Estimation of Cepstral Coefficients for Robust Speech Recognition

by

Kevin M. Indrebo, B.S., M.S.

A Dissertation submitted to the Faculty of the Graduate School, Marquette University, in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

Milwaukee, Wisconsin

August, 2008

Acknowledgement

I would first like to thank my advisor, Dr. Richard Povinelli, whose advice and support have been critical in the development of this work. My thanks also go to Dr. Michael Johnson, whose expertise has been very valuable to me. I also thank my committee members, Drs. George Corliss, Craig Struble, and Edwin Yaz for all their assistance.

Thanks go to Dr. Mohamed Mneimneh for all his help during his time as my labmate in the Knowledge and Information Discovery Lab.

I also thank Marquette University and the Department of Electrical and Computer Engineering for all support financial and otherwise. I would also like to recognize the National Science Foundation and the U.S. Department of Education for their financial support.

I am grateful to my family and all my friends for their moral support and friendship during my time as a student at Marquette University.

Abstract

This dissertation introduces a new approach to estimation of the features used in an automatic speech recognition system operating in noisy environments, namely mel-frequency cepstral coefficients. A major challenge in the development of an estimator for these features is the nonlinear interaction between a speech signal and the corrupting ambient noise. Previous estimation methods have attempted to deal with this issue with the use of a low order Taylor series expansion, which results in a rough approximation of the true distortion interaction between the speech and noise signal components, and the estimators must typically be iterative, as it is the speech features themselves that are used as expansion points. The new estimation approach, named the additive cepstral distortion model minimum mean-square error estimator, uses a novel distortion model to avoid the necessity of a Taylor series expansion, allowing for a direct solution.

Like many previous approaches, the estimator introduced in this dissertation uses a prior distribution model of the speech features. In previous work, this distribution is limited in specificity, as a single global model is trained over an entire set of speech data. The estimation approach introduced in this work extends this method to incorporate contextual information into the prior model, leading to a more specific distribution and subsequently better estimates of the features.

An introduction to automatic speech recognition is presented, and a historical review of relevant feature estimation research is given. The new estimation approach is developed, along with a method for implementing more specific prior distribution modeling, and the new feature estimation approach is evaluated on two standard robust speech recognition datasets.

Table of Contents

Table of Figures

Table of Tables

Chapter 1 Introduction

An approach to the robust estimation of features from noise corrupted speech signals for use in an automatic speech recognition (ASR) system is introduced and developed in this dissertation. This work represents a contribution to the advancement of the robustness of ASR systems in adverse environments, in which current state-of-the-art systems fail to operate satisfactorily [1, 2]. The new method for enhancement of ASR system performance in the presence of ambient noise is based on statistical estimation of the features used in traditional speech recognizers.

This chapter introduces the task of speech recognition and briefly discusses the major problems that prevent current systems from performing adequately in real world environments. It also highlights the contributions of this work and provides an outline of the dissertation.

1.1 Speech Recognition in Adverse Environments

Automatic speech recognition is defined as the task of converting an acoustic signal into a sequence of words. The signal is acquired through one or more microphones and digitally sampled. ASR systems can be used in applications such as dictation, automated call center menu navigation, and voice-activated device control.

Current ASR systems can achieve high word recognition rates (sometimes matching or exceeding 99% accuracy) in an ideal environment [3]. However, when the environmental conditions under which the system is trained do not match those under which the system is tested, the recognition accuracy often degrades significantly. Because ASR systems are expected to be useful in many different environments, by

various users, and in multiple locations, this mismatch between training and testing is quite common. Consequently, many human-computer interfaces that could benefit from the use of high quality speech recognition are unable to take advantage of current ASR systems.

Two major confounding elements that cause a mismatch between training and testing conditions are the speaker and the environment. Because every speaker's voice has individual characteristics, recognition systems trained using speech signals generated from one set of speakers do not perform as well when tested on a different set of speakers. This problem can be mitigated by speaker adaptation [4], in which speech models are built using a large set of data from training speakers and adapted with a smaller amount of data generated by the intended user(s).

This dissertation explores the effects of environmental noise on ASR system performance. Much attention has been given to this problem, and yet it remains unsolved [1, 2].

In any environment, two primary types of noise may be present: background/ambient noise and channel noise. Background noise is often modeled as additive, whereas channel noise is treated as convolutional, as described in equation (1.2). Background noise can come from many sources, including HVAC systems, vehicles, and other speakers, and can vary greatly across environments. Channel noise is defined by the reverberation effects of the room, as well as the impulse response of the microphone in use.

The digitally sampled signal that is recorded by the acquisition system, which consists of the microphone and sampling hardware, is considered to be a corrupted

version of the clean speech signal. The corrupted signal, $y(t)$, is a function of three components:

- $x(t)$ – the clean speech signal,
- $h(t)$ – the impulse response of the channel (microphone and room), and
- $n(t)$ – the ambient noise signal.

Here, *t* is the time sample index. The relationship between these components is

$$y(t) = x(t) \otimes h(t) + n(t), \tag{1.1}$$

where $\otimes$ represents the discrete convolution operation,

$$x(t) \otimes h(t) = \sum_{k=-\infty}^{\infty} x(k) h(t-k). \tag{1.2}$$

The features used in many ASR systems, known as cepstral coefficients, are computed from the power spectrum of the input signal. Applying the discrete Fourier transform to (1.1) and treating the component signals as independent stochastic processes, the relationship is given in the power spectral domain as

$$\left|Y[k]\right|^2 = \left|X[k]\right|^2 \left|H[k]\right|^2 + \left|N[k]\right|^2. \tag{1.3}$$

Here, *k* is the frequency bin index. It is common for algorithms that attempt to remove the noise from the speech signals to ignore the convolutional noise component and apply an operation known as cepstral mean subtraction (CMS) to the features extracted from the noisy signals [5]. While this approach does not completely solve the issue of convolutional distortion, it is a simple technique, and it yields some reduction of distortion caused by channel mismatch. CMS is discussed further in Chapter 3.

No technique has been universally accepted for the reduction of additive noise from a corrupted speech signal, and the degradation on the performance of an ASR system due to additive noise remains a major issue in real ASR systems. This is the problem addressed in this dissertation.

1.2 Problem Statement

The goal of this dissertation is to develop methods to increase the word recognition accuracy of a real ASR system operating under conditions of heavy background noise. To evaluate the performance improvements gained with the proposed feature compensation methods, speech recognition experiments are conducted over various adverse conditions. Small (~10 words) and medium (5000 words) vocabulary datasets are studied under a range of noises with different spectral characteristics and signal-to-noise ratios.

This task is accomplished through the development of a statistical approach for enhancement of the features extracted from speech signals. The purpose of the introduced approach is to recover the features that describe only the speech component, $x(t)$, of a corrupted signal, removing the effects of the ambient noise signal, $n(t)$. The main contributions of this work are outlined in the next section.

1.3 Contributions

Various approaches to improving the robustness of ASR systems to additive (background) noise have been studied, including speech enhancement algorithms [6-9], microphone arrays [10-12], auditory system motivated features [13, 14], adaptation of system parameters [4, 15], and feature compensation algorithms [15-18].

In this dissertation, a new feature compensation algorithm, which is named the additive cepstral distortion model minimum mean-squared error (ACDM-MMSE) estimator, is developed, based on statistical estimation theory. An optimal estimator for the cepstral coefficients, the features extracted from the audio signal, is derived using a small set of assumptions about the statistical distributions of the components of the speech and noise signals. This estimation procedure replaces the typical front end of an ASR system, improving the robustness of the features used. A block diagram of a standard front-end is shown in Figure 1-1A and compared to the new estimation front-end in Figure 1-1B. These systems are described in detail in Chapters 2 and 4, respectively.

The ACDM-MMSE estimator overcomes some problems with previous work in this area. Because traditional speech features are computed from a transformation on the logarithm of the power spectrum of the signal, the relationship between the speech and noise components is nonlinear. Thus, many of the previously studied feature estimation procedures [15, 17] make use of a Taylor series expansion, resulting in the need for multiple estimation iterations, as the expansion points used are the speech and noise powers, values that are not known *a priori*. In the new approach, a set of statistical assumptions leads to a non-iterative solution, allowing for faster computation time. Also, the ACDM-MMSE estimator works entirely in the cepstral domain. Many other approaches model the interaction between the speech and noise through the log spectral domain, so they must compute a (pseudo-) inversion of a non-square matrix, as the number of cepstral coefficients is generally less than the number of filters in the log

spectrum. This can lead to inaccuracies in the estimated features. No pseudo-inverse computation is necessary in the ACDM-MMSE estimation approach.

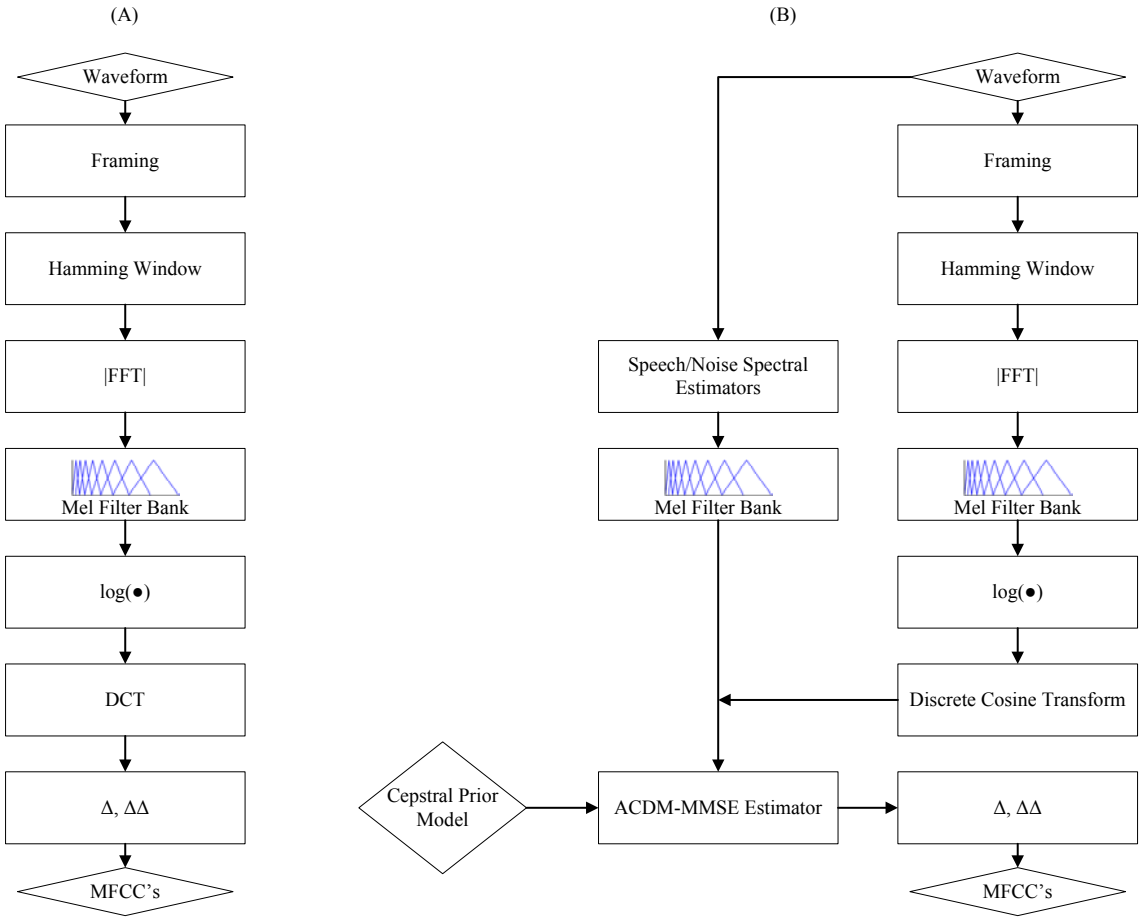

Figure 1-1. (A) A standard front-end. (B) The ACDM-MMSE estimation front-end.

The ACDM-MMSE estimator is derived using two distributions: a conditional distribution of the corrupted feature vector given the clean feature vector and a prior distribution of the clean feature vector. As in many previous estimation methods, the prior is modeled with a Gaussian mixture model (GMM). This distribution can be learned using the well-known expectation maximization (EM) algorithm [19] over all clean speech training data available. This yields a very generic prior model, however, and does

not take advantage of all possible information. A second major contribution of this dissertation is the inclusion of information that can be gained from iterative recognition passes of an ASR system. A recognition pass, as discussed in Chapter 2, yields a set of transcription hypotheses. This output information is used to adapt the prior statistical distributions of the speech features. Because temporal acoustic and language information is fundamental to the generation of the transcription by a recognition system, information that is not contained in the raw audio signal is available for use in the updated prior models. It is shown empirically in Chapter 5 that a substantial improvement in the performance of the estimator is achieved if more informative and reliable prior distributions are used.

This dissertation focuses strictly on the front-end (feature extraction unit) of an ASR system. It does not attempt to improve upon the acoustic or language modeling units, as described in Chapter 2. While some of the experiments involve signals that have corrupted by convolutional noise, this work does not attempt to model the channel distortion.

With the problem and scope of this work defined, the next section provides an outline for the rest of the dissertation.

1.4 Outline

In Chapter 2, the fundamentals of an ASR system are presented. The chapter segments a typical system into subsystems and describes each subsystem. Chapter 3 provides a comprehensive survey of the various approaches that have been studied for improving the robustness of ASR systems. In addition, an introduction to statistical estimation theory is given.

In Chapter 4, the new ACDM-MMSE estimation approach developed as a primary contribution of this dissertation is presented. Chapter 5 extends the estimator by developing methods for incorporating more information into the modeling of the prior distributions of the speech features. Experimental results on standard speech recognition datasets are presented in each chapter, demonstrating improvement in word recognition accuracy over standard baselines.

In Chapter 6, the work presented in this dissertation is summarized, and future extensions to the proposed approach are discussed.

