Design Objectives with Non-Uniform Error Cost

We now develop and extend the method of minimum classification error to satisfy the requirement of non-uniform error cost. The error cost function associated with a recognition task is defined as  $\epsilon_{ij}, i, j \in I_M = \{i, j = 1, 2, \dots, M\}$ , signifying the cost in identifying a pattern from the  $j^{\text{th}}$  class as one of the  $i^{\text{th}}$  class. It is customarily assumed that  $\epsilon_{ii} = 0, \forall i \in I_M$ , meaning no cost is incurred when the decision is the same as the true class identity. Following the development of Bayes' optimal decision, the expected system loss of (2) is minimized when the conditional loss of (1) is minimized due to the fact that  $p(X) \geq 0$ . Given a general set of cost assignments  $\epsilon_{ij}$ , we can explicitly write the optimal decision rule as one that minimizes the conditional cost of (1).

$$C(X) = \arg \min_i R(C_i|X) = \arg \min_i \sum_{j=1}^M \epsilon_{ij} P(C_j|X) \quad (29)$$

and

$$\min \mathcal{L} = \min E\{R(C(X)|X)\} = \int \min \sum_{j=1}^M \epsilon_{ij} P(C_j|X) p(X) dX \quad (30)$$

Execution of (29) obviously requires knowledge of the *a posteriori* probability  $P(C_j|X), \forall j \in I_M$ . As stated in [36], however, the true *a posteriori* probability is rarely available for a number of reasons (e.g., lack of knowledge of the distribution forms or sufficient labeled data for accurate estimation of the distribution parameters). Any decision rule such as the MAP policy that requires precise knowledge of the *a posteriori* probability cannot be accomplished in practice. The decision rule of (29), which involves a weighted combination of the *a posteriori* probabilities for all the

classes, may demand even more crucially the availability of the *a posteriori* probability than the MAP policy (which in effect only requires that the rank order of the *a posteriori* probabilities is accurately preserved in the evaluation). Furthermore, the complexity of the conditional cost of (29) may impose additional difficulties for the system designer in order to associate appropriate training models with the given data in the optimization process. The theoretical minimum of (30), which we continue to name the Bayes risk, is indeed achieved when the *a posteriori* distribution is truly available.

We approach the problem of constructing a system design objective with non-uniform cost in two steps, embedding of decision cost and prescription of the operating (scoring or discriminant) functions.

#### 4.4.1.2 *Embedding of Decision Cost*

To overcome these difficulties in system training, the expected system loss of (2) needs to be expressed in terms of the empirical loss (yet to be defined) with the decision rule embedded in it. For clarity, let  $i_X = C(X)$  be the identity index as decided by the recognizer and  $j_X$  be the true identity index of  $X$ . Also,  $\Omega = \{X^{(n)}\}_{n=1}^N$  is the set of training tokens. A *single token* realized cost is defined as

$$l_{i_X}(X; \Lambda) = \epsilon_{i_X j_X} \quad (31)$$

Therefore if the empirical system loss is defined over the realized token-based costs (rather than the expected cost of realized tokens), an alternative non-uniform cost will result:

$$L = \frac{1}{N} \sum_{X \in \Omega} \epsilon_{i_X j_X} \rightarrow \int \epsilon_{i_X j_X} p(X) dX \quad (32)$$

Suppose each class is prescribed a discriminant function  $g_j(X; \Lambda), \forall j$ . Define the recognizer function as

$$C(X) = i_X = \arg \max_i g_i(X; \Lambda) \quad (33)$$

The empirical system loss of (32) based on  $\Omega$  is then

$$L = \frac{1}{N} \sum_{X \in \Omega} \sum_{i \in I_M} \sum_{j \in I_M} \epsilon_{ij} \mathbf{1}[j_X = j] \mathbf{1}\{i = \arg \max_k g_k(X; \Lambda)\} \quad (34)$$

Note that in the above equation the indicator function  $\mathbf{1}[j_X = j] = \mathbf{1}[X \in C_j]$ .

#### 4.4.1.3 Prescription of Operating Function

What is the proper choice of operating the discriminant function for each class? Obviously, if the true *a posteriori* probability is available, a monotonically decreasing function of the conditional risk of (1) (to switch min into max operation) would be appropriate. For example,

$$g_i(X; \Lambda) = \exp\{-R(C_i|X)\} = \exp\left\{-\sum_{j \in I_M} \epsilon_{ij} P(C_j|X)\right\} \quad (35)$$

In general, since the  $P(C_j|X)$  are not available, one would use the alternative equation  $\tilde{P}(C_j|X) = \tilde{P}(X|C_j)\tilde{P}(C_j)/\tilde{P}(X)$  based on approximated and parameterized models in lieu of the true *a posteriori* distribution. This form of the discriminant function is appropriate in many simple pattern recognition tasks where good approximations to the conditional probabilities and the prior as argued in (28) may be relatively easy to obtain. (That can be accomplished by distribution fitting to class-labeled data.) If one is confident about the closeness of the approximation, the discriminant function of (35) would have the advantage of being intimately (but inversely) related to the conditional risk.

When the above approximation of the *a posteriori* distribution cannot be ensured, it is not particularly advisable to insist on using the weighted summation form of (35) as the discriminant function. For other applications where the form of the *a posteriori* distribution (as a function of  $X$ ) may be complex or hard to ascertain (such as those of speech signals), one may opt for other choices of the discriminant function based on some reasonable convention. One example is the use of hidden Markov models (HMM) as the discriminant functions.

#### 4.4.1.4 Smoothing of the Empirical Cost for Parameter Optimization

The remaining challenge in designing a discriminative training algorithm with non-uniform error cost is to turn the objective function,  $L$  in (34), into an appropriate smooth function of the parameter so as to allow for numeric optimization. Consider

$$L = \sum_{j \in I_M} L_j \quad (36)$$

and

$$L_j = \frac{1}{N} \sum_{X \in \Omega} \left( \sum_{i \in I_M} \epsilon_{ij} \mathbf{1} \left\{ i = \arg \max_k g_k(X; \Lambda) \right\} \right) \mathbf{1}[X \in C_j] \quad (37)$$

That is,  $L_j$  is the empirical error cost collected over all training tokens in  $\Omega$  with  $j_X = j$ . The approximation then needs to be made to the summands. This can be accomplished by

$$\sum_{i \in I_M} \epsilon_{ij} \mathbf{1} \left\{ i = \arg \max_k g_k(X; \Lambda) \right\} \approx \sum_{i \in I_M} \epsilon_{ij} \frac{g_i(X; \Lambda)}{G(X; \Lambda)} \quad (38)$$

where

$$G(X; \Lambda) = \left[ \sum_{i \in I_M} g_i^\eta(X; \Lambda) \right]^{1/\eta} \quad (39)$$

Note that as  $\eta \rightarrow \infty$ ,

$$\frac{g_i(X; \Lambda)}{G(X; \Lambda)} \approx \begin{cases} 1, & G(X; \Lambda) = \max_k g_k(X; \Lambda) \\ 0, & \text{otherwise} \end{cases} \quad (40)$$

Finally, the smoothed empirical system cost is

$$L \approx \frac{1}{N} \sum_{X \in \Omega} \sum_{j \in I_M} \left( \sum_{i \in I_M} \epsilon_{ij} \frac{g_i(X; \Lambda)}{G(X; \Lambda)} \right) \mathbf{1}[X \in C_j] \quad (41)$$

which is a continuous function of the parameter set  $\Lambda$ . Similarly, the parameter  $\eta$  can be chosen as a tradeoff between approximation accuracy and smoothness. We name this method as the *weighted MCE* method.

#### 4.4.1.5 Empirical Bayes Risk

Given a labeled training set  $\Omega = \{X^{(n)}\}_{n=1}^N$ , it is possible to compute the *empirical Bayes risk*, defined as

$$L_B = \frac{1}{N} \sum_{X \in \Omega} R(C_{j_X}|X) = \frac{1}{N} \sum_{X \in \Omega} \left( \sum_{j \in I_M} \epsilon_{j_X j} P(C_j|X) \right) \quad (42)$$

which is obtained when the correct labels are used to compute the conditional risk. The empirical Bayes risk is expected to be identical to the Bayes risk when  $N$ , the size of the training set, is infinite.

#### 4.4.2 Optimization Method for the System Cost Function

In this section, following the GPD algorithm [39][36][37], we present an optimization algorithm for the MCE method with non-uniform error cost. Recall the purpose of the training process here is to find a parameter set to minimize the expected loss of  $\mathcal{L}$  in (30), which can be re-written as

$$\mathcal{L} = \int \sum_{j=1}^M \epsilon_{i_X j} P(C_j|X) p(X) dX = E_X \{l_{i_X}(X; \Lambda)\} \quad (43)$$

where  $l_{i_X}(X; \Lambda) = \sum_{j=1}^M \epsilon_{i_X j} P(X|C_j) P(C_j) / P(X)$ . As we have fixed the error cost  $e_{ij}$  for  $i, j = 1, 2, \dots, M$ , the GPD algorithm can be employed for the minimization of (43). The target function is iteratively optimized in the GPD-based minimization. The convergence property of the GPD method can be established from the following theorem [36][8]:

*Theorem 1:* Suppose the following conditions are satisfied:

$$C1: \sum_{t=1}^{\infty} e_t = \infty, \quad \sum_{t=1}^{\infty} e_t^2 < \infty, \quad e_t \geq 0;$$

$C2: \exists 0 < V < \infty$ , such that for all  $t$ ,

$$R_t(e_t, \theta_t) = \langle \nabla l(X; \Lambda_n), H(X; \Lambda_n) + e_n \theta_n \nabla l(X; \Lambda_n) \nabla l(X; \Lambda_n) \rangle \leq V, \text{ where } H$$

is the Hessian matrix of second-order partial derivatives;

$C3$  :  $\Lambda^* = \arg \min_{\Lambda} E_X l(X; \Lambda)$  is the unique  $\Lambda$  such that

$$\nabla L(\Lambda)|_{\Lambda=\Lambda^*} = \nabla E_X l(X; \Lambda)|_{\Lambda=\Lambda^*} = 0$$

Then,  $\Lambda_t$  given by

$$\Lambda_{t+1} = \Lambda_t - e_t \nabla l(X; \Lambda)|_{\Lambda=\Lambda_t} \quad (44)$$

will converge to  $\Lambda^*$  almost surely (i.e., with probability one).

Condition  $C3$  can be considerably weakened. Even without condition  $C3$  the following is still true

$$E_X \nabla l(X; \Lambda_{t_k}) \rightarrow 0 \quad (45)$$

where  $\Lambda_{t_k}$  is a subsequence of  $\Lambda_t$ . In this case,  $\Lambda_{t_k}$  will converge to a local minimum point  $\Lambda_*$  where  $\nabla L(\Lambda)|_{\Lambda=\Lambda^*} = 0$ . Eq.(44) can also be generalized:

$$\Lambda_{t+1} = \Lambda_t - e_t U_t \nabla l(X_t; \Lambda)|_{\Lambda=\Lambda_t} \quad (46)$$

where  $U_t$  is a positive definite matrix [8]. Other theoretical properties of the GPD algorithm under the name of stochastic approximation can be found in [5][70][15]. However, in order to apply this algorithm to real applications, the GPD algorithm must obey the physical constraints of the application and the constraints imposed on the underlying statistical models. In other words, the GPD algorithm is an unconstrained minimization scheme that needs modification for solving minimization problems with constraints. It should be noted that the underlying probability distributions involved in minimizing (43) are often unknown to the designer. One of the advantages of a GPD-based minimization algorithm is that it does not make any explicit assumption about these unknown probabilities. This feature is important for recognition and adaptive learning problems.

In practice, for a given training data set consisting of  $N$  samples  $\Omega = \{X_1, \dots, X_N\}$ , the expected loss of (43) needs to be approximated by the empirical loss of (32). The empirical probability measure  $P_N$  defined on the training data set is a discrete probability measure that assigns equal mass at each sample [36]. Therefore, the empirical

loss of (32) can be re-written as

$$L = \frac{1}{N} \sum_{X \in \Omega} \sum_{j \in I_M} \left( \sum_{i \in I_M} \epsilon_{ij} \mathbf{1} \left\{ i = \arg \max_k g_k(X; \Lambda) \right\} \right) \mathbf{1}[X \in C_j] = \int \epsilon_{ij} dP_N \quad (47)$$

If we assume that the training data are independently sampled from a space with a fixed probability distribution  $P$ , the empirical probability distribution  $P_N$  will converge to  $P$  in distribution as  $N \rightarrow \infty$ . In other words, for any measurable function  $f$ ,

$$\int f dP_N \rightarrow \int f dP \quad (48)$$

Therefore, the empirical loss defined on the  $N$  independent training samples as of (47) will converge to the expected loss (43) as the sample size  $N$  increases. With sufficient training samples, the empirical loss is an estimate of the expected loss. The goodness of this estimate is determined by the training sample size  $N$  and the convergence rate of the empirical probability measure  $P_N$  to distribution  $P$ . Various upper bounds on the convergence rate of the empirical probability measure can be found in [63].

#### 4.5 Gaussian Mixture Classifiers – An Example

The implementation of non-uniform error training criteria requires reasonable modeling of the discriminant/scoring function  $g(X; \Lambda)$  in (33). The Gaussian mixture model (GMM) [17] has found extensive use as a statistical model for many pattern recognition problems due to its capability to approximate complex distributions. Assume  $X = (x_1, \dots, x_l, \dots, x_D)$ ,  $X \in \Omega$  is an observation vector in the training set  $\Omega$  with  $D$  dimensions. The GMM model for the  $i^{\text{th}}$  class with  $K$  mixture components can be written as

$$b^{(i)}(X) = \sum_{k=1}^K c_k^{(i)} \mathcal{N}[X | U_k^{(i)}, R_k^{(i)}] \quad (49)$$

$$= \sum_{k=1}^K c_k^{(i)} \frac{1}{(2\pi)^{D/2} |R_k^{(i)}|^{1/2}} \exp \left\{ -\frac{1}{2} \sum_{l=1}^D \frac{(x_l - \mu_{kl}^{(i)})^2}{(\sigma_{kl}^{(i)})^2} \right\} \quad (50)$$

$$= \sum_{k=1}^K c_k^{(i)} \frac{1}{(2\pi)^{D/2} (\prod_{l=1}^D \sigma_{kl}^{(i)})} \exp \left\{ -\frac{1}{2} \sum_{l=1}^D \frac{(x_l - \mu_{kl}^{(i)})^2}{(\sigma_{kl}^{(i)})^2} \right\} \quad (51)$$

where  $\mathcal{N}[\cdot]$  denotes a normal distribution,  $c_k^{(i)}$  is the weight for the  $k^{\text{th}}$  mixture component of the  $i^{\text{th}}$  model,  $U_k^{(i)} = [\mu_{kl}^{(i)}]_{l=1}^D$  is the mean vector and  $R_k^{(i)}$  is the corresponding covariance matrix which, for simplicity, is assumed to be diagonal, i.e.,  $R_k^{(i)} = [(\sigma_{kl}^{(i)})^2]_{l=1}^D$ .

Based on the MR decision rule of (28), the ideal discriminant scoring function  $g_i(X; \Lambda)$  is the non-uniform error cost  $\sum_{i \in I_M} \epsilon_{ij} P(C_j|X)$ , as long as we know the true *a posteriori* distribution  $P(C_i|X)$  exactly. This is, of course, rarely possible in any real pattern recognition applications. If we can parameterize and approximate the *a posteriori* distribution accurately, say,  $P(C_j|X) \approx \tilde{P}(X; \Lambda_{X|C}|C_j)\tilde{P}(C_j)/\tilde{P}(X)$ , we can model  $g_i(X; \Lambda)$  according to the original definition of the MR decision rule. To transform the objective of minimizing the error cost to maximizing the scoring function  $g$ , we assume  $g_i(X; \Lambda) = h\left(\sum_{i \in I_M} \epsilon_{ij} \tilde{P}(X; \Lambda_{X|C}|C_j)\tilde{P}(C_j)/\tilde{P}(X)\right)$ , in which  $\tilde{P}(X; \Lambda_{X|C}|C_j) = b^{(j)}(X; \Lambda)$  and  $h(\cdot)$  is a non-negative monotonically non-increasing function (e.g.,  $h(x) = e^{-x}$ ). The other straightforward and simple way of modeling the scoring function is to assume  $g_i(X; \Lambda) = \sum_{i \in I_M} \epsilon_{ij} P(C_j|X) \approx b^{(i)}(X; \Lambda)$  directly. Note that to use this modeling strategy, we need to normalize the error cost  $\epsilon_{ij}$  to assure  $0 < \sum_{i \in I_M} \epsilon_{ij} P(C_j|X) < 1$ . There are many possibilities between these two extreme cases, and the system designer can choose the most suitable one according to the application specifications. In this section, we discuss these two extreme representative modeling techniques for  $g(X; \Lambda)$  under non-uniform error cost criteria. In each case, we provide extensive derivations for parameter optimization equations.

#### 4.5.1 Modeling $g(X; \Lambda)$ as a function of the Non-Uniform Error Cost and $P(X|C_i)$ as GMM

If we use GMM of (51) in lieu of the conditional probability  $P(X|C_i)$  and assume the discriminant scoring function to be

$$g_i(X; \Lambda) = h\left(\sum_{j \in I_M} \epsilon_{ij} P(C_j|X)\right) = \exp\left\{-\sum_{j \in I_M} \epsilon_{ij} b^{(j)}(X; \Lambda) P(C_j)/P(X)\right\} \quad (52)$$

Eq.(36) can be re-written as

$$L = \frac{1}{N} \sum_{n=1}^N \sum_{j \in I_M} l_{X^{(n)}}^{(j)}(X^{(n)}; \Lambda) \mathbf{1}[X \in C_j] \quad (53)$$

$$\approx \frac{1}{N} \sum_{X \in \Omega} \sum_{j \in I_M} \sum_{i \in I_M} \epsilon_{ij} \frac{g_i(X; \Lambda)}{G(X; \Lambda)} \mathbf{1}[X \in C_j] \quad (54)$$

$$= \frac{1}{N} \sum_{X \in \Omega} \sum_{j \in I_M} \sum_{i \in I_M} \epsilon_{ij} \frac{g_i(X; \Lambda)}{\{\sum_{r \in I_M} [g_r(X; \Lambda)]^\eta\}^{1/\eta}} \mathbf{1}[X \in C_j] \quad (55)$$

Optimization of the parameter set  $\Lambda$  follows the algorithm of (46).

In this case, we need to set up some constraints before optimization: 1) the GMM function needs to be non-negative; 2)  $\epsilon_{ij} \geq 0$ ; 3)  $\sum_k c_k = 1$ ; and 4)  $\sigma_{kl} > 0$ . The following parameter transformations allow us to maintain these constraints during parameter adaptation [36]:

$$c_k \rightarrow \tilde{c}_k, \text{ where } c_k = \frac{e^{\tilde{c}_k}}{\sum_k e^{\tilde{c}_k}} \quad (56)$$

$$\mu_{kl} \rightarrow \tilde{\mu}_{kl} = \frac{\mu_{kl}}{\sigma_{kl}} \quad (57)$$

$$\sigma_{kl} \rightarrow \tilde{\sigma}_{kl} = \ln \sigma_{kl} \quad (58)$$

The derivative of  $l_{X^{(n)}}^{(j)}(X; \Lambda)$  with respect to  $\tilde{\Lambda}_m = \{\tilde{c}_k^{(m)}, \tilde{\mu}_{kl}^{(m)}, \tilde{\sigma}_{kl}^{(m)}\}$  can be written as:

$$\frac{\partial l_{X^{(n)}}^{(j)}(X; \Lambda)}{\partial \tilde{\Lambda}_m} = \frac{\partial l_{X^{(n)}}^{(j)}(X; \Lambda)}{\partial g_i(X; \Lambda)} \frac{\partial g_i(X; \Lambda)}{\partial b^{(m)}(X; \Lambda)} \frac{\partial b^{(m)}(X; \Lambda)}{\partial \tilde{\Lambda}_m} = \frac{\partial l_{X^{(n)}}^{(j)}(X; \Lambda)}{\partial b^{(m)}(X; \Lambda)} \frac{\partial b^{(m)}(X; \Lambda)}{\partial \tilde{\Lambda}_m} \quad (59)$$

where

$$\frac{\partial l_{X^{(n)}}^{(j)}(X; \Lambda)}{\partial b^{(m)}(X; \Lambda)} = l_{X^{(n)}}^{(j)}(X; \Lambda) \frac{\sum_{i \in I_M} \epsilon_{im} g_i^\eta(X; \Lambda)}{\sum_{i \in I_M} g_i^\eta(X; \Lambda)} P(C_m) - \frac{\sum_{i \in I_M} \epsilon_{ij} \epsilon_{im} g_i(X; \Lambda)}{G(X; \Lambda)} P(C_m) \quad (60)$$

$G(X; \Lambda)$  is defined as

$$G(X; \Lambda) = \left[ \sum_{i \in I_M} g_i^\eta(X; \Lambda) \right]^{1/\eta} = \left[ \sum_{i \in I_M} \exp \left\{ -\eta \sum_{m \in I_M} \epsilon_{im} b^{(m)}(X; \Lambda) P(C_m) / P(X) \right\} \right]^{1/\eta} \quad (61)$$

For  $\frac{\partial b^{(m)}(X; \Lambda)}{\partial \Lambda_m}$ , we use the same equations derived in [36]. Starting with the updating of the mixture weight  $c_k$ , we obtain the updating equation for  $c_k^{(m)}$  based on the error cost for the  $m^{\text{th}}$  class:

$$\tilde{c}_k^{(m)}(n+1) = \tilde{c}_k^{(m)}(n) - e \frac{\partial l_{X^{(n)}}^{(j)}(X; \Lambda)}{\partial b^{(m)}(X)} \frac{\partial b^{(m)}(X)}{\partial \tilde{c}_k^{(m)}} \quad m, j = 1, 2, \dots, M \quad (62)$$

and

$$\frac{\partial b^{(m)}(X)}{\partial \tilde{c}_k^{(m)}} = c_k^{(m)}(1 - c_k^{(m)}) \frac{1}{(2\pi)^{D/2} (\prod_{l=1}^D \sigma_{kl}^{(m)})} \exp \left\{ -\frac{1}{2} \sum_{l=1}^D \left( \frac{x_l - \mu_{kl}^{(m)}}{\sigma_{kl}^{(m)}} \right)^2 \right\} \quad (63)$$

The last step is to convert  $\tilde{c}_k^{(m)}$  back according to (56),

$$c_k^{(m)} = \frac{e^{\tilde{c}_k^{(m)}}}{\sum_k e^{\tilde{c}_k^{(m)}}} \quad (64)$$

Similarly, we have the updating equations for the mean vector  $U_k^{(m)} = (\mu_{k1}^{(m)}, \dots, \mu_{kD}^{(m)})$ :

$$\tilde{\mu}_{kl}^{(m)}(n+1) = \tilde{\mu}_{kl}^{(m)}(n) - e \frac{\partial l_{X^{(n)}}^{(j)}(X; \Lambda)}{\partial b^{(m)}(X)} \frac{\partial b^{(m)}(X)}{\partial \tilde{\mu}_{kl}^{(m)}} \quad (65)$$

where

$$\frac{\partial b^{(m)}(X)}{\partial \tilde{\mu}_{kl}^{(m)}} = c_k^{(m)} \frac{1}{(2\pi)^{D/2} (\prod_{l=1}^D \sigma_{kl}^{(m)})} \left( \frac{x_l - \mu_{kl}^{(m)}}{\sigma_{kl}^{(m)}} \right) \exp \left\{ -\frac{1}{2} \sum_{l=1}^D \left( \frac{x_l - \mu_{kl}^{(m)}}{\sigma_{kl}^{(m)}} \right)^2 \right\} \quad (66)$$

Finally,

$$\mu_{kl}^{(m)}(n+1) = \sigma_{kl}^{(m)}(n) \tilde{\mu}_{kl}^{(m)}(n) \quad (67)$$

For the variance vector  $\Sigma_k^{(m)} = (\sigma_{k1}^{(m)}, \dots, \sigma_{kD}^{(m)})$ , we have

$$\tilde{\sigma}_{kl}^{(m)}(n+1) = \tilde{\sigma}_{kl}^{(m)}(n) - e \frac{\partial l_{X^{(n)}}^{(j)}(X; \Lambda)}{\partial b^{(m)}(X)} \frac{\partial b^{(m)}(X)}{\partial \tilde{\sigma}_{kl}^{(m)}} \quad (68)$$

where

$$\frac{\partial b^{(m)}(X)}{\partial \tilde{\sigma}_{kl}^{(m)}} = c_k^{(m)} \frac{1}{(2\pi)^{D/2} (\prod_{l=1}^D \sigma_{kl}^{(m)})} \left[ \left( \frac{x_l - \mu_{kl}^{(m)}}{\sigma_{kl}^{(m)}} \right)^2 - 1 \right] \exp \left\{ -\frac{1}{2} \sum_{l=1}^D \left( \frac{x_l - \mu_{kl}^{(m)}}{\sigma_{kl}^{(m)}} \right)^2 \right\} \quad (69)$$

Finally

$$\sigma_{kl}^{(m)}(n+1) = \exp\{\tilde{\sigma}_{kl}^{(m)}(n)\} \quad (70)$$

The optimization algorithm we have discussed before is operated on a per token basis, i.e., it is a method that updates classifier models for each training token. The optimization routine can be also operated in a “batch” fashion, in which the parameters are updated on a per data block basis using the statistics accumulated on the tokens in the block.

