DESIGN OF LOSS FUNCTIONS AND FEATURE TRANSFORMATION
FOR MINIMUM CLASSIFICATION ERROR BASED
AUTOMATIC SPEECH RECOGNITION

by

MADHAVI VEDULA RATNAGIRI

A dissertation submitted to the
Graduate School – New Brunswick
Rutgers, The State University of New Jersey
In partial fulfillment of the requirements
For the degree of
Doctor of Philosophy
Graduate Program in Electrical and Computer Engineering

Written under the direction of
Prof. Lawrence Rabiner
And approved by

____________________________________________________________________

____________________________________________________________________

____________________________________________________________________

____________________________________________________________________

____________________________________________________________________

New Brunswick, New Jersey

January 2011
ABSTRACT OF THE DISSERTATION

DESIGN OF LOSS FUNCTIONS AND FEATURE TRANSFORMATION FOR MINIMUM CLASSIFICATION ERROR BASED AUTOMATIC SPEECH RECOGNITION

by

MADHAVI VEDULA RATNAGIRI

An automatic speech recognition system has two main components, the front end feature processing component, followed by a model training component. A widely used algorithm for training models is Minimum Classification Error (MCE) that designs the model parameters to minimize recognition error. One approach for designing the feature processing component to minimize the error in recognition is transforming the features using the MCE criterion that is most commonly used only to estimate the model parameters. Past efforts that have integrated feature transformation into MCE training assumed the Hidden Markov Model state distributions were represented by diagonal covariance Gaussian mixtures when estimating the model parameters; but assumed full covariance Gaussian mixtures when estimating the feature transformation matrix. We
rectify this discrepancy in assumptions, derive and implement the MCE based feature transformation, using diagonal covariance Gaussian mixtures.

For designing the model parameters, MCE minimizes the recognition error using a standard sigmoid loss function, which is a Parzen window estimate of the Bayes risk. Using different kernels for Parzen Window estimation we developed new loss functions, viz. the Gaussian Kernel based loss and Generalized Savage loss. Investigation into the recognition performance of these loss functions lead to the introduction of a new large margin based loss function for MCE (LM-MCE) where the error is minimized and margin maximized. Minimizing the error, aims to shift the decision boundary so the wrongly classified tokens move to the side of the correct class, (however some of these tokens could lie on or close to the boundary); while increasing the margin between the correctly classified tokens and the decision boundary, removes some of these tokens away from the boundary and improves the robustness of the classification. Unlike previous studies, this effort does not require MCE training prior to maximizing the margin nor does it require non-optimal manual increments of the margin shifts; and since this loss function is bounded, it is not susceptible to outliers. The development of the new LM-MCE loss function has a theoretical basis in the Vapnik and Chervonenkis (VC) theory and was developed using the Bayes risk formulation.
ACKNOWLEDGEMENT

I am grateful to my advisor, Prof. Lawrence Rabiner for his valuable suggestions, large heartedness and understanding. He gave me the freedom to explore and find the problem of interest to me, provided the assistance I needed to evaluate my work and help move things forward. I would like to thank Prof. Biing-Hwang (Fred) Juang for his support with the publication of my papers and sharing his rich insights. I would also like to thank Prof. Chin-Hui Lee for taking the time to read the drafts of my ideas, for his comments and constructive criticism. To Prof. Joseph Wilder and Prof. Ivan Marsic, my committee members, I owe it to their remarks that my presentations at the conference were much appreciated by many peers. I extend my gratitude to Prof. Zoran Gajic, the graduate director and Ms. Barbara Klimkiewicz for helping me navigate through the paperwork and assisting me to stay on track and up to date with the requirements of the program.

I owe a great deal to my parents, parents-in-law and grandparents for their love and support. I feel indebted to my husband who stood by me in the hardest of times, when I myself had lost all hope and now thanks to him I stand a more confident, capable woman than I have ever been. To my kids whose affection and innocent outlook of life has brought me such joy and happiness and given me a whole new perspective on life - Love you always.
Quotes from “IF” by Rudyard Kipling

If you can keep your head when all about you
    Are losing theirs and blaming it on you;
If you can trust yourself when all men doubt you,
    But make allowance for their doubting too:

        ........

If you can dream – and not make dreams your master;
    If you can think – and not make thoughts your aim,
If you can meet with Triumph and Disaster
    And treat those two impostors just the same:

        ........

If you can fill the unforgiving minute
    With sixty seconds’ worth of distance run,
Yours is the earth and everything that’s in it,

        ........
# TABLE OF CONTENTS

ABSTRACT OF THE DISSERTATION ................................................................................... ii

ACKNOWLEDGEMENT ........................................................................................................ iv

Quotes from “IF” by Rudyard Kipling ........................................................................... v

TABLE OF CONTENTS ................................................................................................... vi

CHAPTER 1 ....................................................................................................................... 1
  INTRODUCTION .............................................................................................................. 1
    1.1 Background ............................................................................................................... 2
    1.2 Outline of the Thesis ................................................................................................. 6

CHAPTER 2 ....................................................................................................................... 9
  CLASSIFICATION TECHNIQUES .................................................................................. 9
    2.1 Introduction ............................................................................................................... 9
      2.1.1 Maximum Likelihood Estimation (MLE) ........................................................ 10
      2.1.2 Maximum Mutual Information Estimation (MMIE) ....................................... 11
      2.1.3 Minimum Word Error (MWE)/ Minimum Phone Error (MPE) ................. 11
      2.1.4 Minimum Classification Error (MCE)............................................................. 13
    2.2 Minimization of Classification Error Using Gradient Probability Descent .......... 14
    2.3 Estimation of Hidden Markov Models using MCE ................................................ 17

CHAPTER 3 ..................................................................................................................... 21
  FEATURE TRANSFORMATION TECHNIQUES ........................................................... 21
    3.1 Linear Transformations ........................................................................................... 22
      3.1.1 Principal Component Analysis (PCA) ............................................................. 23
      3.1.2 Linear Discriminant Analysis (LDA) ............................................................... 24
      3.1.3 Heteroscedastic Linear Discriminant Analysis (HLDA).............................. 25
    3.2 Non Linear Feature Transformation Using Artificial Neural Networks ................. 26
    3.3 Integrated Optimization ........................................................................................ 27
      3.3.1 Semi-Tied ML estimation ................................................................................ 28
      3.3.2 MCE Based Integrated Feature Processing and Model Design ....................... 29

CHAPTER 4 ..................................................................................................................... 33
  MCE BASED FEATURE TRANSFORMATION ........................................................... 33
    4.1 MCE Based Feature Transformation Using Diagonal Covariance ...................... 34
    4.2 Experimental Results ............................................................................................ 40
      4.2.1 Aurora2 Database ............................................................................................. 40
      4.2.2 Comparison to Other Studies Done on Aurora database ................................. 44
    4.3 Conclusions and Future Work .............................................................................. 47
CHAPTER 1

INTRODUCTION

Work on automatic speech recognition (ASR) systems has been ongoing for a number of years now and with each passing year more powerful and more successful systems are being developed. However, the one problem that has consistently baffled researchers in this field, since the time the first speech recognition systems were built, is how to resolve the differences between the training samples used to design and build a speech recognition system and the test samples that are need to be recognized out in the real world. Most successful recognition systems are trained on data that is identical to the test data; thus limiting the differences between the train and test sets. With application of ASRs becoming wide spread, it has become more urgent that ASRs perform with high accuracy in environments and test scenarios that they were not trained in, because ASRs cannot be retrained every time there is a change in the background noise or for every new speaker. Somehow humans seem to do it with relative ease; and the question for automated speech recognition is how and which aspect of a general recognition system needs to be modified to incorporate differences between train and test data - the features that are used to train the speech models, the type of models that are used to characterize the features or the model design algorithm. There are a large number of approaches to tackling this problem of mismatch between the train and test sets and this doctoral research work contributes to two approaches that were developed to resolve the mismatch
problem. One popular technique is model adaptation, where the models are trained using the old training data and when there is a change in the environment or speaker, a small dataset that includes the change is used to transform the previously trained models to incorporate the new characteristics. The mean and variances of the previously trained models are transformed so that the modified models are more likely to generate the ‘new’ test data. The research done in this work on feature transformation extends directly to model adaptation.

The work on loss functions relates to another more recent approach to addressing the mismatch problem viz. large margin (LM) classification. Studies in LM classification have suggested that increasing the margin between correctly classified feature vectors (from the training set) and the decision boundary can improve the robustness of the classification, so that test vectors from the same class that have a slightly higher perturbation still fall within the class boundary. This thesis delves into more detail in each of these areas, i.e. feature transformation and large margin classification, with specific regard to the minimum classification error (MCE) training algorithm.

1.1 Background

Speech recognition is a pattern classification task that recognizes the word that was spoken by categorizing a speech signal into one of the word classes. Speech data as such is never used for recognition, because it is too large and has a lot of redundant information. Thus features are extracted from the speech signal that have relevant
information for classification and are compact so as to reduce computational and storage requirements. State of the art speech recognition systems have a front end feature processing section (Fig. 1) that extracts features from the speech signal and also transforms the features, usually to decorrelate them. The statistical properties of features belonging to a word class are generally represented by Gaussian distributions and decorrelating these features makes it possible to use only the diagonal covariance as opposed to a full covariance matrix, thus further reducing the computational complexity. The parameters of the Gaussian distributions (like the mean and diagonal covariance) are estimated in the modeling section, using the features from the feature processing section. The models thus trained are then used to recognize words from the test set, to evaluate the efficacy of the training process.

A background on the standard approach to extraction and transformation of features and an explanation of the fundamental Bayes’ Decision theory used to categorize a feature vector into a class are briefly explained in the next two sections. The model training algorithms however are detailed in Chapter 2 as listed in the Outline section (1.2).
1.1.1 Standard Feature Extraction and Transformation

The most commonly used features and feature transformation technique that is a standard in all baseline speech recognition systems is the Mel-frequency cepstral coefficients (MFCC). MFCCs are extracted from short 25ms windowed segments of speech, with the window shifted every 10 ms along the duration of the speech. These segments of speech are then passed through a bank of triangular filters. It was found that spacing the center frequency of the triangular filters based on the natural mel-frequency scale, seen in the cochlea of the human ear, leads to better performance [1]. The log of the spectral features thus extracted are then decorrelated using Discrete Cosine transform (DCT) to obtain the cepstral coefficients. Usually the first and second order derivatives of the cepstral coefficients are also computed for information about the relative change between successive cepstral values [2]. Once the MFCCs are extracted, these features are used to design the speech models in the modeling section (Fig. 1) and for classifying the extracted features into the various model classes, during the evaluation phase, the Classical Bayes Decision theory [3] was developed.

1.1.2 Classification Using Bayes Risk

Bayes Decision Theory is a fundamental theory that uses the statistical distribution of the features to classify a speech sample into the right word class by reducing the cost of classification.
Consider that there are $M$ word classes, the goal is to develop a classifier with a decision function $h: \mathbb{Y} \rightarrow \mathbb{C}$ that accurately maps a class label $\mathbb{C} \in \{C_1, C_2, ..., C_M\}$ to every feature vector $y \in \mathbb{Y}$, where $\mathbb{Y}$ is the feature space. Ideally we seek a function $h$ that accurately classifies all feature vectors $y$ to their right class; practically we choose a function $h$ to minimize the error in classification. Assume that the feature vectors have a probability density $p(y)$ then the classification error or risk is defined as

$$R = \int_{\mathbb{Y}} R(\mathbb{C}_i/y)p(y)dy$$

(1.1)

where $R(\mathbb{C}_i/y)$ is the conditional risk and defined as

$$R(\mathbb{C}_i/y) = \sum_{j=1}^{M} e_{ij}p(C_j/y)$$

(1.2)

where $M$ is the total number of models, $p(C_j/y)$ is the aposteriori probability and $e_{ij}$ is the cost of wrongly classifying an observation from class $C_j$ into class $C_i$. If the cost is defined as in (1.3),

$$e_{ij} = \begin{cases} 1, & i \neq j \\ 0, & i = j \end{cases}$$

(1.3)

that is, there is no cost when the vector is classified to the right class and the cost is 1 when the vector is wrongly classified, then the conditional risk is

$$R(\mathbb{C}_i/y) = \sum_{i \neq j} p(C_j/y) = 1 - p(C_i/y)$$

(1.4)

and the risk is minimized by the decision rule
Thus the class with maximum aposteriori probability is chosen as the word class for the given feature vector. The risk obtained using the above decision rule (1.5) is called the “Bayes” risk, and if a model can be used to define the true aposteriori probability, the parameters of the model can be estimated by minimizing the Bayes’ risk. However, the Bayes risk cannot be minimized using optimization methods because the cost function (1.3) is discontinuous and the Bayes decision rule (1.5) cannot be implemented because of the lack of knowledge of the true aposterior probability $P(C_j/y)$. For information on the true aposteriori probability complete knowledge of the source of speech is required, which cannot be obtained from the limited amount of speech samples that is usually available. Thus the Bayes Risk remains an unattainable “theoretical lower bound” and hence other classification techniques have been developed over the years. In this work we explore various aspects of one such classification technique called the Minimum Classification Error (MCE).

### 1.2 Outline of the Thesis

Bayes decision theory is a practically unachievable theoretical lower bound, so other classification techniques were explored by many researchers and some of the successful techniques have been summarized in Chapter 2. A detailed explanation of the Minimum Classification Error (MCE), model design technique is given because this is the technique
shown to give the best recognition accuracy on many speech recognition tasks and has been extensively researched in this work. The estimation of the model parameters when Hidden Markov Models are used to represent the distribution of the features is explained as well. Chapter 3 recaps techniques that have been introduced over the years to decorrelate and/or transform features. Specifically techniques that transform features into a space that would optimally separate them to minimize the recognition error are explored. Model training estimates the model parameters to minimize the error, when the same criterion is used to transform the features, the features and the model parameters are optimally estimated and it is a more effective way to develop a speech recognition system (integrated optimization – as shown in Fig. 2).

The feature transformation technique developed as part of this work transforms features using the MCE classification technique for Hidden Markov Models with state distributions that are Gaussian mixtures with diagonal covariance. The mathematical
exposition of this approach along with the experimental results have been presented in Chapter 4. Chapter 5 illustrates the reasoning behind the exploration for new loss functions for MCE classification and introduces two new loss functions that have been developed as part of this work. Chapter 6 develops a new large margin based loss function – that optimally maximizes the distance between the correctly classified vector samples and the decision boundary. Chapter 7 concludes with a summary of the work done as part of this doctoral research, the conclusions drawn from the results that were observed and ideas for future work.
CHAPTER 2

CLASSIFICATION TECHNIQUES

2.1 Introduction

One of the key aspects of speech recognition systems is modeling the distribution of feature vectors belonging to a class; this involves preserving the class information, while at the same time being able to separate the features belonging to different classes. The Bayes risk formulation provides a decision criterion to accurately categorize features into word classes based on the feature distribution. However, the Bayes decision theory requires complete knowledge of the distribution of features and this cannot be obtained for most tasks because only limited amount of speech data is typically available; thus the Bayes risk formulation cannot be practically implemented. In order to develop recognition systems, classification techniques have been developed that employ other estimation criteria to model the feature distribution. Some of the successful estimation techniques are briefly explained followed by a detailed explanation of the standard MCE estimation approach, since that is classification technique researched in this work.
2.1.1 Maximum Likelihood Estimation (MLE)

The standard classification technique is the Maximum Likelihood estimation (MLE) and is based on the Bayes decision theory. As described in Chapter 1 the Bayes decision theory decides a model class by maximizing the aposteriori probability (1.5). Since the true aposteriori probability cannot be obtained, MLE derives the best estimate of aposteriori probability distribution, which is the distribution of the class given the features that belong to it [refer Section 1.1.2], [4] [5]. Using the Bayes rule, the aposteriori probability can be rewritten as shown below.

\[ P(C_j/y) = \frac{P(y/C_j)P(C_j)}{P(y)} \]  \hspace{1cm} (2.1)

Since \( P(y) \) is not dependent on the class, it is ignored and assuming all classes are equally likely, the aposteriori probability can be maximized by maximizing the class conditional likelihood \( P(y/C_j) \). Typically Hidden Markov Models (HMM) are considered to well characterize the transient and spectral characteristics of speech and hence widely used to represent the class conditional likelihood. To estimate the HMM model parameters, the Expectation Maximization formulation and its implementation using the fast and efficient Baum-Welch algorithm [6], is considered as the standard and used as a baseline to compare all other classification techniques.
2.1.2 Maximum Mutual Information Estimation (MMIE)

Unlike the Bayes decision theory that aims to minimize the classification error, Maximum Mutual Information Estimation (MMIE) [7] maximizes the mutual information between the observed feature sequence and its corresponding word sequence. Assume \( X = x_1x_2...x_t \) is the sequence of features extracted from a speech signal and \( \Theta = \{ \theta_1, \theta_2, ..., \theta_M \} \) are the HMM models representing the word sequence, then MMIE aims to estimate the parameters of the HMM model that would maximize the mutual information between the model and its features \( \left( I_\varphi(\theta_t, y_t) \right) \) as shown below

\[
I_\varphi(\theta_t, x_t) = \log \frac{P_\varphi(x_t, \theta_t)}{P(x_t)P(\theta_t)} \tag{2.2}
\]

where \( \varphi \) is the set of HMM parameters.