Chapter 2 Introduction to Speech Recognition

This chapter introduces and develops the fundamental concepts of automatic speech recognition (ASR) systems. Three subsystems are described: the front-end, the acoustic modeling unit, and the language modeling unit. An emphasis is placed on the front-end signal processing, as this dissertation focuses on improving the front-end of ASR systems.

This chapter is divided into six sections, starting with an overview of ASR system organization. The following three sections describe each subsystem in turn, the front-end, the acoustic modeler, and the language modeler. Practical considerations of real and experimental systems are then discussed, followed by a summary. Table 2-1 specifies notation used in this chapter.

2.1 Overview of an ASR System

The objective of an ASR system is to translate an acoustic signal into a sequence of words, a task referred to as speech-to-text. The acoustic signal is first converted into an electrical signal by one or more microphones, then sampled and stored digitally. Prior to sampling, a filter is often applied to the signal before sampling to prevent aliasing. Common sampling rates are 8 kHz and 16 kHz. The former is typical for telephony-based applications.

The stored signal is then processed, and the ASR system produces a transcription, a sequence of words. The quality of the system and operating conditions determine the likelihood that the generated transcription matches the actual set of words spoken by the

speaker. As extra references on the fundamentals of ASR systems, the reader is referred to [20-22].

t	Time sample index
l	Frame index
$y(t)$	Digitally sampled speech signal
y_p	p^{th} log filter bank energy coefficient
P	Number of filters in mel-spaced triangular filter bank
c_k	k^{th} cepstral coefficient
Δ_k	k^{th} delta coefficient
$\Delta\Delta_k$	k^{th} delta-delta coefficient
$\mathcal{M}$	Trained model
S_i	i^{th} State in $\mathcal{M}$
$\mathcal{S}$	Any sequence of states in a Hidden Markov Model
a_{ij}	Transition probability from state i to state j in $\mathcal{M}$
o_l	Observation vector for frame l
O	Observation sequence
$b_j(o_l)$	Observation probability for state j at frame l
$\alpha_i(l)$	Forward variable for state i at frame l
$\beta_i(l)$	Backward variable for state i at frame l

Table 2-1. Signal processing and Hidden Markov Model notation.

An ASR has three primary subsystems, as shown in Figure 2-1.

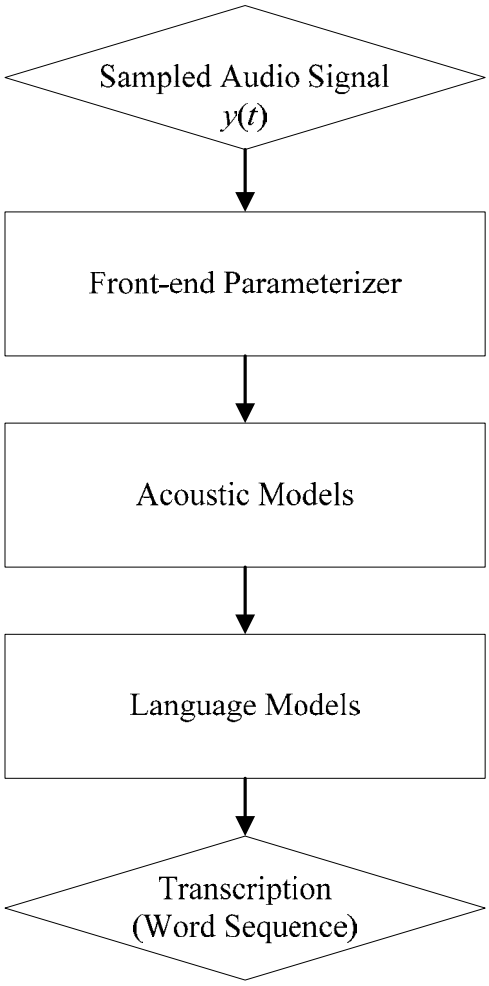

Figure 2-1. An automatic speech recognition system diagram.

In the front-end parameterizer, the sampled input signal is processed to produce speech features. Different types of features have been studied, but the most commonly used features in practice are known as cepstral coefficients [20]. A more detailed description of the computation of these features appears in Section 2.2 Front-end Parameterization.

In the acoustic models, the features computed in the front-end are tracked across time. This task, modeling the dynamics of the acoustic production system, is known as

acoustic modeling. The dominant mechanism for the modeling of acoustic units (discrete elements of speech sounds) is the Hidden Markov Model (HMM). HMM's are discussed in greater detail in Section 2.3 Acoustic Modeling with Hidden Markov Models.

Following the acoustic modeling, a language model often is applied. Language models can vary in their complexity and form, with the specific task (e.g. dictation or call center menu navigation) determining exactly how they are constructed and applied. Generally, language models are statistical, as is usually the case when the task is dictation. In some systems, a grammar is applied to restrict the form of the sentences the recognizer will accept. This is common for systems such as a voice activated automatic call desks. Language models are examined further in Section 2.4 Language Modeling.

2.2 Front-end Parameterization

The input to an ASR system is a sampled speech signal, collected from one or more microphones and possibly filtered to prevent aliasing. In this work, the input is assumed to be a single channel. For further reading on multi-channel speech signal processing, see [11]. Following collection, the signal must be processed to produce features that are used for tracking (modeling of speech dynamics) and recognition of the spoken words.

Cepstral coefficients describe the general shape of the magnitude spectrum or power spectrum of a segment of a signal. There are multiple methods for computing cepstral cofficients, giving rise to more than one type of cepstrum. In real systems, the two prevalent types are mel-frequency cepstral coefficients (MFCC's) and perceptual linear predictive (PLP) cepstral coefficients.

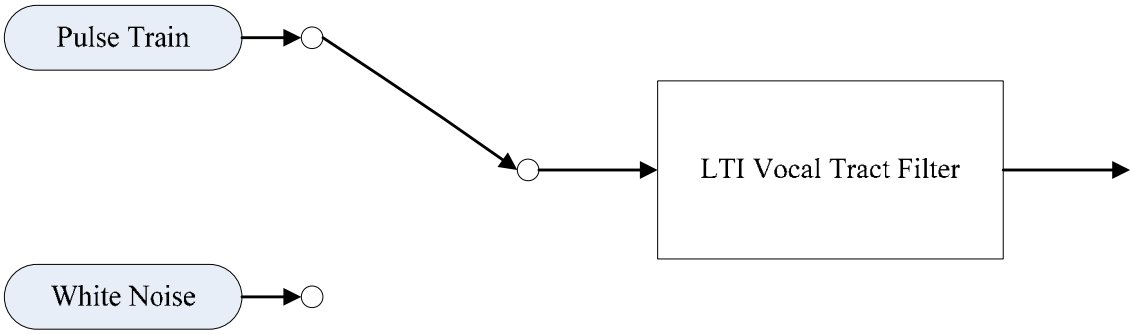

Figure 2-2. Source-filter model of human speech production.

MFCC's were first introduced for speech recognition by Davis and Mermelstein [23]. The motivation for these features is grounded in a simple source-filter model of human speech production. In this model, the speech production system is viewed in two components: a switched excitation source and a linear time-invariant (LTI) filter. The source represents the glottis, while the LTI filter represents the vocal tract, consisting of the mouth and nasal passage, as suggested by Figure 2-2.

In the first block, a signal is generated by one of two components, depending on the type of sound being produced. Generally, speech sounds are classified as either voiced or unvoiced. Vowels (e.g., the "ah" sound in abode) and some other sounds are voiced and have a pitch, whereas many fricatives (e.g., the "f" sound in frog) are unvoiced and have a more strident quality. For voiced sounds, the glottis generates a signal that is modeled as pulse train induced at the microphone, an example of which appears in Figure 2-3. The period between the pulses is determined by the fundamental frequency of the signal. Alternatively, if the sound is unvoiced, the glottis generates white noise, which has a flat power spectrum.

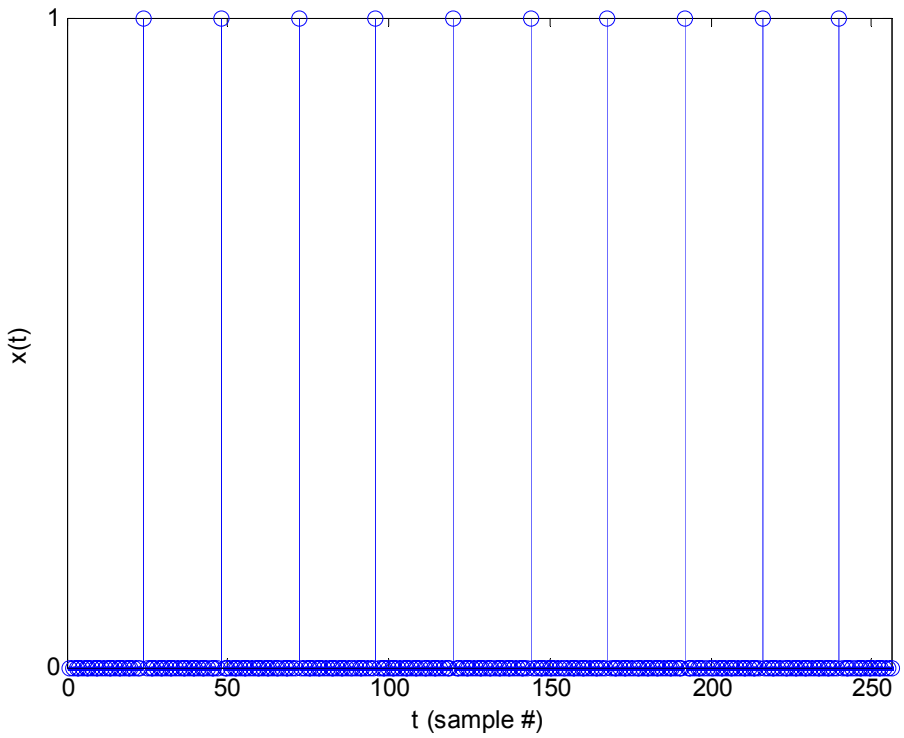

Figure 2-3. A pulse train with a period of 24 samples.

In English and many other languages, most of the information that distinguishes between the basic speech sound units, known as phonemes, is carried in the spectral envelope. Thus, it is primarily the LTI filter that determines the particular sound that is produced. Because of this, most front-end parameterization systems attempt to extract features from the speech signal that describe the LTI filter, ignoring the source signal as much as possible. In Figure 2-4C, the rough shape of the spectrum corresponds to the vocal tract filter, whereas the small ripples are caused by the glottis. The cepstral domain separates these two components [21]. The cepstrum is the inverse Fourier transform of the log of the magnitude of the Fourier transform of a signal, as

$$\text{cepstrum} = \text{IFFT}\left\{\log\left|\text{FFT}\left\{y(t)\right\}\right|\right\}. \tag{2.1}$$

A small number (around 12) of cepstral coefficients describe the spectral envelope well, ignoring the ripples created by the glottis.

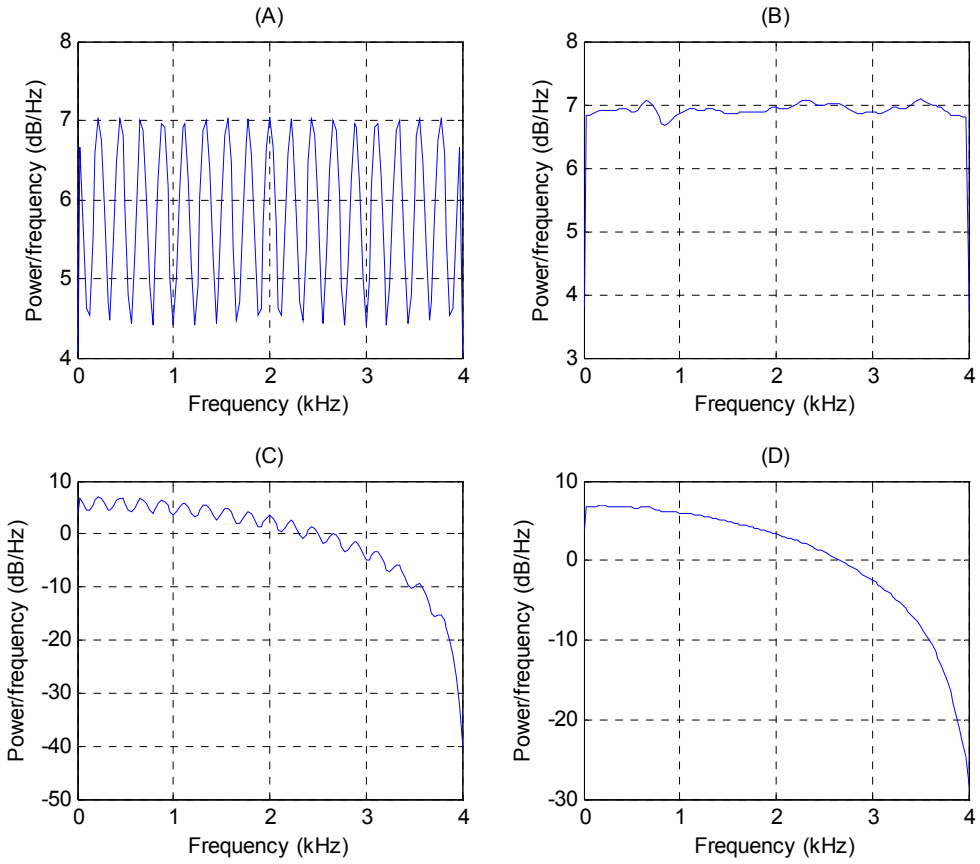

Figure 2-4. A pulse train spectrum (A), a white noise spectrum (B), a filtered pulse train spectrum (C), and a filtered white noise spectrum (D).

As stated, there are several ways to generate cepstral coefficients. The process for computing MFCC features in a typical ASR system front-end is shown in Figure 2-5. The following sections describe the steps in MFCC parameterization. The output of the system is a sequence of feature vectors.