#### 4.5.2 Modeling $g(X; \Lambda)$ as GMM

If knowledge of the *a posteriori* distribution is lacking, one straightforward modeling method is to let

$$g_i(X; \Lambda) = b^{(i)}(X; \Lambda) \quad (71)$$

In other words, we use the GMM model to approximate the non-uniform error cost  $\sum_{j \in I_M} \epsilon'_{ij} P(C_i|X)$  directly. Note that in this case,  $\epsilon'_{ij}$  should be normalized to assure  $0 < b^{(i)}(X; \Lambda) = \sum_{j \in I_M} \epsilon'_{ij} P(C_i|X) < 1$ .

In this case, we can formulate the objective function as

$$L' = \frac{1}{N} \sum_{n=1}^N \sum_{j \in I_M} l_{X^{(n)}}^{(j)}(X^{(n)}; \Lambda) \mathbf{1}[X \in C_j] \quad (72)$$

$$= \frac{1}{N} \sum_{n=1}^N \sum_{j \in I_M} \sum_{i \in I_M} \epsilon_{ij} \frac{g_i(X; \Lambda)}{\{\sum_{r \in I_M} [g_r(X; \Lambda)]^\eta\}^{1/\eta}} \mathbf{1}[X \in C_j] \quad (73)$$

$$= \frac{1}{N} \sum_{n=1}^N \sum_{j \in I_M} \sum_{i \in I_M} \epsilon_{ij} \frac{b^{(i)}(X; \Lambda)}{\{\sum_{r \in I_M} [b^{(r)}(X; \Lambda)]^\eta\}^{1/\eta}} \mathbf{1}[X \in C_j] \quad (74)$$

By imposing the same constraints on the parameters  $\Lambda_m = \{c_k^{(m)}, \mu_{kl}^{(m)}, \sigma_{kl}^{(m)}\}$ ,  $m = 1, 2, \dots, M$  as defined in (56)-(58), we can use the identical updating equations as (62), (65) and (68) to optimize  $\tilde{c}_k^{(m)}$ ,  $\tilde{\mu}_{kl}^{(m)}$  and  $\tilde{\sigma}_{kl}^{(m)}$ . Analogous to the derivation of the last section, the derivative of  $l_{X^{(n)}}^{(j)}(X; \Lambda)$  with respect to  $\tilde{\Lambda}_m$  can be written as

$$\frac{\partial l_{X^{(n)}}^{(j)}(X; \Lambda)}{\partial \tilde{\Lambda}_i} = \frac{\partial l_{X^{(n)}}^{(j)}(X; \Lambda)}{\partial b^{(i)}(X; \Lambda)} \frac{\partial b^{(i)}(X; \Lambda)}{\partial \tilde{\Lambda}_i} \quad (75)$$

where

$$\frac{\partial l_{X^{(n)}}^{(j)}(X; \Lambda)}{\partial b^{(i)}(X; \Lambda)} = \frac{\epsilon_{ij} \sum_{r \in I_M} [b^{(r)}(X; \Lambda)]^\eta - \sum_{r \in I_M} \epsilon_{rj} [b^{(r)}(X; \Lambda)] [b^{(i)}(X; \Lambda)]^{\eta-1}}{G \cdot \sum_{r \in I_M} [b^{(r)}(X; \Lambda)]^\eta} \quad (76)$$

and  $G = [\sum_{r \in I_M} g_r^\eta(X; \Lambda)]^{1/\eta} = [\sum_{r \in I_M} [b^{(r)}(X; \Lambda)]^\eta]^{1/\eta}$ .

The derivatives of the  $b^{(i)}(X; \Lambda)$  with respect to  $\tilde{\Lambda}_i = \{\tilde{c}_k^{(i)}, \tilde{\mu}_{kl}^{(i)}, \tilde{\sigma}_{kl}^{(i)}\}$  follow the equations in the last section. In particular,  $\frac{\partial b^{(i)}(X; \Lambda)}{\partial \tilde{c}_k^{(i)}}$ ,  $\frac{\partial b^{(i)}(X; \Lambda)}{\partial \tilde{\mu}_{kl}^{(i)}}$  and  $\frac{\partial b^{(i)}(X; \Lambda)}{\partial \tilde{\sigma}_{kl}^{(i)}}$  are identical to (63), (66) and (69), respectively. Finally, we can obtain the parameter set  $\Lambda_i = \{c_k^{(i)}, \mu_{kl}^{(i)}, \sigma_{kl}^{(i)}\}$  from  $\tilde{\Lambda}_i = \{\tilde{c}_k^{(i)}, \tilde{\mu}_{kl}^{(i)}, \tilde{\sigma}_{kl}^{(i)}\}$  following (64), (67) and (70).

## 4.6 Experimental Results for General Pattern Recognition

In this section, we present demonstrative experiments to show the effectiveness of the algorithm for recognizer design using non-uniform error cost criteria. As the Gaussian mixture model (GMM) is well-known for its capability to approximate arbitrary probability distributions, our experiments are carried out using data samples generated by GMMs with different numbers of mixtures. In detail, there are three classes in our experiments. The data for each class is two-dimensional and generated by a GMM with four mixture components. We split the entire data set into the training and test set with a ratio of 80/20. The class prior probabilities are assumed to be 0.2, 0.3, and 0.5 respectively. The baseline models are initialized based on maximum likelihood distribution estimation via the EM algorithm [17].

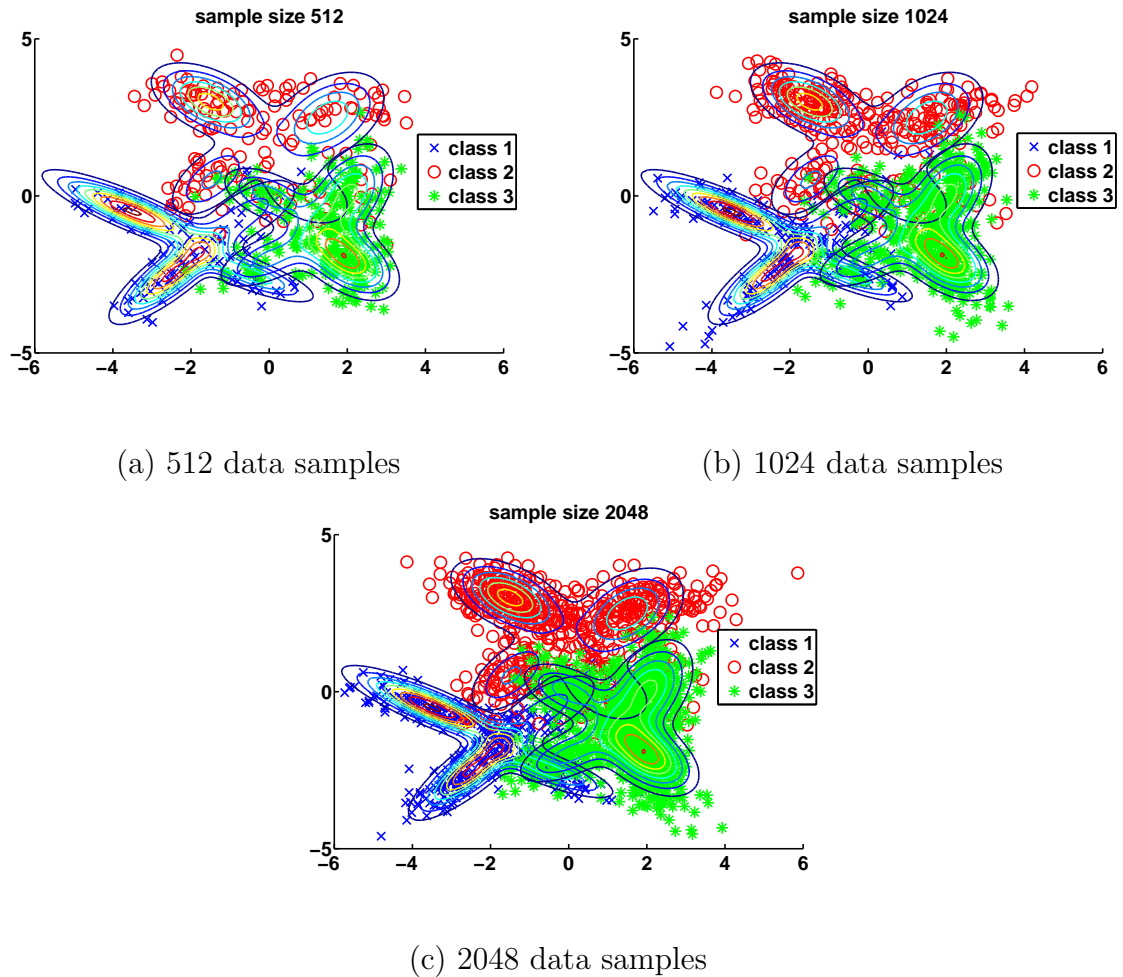
The study of non-uniform error cost criteria consists of two parts. First, in an ideal case, we assume that the form of the data distribution is known and both implementation strategies described in the previous section are examined. We investigate the recognition performance with different error cost matrices and as a function of the training data size. Second, as in most real applications, we investigate the performance of non-uniform cost training criteria in the situation that the classifier models do not match the true data distributions. For simplicity, we create the mismatch

by changing the number of mixtures in the GMM models. Thus, the issue of model mismatch may be rather mild.

#### 4.6.1 True Models and Simulation Settings

##### 4.6.1.1 Models and Data

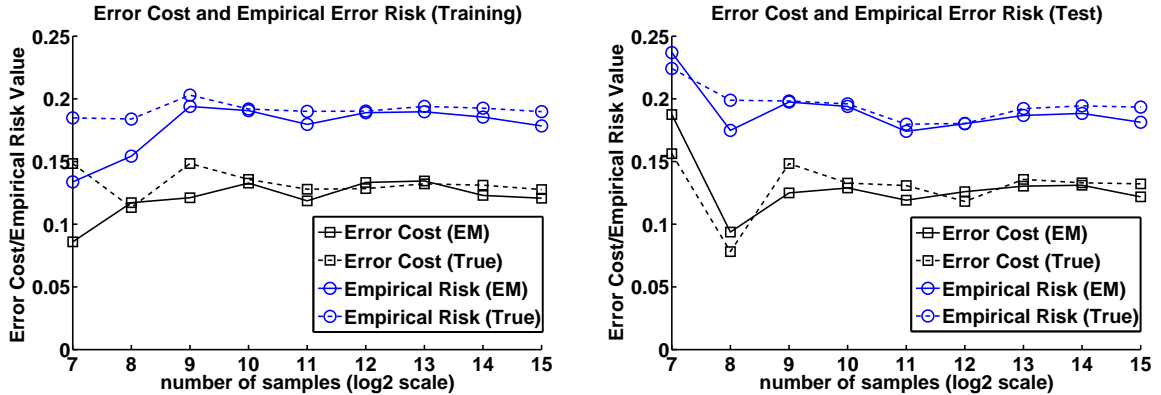
We generate nine data sets for each of the three classes using 4-mixture GMMs, which correspond to the data set size from 128 (which is  $2^7$ ) to 32768 (which is  $2^{15}$ ). Fig.2 shows the GMM model iso-probability contours and overlaid on three sets of data samples. In Fig.2, the subplot (a), (b), and (c) illustrate the 512, 1024, and 2048 data samples, respectively.



**Figure 2:** GMM models and data samples.

#### 4.6.1.2 Baseline Models

For each data set, we initialize our classifier using the EM algorithm as mentioned earlier. Assuming that the classifier models are also 4-mixture GMMs (i.e., in a “matched” condition), we can compare the performances of the baseline model with that of the true model. In Fig.3, we plot the error cost of Eq.(36) and the empirical Bayes risk of Eq.(42) versus the number of training data samples in a log 2 scale. Note that in Fig.3, the error cost function is uniform thus the error cost is equivalent to the recognition error rate. The dashed lines and the solid lines represent the evaluated error costs of the true models and the EM-trained models, respectively. We denote the empirical Bayes risk of Eq.(42) by the circle sign and the empirical error cost of Eq.(36) using the square sign. The left and right panel in Fig.3 are the performance metrics computed on the training and the test set, respectively. We can observe that for the simple distributions tested if the total number of training data samples is larger than  $2^{11} = 2048$ , the EM-trained models are very close to the true models in terms of these performance metrics in this matched model condition.



**Figure 3:** Error cost and empirical Bayes risk computed using the true models and the EM-trained models.

### 4.6.1.3 Objective Function and Performance Measure

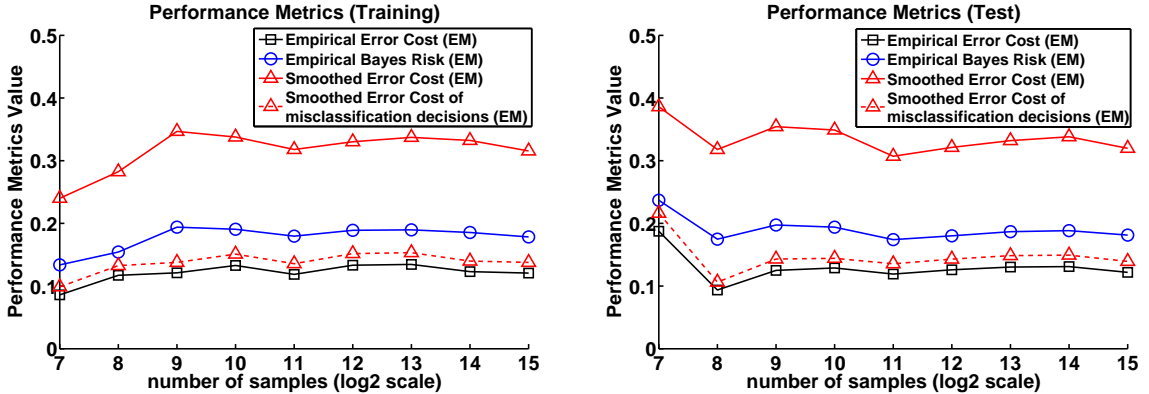
As a natural extension of the conventional pattern recognition metric (the recognition error rate), the empirical error cost (36) is a good choice of performance measure under the non-uniform error cost criterion. As described in Chapter 4.4.1.4, Eq.(36) needs to be smoothed in order to implement the gradient descent method in updating equations such as (46). Hence, the actual objective function for the non-uniform cost MCE training method becomes the smoothed empirical error cost of (41), which can be adjusted between the approximation accuracy and the smoothness according to system requirements. We can also use the empirical Bayes risk of Eq.(42) as a reference for the classifier performance measure.

We plot several performance measures of the baseline models in Fig.4 for the training (left panel) and the test set (right panel) as a function of the data size using a uniform error cost matrix. The solid lines with triangle, circle and square sign denote the smoothed error cost with  $\eta = 2$  using (41), the empirical Bayes risk using (42), and the empirical error cost of (36), respectively. Of course, there are many other possible measures. For example, another possible reference is the embedded error cost of misclassification decisions, which is computed by Eq.(77) below and denoted by the dashed line with triangle signs in Fig.4.

$$L \approx \frac{1}{N} \sum_{X \in \Omega} \sum_{j \in I_M} \left( \sum_{i \neq j, i \in I_M} \epsilon_{ij} \frac{g_i(X; \Lambda)}{G(X; \Lambda)} \right) \mathbf{1}[X \in C_j] \quad (77)$$

In Fig.4, we can see that though the value may be different, the trends of all lines are quite similar. If the number of data samples is large enough to emulate the continuous distribution of  $P(C_j|X)$  (note: no model mismatch here), the empirical error cost of (36) will eventually converge to the empirical Bayes risk of (42). This convergence is hard to observe because the data sample is still too sparse in terms of representing  $P(C_j|X)$  as a continuous function of  $X$ . In our experiments, we will use the empirical error cost as the performance measure with the smoothed empirical error cost as the

optimization objective.



**Figure 4:** Some performance metrics with uniform error cost matrix on the training and test set.

#### 4.6.2 Non-Uniform Error Cost Training Based on Correct Classifier Models

In this section, we present the performance of the recognizer design using non-uniform error cost criteria via two modeling methods of the discriminant function  $g(X; \Lambda)$ , based on the assumption that we have known the form of the true data distribution (i.e., a GMM with a 4-mixture distribution for each data class).

##### 4.6.2.1 Modeling $g(X; \Lambda)$ as a Function of the Non-Uniform Error Cost

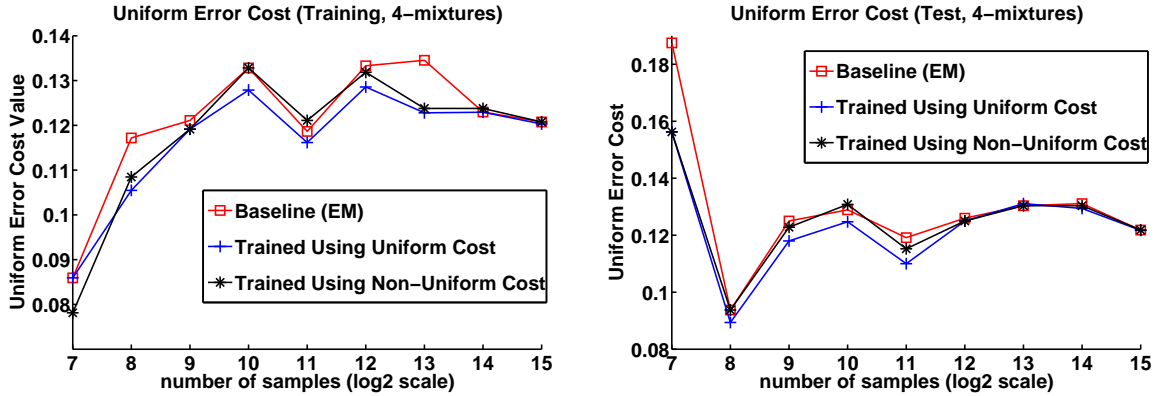
Here, we investigate an ideal case, in which we know the analytical form of the data distribution so that we can model the scoring function  $g(X; \Lambda)$  as a function of the non-uniform error cost as of (52) in which  $P(X; \Lambda|C_j)$  is modeled as a 4-mixture GMM.

Figure 5 compares the performance in terms of the empirical error cost of (36) computed using a uniform cost matrix (i.e., the error rate) between the baseline and the MCE-trained models. The performance comparison assessed on the training set is plotted in the left panel and that on the test set in the right panel. In each panel, the horizontal axis represents the number of training data samples. The square sign,

the “+” sign and the star sign denote the baseline, the model trained by the MCE method with a *uniform* cost matrix, and the model trained by the MCE method with a *non-uniform* cost matrix, respectively. For each situation of the MCE training, parameters are updated for 10 epochs and  $\eta = 5$  during the training process. The non-uniform error cost matrix is assumed to be

$$[\epsilon_{ij}] = \begin{bmatrix} 0 & 7 & 3 \\ 2 & 0 & 8 \\ 4 & 6 & 0 \end{bmatrix}$$

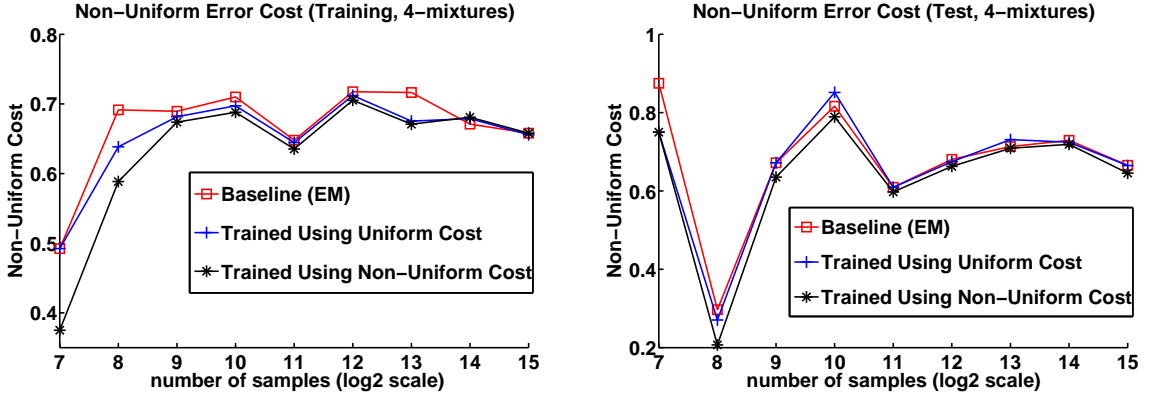
We can observe that since the performance measure is defined by the uniform error cost, the model trained by the MCE method with a uniform cost matrix (i.e., a matched-objective condition) shows the best performance at most operating points due to the consistency between the training objective and the performance measure. The other observation is that when the size of data increases, the baseline model becomes better trained, leading to a performance approaching that of the MCE models. This observation is consistent with the trend showing in Fig.3.



**Figure 5:** Uniform error cost for training and test set when modeling  $g(X; \Lambda)$  as a function of the non-uniform error cost

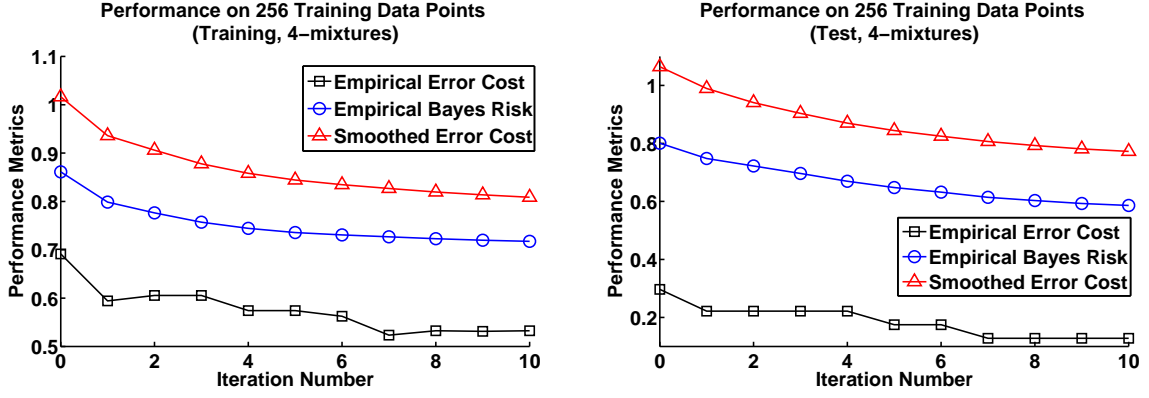
In a similar experiment, the non-uniform error cost is adopted in the training objective. Fig.(6) compares the performance in terms of the empirical error cost of (36) computed using the non-uniform cost matrix of (78) between the baseline and

the MCE-trained models. Since the performance is assessed on the non-uniform cost matrix, the model trained by the MCE method with the non-uniform cost matrix outperforms the baseline and the one using the uniform matrix.



**Figure 6:** Non-uniform error cost for training and test set when modeling  $g(X; \Lambda)$  as a function of the non-uniform error cost

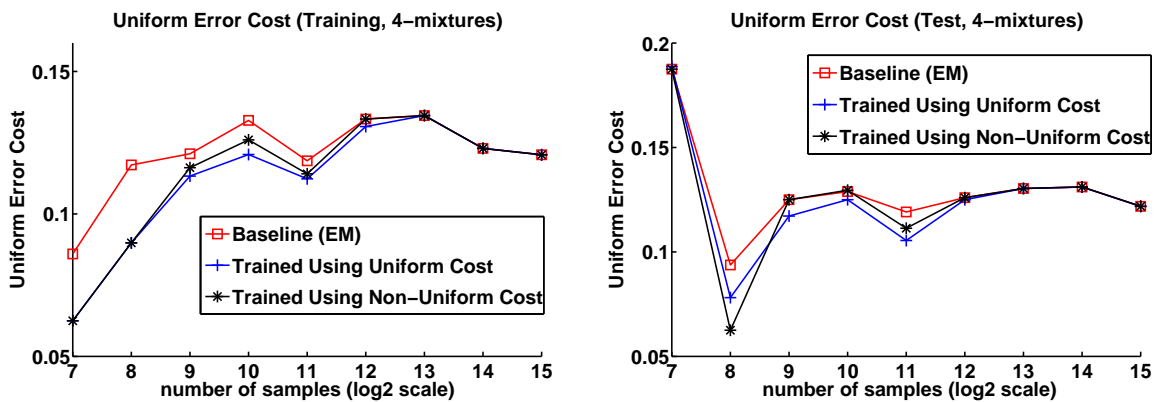
Figure 7 illustrates the classifier performance over 10 non-uniform MCE training iterations with the error cost matrix of (78) and a data set size of 256. One iteration means one cycle of updating the classifier parameters through all data in the training set. The lines with the square sign, the circle sign and the triangle sign represent the empirical error cost of (36), the empirical Bayes risk of (42), and the smoothed error cost of (41) with  $\eta = 5$ , respectively. All quantities are normalized by the number of data. The error cost matrix is defined in (78). The performance assessed on the training set is plotted in the left panel, and that on the test set in the right panel. We can see that all metrics enhance over the training iterations/epochs. The normalized error cost of the EM-trained baseline is 0.70 for the training set and 0.30 for the test set respectively, while the final MCE training achieves 0.53 and 0.13 normalized error cost after 10 iterations.