2.1.3 Minimum Word Error (MWE)/ Minimum Phone Error (MPE)

Minimum word error introduced by Povey et. al. [8] combines the MMIE criterion with word error rate. Word error rate is the performance metric used to test speech recognition systems. When training the speech recognition models the estimation techniques minimize error at the sentence level however when testing the trained models, word error rate as measured using the Levenshtein distance is used. In order to bridge the gap between the sentence based estimation techniques and the word based performance
metric; objective functions that have posterior weighted probability are used as shown in (2.3).

\[ F(\varphi) = \sum_{S} P_{\varphi}(S/y)A(S, R) \]  

(2.3)

where \( S \) is the hypothesis for an utterance whose observation vector is \( x \) and \( R \) is the reference transcription. \( P_{\varphi}(S/y) \) is the posterior probability of the utterance given the observation vector, \( \varphi \) is the model parameter set. \( A(S, R) \) is a variation of the word error rate also called the Raw accuracy, and this should represent the Levenshtein distance which is the number of words that were incorrectly recognized by substitutions, deletions or insertions. Because of the discontinuous nature of the Levenshtein distance, many approximations to the Raw accuracy have been defined, like RawWordAccuracy.

Minimum Phone error (MPE) a variation of the Minimum Word error also introduced by Povey et. al. [8] used RawPhoneAccuracy, as an approximation of the phone level Levenshtein distance, (they found the phone level improved recognition accuracy more that the word level). It was defined as the number of correctly transcribed phones (q) in a hypothesis \( S \) and it was computed based on the extent of overlap in time of each phone with its reference considered over all possible phone hypotheses \( S \). The objective function of this approach is shown below

\[ \text{RawPhoneAccuracy}(S) = \sum_{q \in S} \text{PhoneAcc}(q) \]

where

\[ \text{PhoneAcc}(q) = \begin{cases} 
-1 + 2e & \text{when } q \text{ is the correct label} \\
-1 + e & \text{otherwise}
\end{cases} \]  

(2.4)
where \( e \) represents the contribution of each phone determined by the extent of overlap in time to the reference phones. The objective function used for MPE was

\[
F_{\text{MPE}}(\varphi) = \frac{\sum_S \theta(x/S)P(S) \text{RawPhoneAccuracy}(S)}{\sum_S \theta(x/S)P(S)}
\]  

(2.5)

### 2.1.4 Minimum Classification Error (MCE)

Minimum Classification Error is a classification technique that directly minimizes an empirical risk that is different from the Bayes risk (though it has been shown [35] that under certain assumptions this risk can converge to the Bayes’ risk). MCE estimates the model parameters by minimizing a continuous sigmoid loss function using the Gradient probability descent (GPD) optimization approach. Embedded into the loss function is a misclassification measure, which is a measure of separation between hypothesized class and the nearest competing classes, making this approach discriminative. While the loss function is minimized, the misclassification measure is maximized. As is generally the case, the speech features are assumed to be sufficiently characterized by Hidden Markov Models (HMMs) and the following section explains the standard approach [10, 9] to estimation of the HMM model parameters in more detail [9,10,17,35].
2.2 Minimization of Classification Error Using Gradient Probability Descent

Consider $X$ to be the set of training vectors and $\varphi$ the set of model parameters that need to be estimated. Let $C_j$ be the $j$th class that $X$ belongs to and let $C_i$ be the class it is being classified into. Just as in the case of Bayes risk [refer Chapter 1], MCE also assigns zero cost to correct classification and a cost of 1 to incorrect classification using a cost function as shown below

$$ e_{ij} = I\left(d_j(x) \geq 0\right) \quad (2.6) $$

In order to determine if there has been a misclassification, MCE defines $d_j(x)$, a misclassification measure and a limiting case of $d_j(x)$ is defined in (2.7).

$$ d_j(x) = -g_j(X; \varphi) + \max_{i \neq j} g_i(X; \varphi) \quad (2.7) $$

where $g_i(X; \varphi)$ is the discriminant function. Different discriminant functions have been previously suggested [10], the one that is widely used is the log of the class conditional likelihood,

$$ g_i(X; \varphi) = \log[p_i(X; \varphi)] \quad (2.8) $$
Thus if we consider a feature vector from class $C_i$ which is classified to class $C_j$ because $P_j(X/\phi)$ is greater than $P_i(X/\phi)$ then $d_i$ (2.7) will be positive and thus assigned a cost of 1 (2.6). To minimize the classification error rate the cost function (2.6) is to be minimized, however the cost function being discontinuous cannot be minimized by optimization routines, hence the misclassification measure is embedded into a continuous sigmoid (0-1) loss function as shown below.

$$L_i(X;\phi) = \frac{1}{1+\exp(-\gamma d_i(X)+\rho)}$$  \hspace{1cm} (2.9)$$

and the empirical risk is defined as

$$R_N = E_{X}[l(X;\phi)] = \sum_{X\in C_i} \sum_{i=1}^{M} L_i(X;\phi)p(x)dx$$  \hspace{1cm} (2.10)$$

where $M$ is the total number of classes. It can be observed from the above equations (2.9), (2.10) that minimizing the empirical error, minimizes the loss function, which in turn maximizes the misclassification measure. Maximizing the discriminant function for the current class while minimizing the function for nearest competing class (2.7) increases the separation between the class (it was shown that this distance defined by the misclassification measure is an approximation to the Kullback-Leibler divergence [11]). Thus an important benefit to using MCE is that while it minimizes the cost of classification, it also maximizes the separation distance between classes, i.e it is discriminative by nature.
The model parameters $\phi$ that minimize the MCE empirical cost are estimated using GPD. GPD updates the model parameters as shown in (2.11) where the model parameters are updated along the negative gradient of the loss function, thus choosing model parameters that lower the value of the loss function. With iterative processing the local minimum can be obtained and if the initial model parameters are chosen judiciously, this local minimum can be the global minimum.

$$\phi_{\tau+1} = \phi_\tau - \varepsilon \frac{\partial l_1(X; \phi)}{\partial \phi} \bigg|_{\phi = \phi_\tau}$$

(2.11)

$\tau$ is the above equation is the number of iteration, $\varepsilon$ is the learning rate and the derivative of the loss function is determined using the chain rule as shown in (2.12).

$$\frac{\partial L_1(X; \phi)}{\partial \phi} = \frac{\partial L_1}{\partial d_i} \frac{\partial d_i}{\partial g_j} \frac{\partial g_j}{\partial \phi}$$

and

$$\frac{\partial L_1}{\partial d_i} = \gamma L_1(d_i)(1 - L_1(d_i))$$

$$\frac{\partial d_i}{\partial g_j} = \begin{cases} -1 & \text{if } j = i \\ 1 & \text{if } j \neq i \end{cases}$$

(2.12)

To implement the GPD as described above, a model that characterizes the class conditional likelihood, viz. $P_j(X; \phi)$, is required and the most commonly used models are the Hidden Markov Models (HMMs). A simple left-right HMM is typically used, so as to represent both the time and frequency characteristics, while at the same time keeping the
computational complexity manageable. To get GPD started HMM model parameters are initialized using the ML estimation criterion, which is followed by the MCE estimation as described below.

2.3 Estimation of Hidden Markov Models using MCE

When using Hidden Markov models to characterize each word/sub-word class, as is generally done for most speech recognition systems, the discriminant function and the class-conditional likelihood are defined as shown in (2.13).

\[
g_j(X/\varphi) = \log \left\{ P_j(X/\varphi) \right\} = \log \left\{ \max_q P_j(X,q/\varphi) \right\} \\
= \log \left\{ P_j(X,q/\varphi) \right\} = \log \left[ \prod_{q_0} \prod_{t=1}^T a_j^{q_{t-1}q_t} \pi_j^{q_0} b_j^{q_t}(x_t) \right]
\]  

(2.13)

where \( \pi_j^{q_0} \) is the initial state probability of the HMM model corresponding to class \( C_j \), \( q_0 \) is the initial state, \( a_j^{q_{t-1}q_t} \) is the transition probability from state \( q_{t-1} \) to \( q_t \), \( b_j^{q_t}(x_t) \) is the output state probability distribution, \( T \) is the number of time frames and \( \bar{q} \) is the optimal state sequence. Using the above definition for HMM density the set of model parameters that need to be estimated for an HMM are the initial state probability \( \pi_j^{q_0} \), the transition probability \( a_j^{q_{t-1}q_t} \) and the parameters of the output state distribution \( b_j^{q_t}(x_t) \), which is generally represented using Gaussian mixtures. It is only
the state distribution parameters that are estimated because the other parameters do not change much during the MCE estimation process [17]. Thus the gradient of the discriminant function, required in (2.12) is obtained as shown below.

\[
\frac{\partial g_j}{\partial \phi} = \frac{\partial \log P_j(X|\phi)}{\partial \phi} = \sum_{t=1}^{T} \delta(q_t - j) \frac{\partial \log b^j_{q_t}(x_t)}{\partial \phi} \tag{2.14}
\]

The Gaussian mixture that represents the output state distribution \(b^i_{q_t}(x_t)\) is defined as

\[
b^i_{q_t}(x_t) = \sum_{k=1}^{K} c^i_{k,q_t} \mathcal{N}(x_t; \mu^i_{k,q_t}, \Lambda^i_{k,q_t}) \tag{2.15}
\]

where \(K\) is the total number of Gaussian mixtures, \(c^i_{k,q_t}\) is the weight of each Gaussian mixture and \(\mathcal{N}()\) denotes a normal distribution with mean \(\mu^i_{k,q_t}\) and diagonal covariance \(\Lambda^i_{k,q_t}\) and defined in (2.16).

\[
\mathcal{N}(x_t; \mu^i_{k,q_t}, \Lambda^i_{k,q_t}) = \frac{1}{(2\pi)^{n/2}|\Lambda^i_{k,q_t}|^{1/2}} \exp \left(-\frac{1}{2} (x - \mu^i_{k,q_t})^T \Lambda^i_{k,q_t}^{-1} (x - \mu^i_{k,q_t}) \right) \tag{2.16}
\]

\(n\) is the dimensionality of \(X\). Thus the set of parameters that define the output state distribution and are estimated is \(\phi = \left\{ c^i_{k,q_t}, \mu^i_{k,q_t}, \Lambda^i_{k,q_t} \right\}\). In order to maintain the constraints of the HMM and keep the covariance matrix and the mixture weights positive.
definite and also to handle the issue of small variances [10], the following transformations for the mean (2.17), variance (2.18) and mixture weight (2.19) are used.

\[ \tilde{\mu}_{k, q_t}^j = \frac{\mu_{k, q_t}^j}{\Lambda_{k, q_t}^j} \]  \hspace{1cm} (2.17)

\[ \tilde{\Lambda}_{k, q_t}^j = \log \left\{ \Lambda_{k, q_t}^j \right\} \]  \hspace{1cm} (2.18)

\[ \tilde{c}_{k, q_t}^j = \log \left\{ c_{k, q_t}^j \right\} \]  \hspace{1cm} (2.19)

Using the above transformation, the gradients with respect to the mean, variance and mixture weight are

\[
\frac{\partial \log b_{q_t}^j (x_t)}{\partial \mu_{k, q_t}^j} = \frac{c_{k, q_t}^j}{(2\pi)^{n/2} \left| \Lambda_{k, q_t}^j \right|^{1/2} b_{q_t}^j (x_t)} \exp \left\{ -\frac{1}{2} (x - \mu_{k, q_t}^j)^T \Lambda_{k, q_t}^{-1} (x - \mu_{k, q_t}^j) \right\}
\]  \hspace{1cm} (2.20)

and
\[
\frac{\partial \log b^j_{q_t}(x_t)}{\partial \mathbf{\Lambda}^j_{k,q_t}} = \frac{c^j_{k,q_t}}{(2\pi)^{n/2} \left| \mathbf{\Lambda}^j_{k,q_t} \right|^{1/2} b^j_{q_t}(x_t)}^{*} \left[ \left( \frac{x - \mu^j_{k,q_t}}{\mathbf{\Lambda}^j_{k,q_t}} \right)^2 - 1 \right] \exp \left( -\frac{1}{2} (x - \mu^j_{k,q_t})^T \mathbf{\Lambda}^j_{k,q_t}^{-1} (x - \mu^j_{k,q_t}) \right) \right]
\]

(2.21)

and

\[
\frac{\partial \log b^j_{q_t}(x_t)}{\partial c^j_{k,q_t}} = \frac{c^j_{k,q_t}}{(2\pi)^{n/2} \left| \mathbf{\Lambda}^j_{k,q_t} \right|^{1/2} b^j_{q_t}(x_t)}^{*} \left[ \left( \frac{x - \mu^j_{k,q_t}}{\mathbf{\Lambda}^j_{k,q_t}} \right)^2 - 1 \right] \exp \left( -\frac{1}{2} (x - \mu^j_{k,q_t})^T \mathbf{\Lambda}^j_{k,q_t}^{-1} (x - \mu^j_{k,q_t}) \right) \right]
\]

(2.22)

An additional constraint that the mixture weights sum to one is maintained by renormalizing the weights after the MCE update (2.22). Thus using Gradient probability descent and iterative processing, the model parameters that minimize the classification error can be estimated.
CHAPTER 3

FEATURE TRANSFORMATION TECHNIQUES

In the previous chapter we considered one key aspect of speech recognition systems, which was the design of the models that characterize the feature vectors. Here we consider another key aspect namely, feature transformation. The transformation of features is essential to reduce the number of parameters that are estimated in the modeling section, so as to reduce computational complexity as well as ensure reliable estimates. It can also be used to improve recognition accuracy.

Most transformation techniques decorrelate features so that when using Gaussian mixtures to represent them, diagonal covariances would suffice instead of full covariance matrices. For an NxN diagonal covariance matrix only N terms are estimated, on the other hand for a full covariance matrix N*(N+1) terms need to be estimated. Given limited amount of data, it might not be possible to reliably estimate all the cross terms of a full-covariance matrix. Thus considering only the diagonal elements of a covariance matrix not only reduces computation, but also ensures the terms of the covariance matrix estimated are reliable.

Other transformation techniques optimize features to directly or indirectly minimize the classification error. It is believed that an optimal speech recognition system is obtained when the front end processing is integrated with that of the modeling section.
So, transformation of the features is done using the same classification criteria used to train the classifier models (as described in Chapter 2).

Some linear transformation techniques used to decorrelate feature vectors, a non-linear transformation that uses Artificial Neural Networks (ANNs) and integrated optimization techniques are further discussed in this chapter.

3.1 Linear Transformations

To diagonalize a covariance matrix, the features are decorrelated using linear transformation techniques like Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), or Heteroscedastic Linear Discriminant Analysis (HLDA). These techniques have shown to significantly improve the recognition performance [18], [13] compared to the standard Discrete Cosine Transformation approach (described in Chapter 1).

A basic approach to diagonalizing a feature vector is to consider its eigen decomposition. Let $X$ be a $p$ dimensional feature vector. Then $X$ can be represented by a sum of linearly independent vectors

$$
X = \sum_{m=1}^{p} y_m \phi_m
$$

(3.1)

where $\phi_m$ are the basis vectors of $X$ and $y_m \in Y$ is the orthogonal transformation. Using matrix manipulation, the transformed feature vectors can be obtained as shown in (3.2).
\[
y_m = \phi_m^T X \quad \text{where} \quad m = 1, \ldots, p \Rightarrow Y = \Phi^T X \quad \Rightarrow \Sigma_y = \Phi^T \Sigma_x \Phi = \Lambda_x \tag{3.2}
\]

where \( \Phi \) is the linear transformation matrix, \( \Sigma_x \) is the covariance of \( X \) and \( \Sigma_y \) is the covariance of \( Y \). The columns of \( \Phi \) are the eigen vectors \( \phi_m \) that span the \( p \)-dimensional space of \( X \). Thus the transformed feature vector \( Y \) has a diagonal covariance matrix \( \Lambda_x \), which is the eigen value matrix. This approach does improve recognition accuracy as shown by Ljole et.al. [22], however it is computationally expensive to find all the eigen vector matrices (explained in more detail in Section 3.3.1).