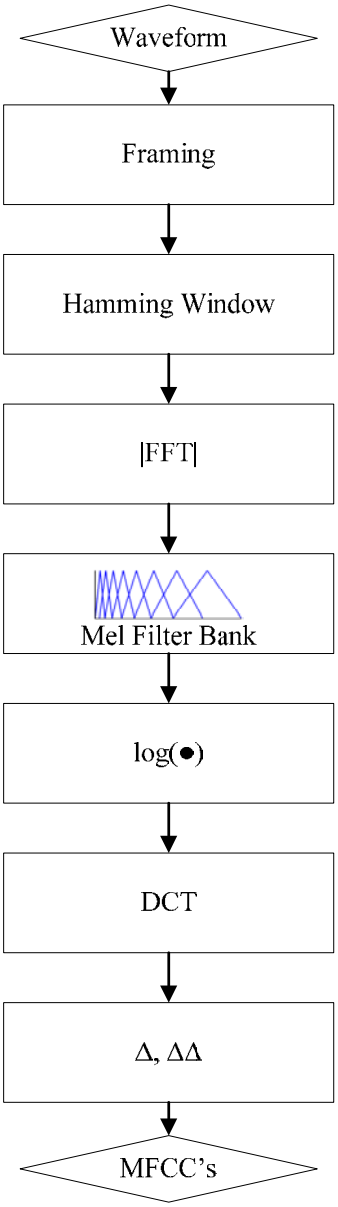

Figure 2-5. Block diagram of a typical ASR front-end [24, 25].

2.2.1 Framing and Windowing

First, a pre-emphasis (low order high-pass) filter may be applied to the sampled signal to reduce spectral tilt. A typical pre-emphasis filter has the response $H(z) = 1 - 0.97z^{-1}$.

Because speech is highly nonstationary, the signal must be segmented into smaller, quasi-stationary frames. Thus, a reasonable power spectral estimate can be obtained for each frame. A typical frame length is 25 – 32 milliseconds. These frames overlap, with the first sample in each frame typically spaced every 10 milliseconds. A smoothing window is then applied to each frame, typically a Hamming window [26].

2.2.2 Power Spectrum Computation

The power (or magnitude) spectrum is computed for each smoothed frame with a fast Fourier transform (FFT). A logarithmically spaced filter bank, consisting of P triangular filters, is applied to the power spectrum, producing a set of mel-frequency filter bank energy coefficients, one for the total power of each filter. Logarithmic (mel) spacing is used to mimic the frequency response of the human auditory system [27].

The endpoints of the triangular filters are determined by equally spacing them in the mel domain and transforming the values of the endpoints into the frequency domain. The mel domain is related to the frequency domain by

$$Mel(f) = 2595 \log_{10}\left(1 + \frac{f}{700}\right). \tag{2.2}$$

where f is frequency in Hz. The number of filters, P, is one of the ASR system parameters and is typically between 20 and 40. The filters are made to overlap by 50%.

2.2.3 Static Feature Computation

Static MFCC's are derived by applying the discrete cosine transform (DCT) to the filter bank energy coefficients. The DCT is defined by

$$c_k = \sum_{p=0}^{P} y_p \cos\left[\frac{\pi}{P}\left(p + \frac{1}{2}\right)k\right], \quad k = 0, \ldots, K-1, \tag{2.3}$$

where k is the index of the cepstral coefficient, and y_p is the p^{th} filter bank energy coefficient. The value of K typically is 13 (although the zeroth coefficient may or may not be used). While the log filter bank energy coefficients could be used as features, the DCT transformation acts as an operation that de-correlates the values. Thus, MFCC's display low cross-correlation, an attractive attribute that allows for simpler statistical modeling by the ASR system.

The static features contain information that is contained only in the local frame. In Section 2.2.5, dynamic features are computed across windows of multiple frames and appended to the static feature vector.

2.2.4 Energy Measures

Once the static cepstral coefficients are computed, an energy measure is appended to the vector of MFCC's. In some ASR systems, the zeroth cepstral coefficient is used. In others, this feature is removed, and log energy is computed and appended to the static feature vector for each frame. This measure is computed for each frame by

$$e(l) = \ln \sum_{t=0}^{T-1} y_l(t)^2, \tag{2.4}$$

where $y_l(t)$ is the t^{th} sample value in frame l, and T is the number of samples in the frame. The energy measure may be normalized over an entire utterance by subtracting the maximum value from all energy coefficients for a given signal.

2.2.5 Derivative Features

First and second derivative features are computed over windows of static features and appended to the static feature vector. These coefficients add information regarding

the temporal nature of the signals. First derivative features, frequently referred to as deltas, are computed over a short window of the static coefficients, as

$$\Delta_k^l = \frac{\sum_{j=1}^{J} j\left(c_k^{l+j} - c_k^{l-j}\right)}{2\sum_{j=1}^{J} j^2}, \tag{2.5}$$

where Δ_k^l is the delta coefficient computed over the k^{th} cepstral coefficient for the l^{th} frame in the window. The parameter J is typically 2 or 3.

The second derivative features, frequently called delta-deltas, are computed over the deltas using an equation of the same form as (2.5), but substituting deltas for static cepstrals:

$$\Delta\Delta_k^l = \frac{\sum_{j=1}^{J} j\left(\Delta_k^{l+j} - \Delta_k^{l-j}\right)}{2\sum_{j=1}^{J} j^2}. \tag{2.6}$$

The value for J used for computation of the delta-deltas is usually one less than the value used for delta computation. If log energy is used as a feature in place of c_0, derivative features are computed on this coefficient in the same manner as the cepstrals.

The final resulting feature vector for each frame of speech typically consists of 39 total coefficients: 13 static coefficients, consisting of 12 MFCC's and a single energy measure,13 first derivative features, and 13 second derivative features. These features are used by the acoustic modeling unit described in Section 2.3 Acoustic Modeling to track the spectral characteristics of a speech signal over time and determine the likeliest sequence of basic speech sounds in the signal.

2.2.6 Variations on Cepstral Features

The procedure for computing speech features described above is among the simplest and most common techniques used in real ASR systems and is the approach adopted for all systems used in this dissertation. However, there are additional methods for speech feature parameterization studied in the literature and built into commercial systems. Among these are linear predictive coefficient (LPC) features [22], perceptual linear predictive (PLP) features [13], and nonlinear or phase based features [28, 29].

Much like cepstral coefficients, LPC's capture the spectral envelope of a segment of a signal. Using the source-filter model described in Section 2.3, the LTI vocal tract filter can be modeled as an all-pole filter with the z-transform

$$H(z) = \frac{G}{1 - \sum_{i=1}^{q} a_i z^{-i}}, \tag{2.7}$$

where G is the gain, and q is the order of the filter. The set of coefficients $a_1, \ldots, a_q$ most commonly is learned by computing autocorrelation coefficients and using the Levinson-Durbin recursion [30] to produce the LPC's.

Unfortunately, the LPC values do not make good features for use in ASR systems because common distance metrics (e.g. Euclidean distance) on these features are not highly correlated with (human) perceptual difference between spectra. It is possible to transform the LPC's to more useful domains, such as the cepstrum, using an efficient recursion formula [31]. The LPC-cepstral coefficient algorithm represents an alternate version of cepstral coefficient computation, although it does support perceptually motivated constructs such as the mel-spaced filter bank described in Section 2.2.2.

PLP features, developed by Hermansky [13], are another set of coefficients that capture information related to the spectral envelope. Like the LPC features, PLP's model the vocal tract filter with an all-pole system. However, additional processes are implemented to warp the signal, motivated by characteristics of the human auditory system. Once the linear predictive coefficients have been computed from the auditory shaped spectrum, the same cepstral recursion applied to LPC's can be used. PLP cepstral coefficients are commonly used in large scale state-of-the-art systems [32, 33].

All the feature sets described so far are based on magnitude or power spectral modeling, in which the phase is ignored. Alternative features that capture nonlinear and phase information have been studied [28, 29]. While these features can represent information that linear predictive and cepstral values do not, it is still true that much of the perceptually important information in human speech lies in the power spectrum. Thus, these alternative features are typically used in conjunction with spectral-based features.

2.2.7 Front-end Summary

This section has introduced front-end feature extraction and described the signal processing methods used in a typical front-end. The front-end subsystem can be viewed as a data reduction system, in which a signal of tens or hundreds of thousands of samples is transformed into a sequence of feature vectors, reducing the size of the data by a factor of around 10. The output of the front-end is used by the acoustical modeling unit to decode the signal into a sequence of discrete sound elements, which in turn compose a sequence of words. The next section describes the popular mechanism for acoustic modeling, the Hidden Markov Model (HMM).

2.3 Acoustic Modeling with Hidden Markov Models

HMM's are the predominant tool for identifying and tracking the spectral characteristics of sequential frames of speech. Many other applications for HMM's have been studied as well [34-37]. The success of HMM's for use in speech (as well as the other applications) depends on the computational efficiency for training and decoding tasks, which is made possible by the Markov property.

HMM's are critical for this work not only because they form the basis for the recognition of sequences of sounds in a speech signal, but also because information for use in feature estimation as presented in Chapter 5 is extracted from the sequence of HMM states that is generated as a transcription hypothesis by the recognizer. Specifically, the token passing scheme described in Section 2.3.6.3 is important, as the data structures known as tokens carry the relevant information used by methods for adaptation of prior distributions of features introduced in Chapter 5.

2.3.1 Markov Property

A Markov process [38] is a stochastic process that obeys the Markov property: previous values of the process have no bearing on the future values; instead, only the current value determines the probability distribution of the next value. In this way, a Markov process is said to be memoryless.

A Markov process can be continuous-time or discrete-time. In the former case, the current value, as described above, is considered to be the process's value for an infinitely small period of time. In the latter, it is simply the value denoted by the "current" time index.

The output of a Markov process can also be either continuous or discrete. For the discrete case, the process is referred to as a Markov chain [38], in which a number (possibly countably infinite) of states $S_0, ..., S_{Q-1}$ are defined. The probability of the process occupying a particular state S_i at time l+1 is determined solely by the state occupied at time l. Using this property, a transition probability matrix can be defined for the Markov chain, specifying the transition probability from any state to any other state (self transitions included), given as

$$A = \begin{bmatrix} a_{00} & a_{01} & \cdots \\ a_{10} & a_{11} & \\ \vdots & & \ddots \end{bmatrix}. \tag{2.8}$$

Here, a_{ij} is the transition probability from state i to state j. For A to be a valid transition matrix, two conditions must be satisfied:

$$0 \le a_{ij} \le 1 \quad \text{for all } i, j, \quad \text{and} \tag{2.9}$$

$$\sum_{j=0}^{Q-1} a_{ij} = 1 \quad \text{for all } i. \tag{2.10}$$

This transition matrix completely characterizes the Markov chain.

2.3.2 Hidden Markov Models

While Markov processes and chains can be used to model some stochastic systems directly, many complex systems such as human speech production include states that are not directly observable. In these systems, even given the observed data, one does not know in which state the process currently resides. Thus, the states are said to be hidden.

Unlike a Markov chain, an HMM is not completely described by the transition probability matrix. In addition, a set of output probability distributions (probability density functions for continuous output models or probability mass functions for discrete output models) is required, one for each state. The probability of a process occupying a particular state S_i at time l+1 is determined by the state occupied at time l (through the transition probability matrix) and the probability of the observed data matching the distribution for state S_i. For the first sample (l=0) in the data, an initial probability distribution (i.e. the probabilities of the process beginning in each state) is needed to compute the occupation probabilities.

As stated, the output probability distribution can be continuous or discrete. In most state-of-the-art ASR systems, including the system described in this dissertation, continuous distributions are used. The probability density function (pdf) of choice in these systems is the multivariate Gaussian Mixture Model (GMM), which is a weighted sum of Gaussian functions. A multivariate Gaussian probability density function is

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{K}{2}} |\Sigma|^{\frac{1}{2}}} e^{(\mathbf{x}-\boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x}-\boldsymbol{\mu})}, \tag{2.11}$$

where $\boldsymbol{\mu}$ is the mean vector, Σ is the covariance matrix (which is often diagonal), and K is the dimension of the feature space. A GMM is expressed as a weighted sum of Gaussian pdf's, with the requirement that the weights sum to unity.

A continuous density HMM is depicted graphically in Figure 2-6. While transitions may be allowed among all pairs of states, many ASR systems use an HMM topology that constrains the transitions to move from left to right (allowing self transitions), which leads to the concept of state ordering, i.e., the states are described in a

linear topology. While the model in Figure 2-6 shows three states (S_1, S_2, S_3), the number of states in an HMM is system-dependent. In Figure 2-6, transitions are represented by the arrows, and the associated transition probabilities are represented by a_{ij}. Depending on the system design, there may or may not be a skip transition from state S_1 to state S_3. Because of the self-transitions, each state can occupy an indefinite number of frames. The output probabilities, which are computed with the output pdf's and the parameterized data for each frame as described in Section 2.3 are labeled as $b_j(o_l)$, in which j is the state index, and o_l is the observation (parameterized) data for frame l.