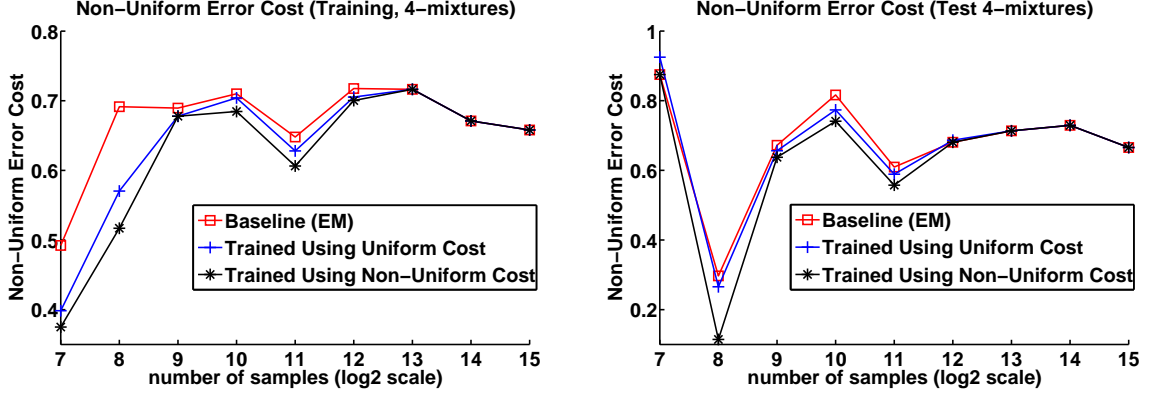
**Figure 7:** Performance over training iterations with 256 training data when modeling  $g(X; \Lambda)$  as a function of the non-uniform error cost.

#### 4.6.2.2 Modeling $g(X; \Lambda)$ as a GMM

Here a simplified modeling strategy is studied. Let the discriminant scoring function  $g(X; \Lambda)$  be defined as Eq.(71), and  $b(X; \Lambda)$  in (71) is assumed to be the 4-mixture GMM. Similar to Fig.5 and Fig.6, Fig.8 and Fig.9 display the performance comparison between the baseline and MCE-trained models. The observation is similar to that of the last section. When the performance measure is chosen to be the uniform error cost, the classifier trained using the uniform cost matrix outperforms the baseline and the model trained using the non-uniform cost matrix and vice versa.

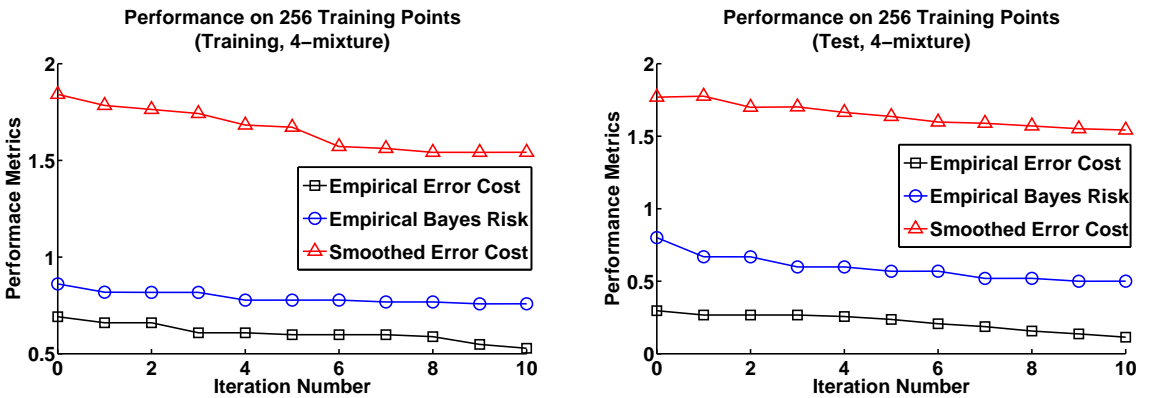


**Figure 8:** Uniform error cost for training and test set when modeling  $g(X; \Lambda)$  as a 4-mixture GMM



**Figure 9:** Non-uniform error cost for training and test set when modeling  $g(X; \Lambda)$  as a 4-mixture GMM

Figure 10 plots the concrete performance improvement over the MCE training iterations with 256 training data points. The left and right panels display the performance improvement over iterations on the training and test set, separately. The solid lines with square, circle and triangle signs denote the empirical error cost, the empirical Bayes risk, and the smoothed error cost with  $\eta = 2$ , respectively. The error cost matrix is defined in Eq.(78). All performance metrics improve along with the number of training iterations. The EM-trained baseline has 0.69 and 0.29 normalized error cost for the training and test set respectively, while the MCE training achieves 0.53 and 0.11 normalized error cost after 10 iterations correspondingly.



**Figure 10:** Performance over training iterations with 256 training data when modeling  $g(X; \Lambda)$  as a 4-mixture GMM.

### 4.6.3 Non-Uniform Error Cost Training for Mismatched Models

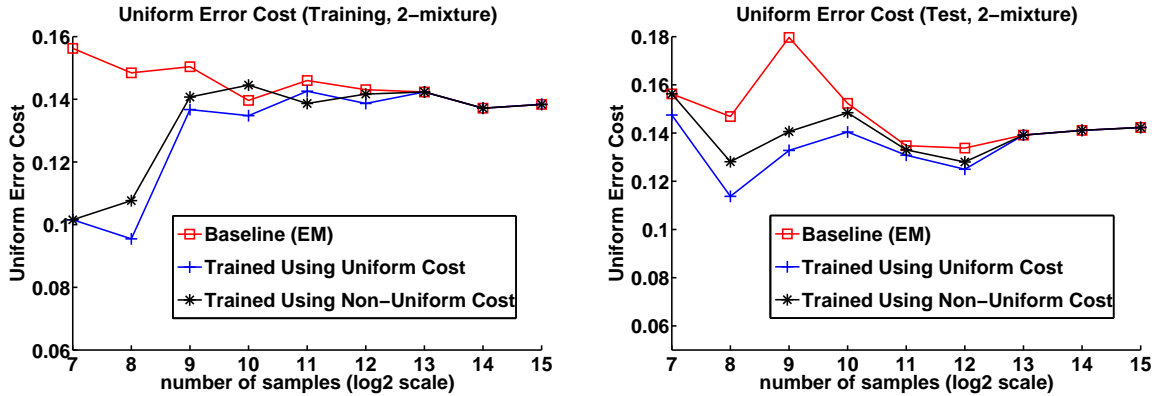
We have investigated the performance of non-uniform error criteria in the last section, under the assumption that the classifier models are formulated identical to the true data distribution (expect the values of the parameters which are then optimized in training). However, true data distribution in most pattern recognition problems is rarely known. In practice, the discriminant function  $g(X; \Lambda)$  of (52) implies a rather strong assumption in its decomposition of functional form. It may involve unnecessary complication if the true analytical form of the data distribution is unavailable. Without sufficient knowledge of the data distribution in this case, modeling the discriminant function  $g(X; \Lambda)$  as one single GMM as (71) may be a reasonable alternative. In fact, modeling of  $g(X; \Lambda)$  as Eq.(52) with an arbitrary form of  $P(C_i|X)$  is not particularly critical in terms of reducing error cost in our experiments.

In this section, we investigate the performance of non-uniform cost training under the scenario which the real data distribution is unknown. The true data is still generated by 4-mixture GMMs as presented before, but we assume the discriminant scoring function  $g(X; \Lambda)$  to be a GMM with a different number of mixtures. In other words, a single GMM model is employed as the scoring function, approximating the “distribution” of the error cost  $\sum \epsilon_{ij}P(C_j|X)$  rather than the posterior distribution  $P(C_j|X)$ . We study the effects of the non-uniform criteria training when the number of mixture components of the classifier models is assumed to be smaller than the correct number (e.g., two components) or larger than the correct number (e.g., six components).

#### 4.6.3.1 Modeling $g(X; \Lambda)$ as 2-mixture GMMs

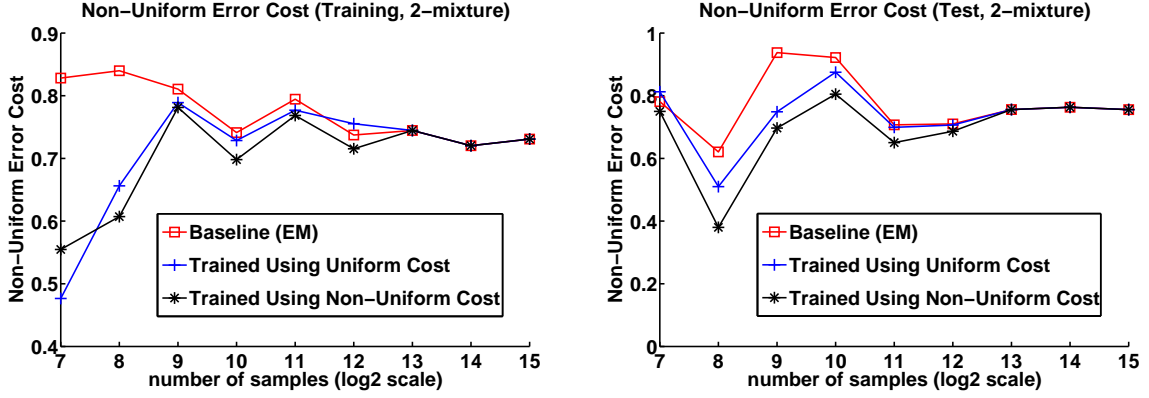
In this section, we investigate the non-uniform error cost training when we model the scoring function  $g(X; \Lambda)$  as 2-mixture GMMs. The baseline, which is a “distribution” based model, is still trained using the EM algorithm.

Figure 11 and 12 compare the performance in terms of the empirical error cost of (36), computed using the uniform cost matrix and the non-uniform cost matrix of (78), respectively. As described before, the performance assessed on the training set is plotted in the left panel and that on the test set in the right panel. In each panel, the horizontal axis represents the number of training data samples and the square sign, the plus sign and the star sign denote the baseline trained by the EM algorithm, the model trained by the MCE method with a *uniform* cost matrix, and the model trained by the MCE method with a *non-uniform* cost matrix, respectively. When the uniform error cost is used to assess the performance, the classifier trained using the uniform cost matrix outperforms the one trained using the non-uniform cost matrix and vice versa.



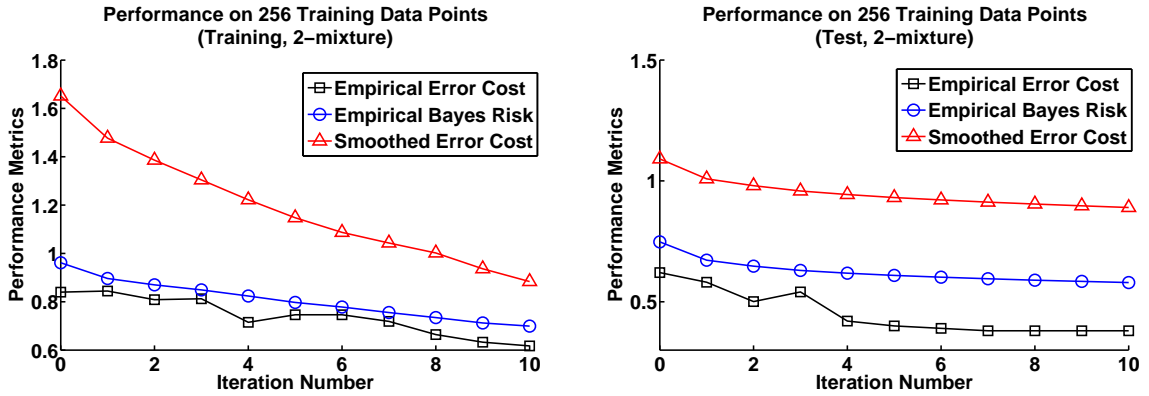
**Figure 11:** Uniform error cost for training and test set when modeling  $g(X; \Lambda)$  as a 2-mixture GMM

Figure 13 plots the non-uniform error cost improvement for 2-mixture GMM models as a function of the MCE training iteration using training set of 256 data points. The left and right panel display the performance improvement over iteration epochs on the training and test set, separately. The solid lines with square, circle and triangle signs denote the empirical error cost, the empirical Bayes risk, and the smoothed error cost with  $\eta = 2$ , respectively. The error cost matrix is defined in (78). All performance measures improve along with the number of training iterations. In particular,



**Figure 12:** Non-uniform error cost for training and test set when modeling  $g(X; \Lambda)$  as a 2-mixture GMM

the EM-trained baseline has 0.84 and 0.62 normalized error cost for the training and test set respectively, while the MCE training achieves 0.62 and 0.38 normalized error cost after 10 iterations correspondingly.

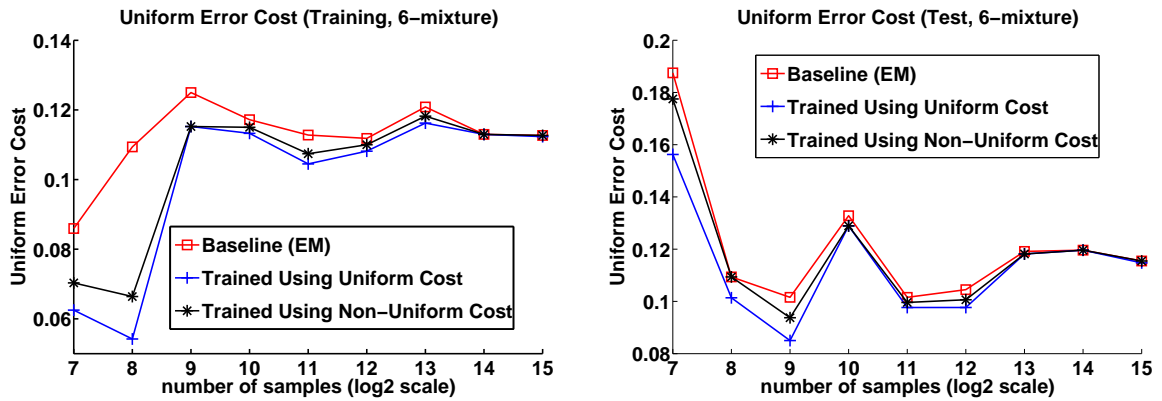


**Figure 13:** Performance over training iterations with 256 training data when modeling  $g(X; \Lambda)$  as a 2-mixture GMM.

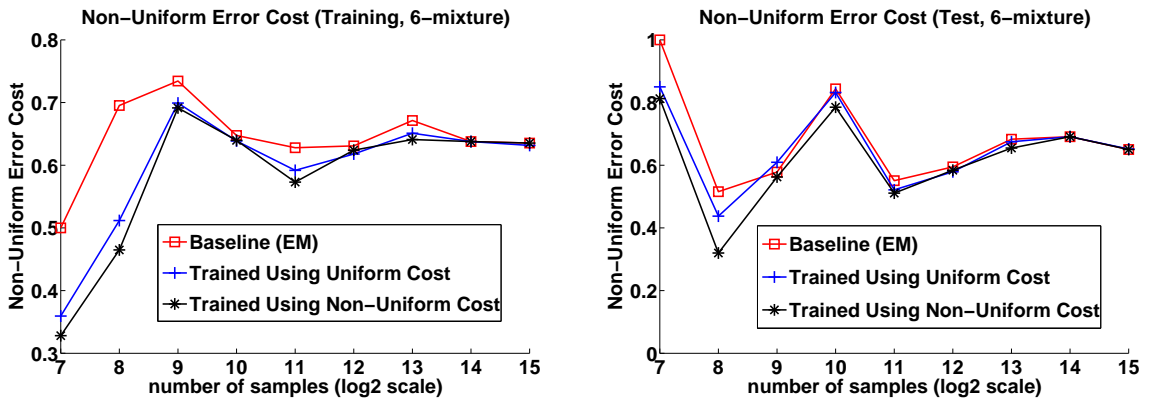
#### 4.6.3.2 Modeling $g(X; \Lambda)$ as 6-mixture GMMs

Figure 14, Fig.15 and Fig.16 are similar figures corresponding to Fig.11, Fig.12 and Fig.13 except that we assume  $g(X; \Lambda)$  to be 6-mixture GMMs. Similar numeric behaviors are observed for all curves in this section. For the training data set with 256 points and the corresponding test set, the EM-trained baseline has 0.70 and

0.52 normalized error cost for the training and test set respectively, while the MCE training achieves 0.52 and 0.31 error cost after 10 iterations correspondingly.



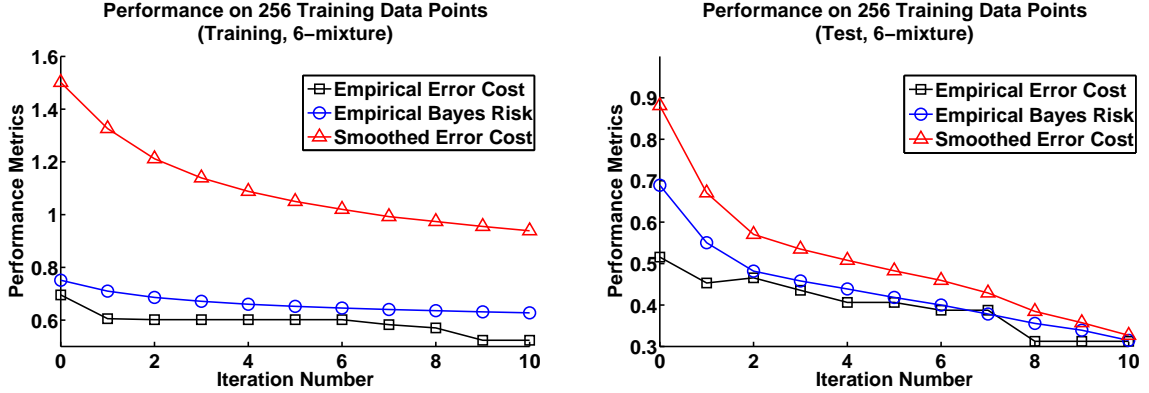
**Figure 14:** Uniform error cost for training and test set when modeling  $g(X; \Lambda)$  as a 6-mixture GMM



**Figure 15:** Non-uniform error cost for training and test set when modeling  $g(X; \Lambda)$  as a 6-mixture GMM

#### 4.6.4 Discussions

We have studied the non-uniform error cost criteria under different scenarios. There are several issues that deserve further discussion. First, the performance of the non-uniform error training is consistent with our expectation when the model of the classifier has the identical form with the true data distribution. We can see that both modeling strategies of the discriminant scoring function  $g(X; \Lambda)$ , either the more



**Figure 16:** Performance over training iterations with 256 training data when modeling  $g(X; \Lambda)$  as a 6-mixture GMM.

complex one as in Eq.(52) or the simplified version as in Eq.(71), show their effectiveness in reducing the empirical error cost and other related performance metrics. Furthermore, a matched-objective condition (i.e., the training objective and the error metric used in performance assessment are consistent) always leads to a better performance as expected. Second, in the model mismatch scenario, the simplified modeling for the discriminant function  $g(X; \Lambda)$  as (71) is flexible and effective in our experiments. Though the classifier model is different from the true data distribution, we still achieve considerable improvement when the ratio of the number of data over the degree of freedom for model parameters is not very large. As an illustrative example, we can observe the obvious decrease in the error cost for the data set with 256 training points. In fact, if we list the performance for different models using 256 training data in Table 2, we can see that when the model matches the real distribution (i.e., 4-mixture GMM), either modeling the scoring function  $g(X; \Lambda)$  as a function of the error cost as in Eq.(52) or modeling it as a single GMM as Eq.(71), the performances are close.

If the model is mismatched but “redundant” (i.e., 6-mixture GMM), the performance is close to the matched case on the training set but the performance on the test set is not as good as the matched case. If the model is mismatched and “worse” (i.e.,

2-mixture GMM) than the real distribution, the performance on both set is worse.

**Table 2:** The normalized error cost comparison between different modeling technologies for the scoring function  $g(X; \Lambda)$  at the operating point of using 256 training data.

	$g(X; \Lambda)$ as a function of Error Cost 4-mix	$g(X; \Lambda)$ as GMM		
		4-mix	2-mix	6-mix
Baseline, Training	0.70	0.69	0.84	0.70
Weighted MCE, Training	0.53	0.53	0.62	0.52
Baseline, Test	0.30	0.29	0.62	0.52
Weighted MCE, Test	0.13	0.11	0.38	0.31

Third, when the number of training data becomes very large (in our experiments, larger than  $2^{11} = 2048$ ), the differences among various training methods diminish. In other words, the size of the training data has an overriding effect on the performance of a recognizer.

## 4.7 Chapter Summary

In this chapter, motivated by the conventional MCE method, we revisited the Bayes decision theory and derived a framework to solve the problem of optimal classifier design with non-uniform error cost. Two types of training methods were introduced. With the knowledge of the form of the data distribution, we could use the gradient descent methods to update the parameters of the classifier model to minimize the error cost. A simplification was possible when the data distribution was not available. An example of Gaussian mixture classifiers with detailed parameter optimization equations for both methods was derived and implemented. Using computer-generated data samples, we conducted demonstrative simulations to verify the effectiveness of our training methods. We showed that the non-uniform error cost training criteria can substantially reduce the error cost even when the model of the classifier was not exactly consistent with the true data distribution.

## CHAPTER V

# NON-UNIFORM ERROR COST CRITERIA FOR SPEECH RECOGNITION

In the last chapter, the non-uniform error cost criterion was proposed and the corresponding system training algorithm was developed. However, to apply this criterion to real applications such as speech recognition tasks, some important implementation issues deserve careful considerations. First, in many cases, the real data distribution is rarely known, thus the evaluation function  $g(X; \Lambda)$  in decision rule (33) can rarely be modeled as a decreasing function of the error cost as of (52). Therefore, the training error has to be accumulated in a *post-decision* manner as of (71). Thus, the non-uniform error cost is being incorporated in system training in an error “weighting” fashion. Usually in speech recognition tasks, a standard solution is to model the evaluation function  $g(X; \Lambda)$  by an HMM model. Second, in real applications with complicated training scenarios such as ASR tasks, the formulation of the objective function and optimization strategies may require task-dependent adjustment. In particular, the selection of the error cost function is critical for the final system performance.

In this chapter, we apply the non-uniform error cost criteria to ASR applications and discuss the implementation details mentioned before. First, we make a minor change on the objective function of the non-uniform error cost criterion so that the numerical behavior is more friendly in ASR tasks. In addition, we explore the relation between the weighted MCE method and the MPE/MWE method. We then present the optimization equations for updating HMM model parameters. Finally, an investigation is conducted for the methodology of building appropriate error cost

functions in different training scenarios.

## 5.1 Revised Non-Uniform Error Cost Framework for ASR Tasks

Though the implementation strategies of the non-uniform error cost criterion may vary in different applications, the essence of building the evaluation function and the objective function has to be consistent. For any specific pattern recognition task, the system designer needs to model the evaluation function task-dependently and an approximation of the classification/recognition error with good numerical behaviors needs to be properly chosen. In speech recognition, the first problem is usually solved by using the HMM model as the evaluation function and we follow the convention in this thesis. For the second concern, we make a minor change to the smoothing function in the objective function. Note that the change of the smoothing function in (38) is entirely technical. Any smoothing function which approximates the indicator function in (38) well can be used in the objective function according to different tasks. We also derive the updating equations for HMM models in this section.

### 5.1.1 Smoothing of the Empirical Cost for Parameter Optimization

Recall the empirical error cost in Eq.(36) and Eq.(37), we need to find an appropriate smoothing function for the objective function of the weighted MCE method. Specifically, The approximation can be accomplished by

$$\sum_{i \in I_M} \epsilon_{ij} \mathbf{1}\{i = \arg \max_k g_k(X; \Lambda)\} \approx \sum_{i \in I_M} \epsilon_{ij} h_i(g_1(X; \Lambda), \dots, g_M(X; \Lambda)) \quad (78)$$

where  $h_i(x)$  is a smoothing mechanism to mimic the indicator function.