### 3.1.1 Principal Component Analysis (PCA)

PCA is a linear transformation technique that transforms the features into a sub-space using the Karhunen-Loeve (KL) transform so that the mean squared error between the transformed features and the original features is minimized. The columns of the KL transformation matrix are eigen vectors, that correspond to the largest eigen values of the covariance matrix. In other words in PCA, only ‘\( q \)’ of the ‘\( p \)’ linearly independent basis vectors are considered.

\[
\hat{X} = \sum_{m=1}^{q} y_m \phi_m \tag{3.3}
\]

In order for \( \hat{X} \) to be a good approximation of the original feature vectors \( X \), the mean squared error between \( X \) and \( \hat{X} \) is minimized and that leads to the condition shown below (refer [12] for more details).
\[ E\left\{ \| X - \hat{X} \|^2 \right\} = \sum_{m=q+1}^{p} \lambda_m \sum_{m=q+1}^{p} \phi_m^T \Sigma \phi_m \]  

(3.4)

where \( \Sigma_x \) is the covariance of \( X \), \( \lambda_m \) are the eigen values. Increasing the number of components of the basis vectors ‘q’ that are considered, to represent the feature vector, reduces the error, but increases the computation and vice versa. The selection of ‘q’ is heuristic where the best trade off between computation vs recognition accuracy is made.

3.1.2. Linear Discriminant Analysis (LDA)

LDA computes eigen vectors that maximize the separation distance between classes. The class separability is measured using the Fisher index \( \left( \Sigma^{-1}_w \Sigma_B \right) \) which is the ratio of the with-in class variance \( (\Sigma_w) \) to between -class variance \( (\Sigma_B) \) [12]. The former is minimized as the latter is maximized. ‘q’ eigen vectors that correspond to ‘q’ of the largest eigen values of \( \left( \Sigma^{-1}_w \Sigma_B \right) \) are considered and the feature vectors are projected onto that space. Campbell showed in his work [20] that the LDA estimates of the vectors were actually the ML estimates under the assumption that the classes had different means but common covariance matrix.
3.1.3 Heteroscedastic Linear Discriminant Analysis (HLDA)

Kumar [21] extended the work by Campbell [20] by considering the between-class variance to be different or heteroscedastic for different classes. In order to incorporate class-specific information and also be able to model all the dimensions of the feature vector, the transformation matrix was partitioned. Consider a $p \times p$ linear transformation matrix and partition it into two matrices, one containing all the class specific information and is $q \times p$ dimensions, while the other $(p-q) \times p$ matrix has the nuisance dimensions and is common to all classes. Considering only the class specific transformation will project vectors into different sub-spaces, thus making comparison of probabilities across classes not possible. By representing all the dimensions of the feature vector, Kumar ensured that the likelihood of the different classes can be compared and by splitting the matrix and considering the class dependent part, class-specific characteristics were represented as well.

\[
Y = A^T X \\
\text{where} \\
A = \begin{bmatrix} A_i & A_g \end{bmatrix} \\
\text{and} \\
\mu^Y_i = \begin{bmatrix} \mu_i \\ \mu_g \end{bmatrix} \\
\Sigma^Y_i = \begin{bmatrix} \Sigma^11_i & 0 \\ 0 & \Sigma^22_g \end{bmatrix} \tag{3.5}
\]
A_i is the q \times p dimensional class-dependent matrix, i is the class label and A_g is the
(p - q) \times p nuisance matrix common to all classes. The mean and covariance of the
transformed features are also split into one corresponding to the class-specific dimensions
\left( \mu_i, \Sigma_i \right) and the nuisance dimension \left( \mu_g, \Sigma_g \right) as shown in (3.5). He showed that
the LDA estimates of these mean and variance terms were ML estimates.

3.2 Non Linear Feature Transformation Using Artificial Neural Networks

Artificial Neural Networks (ANNs) are sometimes used to map the feature vectors to
their respective class labels. Unlike the linear transformation techniques that separate
classes using linear hyperplanes, ANNs can design non-linear hyperplanes. The back
propagation algorithm is commonly used to estimate the parameters of the ANN. The
drawback of ANNs is that the back propagation algorithm requires a supervised learning
that estimates the weights and offset parameters of the ANN by minimizing the mean
squared error between the output values at the output nodes and the target output node
values. The output nodes that correspond to the class that the feature vector belongs to are
assigned a target value of ‘1’ while all other nodes are assigned target values of ‘0’ - so
the output of an ANN, will have values close to 1 or 0, a highly non-Gaussian
distribution, thus making the most common model representation like Gaussian
distributions or HMMs with Gaussian mixture states, inappropriate. Studies [13], [14]
comparing linear transformation techniques like PCA and LDA with non-linear ANN transformation have shown improvement in recognition accuracy with ANNs, where they were able to combine different feature processing algorithms.

### 3.3 Integrated Optimization

Feature transformation and model design are integral parts of a speech recognition system and the design of these two aspects of the recognition system should be integrated and optimized using the same criterion. Many studies [15], [16], [17], [18], [19] have integrated the design of features and classifier models (as shown in Fig. 2) and reported improved performance in recognition accuracy. Gopinath [19] and Gales [15] independently developed approaches to estimate the model and the feature transformation matrix using the Maximum Likelihood (ML) criterion. Use of ML for feature transformation has been possible because of the principle developed by Campbell [20] and Kumar [21] that the eigen vectors obtained using LDA/HLDA are ML estimates. The series of studies by Biem et. al. [23], [24], [25], [26] used the MCE model training algorithm to estimate the filter parameters, the filter that is used to extract features from a speech signal. Wang et. al.[18] compared PCA and LDA, with integrated optimization using MCE and showed feature transformation using MCE performed better when the transformed feature vector dimension wasn’t significantly lower than the original feature dimension. Rahim et.al. [27], [28] used MCE to estimate the parameters of an ANN that they used to non-linearly transform features. So many feature processing tasks can be
done using MCE because of the use of gradient probability descent (GPD). The basic premise of each of the above mentioned approaches are further explained in the following sections.

### 3.3.1 Semi-Tied ML estimation

This is a feature transformation approach developed by Gales [15] based on the ML model estimation criterion. A linear transformation of the feature vectors would transform the covariance matrix as shown in (3.6).

\[
Y = AX \Rightarrow \Sigma_{Y} = A\Sigma_{X}A^T
\]  

(3.6)

Rewriting \(\Sigma_{X}\) using (3.2) and suppose the transformation matrix \(A\) is the transpose of the eigen vector matrix i.e. \(\Phi^T\) then the above equation (3.6) gives

\[
\Sigma_{Y} = \Phi^T \left( \Phi \Phi^T \right)^{-1} \Lambda_{X} \Phi^{-1} \Phi = \Lambda_{X}
\]

(3.7)

Thus choosing \(A\) to be the transpose of the eigen vector matrix gives the option of representing a full covariance matrix \(\Sigma_{Y}\) as a diagonal covariance \(\Lambda_{X}\). This was considered in a study by Ljolje [22].

Typically a speech model employs hundreds of Gaussian mixtures and to compute the diagonal covariance matrix \(\Lambda_{X}\) corresponding to each Gaussian mixture would require just as many full transformation matrices ‘\(A\)’. Computationally this is more even expensive than considering full covariance matrices for all the Gaussian mixtures. Gales
in his formulation of the semi-tied covariances represented the transformed covariance matrix as a combination of a diagonal component $\Lambda^x$ and a non-diagonal transform $A_S$ as shown below.

$$\Sigma^y = A_S A^x A_S^T$$  \hspace{1cm} (3.8)

Instead of using $A_S$ to be the transpose of the eigen vector matrix for each specific Gaussian mixture, the non-diagonal component $A_S$ can be global, which would require the least amount of storage and computation, or specific to each mixture, which would make it the most computationally expensive or anything in between, like tied over all states in a model. To estimate $A_S$, semi-tied estimation uses the derivation by Campbell [20] and N. Kumar [21] that the mean and variance of the vectors projected using LDA and HLDA (i.e., the eigen vectors of (3.7) minimize the Fisher index) are actually the ML estimates. Hence Gales [15] modified the Baum-Welch algorithm, generally used to compute the ML estimates of the model parameters to estimate the semi-tied transformation matrix $A_S$ as well.

### 3.3.2 MCE Based Integrated Feature Processing and Model Design

Other authors [16], [17], [18], [23] used a different estimation approach namely the Minimum Classification Error (MCE) to estimate the model parameters as well as extract and transform features. MCE is a discriminative training criterion that maximizes the separation between competing classes, while minimizing a smoothed error function using
gradient probability descent (GPD). The use of GPD for Minimum Classification Error (MCE) training makes it adaptable to estimating the feature extraction parameters including ANN parameters (when ANNs are used for feature transformation) or the feature transformation matrix itself.

**Feature Extraction Using MCE**

Biem et. al. in a series of studies [23], [24], [25],[26] used MCE to estimate the filter parameters to extract features from the speech signal and also a diagonal feature transformation matrix. They estimated filter parameters like center frequencies, bandwidths, gain etc. This approach makes the filter design optimal for classifier design, but also makes it increasingly sensitive to the training data.

**ANN Feature Transformation Design Using MCE**

Rahim and Lee [27], [28] used ANNs to transform the features before training the models and used MCE to estimate the ANN parameters instead of the standard error back-propagation algorithm. As mentioned earlier, the back propagation algorithm being a supervised algorithm requires target values which are usually 1 or 0, thus the values at the output nodes have a highly non-Gaussian distribution and making HMMs with Gaussian mixture states inappropriate as models. Rahim and Lee [27], [28] by combining the ANN and HMM design alleviated the problem of assigning target values for the ANNs, they estimated the ANN parameters by minimizing the sigmoid loss function using GPD and
thus did not require the target output node values. They extended their work [28] to consider multiple ANNs for different classes, instead of one ANN for all classes [27], but failed to ensure that the probability distributions of feature vectors, projected into different sub-spaces using different class-dependent transformations, can be compared.

**Feature Transformation Using MCE**

Chengalvarayan et. al. [17] and Wang et.al [18] in their work on integrated optimization used the MCE model training algorithm to transform the standard mel-frequency based cepstral coefficients (MFCC). Chengalvarayan et. al. [17] estimated state dependent feature transformations (i.e. used different transformations for the different states in an HMM) and failed to ensure that the class conditional probability (CCP) of different states can be compared across the different sub-spaces that the features were projected into. Comparison of the probabilities is required during MCE’s discriminative training where the CCP of the hypothesized model and competing models are used. In the testing phase as well, the CCPs are compared and the feature vector is assigned to the class with highest probability. Wang et. al. [18] in their work on integrated optimization of features and models used a global feature transformation, so the CCP can be compared across models. They used the Mahanolabis distance as a measure of separability between classes, instead of the misclassification measure (‘d’) which has become more of a standard.

In both these studies by Wang et. al [18] and Chengalvarayan et. al. [17], there was a mathematical inconsistency because they assumed diagonal covariance matrices for the
Gaussian mixtures when estimating the parameters of the model, which is generally the norm (as explained in Chapter 1 due to reasons of computation and reliability of estimates) but assumed full covariance matrices when deriving the estimate for the transformation matrix. Further, it is a known fact that linear transformation of a Gaussian distribution with full covariance, does not transform the features, but only scales the distribution, thereby making the transformation mute.

The following chapter describes in detail MCE based feature transformation, where the Gaussian mixtures with diagonal covariance are considered when estimating both the transformation matrix as well as the classifier model parameters. A global feature transformation matrix that is common to all states and models is estimated, thus making it possible to compare probabilities. Using GPD to estimate the parameters requires the initialization (as explained to Chapter 2, Section 2.2) of the model parameters and the transformation matrix which is done using the integrated approach of semi-tied ML estimation introduced by Gales [15], instead of initializing the transform matrix by an identity as in the case of Chengalvarayan et. al.[17] and Wang et. al.[18]. The commonly used, class conditional probability is considered as the discriminating function [10], as opposed to the Mahanolabis distance used by Wang et. al. [18] or the Euclidean distance used by Torre et. al [16].
CHAPTER 4

MCE BASED FEATURE TRANSFORMATION

Independently optimizing the feature processing and model design aspects of a speech recognition system makes for a sub-optimal system. The ML and MCE model design techniques have been used in the past for feature transformation and these techniques were briefly discussed in the previous chapter. The MCE approaches have shown promise, but made the assumption that the Gaussian mixtures used full-covariance matrices when estimating the transformation matrix and made the assumption of a diagonal covariance matrix when estimating the model parameters. We correct this discrepancy in our optimization process and derive the update equations for the transformation matrix. The results of the mathematical analysis implemented on a speech corpus are also presented, herein.
4.1 MCE Based Feature Transformation Using Diagonal Covariance

Assume that $X$ is the original feature vector which is linearly transformed, $Y = AX$, where $A$ is a non-singular global $n \times p$ transformation matrix, $p$ is the dimension of $X$ and $n$ is the dimension of $Y$. It is assumed that $A$ is a global transformation matrix that is common to all classes. For MCE based optimization, the model parameters and, in the case of this study, the feature transformation matrix, are computed by minimizing the sigmoid loss function. The loss function is defined as shown below:

$$L_i(Y, \varphi) = \frac{1}{1 + \exp(-\gamma d_i(Y, \varphi))}$$  \hspace{1cm} (4.1)

where $i$ is the model, $\gamma$ is a constant $\geq 1$ and $d_i$ is the misclassification measure. As explained in Chapter 2, the misclassification measure $d_i$ is a measure of the distance between the hypothesized model and the closest competing model as shown again in (4.2).

$$d_i = -g_i(Y, \varphi) + \max_{j \neq i} g_j(Y, \varphi)$$  \hspace{1cm} (4.2)

where $g_j(Y, \varphi)$ is the discriminant function of class $C_i$ and is defined as

$$g_j(Y, \varphi) = \log P_j(Y/\varphi)$$  \hspace{1cm} (4.3)
where $P_j(Y/\phi)$ is the class conditional probability. Assuming Hidden Markov Models are used to model the class distributions, the class conditional probability is defined as

$$P_j(Y/\phi) = \log \left( \sum_{j} \prod_{t=1}^{T} a_{j,q_{t-1}q_{t}} b_{j,q_{t}} (y_t) \right)$$  \hspace{1cm} (4.4)$$

where $i$ is the model, $\pi_{j,q_0}$ is the initial state probability, $q_0$ is the initial state, $a_{j,q_{t-1}q_{t}}$ is the transition probability from state $q_{t-1}$ to $q_t$ and $b_{j,q_{t}} (y_t)$ is the output distribution of state $q_t$, $t$ is the current frame and $T$ is the total number of frames in the speech utterance. Each state in the HMM is modeled using mixtures of Gaussians hence the output state distribution is represented as

$$b_{j,q_{t}} (y_t) = \sum_{k=1}^{K} c_{k,q_{t}}^j \mathcal{N}(y_t; \mu_{k,q_{t}}^j, \Lambda_{k,q_{t}}^j)$$  \hspace{1cm} (4.5)$$

where $c_{k,q_{t}}^j$ is the weight of each Gaussian mixture for model $j$ and mixture $k$, $\mathcal{N}(\cdot)$ denotes a normal distribution with mean $\mu_{k,q_{t}}^j$ and diagonal covariance $\Lambda_{k,q_{t}}^j$ and $K$ is the total number of mixtures. Thus the set of model parameters that needs to be estimated is $\phi = \left\{ c_{k,q_{t}}^j, \mu_{k,q_{t}}^j, \Lambda_{k,q_{t}}^j, A \right\}$, which includes the Gaussian mixture parameters and the transformation matrix. The transition probability between states is not considered because it is known to not change very much due to MCE training [29]. The parameters of set $\phi$ are estimated using gradient probability descent (GPD) optimization as shown in (4.6).
\[ \varphi_{\tau+1} = \varphi_{\tau} - \varepsilon \left( \frac{\partial L_i(Y; \varphi)}{\partial \varphi} \right)_{\varphi = \varphi_{\tau}} \]  
(4.6)

where \( \tau \) is the \( \tau^{th} \) iteration, \( \varepsilon \) is the learning rate. The derivative of the model loss is determined as shown in Juang et al. [10] and also shown below

\[ \frac{\partial L_i(Y, \varphi)}{\partial \varphi} = \frac{\partial L_i}{\partial d_i} \frac{\partial d_i}{\partial g_j} \frac{\partial g_j}{\partial \varphi} \]  
(4.7)

where

\[ \frac{\partial L_i}{\partial d_i} = \gamma L_i(1 - L_i) \]

\[ \frac{\partial d_i}{\partial g_j} = \begin{cases} -1 & \text{if } j = i \\ 1 & \text{if } j \neq i \end{cases} \]  
(4.8)

and

\[ \frac{\partial g_j}{\partial \varphi} = \sum_{t=1}^{T} \delta(q_t - j) \frac{\partial \log b_{j, q_t}(y_t)}{\partial \varphi} \]

and \( L_i \) is used instead of \( L_i(Y; \varphi) \) for convenience. \( Y = \{y_1, y_2, \ldots, y_T\} \) is the transformed feature vector \( y_t = [y_{t_1}, y_{t_2}, \ldots, y_{t_n}] \) is the feature vector extracted at time \( t \) and is of dimension \( n \). For update expressions of mean, variance and mixture weights refer Chapter 2 or Juang et al [10], Chengalvarayan et al. [17] or McDermott [29]. Note, the parameter updates given in Chapter 2 and papers [10], [17] & [29] have been derived under the assumption that the covariance matrices for the Gaussian mixtures are diagonal. Here we derive the expression for updating the feature transformation matrix under the same assumption of diagonal covariance.
It is a well known fact that linear transformation of a Gaussian, where the full covariance matrix is considered, does not modify the Gaussian distribution but only scales it by the determinant of the transformation matrix as shown in \((4.9)\), in which case no computation of an optimal transformation matrix can be done and the effect of the transformation matrix is absorbed into the mean and variance updates.