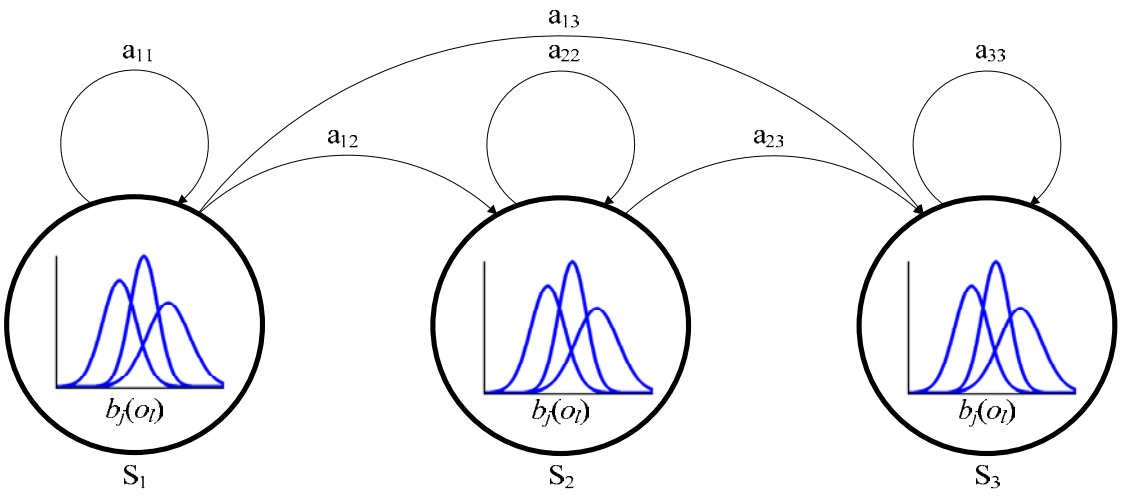

Figure 2-6. A 3-state, left-to-right continuous density HMM.

2.3.3 Recognition Units

An HMM represents a unit (discrete element) of speech. The type of unit represented depends on the system and the task involved. For most medium (1,000 – 10,000 words) to large (10,000+ words) vocabulary systems, the unit of choice is the phoneme, the most basic meaningful acoustic element of speech. A typical system

defines a set of 40 or more phonemes. These phonemes can be categorized into one of five general phonological classes: vowels, fricatives, plosives (or stops), nasals, and glides. In addition, units may be specified for silence, as well as "garbage" sounds, such as laughter or coughing. The phoneme set defined for use in later chapters is given in Appendix A. More discussion of phonemes is found in chapter 23 of [22].

Systems that are designed for tasks that involve smaller vocabularies, such as digit recognition systems, often model the words themselves (i.e., each word is represented by a single HMM unit). In this case, the number of states in each model may be larger than in the case of phoneme models. For phoneme units, 3 to 5 states is typical, while 15 or more states are commonly used for word models.

2.3.4 Chaining Multiple Hidden Markov Models

In continuous speech recognition (where the user is not required to pause between words), an utterance is modeled by a sequence of chained HMM's. For example, the utterance "The dog ran" is represented by the phoneme sequence: *sil dh ah sil d ao g sil r ae n sil*. Here, the silence units inserted between each word (denoted by *sil*) may be optional units.

To accomplish the chaining of HMM's, the model from Figure 2-6 is modified to include an extra "non-emmitting" state, labeled S_{exit}. The non-emitting attribute of this state means that it contains no output distribution. This new model is shown in Figure 2-7. State S_{exit} serves as a mechanism for connecting two consecutive HMM's. As a process exits the final emitting state in a unit, it enters the S_{exit} state. This transition does not consume any frames, however, meaning the transition from the S_{exit} state to the first state in the second model occurs with no time elapsing in the process.

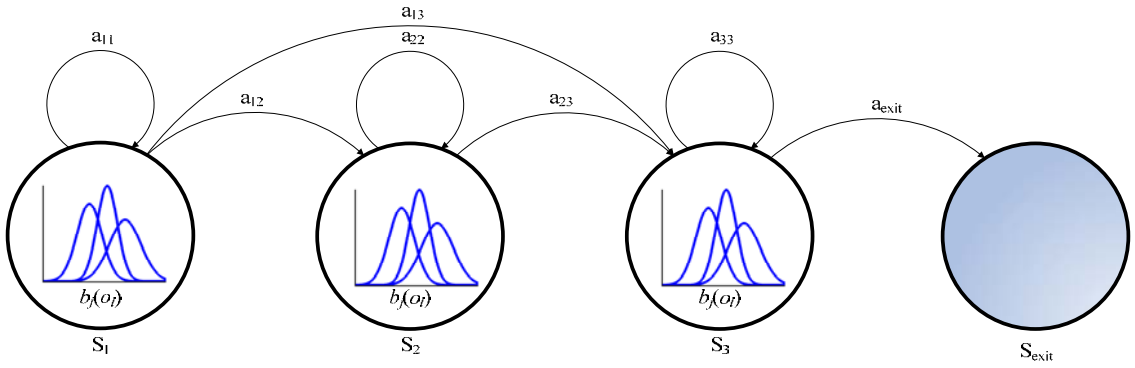

Figure 2-7. Modified HMM containing a non-emitting exit state.

2.3.5 Context-Dependent Units

As stated, most medium to large vocabulary recognition systems use phoneme units. However, the use of 40 (or so) individual phoneme units is insufficient to capture the temporal nature of human speech. A phenomenon known as coarticulation necessitates a more complex model set.

The articulators (jaw, tongue, etc.) in the human speech production system are in constant motion during speech. Subsequently, the actual spectral characteristics of a phoneme are affected by the past and future phonemes in an utterance. For example, a vowel will sound different when preceded by a fricative as opposed to a plosive. This coarticulation effect leads to the need for context-dependent units. Each context-dependent model has a central unit corresponding to one of the phonemes, as well as left and right contexts. If the left and right contexts are both singular (one phoneme each), as is the case in the systems used for the medium vocabulary experiments in this dissertation, these units are known as triphones. An example of a triphone is $b-ae+t$.

The left context is *b*, the right context is *t*, and the central phoneme is *ae*. This triphone represents the vowel sound in the word bat.

While the use of context-dependent units reduces the negative effects on the performance of ASR systems caused by coarticulation, a new problem is introduced, that of data insufficiency. Assuming 40 phonemes, there are 40 possible left contexts, 40 possible central models, and 40 possible right contexts. This gives $40^3 = 64,000$ possible models. The statistical parameters for each model must be learned (this process is described briefly in section 2.3.6.1) from a set of data. Many of the triphone models will not be present in the training data or will have few examples. Consequently, the triphone models must be placed into groups, reducing the number of unique models. This is done a state clustering algorithm.

A decision tree algorithm [39] is used to tie states in the triphone models so that while there are 192,000 individual states (assuming 64,000 3-state models), many of the states share the same output distribution. This method of state-tying can reduce the number of unique state distributions to several thousand. For triphone models for which no examples are present in the training data, similar models are used in their place. The similarity of models is determined using linguistic information and the decision tree.

2.3.6 HMM Training and Recognition Algorithms

In the previous sections, the structure of an HMM is discussed. For use in acoustic modeling in ASR systems, algorithms for the training of HMM parameters and for the application of trained models for recognition purposes are necessary. The Baum-Welch re-estimation algorithm is used for training, while recognition is performed with either

the forward/backward algorithm or the Viterbi algorithm. These algorithms are discussed in the following sections.

2.3.6.1 Baum-Welch Re-estimation

In the process of HMM training, the output probability distribution parameters and the transition probabilities must be learned. For the continuous density HMM's described in previous sections, the distribution parameters are the means and covariances of the Gaussian mixtures, as well as the mixture weights. Parameter training can be accomplished by finding the parameters that maximize the likelihood of the training data given the model set. Unfortunately, while the word sequence (and subsequently the HMM state sequence) corresponding to a training utterance is known, frame level time labels are not available. Baum-Welch re-estimation is able to overcome this issue by an iterative approach of two steps: expectation and maximization [19].

Model parameters are initialized to general values, most often using the statistics of the entire data. In this way, all states have the same mean and covariance values. Following the initialization, an expectation step is run. Here, occupancy likelihoods are computed for all states over each frame in the training data. With this information, new parameters (both the output distribution parameters and the transition probabilities) can be derived with the maximization step. This process is repeated a number of times, and the likelihood of the data given the model parameters converges to a local optimum. The details of the algorithm are not discussed here; a more thorough discussion can be found in [40].

Originally, each state in the model set is represented by a single Gaussian, and only context independent models exist. After a small number of iterations of Baum-

Welch, the decision tree state clustering mentioned in Section 2.3.5 is performed, expanding the model set to include context dependent triphones. Several more iterations are then run using the triphone models before splitting the Gaussians, changing the output probability distributions from simple Gaussians to GMM's. The splitting is performed by duplicating each Gaussian mixture, halving the associated mixture weight (the original weights are set to unity), and perturbing the mean of each Gaussian slightly in opposite directions. After the mixture splitting is applied, several more iterations of Baum-Welch are run. This process is repeated until the desired number of mixtures has been reached. The number of mixtures must not necessarily be the same for each state, and the "ideal" number of mixtures is task dependent.

Once the model parameters have been learned, the model set can be applied to previously unseen data. Two different tasks may be performed: evaluation and decoding. In evaluation, the probability that the data was generated by the model is computed. This is done with the forward/backward algorithm. In decoding, the most likely state path that could have generated the data is found. This is accomplished with the Viterbi algorithm. These processes are discussed in the following two sections.

2.3.6.2 Forward / Backward

Given the trained model, $\mathcal{M}$, and an observation sequence $O = (o_0, o_1, ..., o_{L-1})$, it is possible to determine the probability that the observed data was generated by the model,

$$P(O \mid \mathcal{M}) = \sum_{all\ \mathcal{S}} P(O \mid \mathcal{S}, \mathcal{M}) P(\mathcal{S} \mid \mathcal{M}), \tag{2.12}$$

where $\mathcal{S}$ is any sequence of states of length L. The quantities $P(O|\mathrm{S},\mathcal{M})$ and $P(\mathcal{S}|\mathcal{M})$ are computed by

$$P(O|\mathcal{S},\mathcal{M}) = \prod_{l=0}^{L-1} b_{S_l}(o_l), \quad \text{and} \tag{2.13}$$

$$P(\mathcal{S}|\mathcal{M}) = \pi_{S_0} \prod_{l=0}^{L-2} a_{S_l S_{l+1}}. \tag{2.14}$$

Here, π_{s_0} is the probability of beginning at state S_0. Combining (2.13) and (2.14) gives

$$P(O|\mathcal{M}) = \sum_{all\ S} \pi_{s_0} b_{S_{T-1}}(o_{T-1}) \prod_{t=0}^{T-2} a_{S_t S_{t+1}} b_{S_t}(o_t). \tag{2.15}$$

Direct computation of this value requires approximately $2L$ x Q^L calculations, where Q is the number of (emitting) states, making the problem intractable. However, a more efficient algorithm, the forward procedure, exists for computing $P(O|\mathcal{M})$. In the forward procedure, the overall data likelihood is represented as the sum of the joint densities for all possible final states, given the model, as

$$P(O|\mathcal{M}) = \sum_{i=0}^{Q-1} P(S_{L-1} = i, o_0 o_1 ... o_{L-1} | \mathcal{M}). \tag{2.16}$$

The computation of this quantity uses a forward recurrence variable,

$$\alpha_i(l) = P(S_l = i, o_0 o_1 ... o_l | \mathcal{M}), \tag{2.17}$$

representing the joint density of the observation up to frame t and the occurrence of state i at that frame, given the model. Using the Markov property and assuming that all frames are statistically independent, $\alpha_i(l+1)$ can be expressed as a function of the observation at time l+1, $\alpha_i(l)$, and the transition probabilities,

$$\alpha_j(l+1) = \left(\sum_{i=0}^{N-1} \alpha_i(l) a_{ij} \right) b_j(o_{l+1}). \tag{2.18}$$

For the first frame, the values of α are computed by

$$\alpha_i(0) = \pi_i b_i(o_0). \tag{2.19}$$

Combining equations (2.17) and (2.16) yields

$$P(O \mid \mathcal{M}) = \sum_{i=0}^{N-1} \alpha_i(L-1). \tag{2.20}$$

The overall data likelihood $P(O \mid \mathcal{M})$ is found using the initial condition from (2.19) and the recurrence relation from (2.18). Using this algorithm, the data likelihood can be computed in $O(LQ^2)$ time, much faster than the $O(LQ^L)$ complexity of a direct computation from (2.15).

An analogous algorithm, the backward procedure, exists as well. In the backward algorithm, let

$$\beta_i(l) = P(o_{l+1} o_{l+2} \ldots o_{L-1} \mid S_l = i, \mathcal{M}). \tag{2.21}$$

This variable represents the density of the observation at the following time *l*, given the current state and the model. As in the forward algorithm, a backward recurrence can be developed,

$$\beta_i(l) = \sum_{j=0}^{N-1} \beta_j(l+1) a_{ij} b_j(o_{l+1}). \tag{2.22}$$

Since there is no ending state probability distribution (as is the case for beginning states, π_i), the initial value for $\beta_i(l)$ is unity for all *i*. If the backward recursion is used, the overall data likelihood is

$$P(O \mid \mathcal{M}) = \sum_{i=0}^{N-1} \pi_i b_i(o_0) \beta_i(o_0). \tag{2.23}$$

While either recursion can be used to compute the data likelihood, both algorithms are needed to compute the probability of occurrence of state *i* at time *l*, given the observation and the model. This quantity is computed by

$$P(s_l = i \mid O, \mathcal{M}) = \frac{P(s_l = i, O \mid \mathcal{M})}{P(O \mid \mathcal{M})} = \frac{\alpha_i(l)\beta_i(l)}{\sum_{j=0}^{Q-1} \alpha_j(l)\beta_j(l)}. \tag{2.24}$$

The forward and backward algorithms are necessary for the computation of state occupancy likelihoods, each of which represents the likelihood of the process occupying a particular state *i* at a time *l*. The state occupancy likelihoods are used in Chapter 5 for development of prior statistical distributions on MFCC features in the ACDM-MMSE estimation approach that is introduced in Chapter 4.

2.3.6.3 Viterbi and the Token Passing Algorithm

The forward and backward algorithms are useful for evaluation of a model for a given observation. However, if the sequence of states that are the most likely to have generated the data is desired, a different algorithm is needed. A dynamic programming recursion known as the Viterbi algorithm [41] produces the most likely state path, given a signal.