The sigmoid function with proper argument can be employed to approximate the indicator function in the selection function of Eq.(37).

$$h_i(g_1(X; \Lambda), \dots, g_M(X; \Lambda)) = h_i(d_i(X; \Lambda)) = \frac{1}{1 + \exp\{-\gamma d_i(X; \Lambda) + \theta\}} \quad (79)$$

where  $\gamma$  and  $\theta$  are two constants to decide the curvature and the decision threshold of the sigmoid function. We usually set  $\gamma = 1$  and  $\theta = 0$ . The function of  $d_i(X; \Lambda)$  in (79) is defined as

$$d_i(X; \Lambda) = g_i(X; \Lambda) - G_i(X; \Lambda) \quad (80)$$

and

$$G_i(X; \Lambda) = \log \left[ \frac{1}{M-1} \sum_{k \neq i} \exp\{\eta g_k(X; \Lambda)\} \right]^{1/\eta} \quad (81)$$

in which  $\eta > 0$  can be chosen as tradeoff between approximation and smoothness.  $M$  is the number of classes. Note that as  $\eta \rightarrow \infty$ ,

$$G_i(X; \Lambda) \approx \max_{k \neq i} g_k(X; \Lambda) \quad (82)$$

The smoothing function in Eq.(79) is similar to the smoothing function defined below (see [22]).

$$h_i(g_1(X; \Lambda), \dots, g_M(X; \Lambda)) = \frac{g_i(X; \Lambda)}{[\sum_i g_i^\eta(X; \Lambda)]^{1/\eta}} \quad (83)$$

We have demonstrated that by using Eq.(83), remarkable performance gain can be achieved in general pattern recognition experiments.

We need to note that, though the formulation of the smoothing function  $h_i(g_m(X; \Lambda))$ ,  $\forall m \in M$  is similar to the conventional MCE loss function  $l(\cdot)$  in Eq.(22), the objectives of these two functions are totally different. In the conventional MCE method, the loss function  $l(\cdot)$  calculates the discrepancy of likelihood between recognizing the underlying training token into the label class and other classes. The *class index* can be viewed as a known parameter of the loss function. On the other hand, the smoothing function  $h_i$  here aims at approximating an indicator function, which means that the *class index* is the *output* (or *return value*) of this function.

Finally, the smoothed empirical system cost is

$$\mathcal{F}_{W-MCE} \approx \frac{1}{N} \sum_{X \in \Omega} \sum_{j \in I_M} \left( \sum_{i \in I_M} \epsilon_{ij} h_i(X; \Lambda) \right) \mathbf{1}[X \in C_j] \quad (84)$$

which is a continuous function of the parameter set  $\Lambda$ .

Optimization methods such as the gradient descent methods (e.g., generalized probabilistic descent method (GPD)) [39][36] or the extended Baum-Welch (EBW) method [72][64][29] can be used to update the parameters in (84).

### 5.1.2 Updating Equations for Weighted MCE Training Using GPD Optimization

We assume that the discriminant function  $g(X; \Lambda)$  is modeled by an HMM with  $N$  states in which each state is a Gaussian mixture with  $K$  components. The parameter set for model  $i$  becomes  $\Lambda^{(i)} = \{A^{(i)}, c_{jk}^{(i)}, U_{jk}^{(i)}, R_{jk}^{(i)}\}$  where  $A^{(i)} = [a_{sj}]^{(i)}$  is the transition matrix,  $c_{jk}^{(i)}$  is the weight for the  $k^{th}$  mixture component in the  $j^{th}$  state,  $U_{jk}^{(i)} = [\mu_{jkl}^{(i)}]_{l=1}^D$  is the mean vector and  $R_{jk}^{(i)}$  is the corresponding covariance matrix which, for simplicity, is assumed to be diagonal, i.e.,  $R_k^{(i)} = [(\sigma_{jkl}^{(i)})^2]_{l=1}^D$ . We also assume that  $\mathbf{X} = (X_1, \dots, X_t, \dots, X_T)$  is a series of feature vectors with duration time  $T$  where  $X_t = (x_t^1, \dots, x_t^l, \dots, x_t^D)^T$ .

We have to set up similar constraints as (56)–(58): 1) the transition probability  $a_{sj}$  (from state  $s$  to state  $j$ ) needs to be non-negative; 2) the mixture function needs to be non-negative; 3)  $\sum_k c_{jk} = 1$ ,  $k = 1, \dots, M$ ; and 4)  $\sigma_{jkl} > 0$ ,  $k = 1, \dots, M$ ,  $l = 1, \dots, D$ . The following parameter transformations allow us to maintain these constraints during parameter optimization through Eq.(44):

$$a_{sj} \rightarrow \tilde{a}_{sj}, \text{ where } a_{sj} = \frac{e^{\tilde{a}_{sj}}}{\sum_n e^{\tilde{a}_{sn}}} \quad (85)$$

$$c_{jk} \rightarrow \tilde{c}_{jk}, \text{ where } c_{jk} = \frac{e^{\tilde{c}_{jk}}}{\sum_n e^{\tilde{c}_{jn}}} \quad (86)$$

$$\mu_{jkl} \rightarrow \tilde{\mu}_{jkl} = \frac{\mu_{jkl}}{\sigma_{jkl}} \quad (87)$$

$$\sigma_{jkl} \rightarrow \tilde{\sigma}_{jkl} = \ln \sigma_{jkl} \quad (88)$$

The updating equation can be further written as

$$\frac{\partial h_i(X; \Lambda)}{\partial \tilde{\Lambda}^{(i)}} = \frac{\partial h_i(X; \Lambda)}{\partial d_i(X; \Lambda)} \frac{\partial d_i(X; \Lambda)}{\partial \tilde{\Lambda}^{(i)}} = \gamma h_i (1 - h_i) \sum_j \sum_{t=1}^T \delta(q_t - j) \frac{\partial g_i(X_t; \Lambda)}{\partial \tilde{\Lambda}_j^{(i)}} \quad (89)$$

where  $\delta(\cdot)$  is the Kronecker delta function,  $q_t$  is the state number on time frame  $t$ .

The derivatives of  $\frac{\partial g_i(X_t; \Lambda)}{\partial \tilde{\Lambda}_j^{(i)}}$  for  $\{\tilde{a}_{sj}^{(i)}, \tilde{c}_{jk}^{(i)}, \tilde{\mu}_{jkl}^{(i)}, \tilde{\sigma}_{jkl}^{(i)}\}$  are (see [36])

$$\frac{\partial g_i(X_t; \Lambda)}{\partial \tilde{a}_{sj}^{(i)}} = a_{sj}^{(i)}(1 - a_{sj}^{(i)})[g_i(X_t; \Lambda)]^{-1} \quad (90)$$

$$\frac{\partial g_i(X_t; \Lambda)}{\partial \tilde{c}_{jk}^{(i)}} = \frac{c_{jk}^{(i)}(1 - c_{jk}^{(i)})[g_i(X_t; \Lambda)]^{-1}}{(2\pi)^{D/2}(\prod_{l=1}^D \sigma_{jkl}^{(i)})} \exp \left\{ -\frac{1}{2} \sum_{l=1}^D \left( \frac{x_t^l - \mu_{jkl}^{(i)}}{\sigma_{jkl}^{(i)}} \right)^2 \right\} \quad (91)$$

$$\frac{\partial g_i(X_t; \Lambda)}{\partial \tilde{\mu}_{jkl}^{(i)}} = \frac{c_{jk}^{(i)}[g_i(X_t; \Lambda)]^{-1}}{(2\pi)^{D/2}(\prod_{l=1}^D \sigma_{jkl}^{(i)})} \left( \frac{x_t^l - \mu_{jkl}^{(i)}}{\sigma_{jkl}^{(i)}} \right) \exp \left\{ -\frac{1}{2} \sum_{l=1}^D \left( \frac{x_t^l - \mu_{jkl}^{(i)}}{\sigma_{jkl}^{(i)}} \right)^2 \right\} \quad (92)$$

$$\frac{\partial g_i(X_t; \Lambda)}{\partial \tilde{\sigma}_{jkl}^{(i)}} = \frac{c_{jk}^{(i)}[g_i(X_t; \Lambda)]^{-1}}{(2\pi)^{D/2}(\prod_{l=1}^D \sigma_{jkl}^{(i)})} \left[ \left( \frac{x_t^l - \mu_{jkl}^{(i)}}{\sigma_{jkl}^{(i)}} \right)^2 - 1 \right] \exp \left\{ -\frac{1}{2} \sum_{l=1}^D \left( \frac{x_t^l - \mu_{jkl}^{(i)}}{\sigma_{jkl}^{(i)}} \right)^2 \right\} \quad (93)$$

The last step is to convert  $\tilde{\Lambda}$  back into  $\Lambda$  according to Eq.(85)-(88).

## 5.2 Applying the Non-Uniform Error Criterion to ASR Applications

### 5.2.1 An Example for Non-Uniform Error Cost in ASR

Here is an example of using the non-uniform error cost as the recognition performance measure for better information understanding. Two recognized strings with an identical equal-significance word error rate are displayed as follows:

**0** AT N. E. C. THE NEED FOR INTERNATIONAL MANAGERS WILL KEEP RISING

**1** AT ANY < del > SEE THE NEED FOR INTERNATIONAL MANAGERS WILL KEEP RISING

**2** AT N. E. C. < del > NEEDS FOR INTERNATIONAL MANAGER'S WILL KEEP RISING

Item 0 shown above is a transcription of the first utterance (440c0201.lab) in the test set of the Wall Street Journal (WSJ0) [62][43] database. Item 1 displays a recognition result with two substitution errors and one deletion error. Item 2 is another recognition result with the same error counts. These two recognition results contributed to identical error statistics in terms of the equal-significance word error rate. However, we can retrieve the correct information from the second string with almost no difficulty but the company name is totally lost in the first one. Hence, the second string should be viewed as a better recognition result because it has retained useful information.

Consider the task of acoustic modeling for words and the differentiation in error significance is being applied to words. One straightforward error significance weighting function in this scenario is the Shannon information of word  $-\log P(\text{word})$  in the whole training corpus. This weighting function reasonably assumes that the less frequently a word appears, the more information it contains. We thus define a weighted word error rate as

$$\text{Weighted Error Rate} = \frac{\sum_{s=1}^S [-\log P(w_s)] + \sum_{d=1}^D [-\log P(w_d)] + \sum_{i=1}^I [-\log P(w_i)]}{\sum_{n=1}^N [-\log P(w_n)]} \quad (94)$$

where  $N$  is the total number of words, and  $S$ ,  $D$ , and  $I$  are the number of substitution, deletion and insertion errors, respectively.  $w_s, w_d$ , and  $w_i$  are the words in the corresponding errors in substitution, deletion and insertion, respectively. We obtain the following table in which the Shannon information  $-\log P(\text{word})$  of each word is listed below for each word sequence:

**0** AT N. E. C. THE NEED FOR INTERNATIONAL MANAGERS WILL KEEP  
RISING

2.317 3.138 3.135 2.784 1.275 3.675 2.027 3.259 3.797 2.481 3.689 3.925

**1** AT ANY < del > SEE THE NEED FOR INTERNATIONAL MANAGERS WILL

KEEP RISING

2.317 3.038 < del > 3.503 1.275 3.675 2.027 3.259 3.797 2.481 3.689 3.925

2 AT N. E. C. < del > NEEDS FOR INTERNATIONAL MANAGER'S WILL  
KEEP RISING

2.317 3.138 3.135 2.784 < del > 3.966 2.027 3.259 3.719 2.481 3.689 3.925

Based on Eq.(94), the weighted recognition error rate of the first string is 27.25% while the second one outperforms this number and achieves 25.24%. This example demonstrates the effectiveness of the weighted word error rate. Other significance weighting functions are possible. For example, one could associate proper nouns with substantially higher significance than common words. The error rate differentiation could have been much higher than about 2% as demonstrated.

### 5.2.2 Training Scenarios and Weighting Strategies in ASR

Speech recognition is an important category of pattern recognition applications. In brief, there are at least two scenarios with non-uniform error training criteria in ASR. In the first case, the training and recognition decisions are on the same linguistic level of the performance measure. For example, acoustic models may be trained on the *phone* level and the evaluation metric is the weighted *phone* error rate (PER). In this case, the loss incurred by wrong recognition decisions represents the recognizer's performance directly. We call this scenario the **intra-level training**. The second and the most common circumstance in large vocabulary speech recognition is the **inter-level** training in which the training and recognition decisions are not on the same linguistic level as the performance metric. For example, training and recognition are performed on the *phone* level but the system evaluation measure is defined as the weighted *word* error rate (WER). In this case, the system performance is not

evaluated by the recognition error loss. Hence, minimizing the cost of wrong recognition decisions does not directly optimize the recognizer’s performance in terms of the evaluation measure. To alleviate this inconsistency, the error weighting strategy needs to be built in a cross-level fashion.

In both training scenarios, the error weighting mechanism can be constructed according to two types of error cost: a user-defined cost and a data-defined cost. The user-defined cost is usually characterized by the system requirement and relatively straightforward. For example, for political news recognition, recognition errors like replacing "Iraq" with "Iran" may be viewed as many times more significant than, errors like replacing "baseball" with "basketball". The data-defined cost is more complicated. The Bayes decision theory aims at minimizing the expected error loss, which is generated by the wrong recognition decisions. A wrong decision usually occurs because the underlying data observation deviates substantially from the distribution represented by the corresponding recognizer model. By imposing the data-defined weights, we want to construct such a mechanism that the recognizer is trained in favor of those "reliable" data and those "outliers" are discarded.

### 5.2.2.1 Error Weighting for Intra-Level Training

In the intra-level training situation, the system performance is directly measured by the loss of wrong recognition decisions and it is rather straightforward to accomplish error weighting through the error cost function  $\epsilon_{ij}$ . Assume that the training is on the *phone* level and the evaluation measure is defined as the weighted *phone* error rate. The phone sequence  $PH = (ph_1, ph_2, \dots, ph_{L_n})$  is the label of the  $n^{th}$  training token in a training set with totally  $N$  tokens.  $X_n = \{X_{n,l_n}\}_{l_n}^{L_n}$  is the  $n^{th}$  token that is segmented into  $L_n$  segments corresponding to the phone sequence. Following the objective function of the weighted MCE training as (84), in which  $g(X; \Lambda)$  can be defined as (71) and the error loss function  $\epsilon_{ij}$  contains both types of error cost, the

objective function for the weighted MCE in this case can be written as

$$\mathcal{F}'_{W-MCE} = \frac{1}{N} \sum_{n=1}^N \sum_{l_n=1}^{L_n} \sum_i \epsilon_{ij} h_i \{d_i(X_{n,l_n}; \Lambda)\} \mathbf{1}(X_{n,l_n} \in C_j) \quad (95)$$

where  $h_i \{d_i(X_{n,l_n}; \Lambda)\}$  is defined in Eq.(79). We can interpret Eq.(95) as an accumulation of the error cost for all phones and all classes over the entire training set. After each training epoch, the error cost function  $\epsilon_{ij}$  can be adjusted adaptively if desired.

### 5.2.2.2 Error Weighting for Inter-Level Training

In the inter-level training situation, the system performance is not measured directly by the loss due to wrong recognition decisions. The recognition decisions need to be grouped to form the system output which is on the level of the performance metric. Therefore, we need to use cross-level weighting in this case to break down the higher level cost and impose the appropriate weights upon the lower level models.

Assume that in this case, the training is on the phone model and the performance metric is the weighted word error rate. The first weighting mechanism we consider is the user-defined weighting. Let the word sequence  $W = (w_1, w_2, \dots, w_{L_n})$  be the label of the  $n^{th}$  training token in a training set with totally  $N$  tokens. Each word  $w_{l_n}$  contains a phone sequence as  $ph_{l_n}^1, ph_{l_n}^2, \dots, ph_{l_n}^{K_n}$ .  $X_n = \{X_{n,l_n,k_n}\}_{l_n}^{L_n}$  is the  $n^{th}$  token that is segmented into  $L_n$  segments corresponding to the word sequence. Since the user's demands are normally engaged on the level of the system performance metric, the user-defined cost of each phone in word  $w_{l_n}$  can be set identically using the word-level cost. Hence, the user-defined weighting of the weighted MCE in inter-level training can be written as:

$$\mathcal{F}''_{W-MCE} = \frac{1}{N} \sum_{n=1}^N \sum_{l_n=1}^{L_n} \sum_{k_n=1}^{K_n} \sum_i \epsilon_{ij} h_i \{d_i(X_{n,l_n,k_n}; \Lambda)\} \mathbf{1}(X_{n,l_n,k_n} \in C_j) \mathcal{E}_u(w_{l_n}) \quad (96)$$

where  $\mathcal{E}_u(w_{l_n})$  is the user-defined cost for word  $w_{l_n}$ . Compared to the original definition of the weighted MCE objective (84), this objective function utilizes the class-dependent error cost at the higher level (word level) to control the optimization of

the parameters at the lower level (phone level). One instance of the user-defined error weighting function is the keyword spotting system. As the example we mentioned in Chapter 5.2.2, the cost of misrecognizing keywords is much higher than the cost of non-keyword errors.

The formulation of the data-defined weighting is more complex and flexible in the inter-level training. Since the objective of the data-defined weighting is to find the definitive errors, it can be imposed upon any level of errors. In this situation, based on the training situation assumed above, the objective function of the weighted MCE method can be written as follows:

$$\mathcal{F}''_{W-MCE} = \frac{1}{N} \sum_{n=1}^N \sum_{k_n=1}^{K_n} \sum_i \epsilon_{ij} h_i \{d_i(X_{n,l_n,k_n}; \Lambda)\} \mathbf{1}(X_{n,l_n,k_n} \in C_j) \mathcal{E}_d(m) \quad (97)$$

where  $m = \{k_n, l_n, n\}$  and  $\mathcal{E}_d(m)$  can be the data-defined weighting for the  $k_n^{th}$  phone, the  $l_n^{th}$  word, or the  $n^{th}$  training token. The definition of the data-defined error is very flexible. We can assign the cost of  $l_n^{th}$  word to each phone in it or directly compute the phone-level error cost for the  $k_n^{th}$  phone in the  $l_n^{th}$  word. One example of the data-defined weighting is to use the Shannon information as mentioned in Chapter 5.2.1, i.e.,  $\mathcal{E}_d(w_{l_n}) = -\log P(w_{l_n})$ . Another well-known example is the popular MPE/MWE method.

Finally, a complete weighted MCE objective function for inter-level training with both weighting factors can be written as

$$\mathcal{F}''_{W-MCE} = \frac{1}{N} \sum_{n=1}^N \sum_{k_n=1}^{K_n} \sum_i \epsilon_{ij} h_i \{d_i(X_{n,l_n,k_n}; \Lambda)\} \mathbf{1}(X_{n,l_n,k_n} \in C_j) \mathcal{E}_u(w_{l_n}) \mathcal{E}_d(m) \quad (98)$$

where  $m = \{k_n, l_n, n\}$ .

### 5.2.3 Weighted MCE and MPE/MWE Method

In this section, we discuss the relation between the weighted MCE and the MPE/MWE method in order to obtain a better understanding of the error weighting mechanism. Recall the same inter-level training scenario in the last section. Assume that the

training is on the phone model and the performance metric is the weighted word error rate. Word/phone sequence  $W = (w_1, w_2, \dots, w_{L_n})$  is the label of the  $n^{\text{th}}$  training token in a training set with totally  $N$  tokens.  $X_n = \{X_{n,l_n}\}_{l_n=1}^{L_n}$  is the  $n^{\text{th}}$  token that is segmented into  $L_n$  segments corresponding to word/phone sequence. We also assume that there is no user-defined weighting for simplicity. A weighted MCE objective function can be constructed using the local word/phone probability given the observation as the weighting factor. The objective function can be written as:

$$\mathcal{F}_{W-MCE} = \frac{1}{N} \sum_{n=1}^N \sum_{l_n=1}^{L_n} \sum_i h_i\{d_i(X_{n,l_n}; \Lambda)\} Pr(w_{l_n}|X_n) \quad (99)$$

The minimum phone/word error (MPE/MWE) training method is a popular discriminative training method with a weighted objective function to mimic training errors [66]. The objective function of MPE/MWE is defined in equation (103).

$$\mathcal{F}_{MPE/MWE}(\Lambda) = \frac{1}{N} \sum_{n=1}^N \frac{\sum_{l_n \in L_n} p^\alpha(X_n|w_{l_n}) P^\beta(w_{l_n}) A(W, W_n)}{\sum_{\forall u} p^\alpha(X_n|w_u) P^\beta(w_u)} \quad (100)$$

$$\approx \frac{1}{N} \sum_{n=1}^N \frac{\sum_{l_n \in L_n} p^\alpha(X_n|w_{l_n}) P^\beta(w_{l_n}) A(W, W_n)}{\sum_{u=l_n} p^\alpha(X_n|w_u) P^\beta(w_u) + \sum_{u \neq l_n} p^\alpha(X_n|w_u) P^\beta(w_u)} \quad (101)$$

$$= \frac{1}{N} \sum_{n=1}^N \frac{P_c A(W, W_n)}{P_c + P_w} \quad (102)$$

$$= \frac{1}{N} \sum_{n=1}^N A(W, W_n) P(W_n|X_n) \quad (103)$$

$$= \frac{1}{N} \sum_{n=1}^N A(W, W_n) \left(1 - \frac{P_w}{P_c + P_w}\right) \quad (104)$$

$$= \frac{1}{N} \sum_{n=1}^N A(W, W_n) [1 - \sum_{i \neq l_n} h_i(d_i(X_{n,l_n}; \Lambda))] \quad (105)$$

In equations (100)–(105),  $A(W, W_n)$  is called “raw accuracy”, which is a measure of how many words/phones are correctly recognized in  $W_n$  according to the transcription  $W$  [66]. We may interpret it as a rough estimate of the word/phone error

accumulation for each utterance. Parameters  $\alpha$  and  $\beta$  are the acoustic and language model scale factors, respectively. In this case we use an identical factor for simplicity in expression. We use  $P_c$  to represent the probability of applying the training observations on the transcribed string, and  $P_w$  as the sum of applying the observations on all other recognized strings. i.e.,

$$P_c = \sum_{l_n \in L_n} p^\alpha(X_{n,l_n}; \Lambda | w_{l_n}) P^\beta(w_{l_n}) \quad (106)$$

and

$$P_w = \sum_{u \neq l_n} p^\alpha(X_{n,l_n}; \Lambda | w_u) P^\beta(w_u) \quad (107)$$

Therefore, maximizing the original MPE/MWE objective function in (100) is equivalent to maximizing (105) or minimizing the modified objective function:

$$\mathcal{F}'_{MPE/MWE}(\Lambda) = \frac{1}{N} \sum_{n=1}^N \sum_{i \neq l_n} h_i(d_i(X_{n,l_n}; \Lambda)) A(W, W_n) \quad (108)$$

The MPE/MWE method weights the utterance errors by the “raw accuracy”  $A(W, W_n)$ , therefore builds a objective function that displays the error cost of each training utterance.

Except that the empirical error count is on different linguistic levels, the objective function of the MPE/MWE method in (105) is similar to the objective function of the weighted MCE method in (99). Both of them aim at minimizing training errors weighted by a data-defined cost function. In summary, the relationship between the weighted-MCE and the MPE/MWE can be described as two training algorithms both rooted in the Bayes decision theory, pointing to the same aim of designing the optimal classifier to minimize a weighted error rate.

### ***5.3 Experimental Results for Speech Recognition***

To demonstrate the effectiveness of the weighted MCE training criteria in ASR, we designed two sets of experiments – a small vocabulary task on the TIDIGITS [49]

database and a large vocabulary task on the WSJ [62][43] database. In the small vocabulary task, both the intra-level and inter-level training scenarios were investigated. In the large vocabulary task, we presented some results for the weighted MCE training method. Because of the computational complexity for the large vocabulary task, necessary simplifications were employed for implementation efficiency. In all experiments, we used the GPD method described in [36] and [22] to optimize model parameters. Note that the purpose of the experiments is to verify the correctness of our methodology rather than the design of a particularly superior system.

### 5.3.1 Small Vocabulary Task – TIDIGITS

#### 5.3.1.1 Baseline System

We evaluated two baseline systems on the TIDIGITS database (with 8621 training utterances and 8700 test utterances) for the intra-level and inter-level training, respectively.

For intra-level training, the 11 whole-digit models were constructed using HMMs as the baseline. They are the English digits from 0 to 9 plus the word “oh”. Each HMM had 12 states and each state observation density was represented by a Gaussian mixture. The feature vector consisted of 12MFCCs + energy, and their first and second order time derivatives. The initial models were obtained by maximum likelihood estimation using the EM algorithm for 5 iterations.

The baseline system of inter-level training was built based on 21 HMM models corresponding to all phonemes occurred in digits. Each model consisted of 3 states and each state contained 32 Gaussian mixture components. The training procedure was identical to that for whole-word models.