\[
N\left( y_t; \mu_{j,k,q_t}^j, \Sigma_{j,k,q_t}^j \right) = \frac{1}{(2\pi)^{n/2}} \exp \left\{ -\frac{1}{2} \left( y_t - \mu_{j,k,q_t}^j \right)^T \Sigma_{j,k,q_t}^j \left( y_t - \mu_{j,k,q_t}^j \right) \right\}
\]

\[
= \frac{1}{(2\pi)^{n/2}} \exp \left\{ -\frac{1}{2} \left( A x_t - A \mu_{j,x,k,q_t}^j \right)^T A \Sigma_{j,k,q_t}^j A^T \left( A x_t - A \mu_{j,x,k,q_t}^j \right) \right\}
\]

\[
= \frac{1}{|A|} N\left( x_t; \mu_{j,x,k,q_t}^j, \Sigma_{j,x,k,q_t}^j \right)
\]

\[(4.9)\]

where \( \mu_{j,k,q_t}^j = A \mu_{j,x,k,q_t}^j, \Sigma_{j,k,q_t}^j = A \Sigma_{j,x,k,q_t}^j A^T \) are the transformed mean and full covariance, while \( \mu_{j,x,k,q_t}^j \) and \( \Sigma_{j,x,k,q_t}^j \) are the mean and full covariance of the original features \( X \).

Since it is common practice to consider diagonal covariance Gaussians \( \Lambda_{j,k,q_t}^j \) in order to reduce computational complexity and for reliable estimates, the transformed Gaussian distribution should be written as shown in (4.10).
\[ N\left(y_t; \mu_{k,q_t}^j, \Lambda_{k,q_t}^j\right) = \frac{1}{(2\pi)^{n/2} \left| \text{diag}\left\{ A\Lambda_{k,q_t}^j A^T \right\} \right|^{1/2}} \exp\left(-\frac{1}{2} \left( Ax_t - A\mu_{k,q_t}^j \right)^T \text{diag}\left\{ A\Lambda_{k,q_t}^j A^T \right\}^{-1} \left( Ax_t - A\mu_{k,q_t}^j \right) \right) \] (4.10)

Assuming the X-domain features have been previously diagonalized, so \( \Lambda_{k,q_t}^{j,x} \) is the diagonal covariance in the X-domain and after linear transformation with \( A \), the transformed diagonal covariance \( \Lambda_{k,q_t}^{j,x} \) is \( \text{diag}\left\{ A\Lambda_{k,q_t}^{j,x} A^T \right\} \). From the equation above we see that by explicitly considering the diagonal covariance assumption the effect of the transformation matrix is not absorbed into the mean and variance estimates.

In order to satisfy the constraint that the transformed covariance matrix remains positive definite we scale the transformation matrix as \( \tilde{A} = \log(A) \), then the mean and variance are scaled as well, shown in (4.11) and (4.12), respectively.

\[ \mu_{k,q_t}^j = A\mu_{k,q_t}^{j,x} = e^{\tilde{A}}\mu_{k,q_t}^{j,x} \] (4.11)

\[ \Lambda_{k,q_t}^j = \text{diag}\left\{ e^{\tilde{A}}\Lambda_{k,q_t}^{j,x} \left(e^{\tilde{A}}\right)^T \right\} \] (4.12)

For estimating the transformation matrix using gradient probability descent (4.8) the derivative of the output state distribution with respect to the transformation \( \tilde{A} \) is computed as follows.
\[
\frac{\partial \log\left(b_{j, q_t}(y_t)\right)}{\partial A} = \frac{1}{\sum_{k=1}^{K} c_{k, q_t}^{j}} N\left(y_t; \mu_{k, q_t}^{j}, \Sigma_{k, q_t}^{j}\right) (D1 + D2)
\] (4.13)

where

\[D1 = \frac{\partial}{\partial a_{mn}} \left[ \frac{1}{\text{diag}\left(e^{A} \Lambda_{k, q_t}^{j, x} \left(e^{A}\right)^{T}\right)^{1/2}} \right]
\] (4.14)

\[D2 = \frac{\partial}{\partial a_{mn}} \left[ -\frac{1}{2} U^{T} \text{diag}\left(e^{A} \Lambda_{k, q_t}^{j, x} \left(e^{A}\right)^{T}\right)^{-1} U \right]
\] (4.15)

where \(U = e^{A}\left(x_t - \mu_{k, q_t}^{j, x}\right)\), and \(m\) and \(n\) subscripts in (4.14) and (4.15) are elements of a matrix or vector, lowercase ‘a’ is used to represent matrix elements and matrix \(A\) has been substituted for \(\widetilde{A}\). The derivation of these equations is detailed in the Appendices A and B. Due to the transformation \(\widetilde{A} = \log(A)\), we cannot initialize the transformation matrix by an identity matrix, and instead use ML based semi-tied optimization to initialize both the transformation matrix and the model parameters.
4.2 Experimental Results

The MCE based update of the feature transformation matrix derived above was implemented on a speech corpus. First, the standard Mel-frequency cepstral coefficients (MFCC) including the energy term, along with the first and second order time derivatives were extracted from the speech signal. 39 dimensional MFCC feature vectors extracted from the speech signal were then transformed using a 39X39 dimensional global transformation matrix, the efficacy of the transformation matrix estimate using the equation in (4.13) was tested on the Aurora2 digits speech database.

4.2.1 Aurora2 Database

The Aurora2 database uses the 8440 utterances from TIDigit recordings where the digits like the digits of a telephone number are spoken by male and female speakers with different accents. These utterances are filtered using the G.712 characteristic and different noise signals at different signal to noise ratios (SNRs) are artificially added. Two training sets are available, the clean data set, which has only the TIDigits utterances with no artificial noise added to them and the multi-condition data set, where clean as well as noisy data are used for training. In this study, training was done on the multi-condition dataset; where the 8440 utterances are split into 20 subsets, corresponding to 4 noise scenarios (subway, babble, car, exhibition), at 5 SNR levels (20 dB, 15dB, 10dB, 5dB and 0dB). Similar to the Aurora experimental framework [30], a total of 12 HMM
models were trained, 11 word models for each of the digits (“one, two, ..... , nine including “oh” and “zero”) and a silence model. Each HMM had 16 states with 3 mixtures per state except the silence model which had 3 states with 6 mixtures (- the number of states and mixtures were aligned with the baseline Aurora paper [30] so when results are compared, everything else being equal, the benefit of this approach can be evaluated). The trained models were tested using 3 different test sets (sets A, B &C). In test set A, the speech samples were different from the training set, however 4 noise sequences that are the same as those used in the multi-condition training were artificially added, again at 5 different SNRs, so the noise conditions of this set match the training set. The test set B uses the same utterances as set A but the 4 noise scenarios (restaurant, street, airport, train) added at 5 different noise levels, are different from the training. Finally test set C has 2 noise scenarios added to it, one is the same as that added in Set A (viz. subway noise) and the other is a noise that is added to Set B (viz. street noise). However both the speech and the noise in Set C are filtered using the MIRS characteristic before adding the speech and noise at different SNR levels [30].

Table 4.1 shows the word accuracy on the entire test set after 9 iterations of the GPD update. The standard MCE technique gives a word error rate reduction of 14.77% and 19.66% for set A and set B compared to ML (results from the Aurora experimental paper by Hirsch et. al [30]). While MCE based feature transformation and model estimation reduces the word error rate further, to 26.09% and 27.24% compared to ML. However this approach is not robust to the linear convolution distortion present in Set C.
As shown in Table 4.2, the word error rate reduction using MCE based feature transformation is greater when evaluated on the clean test data, 50.68% and 45.89% word error rate reduction respectively for sets A and C compared to ML. (Clean data is the TIDigit utterances before the noise is added.) The relative reduction in word error rate due to the introduction of the feature transformation can be computed by comparing the word accuracy of the standard MCE model estimation approach and the MCE based feature & model estimation, and we found there was a 10% reduction in word error rate when sets A & B were considered together (test set C wasn’t considered because there was a reduction in recognition accuracy) and a 30% reduction for the clean data in Sets A & C, (since set B uses the same utterances as Set A and hence the clean data is identical to Set A). The reduction in recognition accuracy observed in Set C for both MCE estimation (column 3, Table 4.1) and MCE based model & feature transformation (column 4, Table 4.1) is because MCE, which is a discriminative training approach is not robust to distortions in noise. It is however robust to linear distortions in speech since an increase is observed in the recognition accuracy for Set C of clean data as shown in Table 4.1

<table>
<thead>
<tr>
<th></th>
<th>ML Estimation [12]</th>
<th>MCE ESTIMATION</th>
<th>MCE Based Feature &amp; Model Estimation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set A</td>
<td>87.81</td>
<td>89.61</td>
<td>90.99</td>
</tr>
<tr>
<td>Set B</td>
<td>86.27</td>
<td>88.97</td>
<td>90.01</td>
</tr>
<tr>
<td>Set C</td>
<td>83.77</td>
<td>82.04</td>
<td>81.68</td>
</tr>
</tbody>
</table>

TABLE 4.1 Comparing the average word accuracy for ML, MCE and the MCE based Joint Optimization for the entire test set
4.2 below. Another reason the MCE based model & feature transformation approach did poorly on the Set C with noisy speech data, is because when assuming diagonal covariance matrices the correlation information is simply ignored and hence lost since the feature vectors are not decorrelated or represented by independent vectors as described in Chapter 3.

To compare the MCE based feature & model estimation approach to the semi-tied approach (which is an ML-based feature and model optimization approach developed by Gales [15]), we present the results for ML estimation [30] and semi-tied ML estimation in Table 4.3. We see for noisy data, that the performance actually falls when using semi-tied for Sets A and C, the increase in performance for set B defined in terms of word error rate reduction is 13.55%. A similar comparison between MCE and MCE w/ feature transformation (i.e comparing columns 3 & 4 of Table 4.1) shows a reduction in performance for Set C, but an increase in performance for Set A and Set B and in terms of word error rate it is a reduction of 13.29% and 9.43%. Doing the same comparison for clean data (columns 4 & 5 of Table 4.3) the reduction in word error rate was 16.89% and

<table>
<thead>
<tr>
<th></th>
<th>ML Estimation [12]</th>
<th>MCE Estimation</th>
<th>MCE Based Feature &amp; Model Estimation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set A</td>
<td>98.52</td>
<td>98.98</td>
<td>99.27</td>
</tr>
<tr>
<td>(same as Set B)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Set C</td>
<td>98.54</td>
<td>98.81</td>
<td>99.21</td>
</tr>
</tbody>
</table>

TABLE 4.2 Comparing the average word accuracy for ML, MCE and the MCE based Joint Optimization for Clean data
4.11% for the ML case compared to 28.43% and 33.61% for the MCE case. Thus we see there is a significant improvement in performance when using MCE for feature transformation than ML, in most cases. Thus even though MCE based transformation does not decorrelate the features and hence the correlation information is not explicitly represented by the diagonal covariances, the transformation based on minimizing loss and using MCE discriminative training, leads to a greater improvement in recognition performance.

4.2.2 Comparison to Other Studies Done on Aurora database

The Aurora database was developed as a standard to compare feature processing techniques that are robust to noise. In spite of its drawbacks of not having level variations and linear distortions that occur in a naturally noisy environment, there is variability due
to unmatched training and testing scenarios at different SNRs. In this section we discuss some of the other studies that have used feature vectors robust to noise, noise modeling approaches or other feature transformations for robust speech recognition and see how their approaches compare to the MCE based feature transformation.

The work by Sarma et. al. [14] combined different feature sets (including temporal and spectral processing feature sets) using ANN non-linear transformation. They used error-back propagation to estimate the ANN parameters. The results of this study were presented to compare the relative deterioration of the recognition accuracy based on the SNR levels as shown in Table 4.4. Comparing these results to that obtained using MCE based feature transformation (Row 2, Table 4.4), we see the recognition accuracy on clean data are comparable, but the results of our study on noisy data aren’t as good as that of Sarma et. al.’s work, and it could be because of the lower recognition performance on Test Set C (refer Table 4.1). Results for individual test sets weren’t presented in [14] to do a better comparison.

<table>
<thead>
<tr>
<th></th>
<th>Clean</th>
<th>20 dB</th>
<th>15 dB</th>
<th>10 dB</th>
<th>5 dB</th>
<th>0 dB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-linear feature</td>
<td>99.3</td>
<td>99.1</td>
<td>98.7</td>
<td>97.3</td>
<td>93.5</td>
<td>82.5</td>
</tr>
<tr>
<td>transformation Sarma et. al. [14]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MCE based feature</td>
<td>99.25</td>
<td>98.8</td>
<td>98.2</td>
<td>95.9</td>
<td>87.8</td>
<td>60.9</td>
</tr>
<tr>
<td>and model design</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TABLE 4.4 Comparing results with Sarma et. al. ‘s work on non-linear feature extraction
Zhu et. al.[31] in their work combined MFCCs with features extracted using four different front-end processing techniques, namely, variable frame rate, peak isolation, peak-to-valley ratio locking and harmonic demodulation (for details please refer [31]). These feature processing methods were shown to be more robust to noise than the standard MFCC features. The results of this study are presented in Table 4.5.

Another study on the Aurora database was done by Droppo et. al.[32], where they used stereo data i.e., both clean and noise distorted training data to estimate correction vectors. So the mean and variances of the models are first estimated using clean speech and the noisy speech is then used to train the correction vectors, to develop a mapping between the noise cepstrum and the clean cepstrum. In order to decrease the dependence of this algorithm on the noise statistics in the training data and for better performance on unmatched conditions where the noise in the test set is different from the train set they employed Noise Mean Normalization [32]. The results of their work are compared to the present work on MCE estimation and MCE based feature transformation in Table 4.5.

<table>
<thead>
<tr>
<th></th>
<th>MCE Estimation</th>
<th>MCE Based Feature and Model Design</th>
<th>Zhu et. al. [31]</th>
<th>Droppo et. al. [32]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set A</td>
<td>89.61</td>
<td>90.99</td>
<td>89.22</td>
<td>91.49</td>
</tr>
<tr>
<td>Set B</td>
<td>88.97</td>
<td>90.01</td>
<td>89.34</td>
<td>89.16</td>
</tr>
<tr>
<td>Set C</td>
<td>82.04</td>
<td>81.68</td>
<td>88.00</td>
<td>89.62</td>
</tr>
</tbody>
</table>

TABLE 4.5 Comparing results with Zhu et. al.’s [31] and Droppo et. al.’s [32] work

It can be seen from the comparisons in Table 4.5, that MCE based feature transformation gives a better recognition accuracy than Zhu et. al.’s noise robust front
end processing techniques, except for the test C. Droppo et. al’s work [32] where they use stereo data to separately represent the noise characteristics gives a better recognition result for the matched train and test set (namely Set A results). However for the unmatched case MCE based feature transformation gives a better recognition performance.

4.3 Conclusions and Future Work

The optimization of the feature transformation matrix and the HMM models using minimum classification error shows a significantly improved performance for not only clean data, but also for artificially added noisy data. The relative reduction in word error rate was found to be larger than the semi-tied ML estimation for the matched noise case, but smaller for the unmatched case showing that feature transformation using discriminative MCE training is better than the ML approach, but loosing the correlation information affects the recognition accuracy in unmatched conditions. This approach performed better than most noise robust approaches tested on the Aurora2 database. By using diagonal elements of the covariance matrix, some of the correlation information is lost and the transformed feature vector might not be an adequate approximation to the original feature vectors and yet its performance is better than many other noise robust approaches [14], [31], [32] that were also tested on the Aurora database.