The forward algorithm defines an initialization condition, $\alpha_i(l) = \pi_i b_i(o_t)$, and a recursion relation, $\alpha_j(l+1) = \left(\sum_{i=0}^{Q-1} \alpha_i(l) a_{ij} \right) b_j(o_{l+1})$. The summation in the latter term computes the total likelihood at each frame. However, if only the most likely path is

needed, the summation is replaced with a maximum, taken with respect to the state index *i*. As in the forward algorithm, the Viterbi algorithm begins with an initialization

$$\chi_i(l) = \pi_i b_i(o_t), \tag{2.25}$$

where $\chi_i(l)$ replaces $\alpha_i(l)$ and represents the likelihood of the maximum path to state *i* at time *l*. The recursion is

$$\chi_j(l+1) = \max_i \left\{ a_{ij} \chi_i(l) \right\} b_j(o_l). \tag{2.26}$$

Once the end of the observation sequence has been reached, the likelihood of the best path is

$$\bar{P}(\mathcal{S}, O \mid \mathcal{M}) = \max_i \left\{ \chi_i(L-1) \right\}. \tag{2.27}$$

Because the sequence of states in the most likely path is the desired output of this algorithm, bookkeeping must be performed to ensure the state history is maintained.

Because of the max operation in equation (2.26), only the single best path to a particular state is recorded. In some applications it is desirable to maintain some of the sub-optimal paths. This task is accomplished by a token passing algorithm.

In a token passing algorithm, a recognition structure is built, consisting of states and state arcs coming from the set of HMM's. The valid arcs are determined by the system dictionary, which specifies the model sequences for each word, and possibly by a grammar. Arcs may connect two states in the same model or states from two models that can legally be chained (passing through the non-emitting exit state of the earlier model). In this way, the recognition structure contains all possible word sequence paths at the state level.

The algorithm begins by placing a "token" at each of the states that can legally begin the utterance. A score (output probability) is determined according to (2.25) and

assigned to the token. Each token then propagates to all legal successor states. A new score is determined for each token using the transition probabilities and the observation for the next frame. If only the token with the best score for each state is kept (and all others discarded), this algorithm works in the same manner as the Viterbi algorithm. However, some of the sub-optimal tokens can be saved, allowing for alternate state path hypotheses to be constructed. It is computationally infeasible to retain all sub-optimal tokens, though. Consequently, the list of tokens must be pruned. Only a small number of the best tokens are allowed to survive; all others are discarded. The number of tokens that survive is dependent on ASR system configuration parameters and is chosen based on the speed and performance (accuracy) requirements of the task.

The output of the recognizer is a data structure known as a word lattice, depicted in Figure 2-8. A lattice contains all possible (surviving) word paths and their associated word-level acoustic likelihoods.

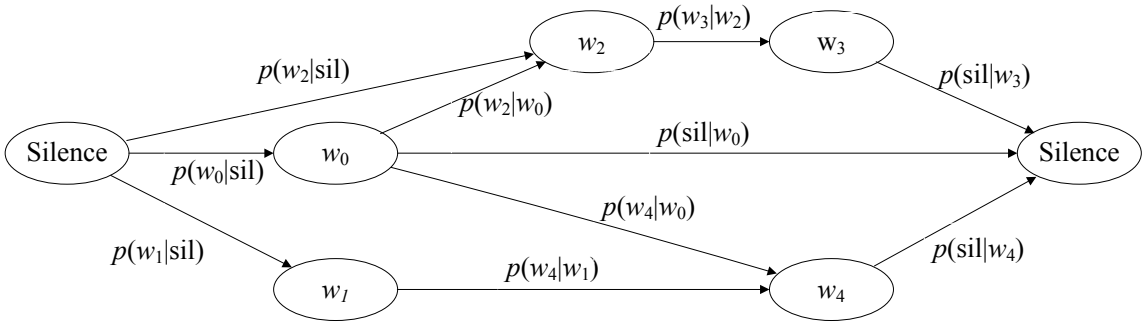

Figure 2-8. A simple example of a word lattice. Each node represents a word with an associated probability, while each edge represents the probability of a word given the previous word.

Another data structure known as an *N*-best list, shown in Figure 2-9, can be extracted from a lattice. This data structure contains a list of the most likely *N* unique

word paths, along with associated likelihoods. Further processing (such as language modeling) can be applied to these data structures, and the ordering of hypotheses may change. This allows for more than just acoustic information to be incorporated into the recognition process, improving the accuracy of the ASR system.

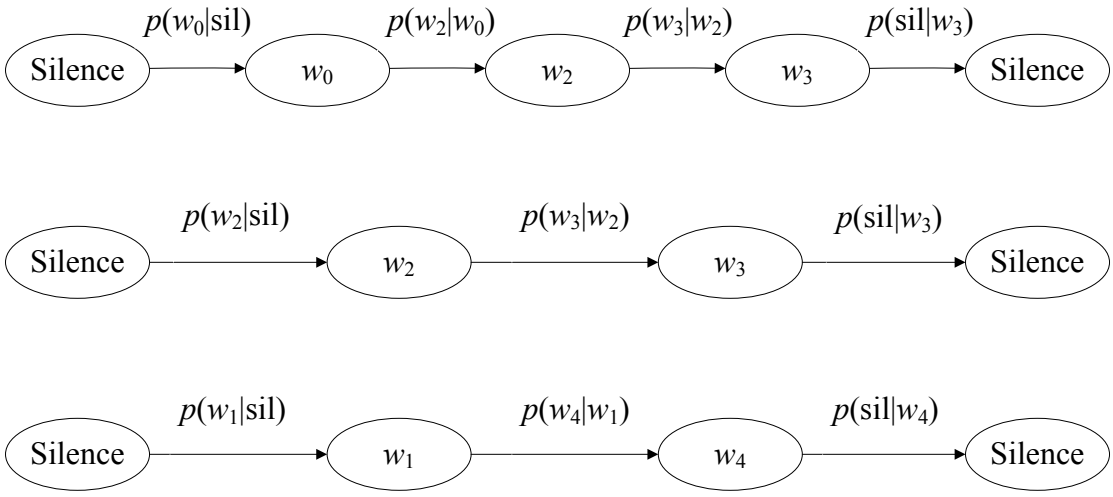

Figure 2-9. An example N-best list extracted from the lattice shown in Figure 2-8 with N=3.

2.4 Language Modeling

The next subsystem in an ASR system is the language model. This unit can operate on the output of the acoustic modeling unit, as is depicted in Figure 2-1, but can also work in conjunction with the acoustic modeler (for practical reasons, as is discussed briefly in the final section of this chapter). In many real systems, the language modeling is performed both during the acoustic modeling, and as a post-processing step.

In real human speech, some grammatical structure is present. Consequently, systems that operate solely on acoustic data ignore important information and can be expected to exhibit performance inferior to systems that take advantage of the structure of

human language. The improvement in system performance enabled by the incorporation of language models has been demonstrated empirically [21] and has been shown to be significant.

The predominant form of language modeling used in ASR systems is statistical. Using statistical language models, probabilities of word co-occurrences are used to modify the likelihoods of state paths either during recognition, as a post-processing step, or both. The mechanism for applying these language models is known as the *n*-gram. A grammar may also be applied [21], but discussion of this instrument is omitted from this dissertation.

An *n*-gram (in the context of language modeling) is a sequence of $n \geq 1$ words. Specific names are given to *n*-grams with $n \leq 3$: unigrams (*n*=1), bigrams (*n*=2), and trigrams (*n*=3). For all other values of *n*, the integer value is inserted in the name (i.e., 4-gram).

An *n*-gram is used to measure the probability of a word w_m occurring, given the previous *n*-1 words, as $P\left(w_m \mid w_{m-n+1}...w_{m-2}w_{m-1}\right)$. In the case of a unigram, the previous sequence of words is not considered. Instead, the probability $P\left(w_m\right)$ is proportional to the frequency with which that word occurs in language.

The *n*-gram probabilities typically are obtained through maximum likelihood estimation. This is done by counting occurrences of word sequences in a set of training data. The probability of a particular *n*-gram is

$$P\left(w_m \mid w_{m-n+1}...w_{m-2}w_{m-1}\right) = \frac{C\left(w_{m-n+1}...w_{m-2}w_{m-1}w_m\right)}{C\left(w_{m-n+1}...w_{m-2}w_{m-1}\right)}, \tag{2.28}$$

where $C\left(z\right)$ is the count of *z* in a dataset.

To see how *n*-grams are useful, the total probability of a sequence of words is

$$P(w_0 w_1 ... w_{M-1}) = P(w_0) P(w_1 \mid w_0) ... P(w_{M-1} \mid w_0 w_2 ... w_{M-2}). \quad (2.29)$$

For large values of *M*, the last terms in the product are conditioned on long sequences of words. To estimate these probabilities from a set of data, each sequence must occur many times. The number of possible sequences grows exponentially with respect to *M*, so even medium length sentences contain probabilities conditioned on word sequences that occur rarely if at all. To solve this problem, each probability is approximated with an *n*-gram. In the case of a bigram, equation (2.29) becomes

$$P(w_0 w_1 ... w_{M-1}) = P(w_0) P(w_1 \mid w_0) P(w_2 \mid w_1) ... P(w_{M-1} \mid w_{M-2}). \quad (2.30)$$

Even with this approximation, some word sequences that can be expected to occur during recognition may appear in the training set rarely or not at all. Because of this, some *n*-grams will be estimated with zero probability. To deal with this issue, discounting (or smoothing) is applied. Discounting methods shift some of the weight from frequently occurring *n*-grams to infrequently occurring (or absent) *n*-grams. Some common discounting methods are Laplacian, which adds a count to every word, Good-Turing [42], and Witten-Bell [43].

In addition to discounting, back-off models [44] may be used. When a previously unseen *n*-gram is encountered, a lower order model (smaller *n*) may be used in place of the unseen *n*-gram, along with a back-off weight. The back-off weight is present to ensure that the *n*-gram probabilities sum to unity.

In an ASR system, the language model and acoustic model are considered independent. The overall likelihood of a word sequence, given the observation, the

acoustic model, and the language model is computed by multiplication of the acoustic and language likelihoods.

After modification of the path likelihood by the language model, a transcription is produced by the ASR system. A transcription is a sequence of words that represents the recognizer's best hypothesis for what was spoken in the input utterance. Depending on the task, the transcription may be used to take action (as in a call center help desk) or may simply be printed (as in a dictation system).

2.5 Practical Considerations

In the evaluation of an ASR system, the two primary measures for performance are word accuracy and run time (in relation to real time). Four measures are considered for the computation of accuracy, the number of word substitutions, S, the number of insertions, I, the number of deletions, D, and the total number of words in the true transcription, W. Substitutions are incorrect replacement of words in the transcription hypothesis, while insertions are extra words and deletions are missing words. The word accuracy is given as a percentage and is computed by

$$Acc\% = \left(1 - \frac{S + I + D}{W}\right) \times 100\%, \tag{2.31}$$

The value in (2.31) is computed by a dynamic programming algorithm, which finds the values for S, I, and D that maximize the word accuracy value.

The run time is computed by the ratio of processing (recognition) time to the length of the audio signal. For example, if the utterance is a speech signal 3 seconds long, and the ASR system requires 6 seconds to parameterize and recognize the signal, the run time is 2x real time.

Another practical consideration of ASR systems involves the computation of path likelihoods. Because each frame is considered independent, the total likelihood for a path is calculated by multiplying the individual frame likelihoods. As these values are often small $\left(p(x) << 1\right)$, numerical problems are often present; the cumulative likelihood value can quickly become zero. To deal with this issue, log likelihoods are used. The multiplication operation then becomes addition in the log domain:

$$\ln\left[p(x)\,p(y)\right] = \ln p(x) + \ln p(y). \tag{2.32}$$

As stated previously, the language model can be used in conjunction with the acoustic modeling unit or as a post-processing step. If used in conjunction, the language model likelihoods are added (in the log domain) to the acoustic model likelihoods in the token passing algorithm. This is useful because of the pruning that occurs during the search, as the extra information can assist the recognizer in selection of the best paths to maintain or prune. Because of complexity issues the language model is restricted to be a bigram or trigram. If a larger *n* is used, the language model must be applied as a rescoring tool to the lattice, the output data structure described in Section 2.3.6.3.

The size of each model (number of states, value of *Q*) depends on the amount of training data available and the system speed required. The available data and task are important considerations that the designer of any ASR system must take into account.

2.6 Summary

In this chapter, the primary components of a speech recognition system have been discussed. These subsystems, the front-end parameterizer, the acoustic modeling unit, and the language modeling unit, each play a critical role in the process of translating speech

to text. While each system works together, they have been presented here as three independent units.

In later chapters, ASR experiments are run using the Sphinx-4 software package [25]. In this system, the features are extracted as described in Section 2.2. For the experiments in Chapter 4, the specific task is connected digit recognition. Consequently, context-independent word models are used, and no language model is applied. In Chapter 5, the task is medium vocabulary (5000 words) continuous speech recognition. These experiments use context-dependent (triphone) HMM's and apply a trigram language model.

The system described here is basic, meaning that many of the enhancements to ASR systems that have been developed are not discussed in this chapter. Some of these enhancements include vocal tract length normalization (VTLN) [45], speaker and environment adaptation [4], and feature compensation [15-17]. While VTLN and adaptation are not covered in this dissertation, feature compensation is the focus of the next chapter.

Chapter 3 Previous Work in Signal Enhancement and Feature Compensation

In the previous chapter, an automatic speech recognition (ASR) system is described. Over the years, many enhancements have been made to the configuration presented. However, the three subsystems discussed, the front-end parameterizer, the acoustic modeling unit, and the language modeling unit, still form the basis for real state-of-the-art systems.

Under ideal conditions, the system introduced in Chapter 2 performs with high accuracy. Unfortunately, ASR systems often must operate under conditions that are less than ideal. The environment in which the recognizer is used has a large impact on performance, and many real environments cause ASR systems to fail.

Two major environmental factors that degrade system performance are background (or ambient) noise and channel noise. Background noise is the result of sources of sound other than the speaker, whereas channel noise is a distortion caused by the impulse response of the room and recording device (microphone). Background noise is considered to be spectrally additive (with respect to speech). Channel noise is convolutional.