#### 5.3.1.2 Intra-Level Training Using Whole Word Models

In this section, we use whole word models for the TIDIGITS database to study the non-uniform error cost criteria in the intra-level training situation in ASR.

One common intra-level weighting scenario in real applications is digits recognition in which some digits may be more important than others. Therefore, the experiments in this section are designed to demonstrate how effective for the weighted MCE method to reduce “important” errors.

In our experiments, the error cost matrix is an  $11 \times 11$  matrix whose rows and columns are framed in the order of  $1, 2, \dots, 9, 0$ , and “oh”. We assume that “4” and “oh” carry unique significance among the digits; “4” is not to be misrecognized as any other digit while it is imperative not to misrecognize any digit as “oh”. Thus, the error cost of misrecognizing “4” as any other digit is set to be 10, and the cost of misrecognizing any other digit as “oh” is also set to be 10. In other words, the value of entries in the error cost matrix is as follows:

$$\epsilon_{ij} = \begin{cases} 1, & i \neq j, i \neq 11, j \neq 4 \\ 0, & i = j \\ 10, & i = 11 \text{ or } j = 4 \end{cases} \quad (109)$$

The error cost matrix in Eq.(109) means that the errors of misrecognizing “4” as other digits and the errors of misrecognizing other digits as “oh” are important errors. We would strongly penalize them to avoid committing such errors.

We compare the results between using the MCE training with a uniform error cost matrix and the one using the weighted MCE method with the non-uniform error cost function of (109). All results are obtained after 5 training iterations. In Table 3, the second column displays the total errors committed by the recognizer. The third and the fourth columns show the number of errors when misrecognizing “4” as other digits and the number of errors when misrecognizing other digits as “oh”, respectively.

We can observe that though the total number of errors decreases from 110 to 75 in both kinds of MCE training, the weighted MCE training reduces the number of important errors more substantially than the conventional MCE training. This experiment illustrates that using the non-uniform error cost matrix can effectively

**Table 3:** Performance comparison between the conventional MCE training method and the weighted MCE training method using the non-uniform error cost matrix as defined in Eq.(109)

	Total Errors	“4” to others	others to “oh”
Baseline	140	6	11
MCE	75	5	8
Weighted MCE	75	2	4

**Table 4:** The confusion matrix of the baseline

	1	2	3	4	5	6	7	8	9	0	oh	Del
one	*	0	0	0	0	0	0	0	0	0	<b>0</b>	0
two	0	*	0	0	0	0	0	0	0	0	<b>4</b>	0
three	0	3	*	0	0	0	0	0	0	0	<b>0</b>	0
four	<b>1</b>	<b>0</b>	<b>0</b>	*	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>5</b>	0
five	0	0	0	0	*	0	0	0	0	0	<b>0</b>	0
six	0	0	0	0	0	*	1	0	0	0	<b>0</b>	0
seven	1	0	0	0	0	0	*	0	0	0	<b>1</b>	0
eight	0	0	0	0	0	1	0	*	0	0	<b>0</b>	1
nine	1	0	0	0	1	0	0	0	*	0	<b>1</b>	0
zero	0	0	0	0	0	0	0	0	0	*	<b>0</b>	0
oh	0	0	0	0	0	0	1	0	0	0	*	18
<b>Ins</b>	2	0	0	0	0	1	0	3	1	1	92	

reduce “important” errors.

In particular, we present the confusion matrix for the results of the baseline models and the models trained by the weighted MCE method in Table 4 and 5, respectively. Note that the value of the entry at the  $i^{th}$  row and  $j^{th}$  column represents the frequency of misclassifying a phone belonging to the  $i^{th}$  class into the  $j^{th}$  class. The column of “Del” and the row of “Ins” represent the deletion and insertion errors, respectively. This denotation is a transpose of the representation of the error cost matrix. The number of correct decisions at the diagonal entries is omitted to avoid distraction. We can see that the “important” errors defined in this case have been substantially decreased though other errors may increase.

**Table 5:** The confusion matrix of the weighted MCE models

	1	2	3	4	5	6	7	8	9	0	oh	<b>Del</b>
one	*	0	0	0	0	0	0	0	0	0	<b>0</b>	0
two	0	*	1	0	0	0	0	0	0	0	<b>3</b>	0
three	0	2	*	0	0	0	0	0	0	0	<b>0</b>	0
four	<b>1</b>	<b>0</b>	<b>0</b>	*	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>1</b>	0
five	0	0	0	0	*	0	0	0	0	0	<b>0</b>	0
six	0	0	0	0	0	*	2	0	0	0	<b>0</b>	0
seven	1	0	0	0	0	0	*	0	0	0	<b>0</b>	0
eight	0	0	0	0	0	1	0	*	0	0	<b>0</b>	1
nine	1	0	0	0	2	0	0	0	*	0	<b>0</b>	0
zero	0	0	0	0	0	0	0	0	0	*	<b>0</b>	0
oh	0	0	0	0	0	0	1	0	0	0	*	34
<b>Ins</b>	1	1	0	0	0	1	0	3	2	1	15	

### 5.3.1.3 Inter-Level Training Using Phone Models

The most common training scenario in ASR is inter-level training because we can not afford to train whole word models in large vocabulary tasks. The main challenge in this scenario, in which the recognition errors are not the direct performance metric, is to adjust the cost of decision error so that the system performance measure is optimized.

In this section, we investigate the performance of inter-level training using a non-uniform error cost function to weight phone recognition errors. The experiments were conducted on the TIDIGITS database, too. The objective of selecting the error cost matrix is to reduce the word error rate. Two methods to construct the non-uniform cost matrix are proposed in this section; the first one is relatively simple and heuristic and the second one has a more rigorous rationale.

The first method transforms the phoneme confusion matrix generated by the baseline decoder into a non-uniform error cost matrix. The transformation is guided by the idea that we want to impose more penalties on frequent errors. One implication behind this idea is that, had we had to tolerate a fixed number of word errors, it

might be preferred to achieve a phoneme confusion matrix with phone errors “uniformly distributed” rather than being concentrated in a few specific entries.

In our experiments, the confusion matrix was generated as an  $N \times N$  matrix, where the entry  $N_{(i,j)}$  signifies the number of substitution errors in identifying a phone from the  $i^{th}$  class as one of the  $j^{th}$  class. The deletion and insertion errors for each phone are listed separately. The construction of the error cost matrix consists of three steps. First, for each entry, the corresponding deletion and insertion errors are normalized by the number of phoneme classes and accumulated into the “total errors” occurring at that entry as well as substitution errors. Second, each entry is smoothed by adding a small number so that for every  $i \neq j$ ,  $N_{(i,j)} > 0$ . This step assures that each phone error has a positive cost. In our experiments, the small number we used is 1.0. Each smoothed entry in the same row is then normalized by the row-wise summation. Finally, we transpose the confusion matrix to form the error cost matrix where entry  $\epsilon_{(i,j)}$  signifies the number of errors in identifying a phone from the  $j^{th}$  class as one of the  $i^{th}$  class. Note that the magnitude of entries in the error cost matrix may need further amplifications for better discriminative capabilities. In our experiments, we took a power of 2 for all entries  $\epsilon_{(i,j)}, \forall i, j$ .

The second method to build the error cost matrix uses the “sensitivity” of word errors for each phone error. The *word error sensitivity* of a phone error is defined as the number of possible word errors caused by this phone error. One possible method to compute the sensitivity can be summarized as follows: Assume phone  $j$  is misrecognized as phone  $i$ . For word  $w_n$  with phone  $j$ , replace the phone  $j$  by phone  $i$  and form a new phone sequence  $w'_n$ . Compare the distance (e.g., the Kullback-Leiber distance [17]) between  $w'_n$  and all words in the lexicon. If the model of  $w_n$  is the closest one to the model of  $w'_n$ , the recognizer is still likely to make the correct word decision even though there exists a phone error. In this case, the word error sensitivity of misrecognizing phone  $j$  to phone  $i$  is 0. Otherwise, the sensitivity is defined as the

number of words whose models are closer to the model of  $w'_n$  than the model of  $w_n$ . In our experiments, all models are assumed to be HMMs.

We can see that the most critical issue in calculating the word error sensitivity of a phone model is to compute the distance between two HMM models. This can be accomplished by the probabilistic model distance measure of [68] or by accumulating distances between corresponding states as suggested in [16]. If both HMM models have the same number of states, the distance between them is computed as a combination of the distances between each pair of corresponding states (e.g., the first state of the first HMM and the first state of the second HMM). The distance between any two states, which are both GMM models, can be computed using the unscented transform mechanism in [16][26].

We also need to consider the effect of the transition probabilities when combining state distances for the distance between HMM models. The transition probabilities control the expected duration for each state and thus affect the global distribution of the HMM model. If we represent a state sequence of an HMM model as

$$\mathbf{q}_{(i,d_i)} = \{\cdots, s_{i-1}, \underbrace{s_i, \cdots, s_i}_{d_i}, s_{i+1}, \cdots\} \quad (110)$$

where  $d_i$  is the duration for state  $i$ , the probability distribution of duration  $d_i$  is

$$P(d_i) = P(\mathbf{q}_{i,d_i}) = (a_{ii})^{d_i-1}(1 - a_{ii}) \quad (111)$$

where  $a_{ii}$  is the transition probability of staying in state  $i$ . The expected duration  $d_i$  for state  $i$  in an HMM model is computed as

$$E(d_i) = \sum_{d_i=1}^{\infty} d_i P(d_i) = (1 - a_{ii}) \sum_{d_i=1}^{\infty} d_i (a_{ii})^{d_i-1} = \frac{1}{1 - a_{ii}} \quad (112)$$

Therefore, the distance between the two HMM models  $D(H^1||H^2)$  is computed by a sum of the distances between the corresponding states weighted by the normalized expected duration. i.e.,

$$D(H^1||H^2) = \sum_{j=1}^N \frac{E(d_j)}{\sum_i^N E(d_i)} D(s_j^1||s_j^2) \quad (113)$$

where  $N$  is the number of states in the HMM models and  $D(s_j^1||s_j^2)$  is the distance between state  $j$  of the first HMM and that of the second HMM computed using the unscented transformation.

If the HMM models do not have an identical number of states, we can align them into state sequences with the same length using dynamic programming techniques [68]. Note that we need to embed the transition probability effect into the alignment. The other issue is that we may have to take an appropriate power for all entries  $\epsilon_{(i,j)}, \forall i, j$  to increase the value of the error cost if necessary. In our experiments, we took the power of 2 for all entries.

We compare the performance of the MCE method using the uniform and the non-uniform error cost matrices for inter-level training in Table 6. Note that in our experiments, MCE training based on the uniform error cost function is carried out on the word-level (i.e., the objective function is constructed to minimize the word error rate). All results were obtained after 5 training iterations. W-MCE I and II in the table correspond to the weighted MCE training using the non-uniform error cost matrices generated by the first method and the second method described above, respectively. We can see that for both the word error rate (which is our system performance measure) and sentence/string error rate, the weighted MCE methods achieve better performance than the conventional MCE using the uniform error cost matrix. In addition, in our experiments on the TIDIGITS database, we do not observe an obvious performance difference between using different error cost matrices in the weighted MCE method.

### 5.3.2 Large Vocabulary Task – WSJ

In this section, we present some results for large vocabulary speech recognition tasks using non-uniform error cost training. Tri-phone models were employed in our experiments, since the number of words for large vocabulary tasks were too many to use

**Table 6:** Performance comparison between the conventional MCE training method and the weighted MCE training method using the non-uniform error cost matrix for phone models

	Word Error Rate (WER)	Sentence/String Error Rate (SER)
Baseline	1.53	4.64
MCE	1.40	4.36
W-MCE I	1.27	3.84
W-MCE II	1.32	3.90

the whole word models. We focus on studying the effect of data-defined weighting in the inter-level training situation in this section.

The experiments were organized based on a similar framework of word-graph based MCE training presented in [24]. The weighted MCE method is constructed by using an estimate of a confidence measure of the correct recognition decision as the data-defined weighting function. Note that the confidence measure we use is called “posterior probability” in [79]. We do not continue to use the term because as a very rough approximation, it may cause unnecessary confusion with the rigorous mathematical definition of the true posterior probability.

Recall the training scenario in Chapter 5.2.3 and assume that the word sequence  $W = (w_1, w_2, \dots, w_{L_n})$  is the label of the  $n^{th}$  training token in the training set with totally  $N$  tokens.  $X_n = \{X_{n,l_n}\}_{l_n}^{L_n}$  is the  $n^{th}$  token that is segmented into  $L_n$  segments corresponding to a word sequence. Recall that we define “target events” as the labeled identity or correct recognition decision for a speech segment while “competing events” refer to the possible recognition errors for the underlying speech segment.

### 5.3.2.1 Baseline System

The experiments were conducted on the WSJ0 database. The baseline recognizer followed the HTK recipe (see <http://www.inference.phy.cam.ac.uk/kv227/htk/>), and was based on continuous density Gaussian mixture hidden Markov models (CDHMM).

A word-internal context-dependent tri-phone set was formed with 7,385 physical models and 19,075 logic models. All models were represented by 3-state strict left-to-right HMMs, with 8 Gaussian mixture components per state. These models were trained first with the Maximum Likelihood (ML) objective implemented by the HTK toolkit. The experiments were then carried out by comparing the performance of the systems trained using different MCE criterion.

We generated feature vectors for all 7,077 utterances by 84 speakers in the training set of the WSJ0 corpus. Each feature vector had 12MFCC+12 $\Delta$ +12 $\Delta^2$  and 3 log energy values. The feature generation process was also applied to the Nov-92 evaluation set with 330 utterances by 8 speakers. The CMU6 recognition lexicon were employed, which contains 126,834 words. The word graphs were generated using the HTK toolkit, too. During the training procedure, a unigram language model was used. Bigram was applied to decode and generate word graphs, where the word insertion penalty and the language model scale factor were set to be  $-4.0$  and  $15.0$ , respectively. At most 3 candidate recognition strings were allowed to survive at the same time during word graph generation. Other baseline system details can be found in [81]. Finally, the word error rate of this baseline is 8.41%.

### 5.3.2.2 Using Confidence Measures as Weighting Function

In [80][75], the concept of generalized posterior probability (GPP) was proposed and GPP was used as a confidence measure for recognition hypotheses. To avoid unnecessary confusion of this term with the real posterior probability, we change the name of the GPP into *generalized confidence measure* (GCM). All training experiments were conducted in the context of word graphs. Assume the labeled word sequence  $W_n = (w_1, w_2, \dots, w_{L_n})$  corresponds to the observation vector  $\mathbf{X}_1^T = (X_1, X_2, \dots, X_T)$ . We denote a word  $w_{l_n}$  starting from time  $s$  and ending at time  $t$  as  $[w_{l_n}; s, t]$  which corresponds to observation  $\mathbf{X}_s^t = (X_s, X_{s+1}, \dots, X_t)$ . Hence, the confidence measure for

this word given  $\mathbf{X}_s^t$  can be written based on the definition in [80]

$$P([w_{l_n}; s, t]|\mathbf{X}_s^t) = \sum_{\forall n, [w_n; s_n, t_n]=[w_{l_n}; s, t]} \frac{P^\alpha(\mathbf{X}_s^t|w_{l_n})P^\beta(w_{l_n}|w_{l_n-1})}{P^\alpha(\mathbf{X}_s^t)} \quad (114)$$

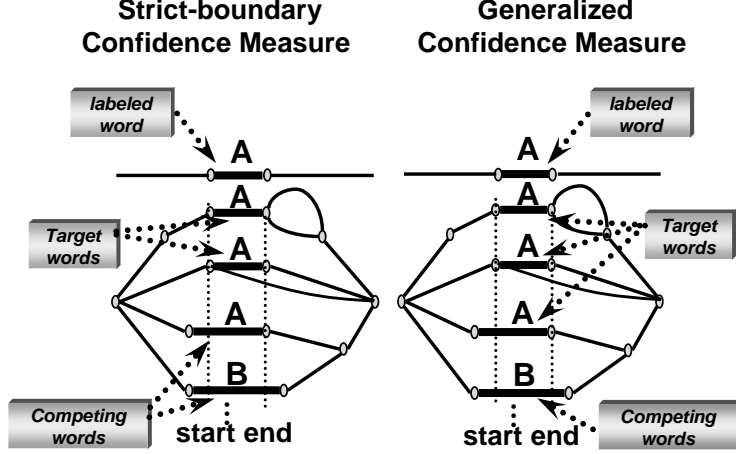
where  $w_n$  is a hypothesized word, and  $P(w_{l_n}|w_{l_n-1})$  is drawn from the language model.  $\alpha$  and  $\beta$  are acoustic and language model scale factors and were set to be 15.0 and 1/15, respectively. The constraint  $[w_n; s_n, t_n] = [w_{l_n}; s, t]$  implies the recognized word has the same identity and exact starting and ending time as the labeled one. Note that Eq.(114) is not identical to the original definition of the ‘‘posterior probability’’ in [80]. The original confidence measure for word  $w_{l_n}$  was defined as  $P([w_{l_n}; s, t]|\mathbf{X}_1^T)$  but we are using the local statistics  $P([w_{l_n}; s, t]|\mathbf{X}_s^t)$ . In recognition, the word might be correctly recognized but the time registration  $s$  and  $t$  of that hypothesis often does not exactly match the labeled segmentation. Since the word identity is more important than the timing information (unless it negatively impacts the recognition decision of the neighboring segments), it is desirable to relax the constraint upon word boundaries at this stage. Therefore, the GCM can be written as:

$$P_G([w_{l_n}; s, t]|\mathbf{X}_s^t) = \sum_{\forall n, w_n=w_{l_n}} \frac{P^\alpha(\mathbf{X}_s^t|w_{l_n})P^\beta(w_{l_n}|w_{l_n-1})}{P^\alpha(\mathbf{X}_s^t)} \quad (115)$$

*when*  $|s - s_n| + |t - t_n| < \tau$

where  $\tau$  is the limit for boundary relaxation. The equation indicates that once the hypothesis and the label have the same identity and a reasonable overlap in time, we will count it as a part of the target events. Figure 17 illustrates this idea. The rule of the strict-boundary confidence measure is shown on the left panel. We can see that words in the word graph with inconsistent boundaries compared to the labeled word are always put into the competing event set. On the right panel, GCM would count all the words with same identity and overlapped duration of the label as the target events.

Based on the derivation in the last section, we can define the objective function



**Figure 17:** Illustration of the strict-boundary confidence measure and the GCM.

of the weighted MCE as:

$$\mathcal{F}_{W-MCE}(\Lambda) = \frac{1}{N} \sum_{n=1}^N \sum_{l_n=1}^{L_n} h_i\{d_i(X_s^t; \Lambda)\} P_G(w_{l_n} | \mathbf{X}_1^T) \quad (116)$$

where  $N$  is the number of total training tokens. We use the “generalized confidence measure” for the word  $w_i$  given the total observation  $\mathbf{X}_1^T$  as the data-defined weighting function. It is defined as:

$$P_G(w_{l_n} | \mathbf{X}_1^T) = \sum_{\forall n, w_n = w_{l_n}} \frac{P^\alpha(\mathbf{X}_1^T | w_{l_n}) P^\beta(w_{l_n} | w_{l_n-1})}{P^\alpha(\mathbf{X}_1^T)} \quad (117)$$

when  $|s - s_n| + |t - t_n| < \tau$

Here the weights  $Pr(w_{l_n} | \mathbf{X}_1^T)$  is fixed during optimization process for simplicity.

### 5.3.2.3 Experimental results

In Table 7, we compared the weighted MCE and the conventional MCE method respectively on three levels: the word level, the phone level and the state level. Word level training means training error cost is computed for each *word*, as are the other levels. We can see that the weighted MCE method outperforms the conventional (non-weighted) MCE method in all categories. Assuming the spoken word occupies the time interval  $[s, t]$ , we allow any words falling into the time interval  $[s - \tau/2, t + \tau/2]$  to be counted in the error calculation. We use a constraint  $\tau = 5$  for the word

**Table 7:** Word Error Rate (WER) and Sentence Error Rate (SER) for WSJ0-eval (Nov-92) using the MCE method and the W-MCE method

	MCE		W-MCE	
Training level	WER	SER	WER	SER
Baseline	8.41	57.88	8.41	57.88
Word-level	8.16	56.97	8.11	56.80
Phone-level	7.96	56.67	7.73	56.67
State-level	8.11	56.97	8.05	56.80

boundary. The corresponding limit for the phone level and the state level are 3 and 0, respectively. The word boundaries are read from the word graph files, and the phone boundaries and state boundaries are set using Viterbi alignment.

In both methods, the phone-level training achieved a better performance than the word level and state level training. One reason is that the time interval for state level training when calculating the GCM may be too short, and the time interval for word level may be too long. Short intervals could lead to over-optimization. Long intervals contain too many parameters and as such the effect for each parameter is weakened when maximizing the corresponding objective.

## 5.4 Chapter Summary

In this chapter, we applied non-uniform error cost training criteria to speech recognition problems and constructed the weighted MCE method. We discussed two important training scenarios, intra-level training and inter-level training, in speech recognition and propose corresponding formulation alternatives for the weighted MCE method. Additionally, the relationship between the weighted MCE method and the popular MPE/MWE method was discussed.

We presented two sets of experimental results to demonstrate the effectiveness of our approach. In the connected digit experiments on the TIDIGITS database, we investigated the intra-level and inter-level training using pre-defined non-uniform error cost matrices. We also introduced two methods to create a non-uniform error

cost matrix. In the large vocabulary task carried out on the WSJ0 database, we studied the data-defined weighting for inter-level training using a confidence measure for correct decision as the error cost. The experiments illustrated that in both tasks, the weighted MCE method outperformed both the baseline and the conventional MCE method.

In the future, more details in regard to the non-uniform error criteria will be discussed based on more complex tasks. Many critical techniques such as how to select a suitable error cost function for specific user requirements need further exploration.

## CHAPTER VI

# A NEW APPROACH TO SELECTING AND ORGANIZING RECOGNITION HYPOTHESES IN THE MCE METHOD

In this chapter, we report a study on the construction of a new approach to selecting and organizing recognition hypotheses in the MCE training method. We propose a new method of *phone-discriminating minimum classification error (P-MCE)*, which performs MCE training at the sub-string or phone level instead of at the traditional string level. Aiming at minimizing the phone recognition error rate, P-MCE nevertheless takes advantage of the well-known, efficient training routine derived from the conventional string-based MCE, using specially constructed one-best lists selected from phone graphs. Extensive investigations and comparisons were conducted between the P-MCE and other discriminative training methods including maximum mutual information (MMI), minimum phone or word error (MPE/MWE), and two other MCE methods. The P-MCE outperforms most experimental approaches on the standard TIMIT database in terms of continuous phonetic recognition accuracy. P-MCE achieves comparable results with the MPE method, which also aims at reducing phone-level recognition errors.

### ***6.1 Motivation***

As we mentioned in Chapter I, the management of recognition hypotheses for each data segment (i.e., the selection and organization of the target and competing events in List 1) is important in the implementation of the MCE method. Recall that we define “target events” as the labeled identity or the correct recognition decision for a

speech segment while the term “competing events” refers to the possible recognition errors for the underlying speech segment.

In phonetic recognition tasks, there are two possible MCE schemes to determine competing events. First, the conventional string-based MCE using N-best lists treats a whole utterance (i.e. a string) as a training “token”, and there is no need to specify the phone boundaries explicitly. This leads to the improvement of string recognition accuracy with improvement in phone accuracy as a potential indirect benefit. In the second scheme, the competing events can be selected using phone graphs, which contain a richer search space and cover more competing candidates than the conventional N-best lists. In this case, for any specific phone in the transcription, an ideal scheme would be to generate a phone graph and compile the arcs with identical segmentation but different identities as the set of competing events. However, the segmentation in phone graphs is neither reliable nor strongly consistent with that of the labeled phones. One compromised solution would be to relax the segmentation-boundary constraint, which, we have found, often leads to inaccurate phone error calculation.

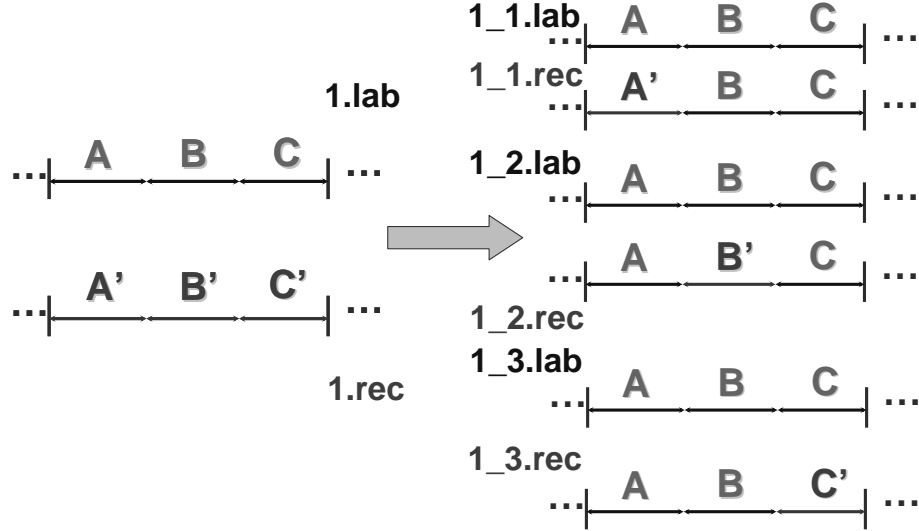
The weaknesses of both schemes above for selecting competing events are overcome by the novel technique of P-MCE introduced in this chapter. While both P-MCE and MPE [66] focus on phone discrimination, P-MCE does not require a number of heuristics used in MPE for defining phone recognition accuracy. In essence, P-MCE selects competing events from phone graphs, forms a large number of one-best lists each containing only a single phone error from the reference, and then carries out the optimization procedure in the same way as in the conventional string-based MCE training. In this way, assignment of phone boundaries is no longer needed, in contrast to MPE which relies on such assignment.