Class-dependent feature transformations have been employed in the past for effective speech recognition where different word classes like consonants and vowels etc, can be
transformed differently [15]. This approach cannot be extended to include different class dependent transformations, because the comparison of class conditional probabilities across different sub-space classes cannot be done. Another disadvantage of this approach is that it is computationally expensive since it requires matrix multiplication to transform the variance (4.14),(4.15). Due to the loss of the correlation information the use of a lower dimension transformation matrix leads to a significant reduction in recognition accuracy.

Previous studies have used MCE [33] and Minimum Verification Error (a variation of MCE [34]) based feature transformation for adaptation, but considered full covariance matrices in their update equations. Adaptation of the trained HMM models to variations in speaker or environment conditions are widely used in speech recognition. For future work, the present approach of using diagonal covariance for MCE based feature transformation can be extended for model adaptation.
CHAPTER 5

LOSS FUNCTIONS FOR MCE ESTIMATION -
BASED ON THE PARZEN WINDOW ESTIMATES
OF THE BAYES RISK

The only loss function that is most commonly used for MCE is the standard sigmoid loss function as defined in (2.7). McDermott’s work [35] showed using Parzen window Estimation that minimization of the standard sigmoid loss function could approach Bayes Risk, when certain assumptions are satisfied. The work presented in this chapter investigated other functions that satisfy the Parzen window constraints to see if they can be used to improve the recognition accuracy for speech recognition tasks.

5.1 Parzen Window Estimate of Bayes Risk

A similar exposition to that done in Chapter 2 for MCE is partly repeated here; this presentation is geared to showing that the MCE loss function can be obtained by considering the Parzen window estimate of the Bayes risk. So once again we consider the case of an M-class classification task, the goal is to develop a classifier with a decision
function \( h: X \rightarrow C \) that accurately maps a class label \( C \in \{C_1, C_2, \ldots, C_M\} \) to every feature vector \( x \in X \), where \( X \) is the feature space. Thus we seek a function \( h \) that minimizes the classification error. The Bayes classification error is defined as

\[
R = \int_X R(C_j / x)p(x)dx = \sum_{j=1}^{M} \int_X e_{ij}P(C_j / x)p(x)dx \tag{5.1}
\]

We assume that the feature vectors have a probability density \( p(x) \) and \( R(C_j / x) \) is the conditional risk defined as,

\[
R(C_j / x) = \sum_{j=1}^{M} e_{ij}P(C_j / x) \tag{5.2}
\]

where \( P(C_j / x) \) is the a posteriori probability and \( e_{ij} \) is the cost of wrongly classifying an observation \( x \) from class \( C_j \) into class \( C_i \). For MCE estimation the cost function as shown in (5.3) is used.

\[
e_{ij} = I \left( d_{ij}(x, \Lambda) \geq 0 \right) \tag{5.3}
\]

where \( I(.) \) is the indicator function and \( d_{ij}(x, \Lambda) \) is a misclassification measure that exploits the discriminant information. It is also a measure of class separability and is defined as

\[
d_{ij}(x, \Lambda) = \begin{cases} -g_j(x, \Lambda) + \max_{i \neq j} g_i(x, \Lambda) \\ \end{cases} \tag{5.4}
\]
substituting for ‘d’ from (5.4), the cost function can be rewritten as in (5.5)

\[ e_{ij} = I \left( \max_{i \neq j} g_i(x, \Lambda) + g_j(x, \Lambda) \geq 0 \right) \]  

(5.5)

If \( g_i(x, \Lambda) \) is greater than \( g_j(x, \Lambda) \) (which is the discriminant function for the hypothesized class \( C_j \)) then \( X \) is wrongly classified to \( C_i \), hence \( d_j(x, \Lambda) > 0 \) and is assigned a penalty of 1 [(5.3) & (5.5)]. So the risk defined in terms of the misclassification measure ‘d’ is

\[ R = \sum_{j=1}^{M} \int_{X} I \left( d_j(x, \Lambda) \geq 0 \right) p(C_j, x) dx \]  

(5.6)

Following along the lines of work by McDermott et. al [35] the risk can be expressed as

\[ R = \sum_{j=1}^{M} P(C_j) \int_{X_j} p(x/C_j) dx = \sum_{j=1}^{M} P(C_j) \int_{0}^{\infty} p(m_j/C_j) dm_j \]  

(5.7)

where \( X_j = \{ x \in X | d_j(x, \Lambda) \geq 0 \} \) is the set of feature vectors that are mis-classified, \( P(C_j) \approx \frac{N_j}{\sum_{j=1}^{M} N_j} = \frac{N_j}{N} \), is the apriori probability, \( N_j \) is the number of frames that belong to the class \( C_j \) and \( N \) is the total number of frames and \( m_j = d_j(x, \Lambda) \). The second equality in (5.7) is obtained by converting the probability density from the feature domain \( x \) to the misclassification measure domain \( m_j \) [35], [38]. The conversion to the
measure domain facilitates the Parzen window estimation of the density $p(m_j/C_j)$ as shown below.

$$p_{N_j}(m_j/C_j) = \frac{1}{N_j} \sum_{r=1}^{N_j} \frac{1}{h} K \left( \frac{m_j - d_j(x_r, \Lambda)}{h} \right)$$

(5.8)

where $K \left( \frac{m_j - d_j(x_r, \Lambda)}{h} \right)$ is the Parzen window or the kernel function of bandwidth $h$, centered at every data point $d_j(x_r, \Lambda)$ and $x_r$ is among the $N_j$ data samples hypothesized as belonging to class $C_j$. The resulting risk, expressed in terms of the Parzen window estimate, is:

$$R_N(L) = \frac{1}{N} \sum_{j=1}^{M} \sum_{l=1}^{\infty} \frac{1}{N_j} K \left( \frac{m_j - d_j(x_r, \Lambda)}{h} \right) dm_j = \frac{1}{N} \sum_{j=1}^{M} L \left( d_j(x_r, \Lambda) \right)$$

(5.9)

where $L(\cdot)$ is the loss function. This empirical risk will approach Bayes risk as the Parzen window estimates converge to the true density, which can be obtained by increasing the number of training samples to infinity [3]. The convergence to Bayes’ risk also depends on making the right assumption about the class conditional likelihood and on the optimization algorithm converging to the global minimum [35]. We see from (5.9) that using different kernel functions, loss functions that minimize the empirical risk $R_N(L)$ can be obtained and if the above mentioned assumptions are satisfied the empirical risk could approach the Bayes’ risk.
5.2 Standard Sigmoid Loss Function as a Parzen Window Estimate

For Parzen window estimation the kernel functions that can be used have to satisfy the following two constraints [3],

\[ K(u) \geq 0 \]
\[ \int K(u) du = 1 \quad \forall u \]  \hspace{1cm} (5.10)

To obtain the sigmoid loss function that is typically used in MCE,

\[ L(d) = \frac{1}{1 + e^{-\gamma d}} \]  \hspace{1cm} (5.11)

the kernel function shown in (5.12) is used.

\[ K(u) = \frac{e^u}{(1 + e^u)^2} \]  \hspace{1cm} (5.12)

where \( u = \left\lfloor \frac{m_j - d_j(x_r, \Lambda)}{h} \right\rfloor \) and \( \gamma = \frac{1}{h} \). This kernel function was obtained by simply taking the derivative of the loss function (5.11). It is straightforward to see that this function is positive for \( u \in [0, \infty] \), thus satisfying the first kernel constraint given in (5.10). The equation below shows that the kernel defined in (5.12) satisfies the second kernel constraint as well.
\[
\int_{u=-\infty}^{\infty} K(u) \, du = \int_{u=-\infty}^{\infty} \frac{e^u}{(1 + e^u)^2} \, du = \frac{1}{(1 + e^u)} \bigg|_{-\infty}^{\infty} = 1 \tag{5.13}
\]

Thus minimizing the standard sigmoid loss function minimizes the classification risk and this classification risk could approach the Bayes risk, provided the assumptions described in Section 5.1 are satisfied. The kernel for the standard sigmoid loss function (5.12) is not the only function that satisfies the Kernel function. In the following section we explore other functions and test their performance on a speech database to see if they improve recognition performance.

### 5.2 Development of Other MCE Loss Functions

#### 5.2.1 Savage Loss

One choice for a loss function is the Savage loss (5.14) introduced by Shirazi et. al. [36].

\[
L(d) = \frac{1}{\left(1 + e^{-\gamma d}\right)^2} 
\tag{5.14}
\]

Shirazi et.al. [36] showed using savage loss and probability elicitation from statistics, that a bounded loss function, with a convex risk performs classification better than convex loss functions like least squares, exponential loss (used for boosting), hinge loss (used in support vector machines) and logistic regression. (Shirazi et. al. showed this for a binary
classification task). Since the Savage loss looks very similar to the standard sigmoid loss function, in this section we investigate the performance of Savage loss on a multi-class speech classification task, by formulating it as a Parzen Window Estimation problem of the Bayes risk.

The kernel of Savage loss for Parzen window estimation, determined by taking the derivative of savage loss (5.14) is shown below

\[ K(u) = \frac{2e^u}{(1+e^u)^3}. \]  

(5.15)

where \( \frac{1}{\lambda} = \gamma \). The kernel has positive values for all \( u \in (-\infty, \infty) \) and \( \int_{-\infty}^{\infty} K(u)du = 1 \) thus satisfying the kernel conditions and showing that the risk minimized using savage loss could approach Bayes risk. A plot comparing the kernels of the standard sigmoid and that of Savage loss is shown in Fig. 5.1. It can be observed from the figure that the kernel for Savage loss has a steeper gradient i.e. the width of the kernel of the Savage loss function is smaller than that for the standard sigmoid, and the magnitude is increased, because the area under both the curves is 1. The effect of increasing the magnitude but reducing the width/ range of the kernel is investigated by testing the performance of the Savage loss on a speech recognition task and the results of this evaluation are presented in Section 5.3.
5.2.2 Gaussian Kernel

A widely used kernel function for Parzen window estimation is the Gaussian Kernel as shown below.

\[
K(u) = \frac{\gamma}{\sqrt{2\pi}} e^{-\frac{1}{2}(\gamma u)^2}
\]  

(5.16)

The integral of a Gaussian function does not have a closed form solution, hence the loss function for a Gaussian kernel cannot be determined. For estimation of the model parameters based on MCE, the GPD optimization routine (2.11) is used and it requires only the derivative of the loss function which is nothing but the kernel, hence it is not necessary to determine the loss function for MCE estimation. MCE estimation using the Gaussian kernel was evaluated on a Speech recognition task and the results are presented in Section 5.3.

5.2.3 Bounded and Smoothed Approximation of the Hinge Loss

Another loss function was developed based on the hinge loss function, commonly used in SVMs and introduced by J. Li. et. al. [37].

\[
L(d) = d^* \frac{1}{1 + e^{-\gamma d}}
\]

(5.17)
The above is the loss function for MCE that was developed by J. Li et. al.[37] where d is the misclassification measure and $\gamma$ is the inverse of the kernel bandwidth h and $\geq 1$. The function (5.17) does not satisfy the bounded nature constraint required for Parzen window estimation. If the misclassification measure, d, is bounded, as shown in (5.18) below, the loss function would satisfy both the kernel constraints (5.10).

$$L(d) = (d*U)*\frac{1}{1+e^{-\gamma d}}$$

$$U = \begin{cases} 
1 & -\rho \leq d \leq \rho \\
0 & \text{elsewhere}
\end{cases}$$

(5.18)

where $[-\rho, \rho]$ defines the range of d

Since d can have negative values and the loss function can take on values between 0 and 1 only; we scaled the misclassification measure d in (5.18) to be positive in the range $[-\rho, \rho]$ and the scaled equation is shown in (5.19).

$$L(d) = \left(\frac{1}{s}d + v\right)*U*\frac{1}{1+e^{-\gamma d}}$$

(5.19)

where $1/s$ is the slope and v is the intercept. The loss function defined in (5.19) is discontinuous and optimization methods cannot be used to minimize it. Thus we approximate the piece-wise linear portion of the loss function above with another sigmoid as shown in (5.20).

$$L(d) = \left(\frac{1}{1+e^{-\gamma d/n}}\right)*\frac{1}{(1+C)}*\left(\frac{1}{1+e^{-\gamma d}}+C\right)$$

(5.20)
where $\gamma_1$ is a constant and $n$ is a constant positive integer. In (5.20), the constant $C$ is added to the second sigmoid to control the value of the loss function at $d=0$, and the scaling with $\frac{1}{(1+C)}$ is to ensure that the value of $L(d)$ is $\leq 1$. To determine the gradient for GPD, the derivative of the loss function (5.20) is calculated below.

$$\frac{\partial L(d)}{\partial d} = \frac{\gamma_1}{n} * \frac{1}{\left(1 + e^{-\gamma_1 d/n}\right)^2} * \frac{1}{(1+C)} * \frac{1}{\left(1 + e^{-\gamma d}\right) + C} + \gamma \frac{1}{\left(1 + e^{-\gamma_1 d/n}\right)} * \frac{1}{\left(1 + e^{-\gamma d}\right)^2}$$

(5.21)

The derivative obtained above can be considered as the kernel of a Parzen window estimate because it satisfies both the constraints of the kernel (5.10), i.e. the function is positive for all values of $u$ and the integral of the above kernel evaluated between the limits of $u = \infty$ and $u = -\infty$ gives 1, where $u = \left\{\frac{m_j - d}{h}\right\}$. When $n=1$, $C=0$ and $\gamma_1 = \gamma$, this loss function is the same as the savage loss. However with the introduction of these extra variables and generalizing the Savage loss, the shape and form of the kernel can be manipulated. Fig. 5.1 shows a plot of the kernels of the standard sigmoid loss ($\gamma = 1.3$) compared to Savage loss (Section 5.2.1 and $\gamma = 1.3$) and the generalized Savage loss function developed in this section (with the variables defined as $C=0.5$, $n=3$, $\gamma_1 = 0.9$ and $\gamma = 1.3$).
It is observed from the Fig. 5.1, that the standard loss function weighs the tokens near the decision boundary (d=0) most heavily, with a diminishing effect on the model parameters for tokens far away from the decision boundary. There is an optimal value of gamma that gives the best recognition performance for the standard sigmoid loss function and either using a higher value or a lower gamma decreases the recognition accuracy. If a higher value is used the kernel becomes steeper and the range of tokens considered for training is reduced, leading to a reduction in the recognition accuracy. Decreasing the value of gamma, in the case of the standard sigmoid, increases the width of the kernel, on the side of d>0 as well. Thus using a lower gamma value increases the range of tokens considered for training, but including misclassified tokens with large positive d, leads to
reduced performance because such tokens are outliers, due to either mis-labeling or mis-pronunciation and lead to erroneous training. A simple heuristic suggested by McDermott et. al. [29] for determining the value of gamma is to chose $\gamma$ such that $L(\sigma_d) = 0.9$, where $L$ is the standard sigmoid function (5.11).

The Savage loss function (represented by the dotted line in Fig. 5.1) has a steeper slope and is shifted to the right and hence weighs tokens a bit removed from the decision boundary most heavily. The range of $d$ values considered under this kernel is reduced as well, especially on the side of $d<0$ (which is the side that the tokens are correctly classified). By using a more generalized form of the Savage loss, we have increased flexibility in controlling the shape and form of the kernel. Of the kernels considered, the kernel of the new generalized Savage loss function has the widest width on the side of the correctly classified tokens only, and assigns maximum weight to the tokens at the decision boundary.

It has been shown [38], [37], [41] that increasing the margin for tokens that have been correctly classified i.e. pushing the correctly classified tokens well away from the decision boundary, so as to remove ambiguity and improve robustness in classification, leads to better generalization ability. As shown by [38], this could translate to shifting the loss function to the left or increasing the width of the kernel (as shown here for the case of the generalized Savage loss) but only on the side of $d<0$ - leads to improved performance.