The effect of noise on ASR system performance is especially severe when the environmental conditions present at the time of system parameter training differ from those present when the system is used for recognition. This training/testing mismatch is an issue that has received much attention in ASR research [46].

In this chapter, previous work in the areas of signal enhancement and feature compensation for the improvement of ASR system performance in the presence of noise

is reviewed. In Section 3.1, the effect of noise distortion on speech features is discussed. The estimation of noise statistics from a corrupted signal is presented in Section 3.2, followed by the enhancement of speech signals in Section 3.3. In Section 3.4, a discussion of feature compensation is given, including an introduction to estimation theory, a tool used by many feature compensation algorithms. Finally, the chapter is summarized in Section 3.5.

3.1 Effects of Noise on Speech Features

The two primary types of corrupting noise that ASR systems must handle, background noise and channel noise, can be either stationary or nonstationary. An example of stationary additive noise is white noise, which has a flat power spectrum. Some background noises that are nonstationary (at varying degrees) are automobile, subway, babble, airport, restaurant, and train station. Channel noise is a function of the microphone used.

For additive noise, the ratio of the speech power to the noise power is known as the signal-to-noise ratio (SNR) and is typically expressed in a log scale, with the unit of decibels,

$$SNR_{dB} = 10 \log_{10} \left(\frac{E\left\{ x(t)^2 \right\}}{E\left\{ n(t)^2 \right\}} \right), \tag{3.1}$$

where $x(t)$ and $n(t)$ are zero-mean speech and noise signals, respectively. The SNR is positive if the speech power is greater than the noise power, negative if the converse is true, and zero if the two powers are equal.

Increased levels of noise lead to decreased performance of ASR systems as measured in word accuracy. As an example, Figure 3-1 shows the accuracy of an ASR system similar to the one described in Chapter 2 as a function of SNR when applied to a task known as connected digit recognition. In this task, each utterance consists of a series of spoken digits, zero through nine, as well as the word "oh." Because the words are simply random sequences of numbers, no language model can be used. However, as the number of unique words in the dictionary is small, this is a relatively easy task. The particular data used for this demonstration comes from the Aurora2 dataset [46]. When there is no noise present, the word accuracy rate as defined in Chapter 2 is 99.12%.

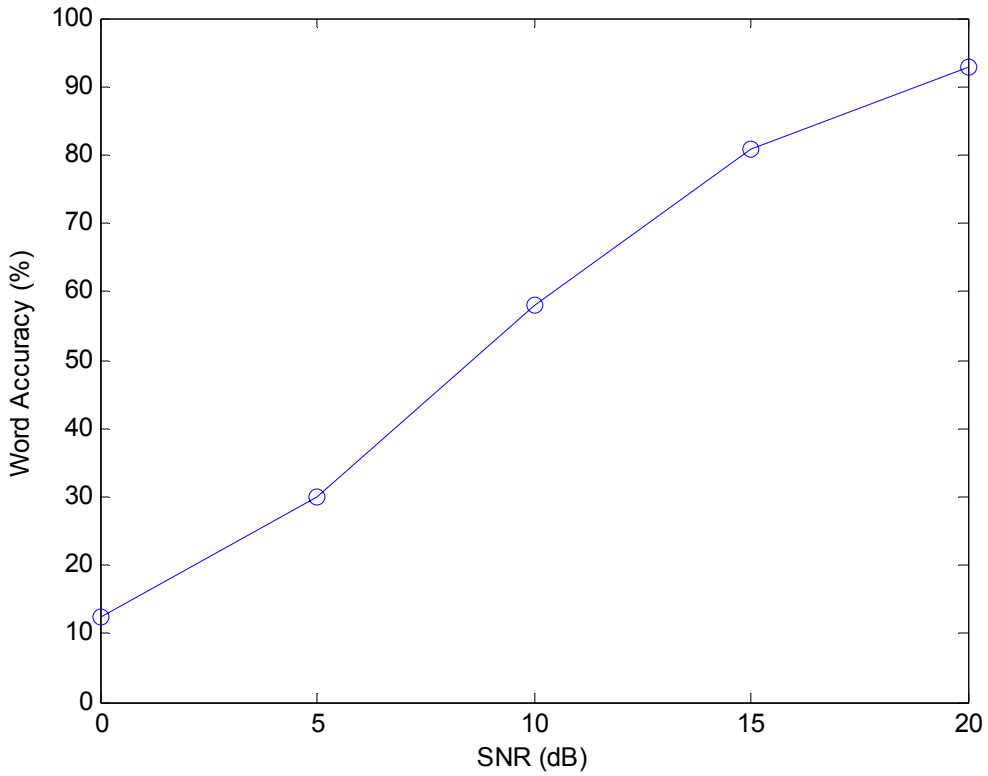

Figure 3-1. Word accuracy vs. SNR for connected digit ASR.

The word accuracies shown in Figure 3-1 are computed by averaging over eight different sets of data, each of which has been corrupted by a different type of noise. As is evident from the graph, the accuracy degrades very quickly as the SNR increases. Even when the speech power remains greater than that of the noise, the system is essentially unusable.

For ASR systems to be useful, the word accuracy in real environments (those in which noise is present) must remain close to that of systems operating in clean environments. Consequently, much effort has been directed at making recognizers more robust to noise, which is the focus of the remainder of this chapter. The work presented in this dissertation builds upon this previous research, following many of the same strategies, but solving some of the shortcomings of the methods described in Section 3.4.

3.1.1 Model for Speech and Noise Interaction

The relationship between a (clean) speech signal, $x(t)$, a noise signal, $n(t)$, the channel impulse response, $h(t)$, and the corrupted signal, $y(t)$, is

$$y(t) = x(t) \otimes h(t) + n(t). \tag{3.2}$$

Here, $\otimes$ represents convolution, and t is a discrete-time sample index. In the spectral magnitude domain, this is expressed as

$$|Y[k]| = |X[k]||H[k]| + |N[k]|. \tag{3.3}$$

In the computation of speech features, namely mel-frequency cepstral coefficients (MFCC's), a triangular filter bank is applied to the spectrum, as described in Chapter 2. Rewriting equation (3.3) after the filter bank application gives

$$\mathbf{y} = \mathbf{x} \bullet \mathbf{h} + \mathbf{n}, \tag{3.4}$$

where each term is a vector of filter bank energy coefficients, and $\bullet$ represents elementwise multiplication. The next step in the parameterization process is the log transform, giving

$$\ln(\mathbf{y}) = \ln(\mathbf{x}\bullet\mathbf{h} + \mathbf{n}). \tag{3.5}$$

The log function in this equation is applied to each vector element individually. Because logarithm function is nonlinear, the right side of (3.5) cannot be split directly. However, the expression inside the log operation can be factored,

$$\ln(\mathbf{y}) = \ln\left[\mathbf{x}\bullet\mathbf{h}\left(\mathbf{i} + \frac{\mathbf{n}}{\mathbf{x}\bullet\mathbf{h}}\right)\right], \tag{3.6}$$

$$\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{h} + \ln\left(\mathbf{i} + e^{\boldsymbol{n}-\boldsymbol{x}-\boldsymbol{h}}\right). \tag{3.7}$$

Here, $\boldsymbol{y} = \ln(\mathbf{y})$, $\boldsymbol{x} = \ln(\mathbf{x})$, $\boldsymbol{n} = \ln(\mathbf{n})$, $\boldsymbol{h} = \ln(\mathbf{h})$, and $\mathbf{i}$ represents the unit vector. This factoring allows the relation given in equation (3.4) to be rewritten in terms of the log filter bank energy coefficients of the four factors. As can be seen from (3.7), the log operation introduces a nonlinearity that can be difficult to handle in the feature domain. As will be seen in a later section, one approach to dealing with this problem uses Taylor series expansions.

The model presented in this section is used by several well-known feature compensation algorithms, including one known as the Vector Taylor Series (VTS) [15] algorithm that is used for empirical comparisons to the new ACDM-MMSE estimator introduced in this dissertation.

3.1.2 Robust Speech Recognition

An ASR system is said to be robust if the performance degradation in the face of adverse conditions is minor. In this dissertation, adverse conditions are considered to be presence of additive and convolutional noise, with the possibility of mismatched noise conditions between the training and testing scenarios.

There are multiple strategies to improving the robustness of the system described in Chapter 2. One approach is to replace or augment the MFCC's with features that are less sensitive to noise corruption. Examples include phase-sensitive features [29], perceptual-based features [13], and multi-band features [47]. Another method involves adaptation of the acoustic model parameters to reduce the mismatch between the HMM parameters and the statistics of the testing environment [15]. The third common strategy for building robust ASR systems relies on compensation of the computed features in an attempt to remove the effect of the noise corruption. This dissertation focuses on this approach.

The goal of feature compensation is to recover the original clean speech information from the corrupted signal. Returning to equation (3.7), the only known quantity is $\boldsymbol{y}$. The desired quantity is $\boldsymbol{x}$, the vector corresponding to the (filtered) spectrum of the clean speech. If this vector can be found successfully with no error, the computed speech features will be equivalent to those computed from an uncorrupted speech signal, and the performance of the recognizer will be equal to that of one operating in a clean environment. Regardless of the particular feature compensation method applied, $\boldsymbol{n}$ and $\boldsymbol{h}$ must first be estimated. In the next section, several approaches to estimating the noise are discussed.

3.2 Estimation of Noise Statistics

Estimates for the additive and convolutional noise factors may be obtained in several different domains, including the spectrum, the filter bank energy domain, the log filter bank energy domain, or the cepstral domain. Estimates generally can be transformed forward along this list with ease (e.g., a spectral estimate can be transformed into a filter bank energy coefficient estimate by application of the filter bank), although the same is not necessarily true of the reverse (because of data reduction, the spectral estimate cannot be obtained directly from the filter bank energies). However, the error distribution of the estimate is likely to vary, depending on the domain of estimation. Consequently, the noise estimation typically is performed in the domain in which the enhancement/compensation of the speech is to be executed.

Several techniques have been developed for estimating the noise spectrum in a corrupted signal, ignoring the channel distortion. These methods generally are performed as the first step in a signal enhancement algorithm. Among these are silence detection and spectrum averaging, minimum statistics [8], and recursive averaging [48].

3.2.1 Silence Detection Methods

Noise estimators that use silence detection are among the simplest available algorithms. In these estimators, the noise is assumed stationary, meaning only one noise spectrum is estimated for the entire signal. Some silence detection algorithm, of which there are many (e.g., [49]), is first applied to the signal. Each frame is labeled as speech or non-speech. Those frames labeled as speech generally contain both speech and noise, whereas those labeled as non-speech consist of noise power only. Following the silence detection, the noise spectrum is estimated by

$$\left|\hat{N}[k]\right| = \frac{1}{C\left(I(l)=1\right)} \sum_{l=0}^{L-1} I(l) \left|Y[l,k]\right|, \tag{3.8}$$

in which l is the frame index, k is the spectral bin index, and $I(l)$ is an indicator variable that is equal to 1 if the frame l is labeled as non-speech and 0 otherwise. The quantity $C\left(I(l)=1\right)$ is the count of frames in which $I(l)=1$.

An even simpler noise estimator, similar to the silence detection approach, uses the average of the first L frames of the signal. This may be performed if it is known that the onset of speech in the utterance does not occur before frame L+1.

3.2.2 Minimum Statistics

The silence detection and spectrum averaging noise estimators are attractive because of their simplicity. However, the assumption of noise stationarity may not hold. If the noise spectrum changes over time, an algorithm that tracks the noise throughout the signal may be more appropriate.

One approach that attempts to track the changing noise envelope is based on minimum statistics [8]. In this approach, power spectral estimates for each frame are obtained using a smoothed periodogram. A noise floor value by frame and by spectral bin is found by taking the minimum spectral value in a fixed window of frames. A multiplicative factor is derived to compensate for a bias in the estimate of the minimum. This approach allows for a smoothing of the noise estimate over time.

3.2.3 Recursive Averaging

Another method that uses smoothing in the context of noise tracking is recursive averaging. In this approach, the noise power estimate is updated recursively, as

$$\left|\hat{N}(l,k)\right| = \alpha\left|\hat{N}(l-1,k)\right| + (1-\alpha)\left|Y(l,k)\right|, \tag{3.9}$$

where l is the frame index, and k is the spectral bin index. The parameter α controls how quickly the noise estimate is allowed to change. An initial estimate for frame 0 must be found first, typically accomplished by assuming the first L frames are noise only and using the average power spectrum of those frames as the estimate.

Depending on the particular algorithm, the noise power estimate may not be updated for each frame. Instead, for some frames, the previous value is used, as $\left|\hat{N}(l,k)\right| = \left|\hat{N}(l-1,k)\right|$. This is done if the frame appears to include speech power. A separate algorithm for determining the probability of speech presence is then needed. One such method, known as the improved minima-controlled recursive average (IMCRA) method [48] uses minimum statistics to accomplish this. This noise estimator is used in the experiments presented in Chapters 4 and 5.

3.2.4 Noise Log Filter Bank Energy Estimators

The methods described so far operate in the spectral domain. Other approaches instead estimate quantities such as the log filter bank energy coefficients. Among these methods are the VTS method [15] and the iterative stochastic approximation to recursive estimation approach of [18]. In these algorithms, a prior model of clean speech is used to aid in the estimation of the noise statistics.

3.3 Signal Enhancement

Speech enhancement is a field that has received much study over the past decades. It is defined as the task of reducing the noise in a corrupted signal while distorting the

speech component of the signal as little as possible. The input and output of a speech enhancement system are both audio signals, but the output should (desirably) be a cleaner signal than the input. A speech enhancement system may be implemented simply to improve the quality of speech signals, or to make the speech more intelligible. An enhancement system can also be used as part of the front-end to an ASR system, with the goal of improving the robustness of the recognizer to noise.