## 6.2 The Phone-Discriminating MCE (P-MCE) Method

We now follow the four-step procedure discussed above to introduce the P-MCE method. Since the performance measure is defined in terms of phone recognition accuracy, we start from the second step.

### 6.2.1 Target and Competing Events

Figure 18 illustrates the principle of our novel competing-token selection in the P-MCE method. Assume the target phone sequence (as the labeled transcription provided by the database) is “ ABC ” and a possible recognized phone sequence is “ A’B’C’ ”. This competing sequence may be the top-one output from the recognizer or be extracted from the phone graph generated by the recognizer. As the figure shows, our goal is to create a competing utterance which differs in *only one phone* from the transcription. That is, for each phone in the transcribed target utterance, we form a new phone sequence which is identical to the target utterance except for that phone. For instance, the competing phone sequence for the first phone “ A ” is “ A’BC ”. Consequently, we can conduct a conventional string-based MCE over “ ABC ” and “ A’BC ”. But since the target and competing sequences differ in only one phone (“ A ” vs. “ A’ ”), the training of model parameters will be focused on the discrimination of phone “ A ” vs. “ A’ ”, reaching the goal of phone-level discrimination. The major advantage for the above P-MCE method is that the statistics used for updating the underlying phone parameters are likely to avoid being contaminated by other models, and that there is no need for estimating the phone boundary explicitly in order to compute the necessary likelihoods. In the current implementation of the P-MCE technique, we only form one competing phone sequence (for each incorrect phone in the original mis-recognized phone sequence) and conduct one-best MCE training. The competing phone “ A’ ” (or “ B’ ” or “ C’ ”) is chosen from a pre-generated phone graph. If there is no suitable competitor found in the phone graph, a competitor is



**Figure 18:** Illustration of creating competing training tokens for the P-MCE method.

selected using a phone confusion matrix generated from the baseline recognizer.

### 6.2.2 Objective Function and Optimization Algorithm

The objective function of the P-MCE is very similar to that of the string-based MCE, which is

$$L_{P-MCE}(\Lambda) = \frac{1}{N'} \sum_{n=1}^{N'} l(d(X_n; \Lambda)) \quad (118)$$

where  $N'$  is the total number of training utterances. Note the number of  $N'$  is no longer the original number of utterances in the training set. According to the procedure described in the preceding subsection, the total number of P-MCE training tokens is expanded from the original  $N$  to  $N' = \sum_{n=1}^N K_n$ , in which  $K_n$  is the number of phones in the  $n^{th}$  utterance. In Eq.(118),  $l(\cdot)$  is the sigmoid loss function and  $d(\cdot)$  is the mis-classification measure, both are defined in the same way as in the conventional MCE training [36]. Naturally, the optimization method can be the GPD method or other gradient descent methods such as Quickprop [59].

We use a modified EBW algorithm proposed in [29] for its stable convergence. EBW is also well suited for a large-scale batch-mode training process essential in large-scale speech recognition. Here, we provide a brief review of the optimization of

the objection function of Eq.(118).

For one-best MCE, we define  $s_n \in \{S_n, s_{n,1}\}$ , where  $S_n$  is the correct label sequence for the  $n^{th}$  utterance and  $s_{n,1}$  is the best competitor (normally the best recognized string but not equal to  $S_n$ ). We can have a misclassification measure function

$$d(X_n|\Lambda) = -\log p(X_n, S_n|\Lambda) + \log p(X_n, s_{n,1}|\Lambda) \quad (119)$$

Note that we use the total likelihood  $\log p(X_n|\Lambda)$  instead of the likelihood of the best state sequence  $\log p(X_n, q|\Lambda)$  in calculating Eq. (119). Consequently, no Viterbi alignment [36] is needed, in contrast to the conventional scheme [36] which requires Viterbi alignment before computing gradient. This eliminates potential problems arising from inaccurate phone segmentations.

We now define

$$\gamma_{m,n,s_n}(t) = p(q_{n,t} = m; \Lambda' | X_n, s_n) \quad (120)$$

as the posterior probability of being in state  $m$  in the corresponding HMM at time  $t$  given utterance  $n$  for word string  $s_n$ . In Eq. (120),  $q_{n,t} = m$  represents that at time  $t$ , model  $n$  stays at state  $m$ .  $\Lambda'$  is the HMM parameter set in the previous iteration, and  $\gamma_{m,n,s_n}(t)$  is computed by the standard forward-backward algorithm [68].

We further define

$$\begin{aligned} \Delta\gamma_{m,n}(t) &= p(S_n; \Lambda' | X_n) p(s_{n,1}; \Lambda' | X_n) \cdot \\ &\quad (\gamma_{m,n,S_n}(t) - \gamma_{m,n,s_{n,1}}(t)) \end{aligned} \quad (121)$$

According to [29], the parameter updating equations are as follows:

$$\mu_m = \frac{\sum_n \sum_t \Delta\gamma_{m,n}(t) x_{n,t} + D_m \mu'_m}{\sum_n \sum_t \Delta\gamma_{m,n}(t) + D_m} \quad (122)$$

$$\begin{aligned}
\Sigma_m &= \frac{1}{\sum_n \sum_t \Delta\gamma_{m,n}(t) + D_m} \times \\
&\quad \left\{ \sum_n \sum_t [\Delta\gamma_{m,n}(t)(x_{n,t} - \mu_m)(x_{n,t} - \mu_m)^T] \right. \\
&\quad \left. + D_m \Sigma'_m + D_m(\mu_m - \mu'_m)(\mu_m - \mu'_m)^T \right\}
\end{aligned} \tag{123}$$

where  $\mu'_m$  and  $\Sigma'_m$  are the HMM parameters from the previous iteration of the algorithm. In Eq. (123),  $D_m$  takes the following functional form:

$$\begin{aligned}
D_m &= E \cdot \sum_{n=1}^{R'} p(S_n|X_n, \Lambda') [p(S_n|X_n, \Lambda') \sum_t \gamma_{m,n,S_n}(t) \\
&\quad + p(s_{n,1}|X_n, \Lambda') \sum_t \gamma_{m,n,s_{n,1}}(t)]
\end{aligned} \tag{124}$$

where  $E$  is set to be 2 in all the experiments.

### 6.2.3 N-best List vs. Graph in Selecting Competing Tokens

One critical issue related to the P-MCE implementation is the selection of competing events using N-best lists versus using phone/word graphs. Conventionally, we adopted the phone graph approach for two reasons. First, conventional “N-best lists” are constructed at the level of competing *utterances*, from which it is difficult to derive the desired phone/word competitors. Second, a graph usually contains a richer search space than an N-best list. One weakness of the phonetic graph approach, however, is the often unreliable arc segmentation in the graph. Another weakness is the possibility of having a sub-string with two or more phones so acoustically cohesive that they should be integrated as a single unit. These problems are particularly severe for phone recognition because the duration of phones is usually shorter than that of words, and hence inaccuracy in phone boundaries will have a direct, negative impact on the quality of the selected competing acoustic events.

For the P-MCE training algorithm, however, all of the above problems are alleviated. The algorithm performs discriminative training on an arbitrary sub-string, as

long as an appropriate competing utterance can be selected. (We showed the selection of phone-level competing “utterances” at the beginning of this section.) For a full exploration of the rich search space afforded by a graph, we can create N-best list from the graph instead of the one-best list as we have currently implemented. With a large value of  $N$  for selecting competing phones or substrings from a phone graph, the richness of the search space in the phone graph would not be curtailed as in the current simplistic implementation. How to optimally organize the competing events in the P-MCE framework is a future research direction.

### **6.3 Experimental Results**

All experiments reported in this section were carried out on the TIMIT database and we used the standard experimental setup as specified in [48, 25]. We experimented with and compared the performance of various discriminative training techniques including the conventional string-based MCE, the P-MCE, the MMI, and the MPE method. The phone graphs in the training are generated using the tool of HVite in the HTK toolkit (<http://htk.eng.cam.ac.uk/>) by setting  $n = 3$  (maximum 3 tokens at one frame)  $p = 0$  (no phone insertion penalty), and  $s = 8$  (language model scale factor).

#### **6.3.1 Baseline system**

A baseline system was built using the HTK, with carefully constructed decision trees to establish triphone HMMs using bigram. A total of 48 monophone units were created following the exact definition of [48, 25]. Triphone HMMs are built by maximum likelihood training using 39 MFCC feature vectors (12MFCC +12 $\Delta$ +12 $\Delta\Delta$ +3 log energy values). All models except for the short pause unit “sp” are 3-state left-to-right HMMs. Each state has 16 Gaussians except for “sil” which contains 28 Gaussians. The short pause model “sp” has only one state with 16 Gaussians. There are a total of 224077 logical triphone models and 7496 physical triphone models, with 917 physical

states after automatic decision-tree tying. Excluding the “sa” utterances in TIMIT, we used a total of 3696 training utterances and 192 core-test utterances according to the standard setup described in [25].

In evaluating the phonetic recognizer, we merge the 48 monophones into 39 monophones according to the standard mapping described in [48, 25] and the confusion among the merged phones is not considered as errors.

### 6.3.2 P-MCE and Other MCE Methods

Table 8 shows the comparison between the conventional string-based MCE, the graph-based MCE [72][24] on phone graphs and P-MCE, both using the EBW optimization method of [29]. The number of training iterations is fixed to be five.

The results in Table 8 show that while the string-based MCE reduces phone recognition errors in the training set (compared with the baseline system), it does not achieve the same for the test set. In contrast, the P-MCE technique outperformed the other two MCE methods in terms of phone recognition accuracy on the test set, although for the training set the improvement is not as much as the string-based MCE.

**Table 8:** Phoneme recognition accuracy (%) for the conventional string-based MCE, graph-based MCE and P-MCE

	Train	Test
Baseline	87.60	72.54
String-based MCE	90.03	70.82
Graph-based MCE	89.90	72.80
P-MCE	89.64	73.01

### 6.3.3 MMI and MPE Methods

The MMI and MPE methods were implemented by the newest HTK3.4 toolkit. The I-smoothed MMI configuration is used for the MMI training, which sets the parameter ISMOOTHTAU=100. The recommended “approximate-error” MPE training

(MPE=TRUE, CALCASERROR=TRUE, INSCORRECTNESS=-0.9) is used for the MPE implementation. To make sure MMI and MPE work correctly, we followed the exact procedures on the HTK tutorial and recorded the values of their objective functions over each training iterations. These values are shown in Table 9 as a function of the training iteration. Consistent increases of the objective functions suggest that the parameters of the algorithms have been set properly.

**Table 9:** Values of the objective functions of the MMI and MPE methods

	iter 1	iter2	iter 3	iter 4	iter 5
MMI	0.815	0.923	0.930	0.949	0.955
MPE	0.882	0.916	0.918	0.930	0.941

Table 10 shows the phonetic accuracy results on the core test set after five iterations of MMI and MPE training. The performance of the P-MCE method on the test set is better than the performance of the MMI method but slightly worse than the one of the MPE method. We examined the earlier work on MMI training [38] for the same TIMIT phonetic recognition task. With a much lower baseline performance of phone accuracy of 66.07%, MMI training only improved the accuracy to 67.50%. Based on the trend of the result figures in [38] and extrapolating them to our comparable higher baseline accuracy of 72.54%, very limited improvement from MMI training would be obtained. So our results reported in Table 10 appear to be consistent with those in [38].

**Table 10:** Phoneme recognition accuracy for the MMI method, the MPE method, and the P-MCE method

	Train	Test
Baseline	87.60	72.54
MMI	89.48	72.85
MPE	89.12	73.03
P-MCE	89.64	73.01

## 6.4 Chapter Summary

We introduced a novel MCE training method, P-MCE, which aims at phone-level discriminative training based on the MCE criterion defined specifically for minimizing phone recognition errors. P-MCE provides a new scheme for selecting competing tokens for each mis-recognized phone in the training utterances, and it takes advantage of the merits associated with both N-best lists and phone graphs. A modified EBW optimization method was described for optimizing the P-MCE objective function.

Compared with another popular phone-level discriminative training method, MPE, P-MCE does not require the same heuristics used in MPE in defining phone recognition accuracy. The objective function of P-MCE is more directly related to the ultimate goal of optimal phone recognition accuracy than MPE. While MPE has been successful for continuous speech recognition where words are recognized as the units, experiments reported in this chapter show that MPE can only achieve limited success on the TIMIT database for continuous phone recognition. We have analyzed several possible reasons for this, among which is MPE's heuristic definition of phone recognition accuracy (from graphs) which is particularly sensitive to phone alignment errors for short-duration units such as phones.

The conventional wisdom suggests that the MPE criterion is superior to the string-based MCE method as it localizes the training errors more accurately. The P-MCE method allows only one error each utterance, which in fact converts the traditional string-based MCE to a phone-based MCE. Though in our experiments P-MCE shows better performance than MMI and two other MCE methods, while being about the same as the MPE method, the improvement over the (relatively high quality) baseline is lower than expected.

Our future work will involve extending the current one-best selection of competing tokens (from the speech event graph) to the richer N-best one within the same

P-MCE framework described here. Another obvious improvement of P-MCE will involve training of selective substrings instead of uniformly single phones as reported. Finally, we plan to incorporate discriminative margins into P-MCE training. This incorporation can be more easily formulated in the P-MCE framework than in other frameworks such as MPE and MMI.

## CHAPTER VII

# DETECTION-BASED SPEECH RECOGNITION AND THE MINIMUM VERIFICATION ERROR (MVE) METHOD

In this chapter, the detection-based automatic speech recognition (detection-based ASR) [47] is introduced, which differentiates itself from the conventional ASR paradigm by conducting bottom-up information fusion through speech attribute detections. To build a set of reliable detectors, we evaluate the performance of the phoneme detectors trained by the minimum verification error (MVE) method [45][71][19], which is a special version of the MCE method for detection/verification problems. We also investigate the performance of rescoreing the conventional decoding candidates using MVE-trained detectors, an intermediate step between the conventional ASR paradigm and a pure detection-based ASR system. Many experiments are carried out based on different training and rescoreing scenarios.

### ***7.1 Detection-Based ASR and Minimum Verification Error (MVE) Method***

#### **7.1.1 Introduction of Detection-Based ASR**

Conventional automatic speech recognition (ASR) techniques are based on the concept of pattern matching, whether it is explicit as in template-matching or implicit as in statistical pattern recognition. These techniques are in general task specific with a fixed system construct (vocabulary, grammar, etc.) which does not permit alteration to adapt to new application environments without completely re-designing the entire system. Performance degradation due to mis-matched design (e.g., the presence of

out-of-vocabulary words or out-of-grammar sentences, different training and test conditions, different background noise or microphones) is often so severe that it renders the system inoperable.

Detection-based ASR was proposed in [47] as an alternative approach, which provides a bottom-up framework based on hypothesis testing supported by the Neyman-Pearson lemma. The biggest advantage of the detection-based ASR is its flexibility to incorporate different knowledge sources and the ability to fuse lower level information into higher level hypotheses. Promising preliminary results of detection-based ASR have been reported (e.g., [40]).

In continuous speech recognition, the recognition errors can be classified into three types in terms of different positions in the alignment between the recognized string and the transcription. They are deletion errors, insertion errors, and substitution errors. These three kinds of errors are named from the point view of **recognition** problems. The MCE/W-MCE method would only minimize the substitution errors because it is hard to present a natural solution for directly minimizing the deletion/insertion errors the framework of the recognition problem. However, if we re-interpret the ASR problem as a **detection** problem, the deletion/insertion/substitution errors can be respectively viewed as miss (type I) errors, false alarm (type II) errors, and dual (both miss and false-alarm) errors. Now we can directly minimize all errors under the framework of the detection theory. This is one of the essence of detection-based ASR [47]. Table 11 shows the comparison between these two paradigms.

The fundamental idea of detection-based ASR is to detect the presence of linguistically relevant information, of various kinds, in a speech signal at a given time and then integrate them to form a higher-level hypothesis. Therefore, an appropriate design of linguistic event detectors is essential in this new approach. Using the methodology of the MCE method, the minimum verification error (MVE) method is a suitable candidate to boost the detector's reliability by directly minimizing the

**Table 11:** Comparison between different problem descriptions in speech recognition

	Error type	Alignment error	Training criterion	Application
Recognition problem	Misclassification errors	Substitution Deletion Insertion	Minimize sub-errors only	Conventional ASR
Detection problem	Type I/II errors	Type I&II Type I Type II	Minimize Type I/II errors	Detection-based ASR

total detection/verification errors.

### 7.1.2 Minimum Verification Error (MVE) Method

#### 7.1.2.1 Motivations

Conventional hypothesis testing is based on the Neyman-Pearson lemma which teaches the use of likelihood ratio to accept or reject a proposed hypothesis. A generalized likelihood ratio is usually computed when test data  $\mathbf{X}$  is observed, and then compared against a decision threshold to decide which one of two hypotheses is to be accepted. The two hypotheses are the *null* hypothesis  $H_0$  with parameters  $\Lambda_0$  and the *alternative* hypothesis  $H_1$  with parameters  $\Lambda_1$ .  $H_1$  is accepted (i.e., an event is detected) when

$$\text{Event happens (accept } H_1) \text{ iff. } \frac{f(X; \Lambda_0)}{f(X; \Lambda_1)} < \theta \quad (125)$$

In order to conduct the test, one needs knowledge about the two probabilistic models, which are conventionally obtained through distribution estimation using pre-labeled data of sufficient amount. As argued in [36], the approach of distribution estimation for constructing hypothesis test is flawed, particularly when the forms of the “true” distributions of the data are unknown. A mis-matched form of distribution will not result in accurate estimate of the model and thus does not lead to any optimal performance as one may predict following the analysis of Neyman-Pearson (or Bayes for the problem of recognition).

Discriminative training methods such as minimum verification error (MVE) training are data-driven approaches that aim at determining optimal boundaries to minimize an empirical estimate of the test error. They have been extensively applied to event verification applications, such as speaker verification. In this chapter, the minimum verification error (MVE) training method is employed in designing the detectors to achieve an optimal performance in terms of composite test error, which is a combination of the type I error (miss) and the type II error (false alarm).

### 7.1.2.2 The MVE Framework

Minimum classification error (MCE) training [36] is a popular discriminative training method which aims at directly minimizing the training errors. To take advantage of the MCE method in detection problems, a special version of MCE called minimum verification error (MVE) training [71] is introduced. In this section, we will review the theoretical framework of MVE.

Analogous to MCE, the essence of MVE is to directly minimize the total detection errors. In detection problems, there are two different kinds of errors: type I error (missing) and type II error (false alarm). Viewed from a classification problem perspective, there are two misclassification measures respectively. Assume there are  $M$  classes and  $N$  training tokens in the training set. For any training token labeled in the  $j^{\text{th}}$  class, a type I error (missing) may result when applied to the detector of the  $j^{\text{th}}$  class, and possibly  $M - 1$  type II errors (false alarm) when applied it to detectors for all the other classes. The type I misclassification measure for an incoming training token  $X$  labeled in the  $j^{\text{th}}$  class can be formulated as

$$d_I = -g_t^j(X; \Lambda_t^j) + g_a^j(X; \Lambda_a^j) + \theta_j \quad (126)$$

where  $g_t = \frac{1}{T}LR_t(X; \Lambda_t^j)$  is the normalized log likelihood of the target model for the  $j^{\text{th}}$  class.  $T$  is the number of frames in the incoming token.  $g_a = \frac{1}{T}LR_a(X; \Lambda_a^j)$  is the normalized log likelihood of the anti-model for the  $j^{\text{th}}$  class.  $\Lambda_t$  and  $\Lambda_a$  are respectively

parameter sets of the target and the anti models.  $\theta_j$  is the decision threshold for class  $j$ .

At the same time, the type II misclassification measure of the  $i^{th}$  class for an incoming training token  $X$  labeled in the  $j^{th}$  class is

$$d_{II}^i(X; \Lambda^i) = +g_t^i(X; \Lambda_t^i) - g_a^i(X; \Lambda_a^i) + \theta_i \quad (127)$$

$$i = 1, 2, \dots, M, i \neq j$$

The two misclassification measures can be embedded into smoothed loss functions written as

$$l_I^j(d_I^j) = \frac{1}{1 + \exp\{-\gamma_j d_I^j\}} \quad (128)$$

and

$$l_{II}^i(d_{II}^i) = \frac{1}{1 + \exp\{-\gamma_i d_{II}^i\}} \quad (129)$$

$$i = 1, 2, \dots, M, i \neq j$$

Finally, the empirical loss for a training set  $\Omega = \{X_1, X_2, \dots, X_N\}$  is given by

$$L_{total}(\tilde{\Lambda}) = \frac{1}{N} \sum_{n=1}^N \sum_{j=1}^M l_{total}^j(X_n; \Lambda^j) 1(X_n \in class j) \quad (130)$$

where the parameter set  $\tilde{\Lambda}$  is defined by  $\tilde{\Lambda} = \{\Lambda_t^j, \Lambda_a^j\}, j = 1, 2, \dots, M$ . The composite error estimation function  $l_{total}^j(X_n; \Lambda^j)$  is a combination of type I and type II errors.

$$l_{total}^j(X_n; \Lambda^j) = PE_I l_I^j(X_n; \Lambda^j) + PE_{II} \sum_{i=1, i \neq j}^M l_{II}^i(X_n; \Lambda^i) \quad (131)$$

$PE_I$  and  $PE_{II}$  are penalty weights for type I and type II errors which can be task-specifically adjusted to bias the preference between these two errors. In equation (130), it can be seen that the total verification errors (weighted sum of type I and type II errors) are represented by a continuous function of the model parameters and decision thresholds. Therefore, the minimization of  $L_{total}$  can be done through the generalized probabilistic descent (GPD) method [36] with respect to all parameters.

Unlike [71], which treats  $\theta_j$  as part of the verification parameter set, we consider  $\theta_j$  to be a “decision policy” which in any discriminative training framework has to be fixed for the performance evaluation to be meaningful. This is because if the policy (i.e., the threshold here) is floating, the significance of data upon the decision boundary can not be correctly and consistently evaluated and the optimization result may not converge. In the following experiments, we have adopted a fixed policy implementation. Note that one can still investigate the ROC curve and determine the operating point (i.e., the threshold) to satisfy the requirements on Type I and Type II errors in a postmortem fashion. Or, one can fix thresholds in (2) and (3) before training.

### 7.1.3 Phoneme Detector Training

We introduce our investigation of segment-based detectors working upon different phoneme categories. Three categories were studied: 1. a 6-class category based on articulatory manner (vowels, fricatives, stops, nasals, silence and others) [68]; 2. a 14-class category based on the broad phonetic definition from [13]; 3. a 48-class category based on monophones defined in [48].

#### 7.1.3.1 System Description

The detectors are based on representing training classes using continuous density Gaussian mixture hidden Markov models (CDHMM). The training units of the first two detector arrays are concluded from knowledge in regard to articulatory manner and place. The third detector array uses 48 context-independent (CI) monophones defined in [48]. Each detector consists of a target model and an anti-model. All models are represented by 3-state strict left-to-right HMMs, with 16 Gaussian mixture components per state. These models are trained first by Maximum Likelihood (ML) method implemented by the HTK toolkit then by Minimum Verification Error (MVE) method. The experiments were carried out by comparing the system trained using

MVE to the baseline system trained by ML. In all experiments we assume equal loss weights of type I and II errors.

The experiments were conducted on the TIMIT database. Starting from the 48 CI phones from [48], we first generated feature vectors for all 4,620 utterances in the training set of the TIMIT database. Each feature vector has  $12\text{MFCC}+12\Delta+12\Delta^2$  and 3 energy values so that total 39 features are used. Then the long feature vectors for each utterances are decomposed into short feature vectors for phonemes according to the transcript. The feature generation process is also applied on the test set with 1,680 utterances.