By following the Parzen window estimate for Bayes risk, we can also incorporate the shifting margin concept (i.e shifting the loss function to the left), introduced by Yu et. al. [38], [39], into the new loss function. Yu et. al. used a cost function as shown in (5.22).
\[ e_{ij} = I(d_j(x, A) \geq -\rho) \quad (5.22) \]

Using this cost function and the same exposition for Parzen window estimation shown previously in Section 5.1, (explained in more detail in Chapter 6 and [38]), the loss function defined in (5.23) is obtained.

\[
L(d) = \frac{1}{1 + e^{-\gamma(d+\rho)}} \quad (5.23)
\]

where \( \rho \) is the margin. Yu et al. [38], [39] in their work showed that increasing the margin by incrementally shifting the loss function to the left as shown in (5.23), leads to improvement in performance. A similar shifting of the margin can be introduced for the new generalized Savage loss function by using the kernel for the generalized savage loss defined in (5.21) and the cost function defined in (5.22).

\[
L(d) = \frac{1}{(1+C) \left[ e^{-\gamma_0(d+\rho)/n} \right]^{\frac{1}{n}} \left[ 1 + e^{-\gamma(d+\rho)} \right] + C} \quad (5.24)
\]

5.3 Experimental Results

The newly introduced loss functions were evaluated on the Aurora2 connected digits database. The Aurora2 database uses 8440 utterances from TIDigit recordings, which are filtered using the G.712 characteristic and to which different noise signals at different
SNRs are added. There are 2 sets of data available for training, the “clean” set trains models using data that does not have any noise added to it, while the other set called “multi”, trains models with clean data as well as with data that has noise added it. For this evaluation the “clean” data with no added noise was used for training. 11 HMM word models for the 9 digits, “oh” and ”zero” with 16 states per model were trained. Initially 1 mixture/state was used and then incremented to the standard 3 mixtures/state [30]. The trained models were tested on data that has both clean and noise added to it. The results tested on clean data are only presented because the results for data with noise were very poor and didn’t have a consistent upward trend in the accuracy with every iteration. This is because the models weren’t trained on the noise data and it is known that classification techniques as presented here where the margin (or the mis-classification measure) is maximized are sensitive to noise.

5.3.1 Generalized Savage and Savage Loss

The evaluation of the Generalized Savage loss and Savage loss on a speech recognition task and the determination of the variables for the loss function are presented in this section. The variables for the generalized Savage loss function were determined using simple heuristics, for instance we chose \( \gamma \) such that \( L(\sigma_d) = 0.9 \), where \( L \) is the standard sigmoid function (5.11) and \( \sigma_d \) is the standard deviation of \( d \). The different values considered for the other variables of the generalized Savage loss and the corresponding recognition performance on the “clean” data of the Aurora2 database have
been tabulated in Table 5.1. Decreasing the value of $\gamma_1 \& \gamma$ increases the width of the kernel and vice versa. It was found that $\gamma = 1.3, \gamma_1 = 0.9$, was the best compromise to extend the range of the kernel on the side of $d>0$. The values for $\gamma_1$ and $n$ were chosen to maximize the range of $d$ on the left side. The value of $C$ determines where the loss function has the steepest slope and since we would like the Kernel, (which determines the slope of the loss) to be maximum at the decision boundary we choose $C$ to be 0.5, i.e. the value of the loss function at $d=0$ (the decision boundary). Overall, the best performance was observed with $\gamma_1 = 0.9, \gamma = 1.3, n = 3, C = 0.5$.

<table>
<thead>
<tr>
<th>$\gamma$</th>
<th>$\gamma_1$</th>
<th>$n$</th>
<th>$C$</th>
<th>String Acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.3</td>
<td>0.9</td>
<td>3</td>
<td>0.5</td>
<td>96.98%</td>
</tr>
<tr>
<td>1.3</td>
<td>0.7</td>
<td>3</td>
<td>0.5</td>
<td>96.8%</td>
</tr>
<tr>
<td>1.3</td>
<td>1.3</td>
<td>3</td>
<td>0.5</td>
<td>96.8%</td>
</tr>
<tr>
<td>2</td>
<td>0.9</td>
<td>3</td>
<td>0.5</td>
<td>96.58%</td>
</tr>
<tr>
<td>1</td>
<td>0.9</td>
<td>3</td>
<td>0.5</td>
<td>96.88%</td>
</tr>
<tr>
<td>1.3</td>
<td>0.9</td>
<td>3</td>
<td>0.2</td>
<td>95.93%</td>
</tr>
<tr>
<td>1.3</td>
<td>1.3</td>
<td>3</td>
<td>0.8</td>
<td>96.75%</td>
</tr>
<tr>
<td>1.3</td>
<td>0.9</td>
<td>4</td>
<td>0.8</td>
<td>96.65%</td>
</tr>
</tbody>
</table>

TABLE 5.1 Comparing the recognition accuracy for different values for the various parameters in the new generalized Savage loss function

The results for clean data obtained with the generalized Savage loss function are compared with the performance using ML estimation, the standard MCE estimation and MCE with Savage loss, in Table 5.2. For the standard MCE (5.11) the parameter $\gamma = 1.3$ was used. The Savage loss system with $\gamma = 1.3$ gave lower accuracy than the standard loss; however the generalized Savage loss function (column 5 of Table 5.2) gave a significant improvement in performance.
The generalized Savage loss function gave an 82.2% reduction in string error rate over the ML estimate and 21.4% improvement over standard MCE. The MCE with generalized Savage loss used a frame-level misclassification measure (introduced by J. Li et. al. [37]) and epoch -based margin shifting (introduced by Yu et. al.[39]), while the MCE with standard loss and Savage loss used a string-level misclassification measure with no margin shifting.

In order to deconstruct the results, we first compared the performance of MCE using the string-level misclassification measure and the frame-level misclassification measure. Though as per the theoretical analysis the normalization by the number of frames is not required it has been practically necessary to do this [35]. The string misclassification measure and the frame misclassification measures are defined in (5.21) and (5.26) respectively. In the case of string misclassification (5.25), the misclassification measure ‘d’ is scaled by the total number of frames in the string being recognized, while in frame misclassification (5.26) the misclassification measure is scaled by the number of frames that are different between the hypothesized string and the closest competing string.
\[
\mathbf{d}^{\text{string}}_{j}(x, \Lambda) = \frac{-g_j(x, \Lambda) + \max_{1 \neq i} g_i(x, \Lambda)}{N}
\]  
(5.25)

\[
\mathbf{d}^{\text{frame}}_{j}(x, \Lambda) = \frac{-g_j(x, \Lambda) + \max_{1 \neq j} g_j(x, \Lambda)}{N_{j}}
\]  
(5.26)

The results comparing frame and string misclassification as tabulated in Table 5.3, show only a 1.43% reduction in the string error rate. Thus suggesting that using frame misclassification is not very beneficial to recognition accuracy.

<table>
<thead>
<tr>
<th>% String Acc.</th>
<th>MCE Standard loss String-level misclassification</th>
<th>MCE Standard loss Frame-level misclassification</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>96.5</td>
<td>96.55</td>
</tr>
</tbody>
</table>

TABLE 5.3 Using the standard MCE to compare the effect of the string-level vs frame level misclassification.

Next we evaluate the effect of margin shifts with results tabulated in Table 5.4. We evaluated the new loss function using a frame-level misclassification measure with and without the margin shifts and the best performance presented in Table 5.4 was observed at a margin \( \rho \) of 5. There is a 8.9% reduction in the string error rate over MCE with no margin shift (Column 2).
<table>
<thead>
<tr>
<th>% String Acc.</th>
<th>MCE w/generalized Savage loss Frame-level misclassification no margin shift</th>
<th>MCE w/generalized Savage loss Frame-level misclassification Margin shifting</th>
</tr>
</thead>
<tbody>
<tr>
<td>96.98</td>
<td></td>
<td>97.25</td>
</tr>
</tbody>
</table>

TABLE 5.4  Evaluating the effect of shifting margin on MCE with the generalized Savage loss using frame-based misclassification measure with and without the shifting margin (the results in the 3rd column are the results when the margin was 5).

Thus it was observed that the increase in recognition accuracy observed in Table 5.1, was due in part to the use of generalized Savage loss function and in part due to the margin shifts.

Finally the number of mixtures/state was increased to 3 as is normally considered for HMM models trained using Aurora2 database [30], and a string accuracy of 98.25% was observed, using the generalized Savage loss, as shown in Table 5.5,. The string accuracy obtained using the new loss function with epoch-based margin shift was 98.43%. We started with a margin=0 and incremented by 1 every 4 iterations and obtained the best performance at a margin of 9. The reduction in string error rate over the standard loss was 20.3%, which is about the same improvement noticed for 1 mixture/state case (refer to Table 5.2).
The MCE estimation implemented using the Gaussian kernel and the results using 11 HMM models with 1 mixture/state are presented in Table 5.6.

<table>
<thead>
<tr>
<th>% String Acc.</th>
<th>MCE w/standard loss</th>
<th>MCE w/Gaussian Kernel $\gamma=1.3$ w/ margin shifts</th>
</tr>
</thead>
<tbody>
<tr>
<td>96.5</td>
<td>95.63%</td>
<td></td>
</tr>
</tbody>
</table>

TABLE 5.6  Comparing MCE using standard loss with MCE using Gaussian Kernel with margin shifts.

It is seen that there is a reduction in string accuracy compared to the standard MCE sigmoid loss function. It was noticed that the Gaussian kernel with $\gamma=1.3$ had a very narrow width, and in order to increase the width to something comparable to the standard sigmoid kernel function’s width, a value of $\gamma$ that is less than 1 had to be chosen, which scaled the magnitude of the kernel exponentially and made it unsuitable for estimation. Hence this loss function was not considered any further.

TABLE 5.5  Comparing MCE using standard loss with MCE using generalized Savage loss with and without margin shifts for HMM models with 3 mixtures/state.

5.3.2 Gaussian Kernel
5.4 Conclusions

New loss functions were introduced using the Parzen Window estimate of the Bayes risk. Though a reduction in recognition accuracy was observed when using the Savage loss and the loss function based on Gaussian kernel, the generalized Savage loss function obtained better recognition performance over MCE with standard loss. We tested the generalized Savage loss function using a frame-level misclassification measure instead of a string-level measure and found some improvement in performance, however observed greater improvement with incremental shifts in margin. Comparing the kernels of these different loss functions and relating the form of the kernel to their respective recognition accuracy it was observed experimentally that increasing the range of the loss function on the side of the correct classification i.e. the side of negative ‘d’, but not on the side of the mis-classification i.e. positive ‘d’ improves recognition accuracy. This observation is corroborated theoretically by the Vapnik and Chervonenkis theory from the large margin classification field of study explained in the following chapter.
CHAPTER 6

LARGE MARGIN MINIMUM CLASSIFICATION ERROR (LM-MCE)

The previous chapters have shown that MCE minimizes the empirical error rate and it improves the recognition accuracy of the classification system on the training data. By using training sets that are similar to the test set, the improved performance on the training set can be reflected in the test set. However differences exist between training and test sets, and the generalization ability of a system to perform just as well on a test set is improved with discriminative training, where the distance between competing classes is maximized while at the same minimizing the risk. Advances in the field of machine learning, particularly relating to VC theory (Vapnik and Chervonenkis theory)[40] have shown that the classification risk on a test set has an upper bound that is the sum of the empirical error rate on the training set and a VC-confidence term, that is inversely dependent on the margin. Thus increasing the margin, can lead to a reduction in the VC-confidence term and consequently reducing the test risk.

X. Li et. al. [41], in their work on large-margin classification, showed that constrained optimization of the margin, between competing models, using only the correctly classified tokens leads to better recognition accuracy than MCE based discriminative
training. Since they were maximizing the margin only for correctly classified tokens, they initially trained the HMMs using the standard ML followed by MCE training to obtain very low training set error rates, so the relatively fewer number of wrongly classified tokens could be discarded.

Yu et. al. [38], using the Bayes risk formulation, developed a large-margin MCE (LM-MCE) approach that incorporated the margin (which they defined as the distance between correctly classified tokens and the decision boundary) into the loss function. They used the general MCE training to decrease the empirical error rate while maximizing the margin as well. Their study showed that increasing the margin between correctly classified samples and the decision boundary led to an increase in recognition performance over the standard MCE approach.

Another study by J. Li et.al.[37] also incorporated the margin into the loss function based on the hinge loss function most commonly used in Support Vector Machines (SVMs). Though the developed function was not bounded and hence susceptible to outliers (caused by mislabeling, mis-pronunciation etc.), they showed considerable improvement in overall classification performance.

Building on the positive results of the aforementioned works, we explored a new loss function that minimizes the empirical risk and maximizes the margin of the correctly classified samples. We show in this chapter that minimization of the empirical risk, based on this loss function, is bounded and could converge to an estimate of the Bayes risk, provided certain assumptions are satisfied. This new loss function, called the sum of shifted sigmoids loss, is an extension to the function developed by Yu et. al [38], where the margin is built into the optimization routine and hence does not require prior MCE
training, as in the case of X. Li et. al.’s work [41]. The new development we have introduced does not require manual incremental adjustment as in the case of Yu et. al.’s work [38], the error is minimized while the margin is maximized in an optimal way. Once the variables have been initialized, no further adjustment of the margin is required and optimization can be done similar to the MCE approach using gradient probability descent (GPD). In this chapter the development of the new sum of the shifted sigmoids loss function using the Parzen Window approximation of the Bayes risk is explained. The recognition results from minimizing the new loss function are tabulated and compared to the results using the standard sigmoid and the LM-MCE loss function developed by Yu et. al.[38].

6.1 LM-MCE Using Manually Incremented Margin Shifts

Similar to the formulation of the Parzen window estimation of Bayes risk in Section 5.2, the case of an M-class classification task is considered, and in order to accurately map every feature vector X to its corresponding class label \( \overline{C} \) the Bayes classification error defined in (6.1) is considered.

\[
R = \int_X R(C_i/x)p(x)dx = \sum_{j=1}^{M} e_j \int_X p(C_j/x)p(x)dx
\]  

(6.1)

The cost function considered in MCE is shown in (6.2).
where $1(.)$ is the indicator function and $d_j(x,\Lambda)$ is a misclassification measure that exploits the discriminant information. Thus all tokens for which $d$ is positive are misclassified and thus given a weight of 1.

Inspired by the positive results of the large-margin classifiers, where improved recognition performance was observed by increasing the margin of the correctly classified samples that are close to the decision boundary, Yu et al. [38] penalized the tokens that have been correctly classified, but are close to the decision boundary by using a cost function as shown below,

$$e_{ij} = l\left(d_j(x,\Lambda) \geq -\rho\right)$$  \hspace{1cm} (6.3)

where the correctly classified tokens that have $d$ in the range $[0, -\rho]$ are also assigned a cost of 1 just like the mis-classified tokens for which $d \geq 0$. Thus the cost function used in Yu et al.’s LM-MCE is

$$e_{ij} = l\left(-g_j(x,\Lambda) + \max_{1 \neq j} g_i(x,\Lambda) \geq -\rho\right)$$  \hspace{1cm} (6.4)

So the risk in (6.1) using the LM-MCE cost function (6.3) is
\[ R = \sum_{j=1}^{M} \int_{X_j} \mathbb{1}\left( d_j(x, \Lambda) \geq -\rho \right) p(C_j, x) \, dx \]

\[ = \sum_{j=1}^{M} P(C_j) \int_{X_j} p(x/C_j) \, dx \]

\[ = \sum_{j=1}^{M} P(C_j) \int_{-\rho}^{\infty} p(m_j/C_j) \, dm \quad (6.5) \]

where \( X_j = \{ x \in X \mid d_j(x, \Lambda) \geq -\rho \} \), it is the set of feature vectors that are given a weight of 1, \( P(C_j) \), is the apriori probability and \( m_j = d_j(x, \Lambda) \). The third equality in (6.5) is obtained by converting the probability density from the feature domain \( x \) to the misclassification measure domain \( m_j \) ([35], [38]). The conversion to the measure domain facilitates the Parzen window estimation of the density \( p(m_j/C_j) \) as shown below.

\[ p_{N_j}(m_j/C_j) = \frac{1}{N_j} \sum_{r=1}^{N_j} K \left( \frac{m_j - d_j(x_r, \Lambda)}{h} \right) \quad (6.6) \]

where \( K \left( \frac{m_j - d_j(x_r, \Lambda)}{h} \right) \) is the Parzen window or the kernel function of bandwidth \( h \), centered at every data point \( d_j(x_r, \Lambda) \) and \( x_r \) is among the \( N_j \) data samples hypothesized as belonging to class \( C_j \). The resulting risk, expressed in terms of the Parzen window estimate, is shown in (6.7).
\[ R_N(L) = \frac{1}{N} \sum_{j=1}^{M} \sum_{r=1}^{N} \frac{1}{h} K \left( \frac{m_j - d_j(x_r, \Lambda)}{h} \right) dm_j \]

(6.7)

\[ = \frac{1}{N} \sum_{j=1}^{M} \left( d_j(x_r, \Lambda) \right) \]

where \( L(.) \) is the loss function. This empirical risk will approach Bayes risk (under the assumptions described previously in Section 5.1).