Some of the more popular traditional speech enhancement algorithms include spectral subtraction [21], Wiener filtering [50], and Ephraim-Malah filtering [7]. Each of these methods operates on the spectral amplitudes of frames of the signal, with the assumption that the phase of the clean signal is equivalent to the corrupted signal. After framing is applied, a fast Fourier transform (FFT) is taken, and the power or magnitude spectrum is found. The spectral values are modified according to the respective algorithm, followed by reconstruction of the (framed) time signal, using the enhanced spectral amplitudes and the original phase values. The entire signal is then rebuilt by adding overlapping frames that have been windowed to ensure they sum correctly [51].

3.3.1 Spectral Subtraction

Spectral subtraction [21] is possibly the simplest speech enhancement method that makes use of a noise spectral estimate. In this approach, a frame of speech is enhanced in the spectral domain by subtracting the estimated noise spectral amplitude values from the corrupted spectral amplitude values according to

$$\left|\hat{X}[l,k]\right| = \left|Y[l,k]\right| - \left|\hat{N}[l,k]\right|, \tag{3.10}$$

where $\left|\hat{X}[l,k]\right|$ and $\left|\hat{N}[l,k]\right|$ are the estimates for the clean speech and noise amplitudes at spectral bin *k* in frame *l*, respectively, and $\left|Y[l,k]\right|$ is the known amplitude for the corrupted speech at bin *k* in frame *l*. Typically, the noise estimate is constant over *l*. Following the computation of the estimate for the clean speech spectra, the estimates for the clean speech values are combined with the corrupted phase to reconstruct the time signal for each frame. The new signal is then constructed from these enhanced frames.

One disadvantage of the spectral subtraction method is the introduction of musical noises into the enhanced signal. Because the noise power is sometimes overestimated, it can be greater than the corrupted signal power. As this can yield a negative value for the clean speech estimate and cannot be allowed, a spectral floor must be implemented. This occasionally injects artificial tones into the enhanced signal. Many modifications to the original spectral subtraction algorithm to deal with this problem have been studied (e.g., [52]). The next two sections discuss two other popular approaches to enhancement that do not add musical tones.

3.3.2 Wiener Filtering

The linear filter that optimally estimates a clean signal in the mean-square error sense under the situation of additive stationary noise is known as a Wiener filter. In the magnitude spectral domain, a Wiener filter is

$$H[k] = \frac{X[k]}{X[k] + N[k]}. \tag{3.11}$$

If the speech and noise are jointly wide-sense stationary stochastic processes, this filter is said to be the linear minimum mean-square error (MMSE) estimator for the speech

signal. While speech and noise signals are never truly stationary, each is assumed to be approximately stationary within a frame of the signal. Therefore, as in the spectral subtraction method, the filter is applied on a frame-by-frame basis, and the frame index *l* must be incorporated into (3.11). Unfortunately, to compute equation (3.11), the true values for the speech and noise power must be known. As the quantities for neither component are available, some modification must be made.

The noise is first estimated using one of the techniques described in Section 3.2. Then, an *a priori* estimate for the speech power must be found. One possibility is to use the corrupted signal power, $Y[k]$. Another is to compute the maximum likelihood estimate for $X[k]$,

$$\hat{X}[l,k] = \max\left\{0, \frac{1}{l+1}\sum_{i=0}^{l} Y[l-i,k] - \hat{N}[l-i,k]\right\}. \tag{3.12}$$

Yet another option is recursive estimation by a method known as the decision-directed approach [6]. In this method, the *a posteriori* value (computed from (3.11)) for the estimate of $\hat{X}[l-1,k]$ is combined with the ratio $\frac{Y[l,k]}{\hat{N}[l,k]}$ in a weighted average to obtain the *a priori* estimate for the speech power at time *l*. A modified version of this decision-directed Wiener filter is used as a component of the estimation system introduced in Chapter 4.

Equation (3.11) may also be applied iteratively. In this manner, the *a posteriori* estimate for $X[l,k]$ from iteration *j* is used as the *a priori* estimate in (3.11) for iteration *j*+1.

3.3.3 Ephraim-Malah Filtering

An alternative MMSE estimator for use in speech enhancement is the spectral amplitude estimator developed by Ephraim and Malah [6]. They derive the MMSE spectral amplitude estimator, assuming that the speech and noise complex frequency coefficients are normally distributed. The estimate for $\left|X\left[l,k\right]\right|$ is derived using the variances of the speech and noise prior distributions (per spectral index). The values for the noise variances are obtained by one of the noise estimators described previously, and the speech variances are determined through either a maximum likelihood estimate or through the decision-directed approach.

Although this estimator provides a rigorous solution to the optimal estimation of spectral amplitudes, a distance measure between two spectra in the linear domain does not correlate well with human perception. Instead, minimizing the distance between the estimate and the true spectrum in the log spectral domain is more appropriate. Consequently, the model in [6] is modified in [7] to operate in the log domain. The final estimation result in this work is

$$\left|\hat{X}[l,k]\right| = \left\{\frac{\xi_{t,k}}{1+\xi_{t,k}} e^{\int_{\upsilon_{t,k}}^{\infty} \frac{e^{-t}}{t} dt}\right\} \left|Y\left[l,k\right]\right|, \tag{3.13}$$

where $\xi_{t,k}$ and $\upsilon_{t,k}$ are the *a priori* and *a posteriori* SNR estimates, respectively, and are obtained in the same manner as in [6].

This optimal estimator has received much study since its introduction. Many recent speech enhancement algorithms are based on the work of Ephraim and Malah,

with further developments of the various components of the filter, such as the estimation of the *a priori* and *a posteriori* SNR estimates [53].

3.3.4 Signal Enhancement for Robust Speech Recognition

Given an estimate for the noise power of a corrupted speech signal and a signal enhancement algorithm, a reduction in the noise component of the corrupted signal is possible. The purpose of the enhancement may be to improve the quality or intelligibility of a signal for audio playback, but the algorithms described in this section may also be used as front-ends to an ASR system operating in noisy environments. In this approach, once the corrupted signal is enhanced, the standard speech features are computed. Recognition is performed with the acoustic and language models, resulting in a hypothesized transcription. If the enhancement system is of high quality, the accuracy of the transcription is expected to be higher than a system that does not include any front-end enhancement.

While this is a perfectly reasonable approach to building an ASR system that is more robust to background noise, it is not necessarily the best approach. Instead, enhancement performed in the domain closest to that used by the recognizer is expected to be more desirable. Consequently, many recognition system front-ends use feature compensation to improve their robustness. In these systems, rather than reducing the noise corruption in the spectrum directly, the speech features are modified to remove the effects of the noise. One major advantage of this strategy is that it allows prior knowledge of human speech to be integrated into the front-end more efficiently. In the next section, various feature compensation procedures are discussed.

3.4 Feature Compensation

Many different strategies have been developed for compensation of features, including statistical normalization [5], codeword-dependent normalization [16], stereo-based compensation [15], and optimal feature estimation without stereo-data [15, 17]. In the following sections, previously developed algorithms based on each of these approaches are discussed.

3.4.1 Statistical Normalization

The methods for feature enhancement discussed to this point have dealt primarily with the additive component of noise corruption. A popular technique for dealing with channel distortion is known as cepstral mean subtraction (CMS) or cepstral mean normalization (CMN). Channel noise is modeled as convolutional in the time domain, and subsequently multiplicative in the spectral domain,

$$Y[l,k] = X[l,k]H[l,k]. \tag{3.14}$$

If the channel impulse response is assumed to be time-invariant, (3.14) is expressed as

$$Y[l,k] = X[l,k]H[k]. \tag{3.15}$$

Because of the log operation in the cepstral feature computation, the product of the speech and channel powers becomes additive in the cepstral domain. Since the channel component of the features is constant over time, the bias introduced by the channel can be removed by de-meaning each feature dimension in an utterance. If the channel response is not truly time-invariant, but moves slowly with respect to the speech signal, a sliding window can be applied to the features of an utterance, de-meaning the feature dimensions within each window.

CMN, which is the normalization of the first-order statistic of cepstral features, is frequently applied as a post-processing step to the computation of MFCC's. In some systems, normalization of statistics of orders higher than one is applied as well. Cepstral variance normalization (CVN), in addition to CMN, has been shown to improve the robustness of ASR systems to noise, especially when an environmental mismatch between training and testing is present. Higher-order (third-order and up) normalization of cepstral coefficients has been explored as well [54].

The different types of cepstral moment normalization do not make use of any estimates of the additive or convolutional noise. Many compensation algorithms that take these quantities into account have been developed. They typically are based on statistical estimation of features. Before the different types of feature estimation algorithms are presented, a brief discussion of parameter estimation is given.

3.4.2 Parameter Estimation

Parameter estimation [55] is important in any task that involves statistical modeling. Given a set of data that is believed to have been sampled from a probability distribution, a model can be fit to the data, provided the form of the underlying distribution is known (or can be well-approximated with a known distribution). To accomplish this task of model fitting, the distribution parameters must be estimated.

3.4.2.1 Maximum Likelihood Estimators

Several types of estimators are available, each with different properties. One of the most commonly used types is the maximum-likelihood (ML) estimator. For a random

variable z with a distribution defined by a parameter set Θ and a set of independent observations $z_1, z_2, ..., z_n$, the likelihood function is

$$L(z_1, z_2, ..., z_n \mid \Theta) = \prod_{i=1}^{n} p_\Theta(z_i), \tag{3.16}$$

where $p_\Theta(z)$ is either the probability density function (pdf) (for continuous distributions) or probability mass function (pmf) (for discrete distributions). The parameter values for which this function is maximized give the maximum likelihood estimate of the true parameters. To make this maximization easier, the log-likelihood function $LL(Z \mid \Theta) = \ln L(Z \mid \Theta)$ is often used in replacement. Because the logarithmic function is monotic, the argument that maximizes the likelihood function is the argument that maximizes the log-likelihood function. The estimate of the parameters is

$$\hat{\Theta} = \arg\max_{\Theta} \ln\left(\prod_{i=1}^{n} p_\Theta(z_i)\right) = \arg\max_{\Theta} \sum_{i=1}^{n} \ln p_\Theta(z_i). \tag{3.17}$$

The parameters can be found by taking the partial derivative of (3.17) with respect to Θ, setting it equal to 0, and solving

$$\frac{\partial \sum_{i=1}^{n} \ln p_\Theta(z_i)}{\partial \Theta} = 0. \tag{3.18}$$

One major advantage to ML estimators is that they often yield computationally simple solutions.

3.4.2.2 Bayes Estimators

Another type of estimator is known as the Bayes estimator. In this estimation approach, values for the parameter set are found to minimize the expected value of a cost function. While the cost function used is chosen based on the particular task, three

popular cost measures are mean-squared error, absolute error, and the delta function. These functions are given in equations (3.19), (3.20), and (3.21), respectively, where θ represents the true value of the parameter to be estimated, and $\hat{\theta}$ is the estimated value. The parameter δ in (3.21) is chosen arbitrarily.

$$C\left(\theta,\hat{\theta}\right)=\left(\theta-\hat{\theta}\right)^2, \tag{3.19}$$

$$C\left(\theta,\hat{\theta}\right)=\left|\theta-\hat{\theta}\right|, \tag{3.20}$$

$$C\left(\theta,\hat{\theta}\right)=\begin{cases}0 & \left|\theta-\hat{\theta}\right|\le\delta\\ 1 & \text{else}\end{cases}. \tag{3.21}$$

Given a cost function and the posterior distribution $p\left(\theta\,|\,Z\right)$, the Bayes risk is

$$R\left(\hat{\theta}\right)=\int_{\theta}C\left(\theta,\hat{\theta}\right)p\left(\theta\,|\,Z\right)\,d\theta. \tag{3.22}$$

The value that minimizes this risk is the estimate, $\hat{\theta}$. Because the posterior distribution is typically unavailable, Bayes' theorem must be invoked, as

$$p\left(\theta\,|\,Z\right)=\frac{p\left(Z\,|\,\theta\right)p\left(\theta\right)}{p\left(Z\right)}. \tag{3.23}$$

For the mean-squared error cost function, the risk function is minimized by the mean of the posterior distribution and is known as a minimum mean-squared error (MMSE) estimator, computed by

$$\hat{\theta}=E\left[\theta\,|\,Z\right]=\frac{\int_{\theta}p\left(Z\,|\,\theta\right)p\left(\theta\right)\,d\theta}{p\left(Z\right)}. \tag{3.24}$$

The estimate that minimizes the absolute error cost is known as a minimum absolute error (MAE) estimate and is computed by the median of the posterior. Typically,

this must be calculated numerically, a trait that makes this type of estimator undesirable in some situations due to computational cost.

An estimator with the delta cost function, known as a maximum *a posteriori* (MAP) estimator, is found as the maximum of the posterior,

$$\hat{\theta} = \arg\max_{\theta} \frac{p(Z \mid \theta)\, p(\theta)}{p(Z)} = \arg\max_{\theta} p(Z \mid \theta)\, p(\theta). \tag{3.25}$$

As the prior distribution of the data is not of function of the parameter, it is irrelevant to the computation of the MAP estimate.

As is described in some of the following sections on feature compensation, many feature estimators use the MMSE estimator. This is done for two reasons. First, empirical results have shown MMSE estimators generally outperform other estimates (specifically the MAP estimator) [56]. Second, the smoothness of the cost function in the MMSE estimator is intuitively desirable compared to the discontinuous function in the MAP estimator, as small perturbations may cause large differences in accumulated cost.

The next three sections describe different classes of feature estimators.

3.4.3 Codeword-Dependent Cepstral Normalization

Acero [16] originally developed a method for incorporating *a priori* information of speech into an MMSE estimator of cepstral coefficients in a discrete HMM ASR system. In this approach, a prior distribution of clean speech is used, represented as a Gaussian mixture model (GMM). The probability density function (pdf) of the speech is

$$p(\mathbf{c}) = \sum_{m=0}^{M-1} P[m]\, \mathcal{N}_c(\mu_m, \Sigma_m), \tag{3.26}$$

where $\boldsymbol{\mu}_m$ and Σ_m are the mean and covariance matrix for the cepstral vector of the m^{th} mixture, respectively, and $P[m]$ is the weight of mixture m. To ensure that this is a valid pdf, the weights must sum to unity. The distribution parameters are learned over a set of clean speech data with the well-known expectation maximization algorithm. Each vector $\boldsymbol{\mu}_m$ represents a codeword in cepstral space, leading to the name for the algorithm, codeword-dependent cepstral normalization (CDCN).