We mapped 48 monophones into 6 and 14 classes respectively using mapping rules illustrated in Table 12 and 13. We did not emerge phones “cl”, “vel” and “epi” into “sil”. In the first two sets of detectors (6-class and 14-class), training tokens belonging to these three phones were simply ignored. Therefore, in the first two experiments we have 119,580 training tokens and 42,657 test tokens. For the last one, we have 140,225 training tokens and 49,762 test tokens. The prior percentage of each class in the test set is also provided in the last column of Table 12 and 13.

**Table 12:** Mapping rule for the 6-class category.

6-class	monophones	percentage(%) in the test set
fricatives	ch dh f jh s sh th v z zh	16.88
vowels	aa ae ah ao aw ax ay eh er ey ih ix iy ow oy uh uw	39.62
nasals	en m n ng	10.32
stops	b d g k p t	13.06
others	dx el hh l r w y	12.95
silence	sil	7.14

*7.1.3.2 Experiment Results*

Before providing our results, we would like to recall the objective of these experiments. The aim of these evaluations is not to report optimal detector performance.

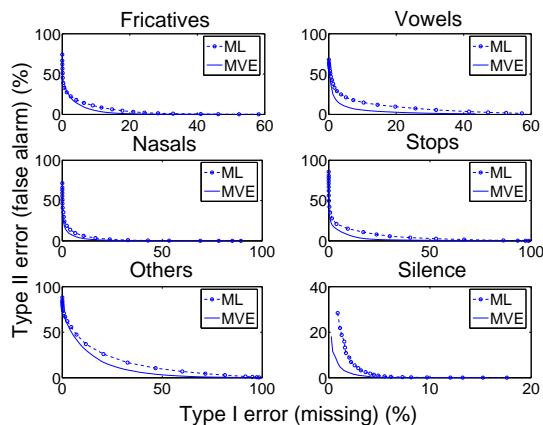
**Table 13:** Mapping rule for the 14-class category.

14-class	monophones	percentage(%) in the test set
front vowels	ae eh ey ih ix iy	20.06
mid vowels	ah ax er	9.33
back vowels	aa ao ow uh uw	7.02
diphthongs	aw ay oy	2.41
voiced fricatives	dh v z	6.50
unvoiced fricatives	f th s sh zh	9.08
affricates	ch jh	1.30
voiced consonants	b d g	3.73
unvoiced consonants	k p t	7.91
nasals	en m n ng	10.32
liquids	dx el l r	10.91
glides	w y	2.98
whispers	hh	1.31
silence	sil	7.14

Although, theoretically, our detectors can be optimal in terms of minimizing training detection errors, we only use spectral knowledge (in this section, MFCC) as diagnosis information. Furthermore, parameter tuning is far from mature. The generic motivation is to demonstrate the great potential of knowledge integration under this detector-based structure using MVE. In the future, more different features from various speech attributes (e.g., landmarks, articulatory knowledge) would be introduced to improve the performance of current ASR systems.

Figure 19 shows the ROC curves of all 6 detectors. Table 14 shows the minimum total error rate. Note that the minimum total error rates do not necessarily locate at the equal-error rate point. They are simply the minimum error rates based on the exhausted search by shifting the thresholds when applying the test tokens onto detectors. Hence, these error rates can be seen as lower bounds or the “best performance” of detectors.

From Fig. 19 and Table 14, it is sufficiently supported that MVE outperforms ML in all detectors. The average error rates weighted by the prior phonetic distribution



**Figure 19:** ROC curves for the 6-class category.

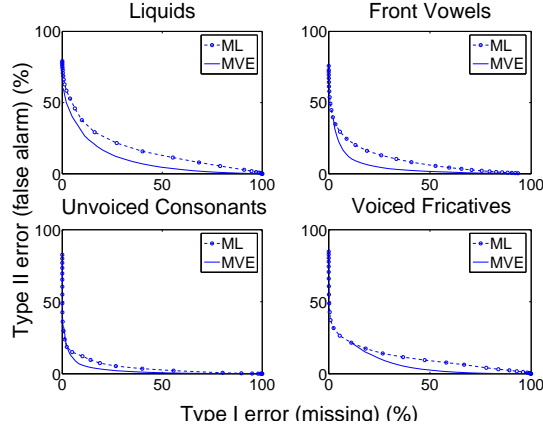
**Table 14:** Minimum total error rate (%) for the 6-class category based on grid search on threshold

6-class	ML	MVE
fricatives	5.89	3.91
vowels	12.94	7.11
nasals	4.14	2.97
stops	8.77	4.60
others	13.04	9.19
silence	0.75	0.53
<b>WEIGHTED AVERAGE</b>	<b>9.44</b>	<b>5.61</b>

in the test set are 9.44% and 5.61% for ML and MVE, respectively.

We select four classes of total 14 classes to show the performance evaluation in term of ROC curves. They are liquids, front vowels, unvoiced consonants, and voiced fricatives. Figure 20 shows their ROC curves. Table 15 shows the minimum total error rates of all 14 classes. We can observe that the weighted average error rate drops from 7.61% to 5.65%.

Table 16 displays the error rate reduction of the 48-class category alphabetically. The prior percentage of each phoneme in the test set is displayed in the parenthesis in the first column. The weighted average values of the minimum error rates for ML and MVE are 2.65% and 2.30%, respectively.



**Figure 20:** ROC curves for 4 representative classes from the 14-class category.

## 7.2 Rescoring Using MVE-Trained Detectors

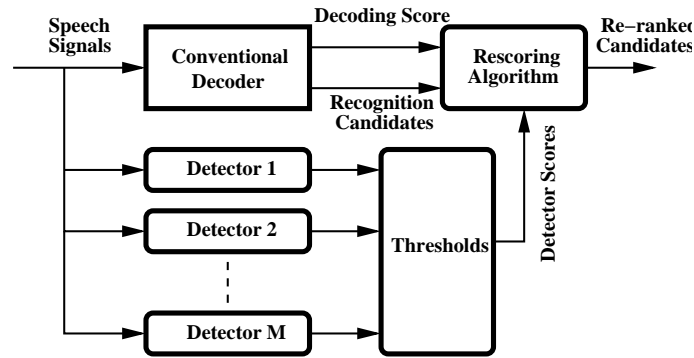
We have witnessed many related research in regard to the detector design methodology and information integration approaches [54][20][51][31]. We are reminded by the previous exploration that before building a real and complete detection-based system, it is helpful to incrementally investigate the effect of combining detectors and conventional decoders. Thus a rescoring system, a hybrid of attribute detectors and a conventional decoder, is the research objective here. In [51], a frame-based detector was introduced and "knowledge-based" front-end features are utilized to accomplish enhanced recognition accuracy. A segment-based rescoring system was reported in [20], showing preliminary improvements; it exploits a set of HMM-based detectors to help a conventional recognizer in reaching the final decision. Two or more relatively independent "inference" measures were integrated in the same observation space.

According to Fig. 21, the performance of a rescoring system is decided by two key factors: the reliability of the detectors and the effectiveness of the rescoring algorithms under different scenarios. In this section, a systematic investigation upon these two key issues has been organized.

First, the MVE method is a well-suited approach to enhancing the reliability

**Table 15:** Minimum total error rate (%) for the 14-class category

14-class	ML	MVE
front vowels	14.70	8.71
mid vowels	9.33	8.97
back vowels	7.02	6.50
diphthongs	2.41	2.28
voiced fricatives	6.30	5.09
unvoiced fricatives	4.44	3.46
affricates	1.30	1.17
voice consonants	3.72	3.49
unvoiced consonants	6.37	4.07
nasals	4.11	3.02
liquids	10.91	9.16
glides	2.88	2.78
whispers	1.15	0.99
silence	0.72	0.53
<b>WEIGHTED AVERAGE</b>	7.61	5.65



**Figure 21:** The rescoreing diagram using HMM-based detectors.

of the detectors, some effective discriminative training criteria are employed. However, the original MVE training is designed for using isolated speech segments hence not consistent to be combined with most of the rescoreing algorithms for continuous speech recognition tasks. To alleviate the mismatch between the detector training and rescoreing scenario, we propose two modified versions of the MVE training method. Second, we examine various rescoreing algorithms under multiple rescoreing scenarios. We investigate two types of rescoreing algorithms in terms of their information fusion strategy – the scoring fusion rescoreing and the decision fusion rescoreing. For the

**Table 16:** Minimum total error rate (%) for the 48-class category

name(%)	ML	MVE	name(%)	ML	MVE
aa(1.70)	1.70	1.70	ae(1.55)	1.55	1.50
ah(1.72)	1.72	1.72	ao(1.53)	1.53	1.50
aw(0.43)	0.43	0.43	ax(2.87)	2.87	2.78
ay(1.38)	1.34	1.18	b(1.09)	1.09	0.95
ch(0.52)	0.52	0.50	cl(8.55)	6.03	3.83
d(1.30)	1.30	1.29	dh(1.66)	1.66	1.60
dx(1.24)	1.24	1.05	eh(2.51)	2.51	2.51
el(0.69)	0.69	0.69	en(0.43)	0.43	0.43
epi(0.66)	0.65	0.54	er(3.40)	2.82	2.79
ey(1.61)	1.54	1.51	f(1.83)	1.18	0.85
g(0.80)	0.80	0.76	hh(1.13)	1.01	0.86
ih(2.89)	2.89	2.89	ix(5.00)	5.00	4.68
iy(3.64)	2.93	2.45	jh(0.59)	0.59	0.59
k(2.35)	2.26	1.85	l(3.73)	3.73	3.14
m(2.80)	2.47	2.46	n(4.86)	4.56	3.55
ng(0.75)	0.75	0.75	ow(1.21)	1.21	1.20
oy(0.26)	0.24	0.24	p(1.79)	1.71	1.32
r(3.69)	3.69	3.12	s(4.36)	2.57	2.13
sh(0.92)	0.61	0.51	sil(6.12)	1.13	0.75
t(2.64)	2.58	2.03	th(0.52)	0.52	0.52
uh(0.43)	0.43	0.43	uw(1.15)	1.14	1.14
v(1.42)	1.40	1.20	vcl(5.07)	4.59	3.57
w(1.80)	1.73	1.53	y(0.75)	0.75	0.74
z(2.48)	2.48	2.01	zh(0.15)	0.15	0.14
<b>WEIGHTED AVERAGE</b>	2.65	2.29			

first type of algorithm, we combined the scores from different information sources and make the final rescoreing decision based on those scores. For the second one, the independent decisions are made before being transformed into the final result. The study of the rescoreing algorithms is conducted on both the intra-linguistic level rescoreing (e.g., the rescoreing is made on the phone level and the objective of rescoreing is to improve phone recognition accuracy) and the inter-linguistic level rescoreing (e.g., the rescoreing is made on the phone level and the objective of rescoreing is to improve word recognition accuracy) to find the appropriate rescoreing configurations under different scenarios. Note that we are not claiming any “optimal” system. The

objective is to justify suitable combinations of the detector design and the rescoring algorithms for future tasks. Further more, it is a helpful intermediate step towards pure detection-based ASR applications.

### 7.2.1 Variants of the MVE Training

In most ASR tasks, the MVE training routine is applied to the phone level using isolated speech tokens which are usually not segmented as consistent and accurate as expected. Furthermore, the decoded candidates are generated in a fashion of continuous recognition that may cause a mismatch between the detector construction and rescoring conditions. Therefore, in this section, we propose two modifications of the MVE method that are more suitable in the context of continuous ASR but still inherit the merits from the original MVE criterion.

#### 7.2.1.1 Substring MVE Training

The first modification of the MVE training is named *substring MVE training (S-MVE)*. This method concatenates the target and anti-target models of contiguous phones respectively to form a set of substring detectors which may contain arbitrary number of phones. The MVE training is conducted from the start and shifts all along the utterance. For example, if the utterance is “sil sh iy hh aa s”, the first substring training could be applied to the detector model “sil+sh+iy” and the second one could be applied to the model “sh+iy+hh”, etc. This method inherits the discriminative ability of the MVE criterion and avoids setting the phone boundaries inside the substring explicitly. The other advantage of this modification is that it exploits some context dependency.

#### 7.2.1.2 Relaxed-Boundary MVE Training

Though the substring MVE method could alleviate the effects of the unreliable phone boundaries, the start and end time of each substring are still subject to errors. Thus,

we develop the second modification of the MVE criterion, the *relaxed-boundary MVE training (RB-MVE)*. This is a more advanced modification of the original MVE criterion based on the S-MVE method. The essence of the RB-MVE method is that the phone boundaries are re-defined by the detector models. We form the utterance target model and anti-model as the concatenation of all phone models in the utterance. For each state  $j$  in the sequence of the phone models, the forward and backward likelihood ratio vector  $\alpha_t^j$  and  $\beta_t^j$  can be computed for each frame  $t$ . Assume there are  $N$  states in each phone model, we then can determine the “best” segment  $[t_s, t_e]$  for each phone in terms of the highest forward likelihood ratio represented by  $\alpha_{t_e}^N - \alpha_{t_s}^1$ . The S-MVE method is then carried out based on the adjusted segments. It is a data-driven procedure to set up better phone boundaries based on the best detector models we have. The training and boundary determination can be repeated iteratively until satisfaction is achieved.

## 7.2.2 Rescoring Algorithms

In this section, we investigate two types of rescoring algorithms: the score-fusion algorithm and the decision-fusion algorithm. Three algorithms are proposed respectively for each type.

### 7.2.2.1 Score-Fusion Algorithms

Score fusion is a technique that combines the detectors scores and the decoder scores. The decoding candidates are re-ranked based on the new scores. In this section, we review three score-fusion methods proposed in [20]. Suppose there are  $M$  corresponding detectors such that each of them consists of a target model and an anti-model. For a speech segment that is decoded as the  $j^{th}$  class with log likelihood  $S_{decode}^{(j)}$ , its  $i^{th}$  ( $i = 1, 2, \dots, M$ ) detector scores are  $S_{tgt}^{(i)}$  and  $S_{anti}^{(i)}$ , respectively. Namely, the likelihood ratio for the  $i^{th}$  detector is  $ratio^{(i)} = S_{tgt}^{(i)} - S_{anti}^{(i)}$ . We call the score for the test segment belonging class  $j$  after combination  $S_{new}^{(j)}$ .

The first method is called *Naive-Adding (NA)*. From its name we can know that it is a quite naive score combination scheme. In this approach, the new score of each segment being decoded as the  $i^{th}$  class is

$$S_{new}^{(j)} = S_{decode}^{(j)} - S_{anti}^{(j)} + ratio^{(j)} \quad (132)$$

The reason for subtracting  $S_{anti}^{(j)}$  is to scale the decoding score into a relatively close dynamic range with the likelihood ratio. This procedure is also taken in the following two methods.

The second method is named *Competitive-Rescoring (CR)*. In this approach, we define a new “competitive” score  $S_c^{(j)}$ .

$$S_c^{(j)} = ratio^{(j)} - \log \left\{ \frac{1}{M-1} \sum_{i \neq j}^M \exp(\eta \cdot ratio^{(i)}) \right\}^{1/\eta} \quad (133)$$

and

$$S_{new}^{(j)} = S_{decode}^{(j)} - S_{anti}^{(j)} + S_c^{(j)} \quad (134)$$

In the first method only the likelihood ratio from the underlying class of detectors are used for rescoring. But in this case, we first compute a distance measure between the claimed class to a geometric average of the other competitive classes. This quantity  $S_c^{(j)}$  is similar to the “misclassification measure” function  $d$  in MCE training [36] but using the corresponding detectors’ likelihood ratio and there is a sign difference.

The third method is called *Remodeled Confidence Measure (RCM)*. Borrowing from the idea of the recognition phone graph, we formed a pseudo-graph for each phoneme segment using detector arrays. We can consider that the detection results of the total  $M$  detectors are  $M$  extra paths for the test speech segment. A remodeled posterior probability of the claimed class  $j$  is defined as the ratio of two scores. The score on the numerator is the scaled decoding score of claimed class  $j$  plus the likelihood ratio of the detector for class  $j$ . The score on the denominator is the sum

of the numerator score and all the other detection scores. i.e,

$$S_{new}^{(j)} = \log \left[ \frac{\exp(S_{decode}^{(j)} - S_{anti}^{(j)}) + \exp(ratio^{(j)})}{\exp(S_{decode}^{(j)} - S_{anti}^{(j)}) + \sum_{i=1}^M \exp(ratio^{(i)})} \right] \quad (135)$$

#### 7.2.2.2 Decision-Fusion Algorithms

In many rescoreing tasks, the detector design is on a different linguistic level compared to the recognized candidates, thus inter-linguistic level rescoreing is required. The score-fusion mechanisms such as the RCM method may only gain incremental impact in inter-linguistic level rescoreing because they do not affect the rescoreing decisions directly. One alternative rescoreing method is to fuse the independent decisions from both the decoder and the detectors to prune the recognized candidates. In this section, three decision-fusion methods are proposed and compared under the cross-linguistic level rescoreing scenario in which phone-level detectors are employed to improve the word accuracy.

To apply phone-level detectors upon the word graph, the decision-fusion methods prune the candidates in word graphs based on the reliability of the phone sequence in each word. In other words, each phone in every decoded word is examined by the corresponding detectors. We define a “miss” error in this situation if a recognized phone belongs to the  $i^{th}$  class but the likelihood ratio of the corresponding  $i^{th}$  detector is less than the threshold. Similarly, a “false alarm” error occurs when a recognized phone belongs to the  $i^{th}$  class but the likelihood ratio of any detectors other than the  $i^{th}$  detector is larger than the threshold.

The first method, the *strict-pruning (SP)* method prunes the whole word if any phone in the word is detected as a “miss” error. This method maps the phone errors and the word errors directly in a strict one-by-one manner.

The second method prunes the word only if at least two or over half of the phones are detected as “miss” errors. This method relaxes some constraints compared to the

first method (SP), but still concentrates on the “miss” errors. We name it *Relaxed-Pruning I (RP-I)*.

The third method is similar to the second one except for an additional provision in which the pruning shall not occur unless there exist “false alarm” errors at the same time. We call this method *Relaxed-Pruning II (RP-II)*.

### 7.2.3 Experimental Results

The experiments are conducted on the TIMIT database. The training set has 3,696 utterances and the test set has 1,344 utterances (the utterances for speaker adaptation are ignored). The acoustic model of the baseline decoder consists of 41 CI phones that are folded from the 48-monophone set defined in [48]. The phones “vcl cl epi” are folded into “sil”. The phones “ix el em en” are folded into “ih l m n” respectively and there is no phone labeled as “dx”. Each phone is modeled by a 3-state HMM. The decoder uses 32 mixtures for each state in the intra-linguistic level rescoring and 70 mixtures in the inter-linguistic level rescoring. The model parameters are trained by embedded Baum-Welch algorithm [68] using 39 dimensional feature vectors with 12MFCC, 12 $\Delta$ , 12 $\Delta^2$  and 3 log energy values. The recognition candidates are organized using phone/word graphs rather than N-best lists because phone/word graphs represent more information in a much compact topology than N-best lists. The phone/word graphs are generated using HVite in the HTK toolbox (<http://htk.eng.cam.ac.uk/>) in way that the pruning criterion is set such that only 3 recognition candidates can survive simultaneously. In one graph, each node represents a time instance and each arc represents a phone/word.

Three taxonomical phonetic category detectors are defined and trained first by the Baum-Welch algorithm [68] then adjusted by the variations of the MVE method. These categories include 6 classes [68], 14 classes [13], and 41 classes phonemes respectively. Tables 12 and 13 show the mapping rules from the 41-class phone set to the

6-class and 14-class set, respectively. The target models and anti-models in detectors are constructed using 3-state HMM with 32 Gaussian mixtures in each state. In our experiments, we employ these three detectors separately to conduct the cross-category rescoring for the 41-class phone/word graphs in each scenario.

Based on the decoder and the detectors described above, we conduct extensive investigations on rescoring performance under different detector building and information fusion scenarios. We examine two prevalent rescoring situations – intra-linguistic level rescoring and inter-linguistic level rescoring. The experiments for intra-linguistic level rescoring are organized using the phone-graph rescoring to enhance the phone recognition accuracy. Comparative results are organized to show the performance difference between various combinations of different detector training methods and score-fusion algorithms. On the other hand, the inter-linguistic level rescoring experiments are presented using the phone-level information integration on word-graphs in order to boost the word recognition accuracy. We concentrate on the result comparison between the decision-fusion algorithms in this case.

### *7.2.3.1 Intra-Linguistic Level Rescoring Using Phone Graphs*

The system performance reaches its upper bound when selecting the candidate from the phone graph which best matches the reference phone transcription. To evaluate a rescoring algorithm, the relative accuracy improvement is defined by the ratio of the absolute improvement over the offset between the upper bound accuracy and the baseline accuracy. In this section, we only focus on the score-fusion methods. Table 17 shows phone recognition accuracy of the baseline decoder and the upper bound of the phone graph using 0-gram and bigram, respectively.

We first compare the results between the different rescoring algorithms using the conventional MVE training. Then, three variations of MVE training methods are compared using the best rescoring method.

**Table 17:** Baseline phone accuracy and upper bounds for intra-linguistic level rescoring.

Acc(%)	0-gram	bigram
Baseline	56.78	63.93
Upper bound	63.27	70.75

Table 18 displays the performance of all three rescoring approaches by using all three taxonomical phonetic detectors upon phone graphs for cross-category rescoring. In Table 18, the first row of results is the rescored accuracy and the second row contains the relative improvement (%). For all three detectors, with the detectors trained using the conventional MVE method, we tried three rescoring algorithms: the naive-adding (NA) method, the competitive rescoring (CR) method and the remodeled confidence measure (RCM) method. In addition, the rescoring effect under two kinds of different language models, 0-gram and bigram, are respectively investigated in the experiments.

Since the phone graphs are generated over the 41-class phone set, we map each phone back to the 6-class and 14-class phone set and compute detection scores when conducting cross-category rescoring. Based on the experiment results, first, we can see that the naive-adding (NA) method has the least performance boosting and the remodeled confidence measure (RCM) method obtains the most gain. It is not surprising since NA is the most naive approach among the three, while the RCM method tries to find a candidate with maximum value of a remodeled posterior probability, which bears relationship to Bayes risk. Second, the 41-class detector displayed the highest performance in cross-category rescoring. Third, rescoring techniques showed much higher performance improvement when 0-gram is used. The reason of this observation might be that the use of better language model eliminates some errors due to inaccurate acoustic modeling.

On the phone graph, the rescoring results of using all three taxonomical phonetic detectors with different detector training strategies are presented in Table 19. In Table

**Table 18:** Intra-linguistic level rescoring performance for different rescoring algorithms (The first row is the rescored accuracy and the second row is the relative improvement).

	NA		CR		RCM	
	0gram	bigram	0gram	bigram	0gram	bigram
6	57.04 <b>4.01</b>	63.93 <b>0.0</b>	57.29 <b>7.86</b>	63.94 <b>0.15</b>	58.04 <b>19.41</b>	64.05 <b>1.76</b>
14	57.08 <b>4.62</b>	63.93 <b>0.0</b>	57.40 <b>9.55</b>	63.96 <b>0.44</b>	58.01 <b>18.95</b>	64.20 <b>3.96</b>
41	57.62 <b>12.94</b>	63.95 <b>0.29</b>	57.94 <b>17.87</b>	63.99 <b>0.88</b>	58.41 <b>25.12</b>	64.35 <b>6.16</b>

19, the first row of results are the rescored accuracy and the second row contains the relative improvement. For all three detectors, we tried three MVE training methods: the original one, the substring MVE (S-MVE) and the relaxed-boundary MVE (RB-MVE). The rescoring algorithm is the RCM method. In addition, the rescoring effect under two kinds of different language models, zero-gram and bigram, are respectively investigated in the experiments.

From Table 19 we can also make some conclusive observations as in the last section. First, the RB-MVE method and S-MVE method outperform the original MVE method no matter what kind of detector is employed. Second, the 41-class detector displayed the highest performance as expected. Finally, as we observed before, the improvement of rescoring using 0-gram graphs is higher than that of bigram phone graphs.

### 7.2.3.2 Inter-Linguistic Level Rescoring Using Word Graphs

The inter-linguistic level rescoring is conducted by using the phone-level detectors to rescore phones inside each recognized word candidate to improve the word recognition accuracy. In this section, we study the rescoring performance for both the score-fusion and decision-fusion methods. In the score-fusion rescoring part, as the RB-MVE method and RCM method outperformed their competitors in our previous research,

**Table 19:** Intra-linguistic level rescoring performance for different detector training methods (The first row is the rescored accuracy and the second row is the relative improvement).

	MVE		S-MVE		RB-MVE	
	0gram	bigram	0gram	bigram	0gram	bigram
6	58.04 <b>19.41</b>	64.05 <b>1.76</b>	58.10 <b>20.34</b>	64.20 <b>3.96</b>	58.98 <b>33.90</b>	64.25 <b>4.69</b>
14	58.01 <b>18.95</b>	64.20 <b>3.96</b>	58.10 <b>20.34</b>	64.35 <b>6.16</b>	59.01 <b>34.36</b>	64.60 <b>9.82</b>
41	58.41 <b>25.12</b>	64.35 <b>6.16</b>	58.90 <b>32.67</b>	64.40 <b>6.89</b>	59.21 <b>37.44</b>	64.78 <b>12.46</b>

we employ them as the detector training and score combination approach respectively for cross-category rescoring using three taxonomical phonetic detectors. The focus of this section is the decision-fusion rescoring, in which we compare three decision-fusion methods in terms of the final word accuracy using the best configuration obtained from the previous experiments. Table 20 shows phone recognition accuracy of the baseline decoder and the upper bound of the phone graph using bigram.