When the kernel function defined in (6.8) is used with the MCE cost function (6.2) the standard sigmoid loss function (6.9) is obtained.

\[ K(u) = \frac{e^u}{(1 + e^u)^2} \]

(6.8)

\[ L_{\text{standard MCE}}(d) = \frac{1}{1 + e^{-\gamma(d)}} \]

(6.9)

where \( u = \left( \frac{m_j - d_j(x_r, \Lambda)}{h} \right) \) and \( \gamma = 1/h \).

If the kernel defined in (6.8) is used instead with the cost function where the margin is incorporated into it i.e. (6.3), a shifted sigmoid loss function as shown in (6.10) is obtained.

\[ L(d) = \frac{1}{1 + e^{-\gamma(d + \rho)}} \]

(6.10)

This LM-MCE loss function (6.10) can be minimized similar to the MCE loss function (6.9) using GPD. When using GPD to minimize the loss function the gradient of the loss
function is used in the update term (2.11) and Fig. 6.1 shows a comparison of the gradient of the standard MCE loss function and with that of the large margin MCE (LM-MCE) loss function with fixed margin shift (a fixed margin of $\rho = 2$ was considered for the LM-MCE function in Fig. 6.1). The standard sigmoid gradient function shows that MCE estimation assigns maximum weight to tokens near the decision boundary, with diminishing effect for larger $d$’s. So tokens that are misclassified and close to the decision boundary contribute to the update term and hence effect maximum influence on the model parameters. When the sigmoid is shifted, tokens that were correctly classified and slightly removed from the decision boundary like tokens with $d = -5$ are also considered for training. However, tokens that are misclassified like a data point at $d=1$ (represented by a circle) would not contribute to a significant update in the model parameters, thus minimizing their effect on the training of the models. Ideally all of the misclassified tokens (except outliers with large positive values of ‘d’) and the correctly classified tokens relatively close to the decision boundary should be considered for training, so that the misclassified tokens can be shifted on to the side of the correct classification and the ambiguity in the correctly classified tokens that lie close to the decision boundary is removed by shifting them further away from the boundary. In order to incorporate this ideology, Yu et. al. [38] trained the models using the sigmoid loss function with no margin shift or even a negative margin shift, for a couple of iterations and then manually shifted the margin by a small, positive and fixed amount and trained the models with the new shifted sigmoid loss function for a couple more iterations before further shifting the sigmoid further to the left. Before every shift in the margin the recognition accuracy of the trained models was computed by testing on a test data set and this iterative process of
shifting and repeated training was continued till the best recognition accuracy was observed. Training the models for a couple of iterations before applying the margin shift ensured that the misclassified tokens and tokens close to the decision boundary were given due consideration in effecting an influence on the model parameters, before the correctly classified tokens with larger ‘d’ were also considered into the training process. The iterative process of training the models and shifting by a margin increases the recognition performance up to a point and beyond that any further shift in margin decreases the recognition accuracy because the effect of the misclassified tokens on the model parameters is erased by the correctly classified tokens. Thus this iterative process needs to be tested with the test set at every iteration and stopped when the best recognition performance is obtained.

6.2 LM-MCE Using Sum of Shifted Sigmoid

The problem of manually shifting the sigmoid by a fixed margin every couple of iterations is not optimal. However, a loss function that has a wider range, especially on the side of negative d (the side of the correctly classified samples) would eliminate the need for fixed increments of the margin. Such a kernel, as represented by the thick solid line in Fig. 6.1, assigns maximum weight not only to tokens that are close to the decision boundary, but to tokens that have been correctly classified as well.
Such a kernel was obtained using the summation of shifted sigmoids as shown in (6.11).

\[ K = \frac{1}{N} \sum_{i=0}^{N} K_i \]

where \( N = 19 \), \( K_i = \frac{e^{u+\rho_i}}{\left(1 + e^{u+\rho_i}\right)^2} \) (6.11)

and \( \rho_i = \{-3,-2,-1,0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15\} \)

where \( \rho_i \) is the margin shift. The range of \( \rho_i \) can be chosen based on the range of \( d \) for the given training data. As shown in the histogram plot in Fig. 6.2, the range of the margin was chosen to extend almost the entire range of \( d \).
The curve has been shifted more to the left to consider the correctly classified samples and not the wrongly classified outlier samples to the far right of the decision boundary. This kernel satisfies the kernel conditions for Parzen Window Estimation (5.10) (shown again in (6.12) below) and hence the sum of shifted sigmoids loss function could approach the Bayes risk.

\[
K(u) \geq 0 \\
\text{and} \\
\int K(u)du = 1 \quad \forall u
\]

The second condition of the kernel constraints (6.12) requires the area under the kernel be a constant of 1, hence widening the width of the kernel so as to weigh more tokens with a
wider range of misclassification measures ‘d’, decreases the absolute value of the kernel and it was found that the absolute value of the kernel function does not affect the recognition accuracy.

The loss function for such a kernel is given in (6.13) and was obtained using a similar mathematical exposition as McDermott et. al.’s [35] and Yu et. al.’s [38] work. The kernel function as defined in (6.12) with the standard MCE cost function (6.2) was used to obtain this loss function and a plot of the same is shown in Fig. 6.3. It can be seen that this loss function (6.13) is the sum of shifted LM-MCE loss function introduced by Yu et.al. (6.10).

\[
L = \frac{1}{N} \sum_{i=0}^{N} L_i
\]

where \( N = 15, \quad L_i = \frac{1}{1 + e^{-\gamma d + \rho_i}} \) \quad (6.13)

and \( \rho_i = \{ -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 \} \)

We see from Fig. 6.3 that the above loss function (6.13) is bounded and hence not susceptible to outliers and can be minimized efficiently using optimization techniques previously developed for MCE, like Gradient Probability Descent.
6.4. Experimental Results

The LM-MCE estimation using the new sum of shifted sigmoids loss function was evaluated on the TIDigits database, consisting of 8623 utterances for training and 8700 test utterances spoken by men and women. 11 whole word Hidden Markov Models (HMMs) for the 9 digits (“one”, “two”, … “nine”) along with “oh” and “zero” were trained, with 12 states and 32 mixtures/state. These models were trained using 12 Mel frequency Cepstral Coefficients (MFCCs) plus energy features including their first and
second order time derivatives. The models were initialized using ML estimation, before applying the MCE estimation.

The string accuracy of the recognition system evaluated on the test set using ML estimation, MCE with standard loss, MCE with manual margin shifts and MCE with sum of shifted sigmoids loss function are presented in Table 6.1.

<table>
<thead>
<tr>
<th></th>
<th>ML Estimation</th>
<th>MCE Estimation w/standard sigmoid loss</th>
<th>LM-MCE Estimation w/ margin shifts</th>
<th>LM-MCE w/ sum of shifted sigmoid loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>% String Acc.</td>
<td>97.77</td>
<td>98.31</td>
<td>98.49</td>
<td>98.57</td>
</tr>
</tbody>
</table>

Table 6.1 Comparison of string accuracy using ML, MCE with standard loss, LM-MCE w/ margin shifts and MCE w/ the new sum of shifted sigmoids loss

MCE with standard loss is the conventional MCE approach that uses the loss function shown in (6.9). MCE with margin shifts as defined in (6.10) is the approach developed by Yu et. al [38], in this case we shifted the margin by 1 every 4 iterations. We had to shift by 1 instead of 0.1 as Yu et. al. [39] did because we used frame-discriminant rather than sentence discriminant. (introduced by J. Li et. al. [37] and described in Section 5.3, (5.25), (5.26)). The performance of 98.49% obtained using LM-MCE with fixed margin shifts was obtained for a margin of 3. The best recognition accuracy was obtained with the LM-MCE using the sum of sigmoids loss as defined by (6.13) which was minimized just like the standard MCE was minimized where 20 iterations of GPD optimization was run starting at a step size of 0.1 and $\gamma=1.3$. When a shorter range of the loss function was considered with $\rho_1 = \{-3,-2,-1,0,1,2,3,4,5,6,7\}$ the recognition accuracy was found to be 98.57%, as shown in Table 6.2 same as for LM-MCE with the wider range. This
suggests that not all the correctly classified tokens needs to be considered for training, but considering all of them, does not harm the recognition accuracy either. Further reducing the range of the margin to $\rho_1 = \{-3, -2, -1, 0, 1, 2, 3, 4\}$ reduces the recognition accuracy to 98.48%, thus suggesting that using a wider range of margin shifts improves the performance, but using the entire range of positive $d$ does not diminish the performance. This makes it convenient to determine the range of the margin shifts, prior to running the MCE training. The range of margin shifts will simply be the range of $d$ (excluding any outliers) as determined from the histogram of $d$.

<table>
<thead>
<tr>
<th>% String Acc.</th>
<th>Standard Sigmoid</th>
<th>Margin Range (-3 to 4)</th>
<th>Margin Range (-3 to 7)</th>
<th>Margin Range (-3 to 15)</th>
</tr>
</thead>
<tbody>
<tr>
<td>98.31</td>
<td>98.48</td>
<td>98.57</td>
<td>98.57</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.2 Comparison of string accuracy using MCE w/ the new sum of shifted sigmoids loss with different margin ranges and the standard MCE loss function.

Looking at the variation of the mean and variance of the $d$ values shows the trend of tokens shifting to the left due to LM-MCE training. After the standard MCE estimation the mean of the $d$-values for all the tokens was -7.68 while the variance was 19.50. The same values after LM-MCE with manual increments of the margin were -8.66 and 22.07. Thus we see that the histogram of the $d$ values shifted to the left while increasing the variance. The mean and variance after LM-MCE using the sum of shifted sigmoids using a range of -3 to 15 was -8.99 and 26.69 (for a range of -3 to 4 it was -8.77 and 23.48), thus there is a further shift in the $d$ histogram to the left.
Figure 6.4. Plot showing the kernel of the new loss function with different ranges for the sigmoids with respect to the histogram of the values of $d$

### 6.5 Conclusions

We have introduced a new loss function that is a shifted sum of sigmoids that optimally maximizes the margin of tokens that are correctly classified. The new loss function was developed from a formulation of the Bayes risk using Parzen window estimation. It has been shown to give improved performance on the TIDigits database and does not require any more iterations than the conventional MCE approach, and lesser number of iterations than Yu et. al.’s LM-MCE with fixed margin shifts [38]. Since it uses GPD to do the
minimization, this loss function can be easily applied to large vocabulary tasks just like the conventional MCE is applied.
CHAPTER 7

SUMMARY, CONCLUSIONS AND SUGGESTIONS FOR FUTURE WORK

This thesis presented work done on two aspects of a speech recognition system, the feature transformation aspect of the front end processing and the model design aspect. The aim of these efforts has been to improve the performance of a speech recognition system when trained with speech data that has noise added and increase the generalization ability of a speech recognition system to differences between training and test sets. These efforts used the minimum classification error criterion for optimization.

Most feature transformation efforts aim to decorrelate the feature vectors so that when characterizing these vectors with Gaussian distributions, only diagonal covariance need be used. This eliminates the need for estimating all the cross-terms in a covariance matrix that might not have sufficient training data and reduces the number of parameters that are estimated. The decorrelated feature vectors are estimates of the original feature vectors and the criterion to obtain the best estimate is not optimized to improve the recognition accuracy as is required for speech recognition. In this work an MCE based feature transformation was developed that was optimized to minimize the classification error and thus improve recognition performance. This work has corrected a mathematical anomaly
observed in all studies on MCE based feature transformation published so far. Previous studies integrating the feature transformation and model design based on the MCE classification criterion, used full-covariance Gaussian distributions when estimating the feature transformation matrix and used diagonal covariance Gaussian distributions when estimating the model parameters. This discrepancy in the type of covariance matrix used to optimize the features as well as model parameters using MCE was corrected and implemented on a speech corpus resulting in significant improvement in recognition performance. Inspite of the limitations of this approach like increased computational complexity and loss of cross-correlation information; it has shown to significantly improve the recognition accuracy when tested on the Aurora2 database compared to almost all approaches that were also tested on the Aurora2 database. When compared to the feature transformation approach using the ML classification criterion viz. semi-tied modeling, the MCE based feature transformation performed better as well. Thus highlighting the significance of optimizing feature transformation using the MCE based discriminative training approach.

Aurora2 speech database that the MCE feature transformation was tested on is a database in which digits are spoken by different male and female speakers with different accents and different kinds of background noise that were artificially added. The improvement in performance on the Aurora2 database, suggests the application of this approach in feature adaptation techniques, where there are changes in the background noise or new speakers. With the corrected assumption in the feature transformation, model adaptation based on MCE can be revisited.
The latter part of this work improved the model design by developing better loss functions for minimizing the classification error and maximizing the margin using the MCE criterion. The only loss function that has been used in the past is the standard sigmoid loss function. Recently, a variant of the sigmoid loss viz. the Savage loss was introduced by Shirazi et. al. [36] for binary classification tasks, and shown to converge to the Bayes risk. In this work we showed using Parzen window estimation that the Savage loss can converge to the Bayes risk even for a multi-class classification task, provided certain assumptions are satisfied. The assumptions are that infinite amount of data be available, the minimization technique reaches global minimum and that the models characterize the speech source accurately. Further analysis on the Parzen Window Estimation approach to Bayes Risk, led to the introduction of two new loss functions, the generalized Savage loss and loss based on Gaussian kernel. These new loss functions along with the Savage loss were implemented and tested on the Aurora2 database and compared with the standard sigmoid loss. Though the Savage loss and Gaussian kernel loss gave lower recognition accuracies than the standard sigmoid, the generalized Savage loss introduced as part of this work improved the classification accuracy. The generalized Savage loss function provided flexibility in controlling the shape of the sigmoid function and with the inclusion of margin shifts introduced by Yu et.al. [38] resulted in significant improvement in classification accuracy. Comparing the kernels of these new loss functions showed that increasing the range of the kernel on the side of the negative ‘d’, i.e. optimizing the models using tokens that were correctly classified improves the recognition accuracy. This investigatory work substantiated the Vapnik and Chervonekis
theory of large margin classification, that increasing the distance between the correctly classified tokens and the decision boundary improves the recognition on the test set.

Based on the above observation a loss function for large margin MCE (LM-MCE) classification was developed which widens the range of the loss function using the sums of margin shifted sigmoids. This new loss function uses the standard GPD approach to minimization of the loss function to optimally minimize the classification error as well as maximize the margin. Large margin classification improves the generalization ability of the system to handle differences between train and test data, (like differences in pronunciation, dialects, etc.) by optimally maximizing the margin between the correctly classified tokens and the decision boundary. The formulation of this loss function is consistent with the Parzen Window estimation approach to Bayes risk and thus is an approximation to the Bayes error. When tested on TIDigits database, the performance was better than an LM-MCE approach introduced by Yu et.al. [38], that shifted the margin manually by fixed amounts.

This work lays the ground work for further development of other loss functions, some that satisfy the Parzen Window Estimation to Bayes Risk and some that do not but increase the generalization ability of a classification system. All improvements suggested for MCE, like scaling the error function based on the relevance of the word being misclassified [42], extending it for large vocabulary speech tasks [35], etc. can be done using the sum of shifted sigmoids LM-MCE approach.
APPENDIX A

The derivation for (4.14) shown again below is derived in this appendix

\[
\frac{\partial}{\partial a_{mn}} \left[ \frac{1}{\text{det} \left( \mathbf{e}^{\mathbf{A}^{\Lambda_{i,x}}_{k qt}} \right)^{1/2}} \right] = 2e^{2a_{mn}} \left( \Lambda_{i,x}^{k qt} \right)_{nn} \left( \mathbf{e}^{\mathbf{A}^{\Lambda_{i,x}}_{k qt}} \right)_{mn} \left( \mathbf{e}^{\mathbf{A}^{\Lambda_{i,x}}_{k qt}} \right)_{mm}^{-1}
\]

(A.1)

Derivation of matrix differentiation is as follows

\[
\frac{\partial}{\partial a_{mn}} \left[ \frac{1}{\text{det} \left( \mathbf{e}^{\mathbf{A}^{\Lambda_{i,x}}_{k qt}} \right)^{1/2}} \right] = -\frac{1}{2} \left[ \text{det} \left( \mathbf{e}^{\mathbf{A}^{\Lambda_{i,x}}_{k qt}} \right)^{3/2} \right] \frac{\partial}{\partial a_{mn}} \left( \mathbf{e}^{\mathbf{A}^{\Lambda_{i,x}}_{k qt}} \right)\left( \mathbf{e}^{\mathbf{A}^{\Lambda_{i,x}}_{k qt}} \right)^T
\]

(A.2)

The determinant term when expanded is
\[
\begin{align*}
\left| \text{diag} \left( e^{\tilde{A}_{i,x}} \Lambda_{k,q_t} \left( e^{\tilde{A}} \right)^T \right) \right| &= e^{2\tilde{a}_1 \Lambda_{11}} + e^{2\tilde{a}_2 \Lambda_{22}} + \ldots + e^{2\tilde{a}_n \Lambda_{nn}} \\
&= \left( e^{2\tilde{a}_1 \Lambda_{11}} + e^{2\tilde{a}_2 \Lambda_{22}} + \ldots + e^{2\tilde{a}_n \Lambda_{nn}} \right) \times \ldots \times \left( e^{2\tilde{a}_n \Lambda_{11}} + e^{2\tilde{a}_n \Lambda_{22}} + \ldots + e^{2\tilde{a}_n \Lambda_{nn}} \right)
\end{align*}
\]

So the derivative of the determinant

\[
\frac{\partial}{\partial a_{mn}} \left| \text{diag} \left( e^{\tilde{A}_{i,x}} \Lambda_{k,q_t} \left( e^{\tilde{A}} \right)^T \right) \right| = 2e^{2\tilde{a}_{mn}} \Lambda_{nn} \text{Adj} \left( \text{diag} \left( e^{\tilde{A}_{i,x}} \Lambda_{k,q_t} \left( e^{\tilde{A}} \right)^T \right) \right)_{mn}
\]

\[
= 2e^{2\tilde{a}_{mn}} \Lambda_{nn} \left| \text{diag} \left( e^{\tilde{A}_{i,x}} \Lambda_{k,q_t} \left( e^{\tilde{A}} \right)^T \right) \right| \left( \text{diag} \left( e^{\tilde{A}_{i,x}} \Lambda_{k,q_t} \left( e^{\tilde{A}} \right)^T \right) \right)^{-1}
\]

(A.3)

Thus the derivative of the reciprocal of the determinant
\[
\frac{\partial}{\partial a_{mn}} \left[ \frac{1}{\text{diag} \left( e^{\tilde{A}_{i,x}} e^{\tilde{A}} \right)^{1/2}} \right] = -\frac{1}{2} \left[ \text{diag} \left( e^{\tilde{A}_{i,x} e^{\tilde{A}} \left( e^{\tilde{A}} \right)^T} \right) \right]^{-3/2} \frac{\partial}{\partial a_{mn}} \left[ \text{diag} \left( e^{\tilde{A}_{i,x} e^{\tilde{A}} \left( e^{\tilde{A}} \right)^T} \right) \right] \\
= -\frac{1}{2} \left[ \text{diag} \left( e^{\tilde{A}_{i,x} e^{\tilde{A}} \left( e^{\tilde{A}} \right)^T} \right) \right]^{-3/2} \\
* 2e^{2\tilde{a}_{mn} A_{nn}} \left[ \text{diag} \left( e^{\tilde{A}_{i,x} e^{\tilde{A}} \left( e^{\tilde{A}} \right)^T} \right) \right] \left[ \text{diag} \left( e^{\tilde{A}_{i,x} e^{\tilde{A}} \left( e^{\tilde{A}} \right)^T} \right)^{-1} \right]_{mm} \\
= -\frac{2e^{2\tilde{a}_{mn} A_{nn}}}{\left[ \text{diag} \left( e^{\tilde{A}_{i,x} e^{\tilde{A}} \left( e^{\tilde{A}} \right)^T} \right) \right]_{mm}^{1/2}} \left[ \text{diag} \left( e^{\tilde{A}_{i,x} e^{\tilde{A}} \left( e^{\tilde{A}} \right)^T} \right) \right]^{-3/2} (A.5) 
\]
APPENDIX B

The derivation for (4.15) shown again below is derived in this appendix

\[
\frac{\partial}{\partial \tilde{a}_{mn}} \left( -\frac{1}{2} (e^\tilde{A} x_t - e^\tilde{A} \mu_{k,q_t})^T \text{diag} \left( e^\tilde{A} \Lambda_{k,q_t} \left( e^\tilde{A} \right)^T \right)^{-1} (e^\tilde{A} x_t - e^\tilde{A} \mu_{k,q_t}) \right) = \]

\[
= \left( x_t - \mu_{k,q_t} \right)^T \text{diag} \left( \Lambda X \left( X^T \right)^T \right) \left( x_t - \mu_{k,q_t} \right) \]

\[
\text{diag} \left( \Lambda X \left( X^T \right)^T \right) \left( x_t - \mu_{k,q_t} \right) \]  

The term in the derivative can be expressed as

\[
(e^\tilde{A} x_t - e^\tilde{A} \mu_{k,q_t})^T \text{diag} \left( e^\tilde{A} \Lambda_{k,q_t} \left( e^\tilde{A} \right)^T \right)^{-1} (e^\tilde{A} x_t - e^\tilde{A} \mu_{k,q_t}) = \]

\[
(x_t - \mu_{k,q_t})^T \left( e^\tilde{A} \right)^T \text{diag} \left( e^\tilde{A} \Lambda_{k,q_t} \left( e^\tilde{A} \right)^T \right)^{-1} e^\tilde{A} (x_t - \mu_{k,q_t}) \]  

\[
= (X)^T \left( e^\tilde{A} \right)^T \text{diag} \left( e^\tilde{A} \Lambda_{k,q_t} \left( e^\tilde{A} \right)^T \right)^{-1} e^\tilde{A} (X) \]

where \( x_t - \mu_{k,q_t} = X \). Expanding derivative term obtained above in terms of matrices
\[(X)^T \left( e^{\tilde{A}} \right)^T \left( \Lambda Y \right)^{-1} e^{\tilde{A}} (X) \]

\[= x_1 \left[ \begin{array}{c}
\frac{\tilde{a}_{11}}{\Lambda_{11}} \left( e^{\tilde{a}_{11}} x_1 + e^{\tilde{a}_{12}} x_2 + \ldots + e^{\tilde{a}_{1n}} x_n \right) + \frac{\tilde{a}_{21}}{\Lambda_{22}} \left( e^{\tilde{a}_{21}} x_1 + e^{\tilde{a}_{22}} x_2 + \ldots + e^{\tilde{a}_{2n}} x_n \right) \\
+ \ldots + \frac{e^{\tilde{a}_{1n}}}{\Lambda_{1n}} \left( e^{\tilde{a}_{11}} x_1 + e^{\tilde{a}_{12}} x_2 + \ldots + e^{\tilde{a}_{1n}} x_n \right) \\

\frac{e^{\tilde{a}_{2n}}}{\Lambda_{2n}} \left( e^{\tilde{a}_{21}} x_1 + e^{\tilde{a}_{22}} x_2 + \ldots + e^{\tilde{a}_{2n}} x_n \right) \\
+ \ldots + \frac{e^{\tilde{a}_{nn}}}{\Lambda_{nn}} \left( e^{\tilde{a}_{11}} x_1 + e^{\tilde{a}_{12}} x_2 + \ldots + e^{\tilde{a}_{nn}} x_n \right)
\end{array} \right] \]

\[+ \ldots \]

\[= x_2 \left[ \begin{array}{c}
\frac{\tilde{a}_{12}}{\Lambda_{11}} \left( e^{\tilde{a}_{11}} x_1 + e^{\tilde{a}_{12}} x_2 + \ldots + e^{\tilde{a}_{1n}} x_n \right) + \frac{\tilde{a}_{22}}{\Lambda_{22}} \left( e^{\tilde{a}_{21}} x_1 + e^{\tilde{a}_{22}} x_2 + \ldots + e^{\tilde{a}_{2n}} x_n \right) \\
+ \ldots + \frac{e^{\tilde{a}_{1n}}}{\Lambda_{1n}} \left( e^{\tilde{a}_{11}} x_1 + e^{\tilde{a}_{12}} x_2 + \ldots + e^{\tilde{a}_{1n}} x_n \right) \\

\frac{e^{\tilde{a}_{2n}}}{\Lambda_{2n}} \left( e^{\tilde{a}_{21}} x_1 + e^{\tilde{a}_{22}} x_2 + \ldots + e^{\tilde{a}_{2n}} x_n \right) \\
+ \ldots + \frac{e^{\tilde{a}_{nn}}}{\Lambda_{nn}} \left( e^{\tilde{a}_{11}} x_1 + e^{\tilde{a}_{12}} x_2 + \ldots + e^{\tilde{a}_{nn}} x_n \right)
\end{array} \right] \]

\[+ \ldots \]

\[= x_n \left[ \begin{array}{c}
\frac{e^{\tilde{a}_{1n}}}{\Lambda_{11}} \left( e^{\tilde{a}_{11}} x_1 + e^{\tilde{a}_{12}} x_2 + \ldots + e^{\tilde{a}_{1n}} x_n \right) + \frac{e^{\tilde{a}_{2n}}}{\Lambda_{22}} \left( e^{\tilde{a}_{21}} x_1 + e^{\tilde{a}_{22}} x_2 + \ldots + e^{\tilde{a}_{2n}} x_n \right) \\
+ \ldots + \frac{e^{\tilde{a}_{nn}}}{\Lambda_{nn}} \left( e^{\tilde{a}_{11}} x_1 + e^{\tilde{a}_{12}} x_2 + \ldots + e^{\tilde{a}_{nn}} x_n \right)
\end{array} \right]
\]

where

\[\Lambda^Y = \text{diag} \left\{ \begin{array}{c}
e^{\tilde{A}} A_{i,x}^T e^{\tilde{A}} \\
e^{\tilde{A}} A_{k,q}^T e^{\tilde{A}} \\
e^{\tilde{A}} A_{n,n}^T e^{\tilde{A}}
\end{array} \right\}
\]

\[= \begin{bmatrix}
e^{2\tilde{a}_{11}} \Lambda_{11} + e^{2\tilde{a}_{12}} \Lambda_{22} + \ldots + e^{2\tilde{a}_{1n}} \Lambda_{nn} \\
e^{2\tilde{a}_{21}} \Lambda_{11} + e^{2\tilde{a}_{22}} \Lambda_{22} + \ldots + e^{2\tilde{a}_{2n}} \Lambda_{nn} \\
\ldots \\
e^{2\tilde{a}_{n1}} \Lambda_{11} + e^{2\tilde{a}_{n2}} \Lambda_{22} + \ldots + e^{2\tilde{a}_{nn}} \Lambda_{nn}
\end{bmatrix}
\]

Now derivative of the Y-domain diagonal covariance (B.4)

\[
\frac{\partial}{\partial \tilde{a}_{mn}} \frac{1}{\Lambda_{mn}^Y} = - \frac{1}{\left( \Lambda_{mn}^Y \right)^2} \frac{\partial \Lambda^Y}{\partial \tilde{a}_{mn}} = - \frac{2e^\tilde{a}_{mn} \Lambda_{nn}^Y}{\left( \Lambda_{mn}^Y \right)^2}
\]
Consider the derivative of (B.3)(B.3)
\[
\frac{\partial}{\partial a_{mn}} \left( X^T \left( e^{\tilde{A}} \right)^T \left( \Lambda^y \right)^{-1} e^{\tilde{A}}(X) \right)
\]
\[
= x_1 \left[ \frac{\tilde{a} m_1}{\Lambda_{mm}} \frac{\partial}{\partial a_{mn}} \left( e^{\tilde{a} m_1 x_1 + e^{\tilde{a} m_2 x_2} + \ldots + e^{\tilde{a} m_{mn} x_n}} \right) + \frac{e^{\tilde{a} m_2}}{\Lambda_{mm}} \frac{\partial}{\partial a_{mn}} \left( e^{\tilde{a} m_1 x_1 + e^{\tilde{a} m_2 x_2} + \ldots + e^{\tilde{a} m_{mn} x_n}} \right) \right]
\]
\[
+ x_2 \left[ \frac{\tilde{a} m_1}{\Lambda_{mm}} \frac{\partial}{\partial a_{mn}} \left( e^{\tilde{a} m_1 x_1 + e^{\tilde{a} m_2 x_2} + \ldots + e^{\tilde{a} m_{mn} x_n}} \right) + \frac{e^{\tilde{a} m_2}}{\Lambda_{mm}} \frac{\partial}{\partial a_{mn}} \left( e^{\tilde{a} m_1 x_1 + e^{\tilde{a} m_2 x_2} + \ldots + e^{\tilde{a} m_{mn} x_n}} \right) \right]
\]
\]
\[
+ \ldots
\]
\[
+ x_n \left[ \frac{\tilde{a} m_1}{\Lambda_{mm}} \frac{\partial}{\partial a_{mn}} \left( e^{\tilde{a} m_1 x_1 + e^{\tilde{a} m_2 x_2} + \ldots + e^{\tilde{a} m_{mn} x_n}} \right) + \frac{e^{\tilde{a} m_2}}{\Lambda_{mm}} \frac{\partial}{\partial a_{mn}} \left( e^{\tilde{a} m_1 x_1 + e^{\tilde{a} m_2 x_2} + \ldots + e^{\tilde{a} m_{mn} x_n}} \right) \right]
\]
Using the expression from (B.5) the above expression simplifies to
\[
\frac{\partial}{\partial a_{mn}} \left( X^T \left( e^{\tilde{A}} \right)^T \left( \Lambda^y \right)^{-1} e^{\tilde{A}}(X) \right)
\]
\[
= x_1 \left[ \frac{e^{\tilde{a} m_1}}{\Lambda_{mm}} e^{\tilde{a} m_{mn} x_n} - e^{\tilde{a} m_1} \left( e^{\tilde{a} m_1 x_1 + e^{\tilde{a} m_2 x_2} + \ldots + e^{\tilde{a} m_{mn} x_n}} \right) \right] \frac{2e^{2\tilde{a} m_{mn} \Lambda_{nn}}}{\Lambda_{mm}}
\]
\[
+ x_2 \left[ \frac{e^{\tilde{a} m_1}}{\Lambda_{mm}} e^{\tilde{a} m_{mn} x_n} - e^{\tilde{a} m_2} \left( e^{\tilde{a} m_1 x_1 + e^{\tilde{a} m_2 x_2} + \ldots + e^{\tilde{a} m_{mn} x_n}} \right) \right] \frac{2e^{2\tilde{a} m_{mn} \Lambda_{nn}}}{\Lambda_{mm}}
\]
\[
+ \ldots
\]
\[
+ x_n \left[ \frac{e^{\tilde{a} m_1}}{\Lambda_{mm}} e^{\tilde{a} m_{mn} x_n} - e^{\tilde{a} m_2} \left( e^{\tilde{a} m_1 x_1 + e^{\tilde{a} m_2 x_2} + \ldots + e^{\tilde{a} m_{mn} x_n}} \right) \right] \frac{2e^{2\tilde{a} m_{mn} \Lambda_{nn}}}{\Lambda_{mm}}
\]
Combining the terms we get

\[
\frac{\partial}{\partial a_{mn}} (X)^T \left( \Lambda^T \right)^{-1} e^{\tilde{A}} (X) = \frac{2}{\Lambda_{mn}} \left( e^{\tilde{a}_n x_n} \right) \quad (B.8)
\]

Thus the final expression is

\[
\frac{\partial}{\partial a_{mn}} \left( -\frac{1}{2} (X)^T \left( \Lambda^T \right)^{-1} e^{\tilde{A}} (X) \right) = -\frac{x_n e^{\tilde{a}_{mn}} [e^{A X}]_m e^{2\tilde{a}_{mn} \Lambda_{nn} [e^{A X}]_m} \Lambda_{mn}^Y \Lambda_{nn}^Y - \left( \Lambda_{mn}^Y \right)^2 \quad (B.9)
\]

where the terms with the subscripts are elements of a matrix and the capitalized terms are matrices or vectors.
REFERENCES


CURRICULUM VITAE

Madhavi Vedula Ratnagiri

Education

2002-2011 PhD Electrical and Computer Engineering, 
Rutgers, The State University of New Jersey, Piscataway NJ.

1997-2000 M.S., Electrical Engineering
Boston University, Boston MA.

1994-1997 M.S., Biomedical Engineering
Indian Institute of Technology, Chennai, India.

1989-1993 B.E. Electrical Engineering
Andhra University, India

Research and Other Positions

2003-2004 Graduate Assistant, 
CAIP center, Rutgers University

2000-2001 Research Engineer, 
Songbird Hearing Inc.

1998-2000 Graduate Assistant, 
Biomedical Engineering Department, Boston University

1997-1998 Teaching Assistant, 
Biomedical Engineering Department, Boston University

Publications

Madhavi V. Ratnagiri, Biing-Hwang (Fred) Juang, Lawrence Rabiner, “Large Margin - Minimum Classification Error Using Sum of Shifted Sigmoid as the Loss Function”, submitted for publication to ICASSP 2011.


Madhavi V. Ratnagiri, Lawrence Rabiner, Biing-Hwang (Fred) Juang, “Multi-Class Classification Using a New Sigmoid Loss Function for Minimum Classification Error
(MCE)”, 9th International Conference on Machine Learning and Applications (ICMLA) 2010 IEEE Computer Society.