**Table 20:** Baseline word accuracy and upper bounds with bigram for inter-linguistic level rescoring

Acc(%)	bigram
Baseline	50.28
Upper bound	65.46

We use a similar rescoring method as we did for the intra-linguistic level rescoring. The detectors are trained using the RB-MVE criterion. The RCM method was applied to calculate new scores for each phone in every word in the word graphs. The final rescored word score is the summation of all rescored phone scores in the word. As we did in phone graph rescoring, we mapped each phone back to the 6-class and 14-class phone set and computed detection scores when conducting cross-category rescoring. Table 21 shows the performance of the inter-linguistic level rescoring using the best score-fusion method selected from the intra-linguistic level rescoring experiments. Still, among all cross-category rescoring experiments, the 41-class phonetic detectors

**Table 21:** Inter-linguistic level rescoring performance using the best score-fusion method (RCM) in our experiments

Detector		Rescored Acc(%)
6-class	Rescored	50.82
	Relative	<b>3.56</b>
14-class	Rescored	50.70
	Relative	<b>2.77</b>
41-class	Rescored	51.25
	Relative	<b>6.39</b>

**Table 22:** Inter-linguistic level rescoring performance using decision-fusion methods.

phone class	Acc(%)	SP	RP-I	RP-II
6-class	Rescored	50.20	50.98	51.27
	Relative	<b>-0.53</b>	<b>4.61</b>	<b>6.52</b>
14-class	Rescored	50.01	50.87	51.46
	Relative	<b>-1.78</b>	<b>3.89</b>	<b>7.77</b>
41-class	Rescored	50.24	51.02	52.39
	Relative	<b>-0.26</b>	<b>4.87</b>	<b>13.90</b>

displays the highest improvement of the word accuracy.

Table 22 displays the experimental results of using three taxonomical phonetic detectors with all three decision-fusion methods. As we did in the intra-linguistic level rescoring, we mapped each phone in the 41-class category back to the 6-class and 14-class phone set when making cross-category decisions. We can see that for all types of detectors, the Strict-Pruning method (SP) overprunes and leads to a slight performance drop because of its strict constraint. The Relaxed-Pruning I (RP-I) method shows some positive gains and the Relaxed-pruning II (RP-II) method achieves the best performance in the inter-level experiments. Though there is no substantial improvement as we expected, the decision-fusion methods do show higher performance enhancement than the score-fusion approaches.

### **7.3 Chapter Summary**

In this chapter, we investigated the performance of MVE training in the application of designing different speech attributes detectors. The parameters of target models

and anti-models were estimated according to the criterion of directly minimizing total verification errors. Experiments were conducted on the TIMIT database, suggesting that MVE can more effectively improve detection performance than ML. This is a promising result for using a discriminative training criterion (especially MVE) in future research such as detection-based automatic speech recognition.

We also presented an extensive investigation for rescoring performance on continuous speech recognition tasks. The study was based on the general framework depicted in Fig.21. Two key components of the rescoring system, the attribute detector design and the rescoring algorithms, were examined under the intra-linguistic level rescoring and the inter-linguistic level rescoring, respectively.

For detector design methods, two variations of the MVE training criterion, the S-MVE method and the RB-MVE method, were introduced for continuous speech recognition scenarios. We discovered that the RB-MVE criterion achieves the best result in the performance comparison. We introduced two types of rescoring algorithms, the score-fusion algorithms and the decision-fusion algorithms. The score-fusion algorithms were tested in the intra-linguistic level rescoring, in which the RCM method shows the best performance. The decision-fusion algorithms were examined in the inter-linguistic level rescoring and the RP-II method displayed the best performance over other decision-fusion methods.

## CHAPTER VIII

### CONCLUSION, CONTRIBUTIONS, AND FUTURE WORK

#### *8.1 Conclusion and Contributions*

The minimum classification error (MCE) method is a discriminative training method, which provides rigorous guidance for approaching the optimal classifier/recognizer design based on the Bayes decision theory. In this thesis, we developed an extensive generalization of the MCE framework on a variety of speech recognition and detection problems. Through many mathematical derivations and experimental results, the generalized MCE framework demonstrated theoretical optimality and achieved encouraging results in various applications. The major contributions of this thesis can be concluded as:

- **One framework** - The generalized MCE framework
- **Two topics** - Conventional speech recognition and detection-based speech recognition
- **Three applications** - The non-uniform error cost criterion, a method to selecting and organizing the recognition hypotheses in the MCE method, and the discriminative detector design for the detection-based ASR.

##### **8.1.1 The Non-Uniform Error Criterion**

An optimal classifier design framework with non-uniform error cost functions was proposed in this thesis. Extensive experiments were carried out on automatic pattern and speech recognition problems and justified the effectiveness of this new paradigm.

The contribution for this topic is:

- A theoretical framework for optimal classifier/recognizer design with non-uniform error cost functions - this framework was derived from the Bayes decision theory and overcame the incapability of the distribution estimation methods when facing non-uniform error costs;
- Practical implementation strategies for different training scenarios - extensive discussions for more complicated training situations such as speech recognition tasks were provided, and the optimization equations for model parameters were derived for Gaussian mixture models and hidden Markov models;
- An investigation of the non-uniform error cost functions - a preliminary study was proposed regarding the techniques to select an appropriate error cost function to optimize the user-specified system objective. Two approaches for designing the evaluation function were introduced for general pattern recognition problems.

### **8.1.2 A New Approach to Selecting and Organizing the Recognition Decisions in the MCE Method**

For this topic, we developed a new approach to selecting and organizing the recognition hypotheses in the MCE method. We proposed a new method of *phone-discriminating minimum classification error (P-MCE)*, which performed MCE training at the sub-string or phone level instead of at the traditional string level. Aiming at minimizing the phone recognition error rate, P-MCE nevertheless took advantage of the well-known, efficient training routine derived from the conventional string-based MCE, using specially constructed one-best lists selected from phone graphs. Extensive investigations and comparisons were conducted between the P-MCE and other

discriminative training methods including maximum mutual information (MMI), minimum phone or word error (MPE/MWE), and two other MCE methods. P-MCE outperformed most experimental approaches on the standard TIMIT database in terms of continuous phonetic recognition accuracy. P-MCE achieved comparable results with the MPE method which also aims at reducing phone-level recognition errors.

### **8.1.3 Discriminative Detector Design for Detection-Based Speech Recognition**

The minimum verification error (MVE) method was applied to improve the accuracy of speech attribute detectors in this thesis. We investigated the phoneme detector’s performance and applied these detectors to reduce the errors of conventional speech recognizers. This was an intermediate step from the conventional ASR paradigm to a pure detection-based ASR system.

The major contribution of this topic is:

- The investigation of the phoneme detector performance; the detectors for three taxonomical broad phonetic classes were trained using the MVE method and their performance was studied on the TIMIT database;
- The investigation of the MVE variants and rescoring methods; for continuous speech recognition tasks, we modified the original definition of the MVE method and proposed two variants. In addition, various rescoring methods were compared under different training scenarios.

## **8.2 Future Work**

We plan to explore more in generalizing the MCE framework. First, for the non-uniform error cost criterion, there are still many open problems. One interesting problem is how to set up an “optimal” error cost matrix so that the system objective can be efficiently optimized. The relation between the error cost matrix and

the confusion matrix is another attractive topic. Second, MVE-trained detectors can be applied to more complicated detection-based ASR applications such as conversational telephony speech signals. In fact, spontaneous speech is the best platform to show the advantage of detection-based ASR, which focuses only on spotting “useful” speech attributes rather than aiming at completed transcriptions. Third, as we have mentioned, the MCE framework has four steps. We only partially generalized the first two issues in this thesis, while the last two issues are left to future research. Among the last two issues, the objective construction and optimization algorithms, the idea of combining the “margin” concept from SVM literatures (e.g., [52][53]) into the discriminative learning is popular and promising.

## REFERENCES

- [1] AMARI, S., “A theory of adaptive pattern classifiers,” *IEEE Transaction on Electronic Computers*, vol. EC-16, no. 3, pp. 299–307, 1967.
- [2] BAHL, L. R., BROWN, P. F., DESOUSA, P. V., and MERCER, R. L., “Maximum mutual information estimation of hmm parameters for speech recognition,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, (Tokyo, Japan), Apr. 1986.
- [3] BAUM, L. E., PETRIE, T., SOULES, G., and WEISS, N., “A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains,” *Ann. Math. Stat.*, vol. 41, pp. 164–171, 1970.
- [4] BAZZI, I. and KATABI, D., “Using support vector machines for spoken digit recognition,” in *Proceedings of the International Conference on Speech and Language Processing*, (Beijing, China), pp. 433–436, 2000.
- [5] BLUM, J. R., “Multidimensional stochastic approximation methods,” *Ann. Math. Stat.*, vol. 25, pp. 737–744, Apr. 1954.
- [6] CHAPELLE, O., HAFFNER, P., and VAPNIK, V., “Svms for histogram-based image classification,” *IEEE Transaction on Neural Networks*, vol. 10, pp. 1055–1064, 1999.
- [7] CHOU, W., “Discriminant-function-based minimum recognition error rate pattern-recognition approach to speech recognition,” *Proceedings of the IEEE*, vol. 88, no. 8, pp. 1201–1223, 2000.
- [8] CHOU, W. and JUANG, B. H., “Adaptive discriminative learning in pattern recognition,” *Tech. Rep., AT&T Bell Labs*.
- [9] CHOU, W., JUANG, B. H., and LEE, C. H., “Segmental gpd training of hmm-based speech recognizer,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, (San Francisco, CA), pp. 473–476, Mar. 1992.
- [10] CHOU, W., LEE, C. H., and JUANG, B. H., “Minimum error rate training based on n-best string models,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, (Minneapolis, MN), pp. 652–655, Apr. 1993.
- [11] CHOW, Y. L., “Maximum mutual information estimation of hmm parameters for continuous speech recognition using the n-best algorithm,” in *Proceedings of*

- the IEEE International Conference on Acoustics, Speech and Signal Processing*, (Albuquerque, NM), pp. 701–704, Apr. 1990.
- [12] CLARKSON, P. and MORENO, P. J., “On the use of support vector machine for phonetic classification,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 585–588, 1999.
- [13] DELLER, J. R., HANSEN, J. H. L., and PROAKIS, J. G., *Discrete-Time Processing of Speech Signals*. New York, NY: IEEE Press, 1999.
- [14] DEMPSTER, A., LAIRD, N., and RUBIN, D., “Maximum likelihood from incomplete data via the em algorithm,” *Journal of Royal Statistical Society B*, vol. 39, no. 1, pp. 1–38, 1977.
- [15] DOOB, J. L., *Stochastic Process*. New York, NY: Wiley, 1953.
- [16] DU, J., LIU, P., SOONG, F. K., ZHOU, J. L., and WANG, R. H., “Minimum divergence based discriminative training,” in *Proceedings of the International Conference on Speech and Language Processing*, (Pittsburgh, Pennsylvania), Sep. 2006.
- [17] DUDA, R., HART, P., and STORK, D., *Pattern Classification, 2nd edition*. John Wiley, 2001.
- [18] FU, Q., HE, X., and DENG, L., “Investigation of phone-discriminating minimum classification error (p-mce) training for continuous phone recognition on the timit database,” in *Interspeech-2007*, (Antwerp, Belgium), Aug. 2007.
- [19] FU, Q. and JUANG, B. H., “Segment-based phonetic class detection using minimum verification error (mve) training,” in *Interspeech-2005*, (Lisbon, Portugal), Sep. 2005.
- [20] FU, Q. and JUANG, B. H., “Investigation on rescoring using minimum verification error (mve) detectors,” in *Proceedings of the International Conference on Speech and Language Processing*, (Pittsburgh, PA), Sep. 2006.
- [21] FU, Q. and JUANG, B. H., “An investigation of the non-uniform error cost function design in automatic speech recognition,” in *submitted to ICASSP-2008*, (Las Vegas, NV), Apr. 2008.
- [22] FU, Q., MANSJUR, D., and JUANG, B. H., “Pattern recognition with non-uniform error criteria,” *submitted to IEEE Transaction on Pattern Analysis and Machine Intelligence*.
- [23] FU, Q., MANSJUR, D., and JUANG, B. H., “Non-uniform error criteria for automatic pattern and speech recognition,” in *submitted to ICASSP-2008*, (Las Vegas, NV), Apr. 2008.

- [24] FU, Q., MORENO, A., JUANG, B. H., ZHOU, J. L., and F.SOONG, “Generalization of the minimum classification error (mce) training based on maximizing generalized posterior probability (gpp),” in *Proceedings of the International Conference on Speech and Language Processing*, (Pittsburgh, PA), Sep. 2006.
- [25] GLASS, J., “A probabilistic framework for segment-based speech recognition,” *Computer Speech and Language*, vol. 17, no. 2-3, pp. 137–152, 2003.
- [26] GOLDBERGER, J., “An efficient images similarity measure based on approximations of kl-divergence between two gaussian mixtures,” in *Proceedings of the International Conference on Computer Vision*, (Nice, France), pp. 370–377, 2003.
- [27] GOPALAKRISHNAN, P. S., KANEVSKY, D., NADAS, A., and NAHAMOO, D., “An inequality for rational functions with applications to some statistical estimation problems,” *IEEE Transaction on Information Theory*, vol. 37, no. 1, pp. 107–113, 1991.
- [28] HAEB-UMBACH, R. and NEY, H., “Linear discriminant analysis for improved large vocabulary continuous speech recognition,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 1, (San Francisco, CA), pp. 13–16, Mar. 1992.
- [29] HE, X., DENG, L., and CHOU, W., “A novel learning method for hidden markov models in speech and audio processing,” in *IEEE International Workshop on Multimedia Signal Processing (MMSP)*, (Victoria, BC, Canada), Oct. 2006.
- [30] HERMANSKY, H., “Perceptual linear predictive (plp) analysis of speech,” *Journal of the Acoustical Society of America*, vol. 87, pp. 1738–1752, Apr. 1989.
- [31] HOU, J., RABINER, L. R., and DUSAN, S., “Automatic speech attribute transcription (asat) - the front end processor,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, (Toulouse, France), May 2006.
- [32] HUANG, X. D. and JACK, M. A., “Semi-continuous hidden markov models for speech signals,” *Computer Speech and Language*, no. 3, pp. 329–352, 1989.
- [33] JELINEK, F., “Continuous speech recognition by statistical methods,” *Proceedings of the IEEE*, pp. 532–556, Apr. 1976.
- [34] JELINEK, F., “The development of an experimental discrete dictation recognizer,” *Proceedings of the IEEE*, pp. 1616–1624, Nov. 1985.
- [35] JEON, W., FU, Q., and JUANG, B. H., “A hierarchical classification method using category-dependent features,” in *submitted to ICASSP-2008*, (Las Vegas, Nevada), Apr. 2008.

- [36] JUANG, B. H., CHOU, W., and LEE, C. H., “Minimum classification error rate methods for speech recognition,” *IEEE Transactions on Speech and Audio Processing*, vol. 5, pp. 257–265, May 1997.
- [37] JUANG, B. H. and KATAGIRI, S., “Discriminative learning for minimum error training,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, pp. 1639–1641, Dec. 1992.
- [38] KAPADIA, S., VALTCHEV, V., and YOUNG, S. J., “Mmi training for continuous phoneme recognition on the timit database,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, (Minneapolis, MN), Apr. 1993.
- [39] KATAGIRI, S., JUANG, B. H., and LEE, C. H., “Pattern recognition using a family of design algorithms based upon the generalized probabilistic descent method,” *Proceedings of the IEEE*, vol. 86, Nov. 1998.
- [40] KAWAHARA, T., LEE, C. H., and JUANG, B. H., “Flexible speech understanding based on combined key-phrase detection and verification,” *IEEE Transactions on Speech and Audio Processing*, vol. 6, pp. 558–568, Nov. 1998.
- [41] KAY, S. M., *Fundamentals of Statistical Signal Processing I: Estimation Theory*. Englewood Cliffs, NJ: Prentice Hall, 1993.
- [42] KAY, S. M., *Fundamentals of Statistical Signal Processing II: Detection Theory*. Englewood Cliffs, NJ: Prentice Hall, 1998.
- [43] KUBALA, F., “Design of the 1994 csr benchmark tests,” in *ARPA Human Language Technology Workshop*, (Austin, Texas), pp. 41–46, Jan. 1995.
- [44] KUMAR, N. and ANDREOU, A., “Heteroscedastic discriminant analysis and reduced rank hmms for improved speech recognition,” *Speech Communication*, vol. 26, pp. 283–297, 1998.
- [45] LEE, C. H., “A unified statistical hypothesis testing approach to speaker verification and verbal information verification,” in *Proceedings of COST Workshop on Speech Technology in the Public Telephone Network: Where are we today?*, (Greece), pp. 62–73, Sep. 1997.
- [46] LEE, C. H. and HUO, Q., “On adaptive decision rules and decision parameter adaptation for automatic speech recognition,” *Proceedings of the IEEE*, vol. 88, Aug. 2000.
- [47] LEE, C. H. and JUANG, B. H., “A new detection paradigm for collaborative automatic speech recognition and understanding,” in *Special Workshop in MAUI (SWIM)-2004*, (Maui, Hawaii), Jan. 2004.

- [48] LEE, K. F. and HON, H. W., “Speaker-independent phone recognition using hidden markov models,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 11, pp. 1641–1648, 1989.
- [49] LEONARD, R. G., “A database for speaker-independent digit recognition,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, (San Diego, CA), pp. 42.11.1–42.11.4, March 1984.
- [50] LI, J. and LEE, C. H., “On designing and evaluating speech event detectors,” in *Interspeech-05*, (Lisbon, Portugal), Sep. 2005.
- [51] LI, J., TSAO, Y., and LEE, C. H., “A study on knowledge source integration for candidate rescoring in automatic speech recognition,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, (Philadelphia, Pennsylvania), March 2005.
- [52] LI, J., YUAN, M., and LEE, C. H., “Soft margin estimation of hidden markov model parameters,” in *Proceedings of the International Conference on Speech and Language Processing*, (Pittsburgh, Pennsylvania), Sep. 2006.
- [53] LI, X. W., JIANG, H., and LIU, C., “Large margin hmms for speech recognition,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, (Philadelphia, Pennsylvania), March 2005.
- [54] MA, C. and LEE, C. H., “A study on detection based automatic speech recognition,” in *Interspeech-2006*, (Pittsburgh, PA), Sep. 2006.
- [55] MACHEREY, W., HAFERKAMP, L., SCHLUTER, R., and NEY, H., “Investigations on error minimizing training criteria for discriminative training in automatic speech recognition,” in *Interspeech-2005*, (Lisbon, Portugal), pp. 2133–2136, Sep. 2005.
- [56] MCDERMOTT, E., *Discriminative training for speech recognition*. PhD thesis, Waseda University, Tokyo, Japan, March 1997.
- [57] MCDERMOTT, E. and HAZEN, T. J., “Minimum classification error training of landmark models for real-time continuous speech recognition,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, (Montreal, Canada), pp. 937–940, May 2004.
- [58] MCDERMOTT, E. and KATAGIRI, S., “A new formalization of minimum classification error using a parzen estimate of classification chance,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, (Hong Kong, China), 2003.
- [59] MCDERMOTT, E. and KATAGIRI, S., “Minimum classification error for large scale speech recognition tasks using weighted finite state transducers,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, (Philadelphia, PA), pp. 113–116, March 2005.

- [60] NORMANDIN, Y., *Hidden Markov models, maximum mutual information estimation, and the speech recognition problem*. PhD thesis, McGill University, Montreal, Canada, 1991.
- [61] NORMANDIN, Y. and MORGERA, S. D., “An improved mmie training algorithm for speaker-independent small vocabulary, continuous speech recognition,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, (Toronto, Canada), May 1991.
- [62] PALLET, D. S., FISCUS, J. G., FISHER, W. M., GAROFOLO, J. S., LUND, B. A., and PRZYBOCKI, M. A., “1994 benchmark test for the arpa spoken language program,” in *ARPA Human language Technology Workshop*, (Austin, Texas), pp. 5–36, Jan. 1995.
- [63] POLLARD, D., *Convergence of Stochastic Process*. New York:Springer-Verlag, NY: Springer Series in Statistics, 1984.
- [64] POVEY, D., *Discriminative training for large vocabulary speech recognition*. PhD thesis, Cambridge University, Cambridge, Britain, Jul. 2004.
- [65] POVEY, D., KINGSBURY, B., MANGU, L., SAON, G., SOLTAU, H., and ZWEIG, G., “fmpe: Discriminatively trained features for speech recognition,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 1, (Philadelphia, PA), pp. 961–964, Mar. 2002.
- [66] POVEY, D. and WOODLAND, P. C., “Minimum phone error and i-smoothing for improved dsicriminative training,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, (Orlando, FL), pp. 105–108, May 2002.
- [67] RABINER, L. R., “A tutorial on hidden markov models and selected applications in speech recognition,” *Proceedings of the IEEE*, vol. 77, Feb. 1989.
- [68] RABINER, L. R. and JUANG, B. H., *Fundamentals of Speech Recognition*. Englewood Cliffs, NJ: Prentice-Hall, 1993.
- [69] RAHIM, M. G. and LEE, C. H., “String-based minimum verification error (sb-mve) training for speech recognition,” *Computer Speech and Language*, vol. 11, pp. 147–160, 1997.
- [70] ROBBINS, H. and MONRO, S., “A stochastic approximation method,” *Ann. Math. Stat.*, vol. 22, pp. 400–407, Apr. 1951.
- [71] ROSENBERG, A. E., SIOHAN, O., and S.PARTHASARATHY, “Speaker verification using minimum verification error training,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, (Seattle, WA), pp. 105–108, May 1998.

- [72] SCHLUTER, R., MACHEREY, W., MULLER, B., and NEY, H., “Comparison of discriminative training criteria and optimization methods for speech recognition,” *Speech Communication*, vol. 34, pp. 287–310, May. 2001.
- [73] SCHOLKOPF, B. and SMOLA, A. J., *Learning With Kernels: Support Vector Machine, Regularization, Optimization, and Beyond*. Cambridge, MA: MIT Press, 2002.
- [74] SHA, F. and SAUL, L. K., “Large margin gaussian mixture modeling for phonetic classification and recognition,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, (Toulouse, France), pp. 265–268, Mar. 2006.
- [75] SOONG, F. K., LO, W. K., and NAKAMURA, S., “Generalized word posterior probability (gwpp) for measuring reliability of recognized words,” in *Special Workshop in MAUI (SWIM)-2004*, (Maui, Hawaii), Jan. 2004.
- [76] VALTCHEV, V., ODELL, J. J., WOODLAND, P. C., and YOUNG, S. J., “Mmie training of large vocabulary recognition systems,” *Speech Communication*, vol. 22, no. 4, pp. 303–314, 1997.
- [77] VAPNIK, V., *Statistical Learning Theory*. Wiley, New York, 1998.
- [78] WALD, A., *Sequential analysis*. New York, NY: John Wiley and Sons, 1947.
- [79] WESSEL, F., MACHEREY, K., and SCHLUTER, R., “Using word probabilities as confidence measures,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, (Seattle, WA), pp. 225–228, May 1998.
- [80] WESSEL, F., SCHLUTER, R., MACHEREY, K., and NEY, H., “Confidence measures for large vocabulary continuous speech recognition,” *IEEE Transactions on Speech and Audio Processing*, vol. 9, pp. 288–298, Mar. 2001.
- [81] WOODLAND, P. C., ODELL, J. J., VALTCHEV, V., and YOUNG, S. J., “Large vocabulary continuous speech recognition using htk,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, (Adelaide, Australia), pp. 125–128, Apr. 1994.
- [82] YU, D., DENG, L., HE, X., and ACERO, A., “Use of incrementally regulated discriminative margins in mce training for speech recognition,” in *Proceedings of the International Conference on Speech and Language Processing*, (Pittsburgh, PA), Sep. 2006.

## VITA

Qiang Fu was born in Xi'an, Shaanxi, P.R.China on June 1, 1978. He received his Bachelor of Engineering degree Department of Electronic Engineering from Tsinghua University, China in 2000. He got his Master of Science degree from Department of Electrical and Computer Engineering at University of Rhode Island in 2002. In August 2002, he began his study at Georgia Institute of Technology, Atlanta, GA, where he is pursuing his doctoral study in Speech Recognition and Digital Signal Processing. He worked as a Graduate Research Assistant from 2002 to 2007.