In CDCN, an MMSE estimator for the clean (desired) cepstral vector, **c**, is computed as a function of the observation (corrupted cepstral vector), **d**, the additive noise cepstral vector, **e**, and the channel response cepstral vector, **q**,

$$\hat{\mathbf{c}}_{MMSE} = E[\mathbf{c} \mid \mathbf{d}, \mathbf{e}, \mathbf{q}] = \frac{\sum_{m=0}^{M-1} P[m] \int \mathbf{c}\, p(\mathbf{d} \mid \mathbf{c}, \mathbf{e}, \mathbf{q}, m)\, p(\mathbf{c} \mid m)\, d\mathbf{c}}{\sum_{m=0}^{M-1} P[m] \int p(\mathbf{d} \mid \mathbf{c}, \mathbf{e}, \mathbf{q}, m)\, p(\mathbf{c} \mid m)\, d\mathbf{c}}. \tag{3.27}$$

The noise vectors, **e** and **q**, are assumed to be stationary for the entire utterance. The solution to (3.27) is

$$\hat{\mathbf{c}}_{MMSE} = \mathbf{d} - \sum_{m=0}^{M-1} f[m] \left\{ \Sigma_m (\Sigma_m + \Gamma)^{-1} (\mathbf{q} + \mathbf{r}[m]) + \Gamma (\Sigma_m + \Gamma)^{-1} (\mathbf{d} - \boldsymbol{\mu}_m) \right\}. \tag{3.28}$$

Here, $f[m]$ is a measure of *a posteriori* probability of mixture m given the acoustic observation **d** and the environmental parameters. The matrix Γ is the covariance matrix of the noise statistics, computed along with **e** and **q** using an iterative maximum likelihood estimation algorithm. The correction vector is

$$\mathbf{r}[m] = IDFT \left\{ \ln \left(1 + e^{DFT[\mathbf{e} - \mathbf{q} - \mathbf{u}_m]} \right) \right\}. \tag{3.29}$$

The correction vector for each codeword is constant, i.e., not a function of the local SNR in the test utterance.

3.4.4 Stereo-Based Feature Compensation

The CDCN algorithm does not take into account the instantaneous SNR of the signal and approximates the correction vector, $r[m]$, as in (3.29) by using the mean of a Gaussian in place of the clean speech cepstrum itself. Subsequently, a modification to this algorithm, known as fixed codeword-dependent cepstral normalization (FCDCN), was introduced by Acero and Stern [57]. In FCDCN method, stereo-training data is used. Here, each utterance in the training set has multiple versions: one clean signal and many noisy signals. The clean signal component is the same for each version; only the noise component differs.

The computation for the correction vector from (3.29) is replaced by an iterative estimation algorithm that is carried out during training. The new correction vectors, which are codeword and SNR-dependent, are found by maximizing the likelihood of the corrupted data (in the cepstral domain), given the clean speech data, each correction vector, and the estimated instantaneous SNR.

One disadvantage of the FCDCN method is that the covariance matrices of the Gaussians in the clean speech model are assumed to be equivalent. To relax that assumption, Moreno et al. [58] developed the multivariate-Gaussian-based cepstral normalization (RATZ) method for feature compensation. Like the previous methods, clean speech is represented by a GMM in the cepstral domain. The corrupted speech statistics are modeled as shifted versions of the clean speech cepstral vectors

$$\boldsymbol{\mu}_{d,m} = \boldsymbol{\mu}_{c,m} + \mathbf{r}_m , \tag{3.30}$$

$$\Sigma_{d,m} = \Sigma_{c,m} + R_m , \tag{3.31}$$

in which $\boldsymbol{\mu}_{c,m}$ and $\boldsymbol{\mu}_{d,m}$ are the clean and corrupted speech means for the m^{th} Gaussian mixture, $\Sigma_{c,m}$ and $\Sigma_{d,m}$ are the clean and corrupted speech (diagonal) covariance matrices for the m^{th} Gaussian mixture, and $\mathbf{r}_m$ and R_m are the mixture dependent correction factors. With stereo training data, these factors are computed by

$$\mathbf{r}_m = \frac{\sum_{l=0}^{L-1} P[m \mid \mathbf{c}_l](\mathbf{d}_l - \mathbf{c}_l)}{\sum_{t=0}^{T-1} P[m \mid \mathbf{c}_l]}, \tag{3.32}$$

$$R_m = \frac{\sum_{l=0}^{L-1} P[m \mid \mathbf{c}_l]\left((\mathbf{d}_l - \boldsymbol{\mu}_{c,m} - \mathbf{r}_m)(\mathbf{d}_l - \boldsymbol{\mu}_{c,m} - \mathbf{r}_m)^T\right)}{\sum_{t=0}^{T-1} P[m \mid \mathbf{c}_l]} - \Sigma_{c,m}. \tag{3.33}$$

Here, $P[m \mid \mathbf{c}_l]$ is the *a posteriori* probability of mixture *m* given the clean cepstral vector at frame *l.* The summation in each equation is computed over all frames in the stereo data training set. As only one correction factor pair ($\mathbf{r}_m$ and R_m) exists per Gaussian mixture, the noise statistics are assumed to be constant.

Once the correction factors are computed, an MMSE estimate of the clean speech vector, $\mathbf{c}$, is

$$\hat{\mathbf{c}}_{MMSE} = \mathbf{d} - \int_C \mathbf{r}(\mathbf{c}) p(\mathbf{c} \mid \mathbf{d})\, d\mathbf{c} = \mathbf{d} - \int_C \sum_{m=0}^{M-1} \mathbf{r}(\mathbf{c})\, p(\mathbf{c}, m \mid \mathbf{d})\, d\mathbf{c}. \tag{3.34}$$

Under the assumption that the correction vector is well approximated by a set value for a particular region in the cepstral space, i.e., $\mathbf{r}(\mathbf{c}) \cong \mathbf{r}_{c,m}$,

$$\hat{\mathbf{c}}_{MMSE} \cong \mathbf{d} - \sum_{m=0}^{M-1} \mathbf{r}_m P[m \mid \mathbf{d}]. \tag{3.35}$$

Alternatively, the parameters of the acoustic models can be adjusted using $\mathbf{r}_m$ and R_m. This approach is referred to as a STAR algorithm [15].

If no stereo training data is available, the equations for computing the correction factors must be modified and made iterative. This approach is named Blind RATZ [15] and is more general than RATZ, though it does not perform as well if stereo data is indeed available. Another enhancement to the basic RATZ algorithm involves the inclusion of local SNR estimates [15].

A further extension to the RATZ formulation is the noise-normalized stereo-based piecewise linear compensation (SPLICE) algorithm [18]. The correction vector used in this approach is

$$\mathbf{r}_m = \frac{\sum_{l=0}^{L-1} P\left[m \mid \mathbf{c}_t - \hat{\mathbf{n}}_l\right]\left(\mathbf{d}_l - \mathbf{c}_l\right)}{\sum_{l=0}^{L-1} P\left[m \mid \mathbf{c}_l - \hat{\mathbf{n}}_l\right]}, \tag{3.36}$$

where $\hat{\mathbf{n}}_l$ is the estimate of the noise cepstrum for frame l. Empirical tests show improvement in performance of the compensation algorithm when the noise normalization is included [18].

3.4.5 Optimal Estimation without Stereo Data

The algorithms presented in the previous section work well when stereo training data is available and the noise found in the training data is similar to that in the test data. Additionally, with the exception of the noise-normalized SPLICE algorithm, the stereo-based methods do not require explicit noise estimation. However, in practice, stereo training data is not always available. Also, the noise conditions between the training and testing data frequently differ, reducing the effectiveness of these algorithms.

Two methods that have been developed for optimal estimation of speech features given an estimate of the noise in the test utterance are the Vector Taylor Series (VTS) method [15] and the MMSE estimator introduced by Deng et al. [17].

The development of the VTS estimator starts with the acoustic distortion model

$$\boldsymbol{y} = \boldsymbol{x} + g(\boldsymbol{x}, \boldsymbol{n}, \boldsymbol{h}), \tag{3.37}$$

where $\boldsymbol{y}$, $\boldsymbol{x}$, $\boldsymbol{n}$, and $\boldsymbol{h}$ are the log filter bank energy vectors for the corrupted speech, clean speech, noise, and channel response, respectively, and

$$g(\boldsymbol{x}, \boldsymbol{n}, \boldsymbol{h}) = \boldsymbol{h} + \ln\left(\mathbf{i} + e^{\boldsymbol{n}-\boldsymbol{x}-\boldsymbol{h}}\right). \tag{3.38}$$

Either the zeroth- or first-order vector Taylor series expansion is the applied to (3.37), expanding around $\boldsymbol{x}_0$, $\boldsymbol{n}_0$, and $\boldsymbol{h}_0$ (each expansion point is in the log domain). The order of expansion depends on the version of VTS: zeroth-order for VTS-0 and first-order for VTS-1. The first order expansion is

$$\begin{aligned}\boldsymbol{y} = \boldsymbol{x} + g(\boldsymbol{x}, \boldsymbol{n}, \boldsymbol{h}) \cong \boldsymbol{x} + g(\boldsymbol{x}_0, \boldsymbol{n}_0, \boldsymbol{h}_0) + \nabla_{\boldsymbol{x}} g(\boldsymbol{x}_0, \boldsymbol{n}_0, \boldsymbol{h}_0)(\boldsymbol{x} - \boldsymbol{x}_0) \\ + \nabla_{\boldsymbol{n}} g(\boldsymbol{x}_0, \boldsymbol{n}_0, \boldsymbol{h}_0)(\boldsymbol{n} - \boldsymbol{n}_0) + \nabla_{\boldsymbol{h}} g(\boldsymbol{x}_0, \boldsymbol{n}_0, \boldsymbol{h}_0)(\boldsymbol{h} - \boldsymbol{h}_0).\end{aligned} \tag{3.39}$$

As in many of the previously discussed feature compensation algorithms, the clean speech is modeled with a GMM, and the noise and channel response are assumed stationary. An MMSE estimator for the clean speech, given the corrupted speech, can be found as

$$\hat{\boldsymbol{x}}_{MMSE} = E[\boldsymbol{x} \mid \boldsymbol{y}] = \int_X \boldsymbol{x}\, p(\boldsymbol{x} \mid \boldsymbol{y})\, d\boldsymbol{x}, \tag{3.40}$$

$$\hat{\boldsymbol{x}}_{MMSE} = \sum_{m=0}^{M-1} P[m \mid \boldsymbol{y}] \int_X \boldsymbol{x}\, p(\boldsymbol{x} \mid m, \boldsymbol{y})\, d\boldsymbol{x}. \tag{3.41}$$

After using the Taylor series approximation with $\boldsymbol{x}_0 = \boldsymbol{\mu}_{m,x}$ and fixing the noise and channel estimates so that $\boldsymbol{n} = \boldsymbol{n}_0$ and $\boldsymbol{h} = \boldsymbol{h}_0$, equation (3.41) becomes

$$\begin{aligned}\hat{\boldsymbol{x}}_{MMSE} \cong \boldsymbol{y} &- \sum_{m=0}^{M-1} P[m \mid \boldsymbol{y}] \int_X g\left(\boldsymbol{\mu}_{m,\boldsymbol{x}}, \boldsymbol{n}, \boldsymbol{h}\right) p\left(\boldsymbol{x} \mid m, \boldsymbol{y}\right) d\boldsymbol{x} \\ &- \sum_{m=0}^{M-1} P[m \mid \boldsymbol{y}] \int_X g'\left(\boldsymbol{\mu}_{m,\boldsymbol{x}}, \boldsymbol{n}, \boldsymbol{h}\right)\left(\boldsymbol{x} - \boldsymbol{\mu}_{m,\boldsymbol{x}}\right) p\left(\boldsymbol{x} \mid m, \boldsymbol{y}\right) d\boldsymbol{x}.\end{aligned} \tag{3.42}$$

The solution to (3.42) is

$$\hat{\boldsymbol{x}}_{MMSE} \cong \boldsymbol{y} - \sum_{m=0}^{M-1} P[m \mid \boldsymbol{y}]\, g\left(\boldsymbol{\mu}_{m,\boldsymbol{x}}, \boldsymbol{n}, \boldsymbol{h}\right). \tag{3.43}$$

A similar result is found for the zeroth-order expansion.

The estimator introduced by Deng et al. [17] begins with an equation similar to (3.37). In this approach, the speech and noise signals are not assumed to be uncorrelated. Consequently, an extra term is present:

$$\boldsymbol{y} = \boldsymbol{x} + g\left(\boldsymbol{x}, \boldsymbol{n}, \boldsymbol{h}\right) + \boldsymbol{r}. \tag{3.44}$$

The term $\boldsymbol{r}$ is a residual that is modeled as a multivariate Gaussian random vector with zero mean and covariance Ψ. The covariance matrix is learned over a set of corrupted speech data.

Again, the clean speech is modeled as a GMM. The MMSE estimator for $\boldsymbol{x}$ is

$$\hat{\boldsymbol{x}}_{MMSE} = \frac{\sum_{m=0}^{M-1} P[m] \int \boldsymbol{x}\, p\left(\boldsymbol{x} \mid m\right) p\left(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{n}\right) d\boldsymbol{x}}{p\left(\boldsymbol{y}\right)}. \tag{3.45}$$

This method deals only with additive noise, and so the channel response is ignored. Assuming that $\boldsymbol{y}$ and $\boldsymbol{n}$ are known quantities and representing (3.44) with a zeroth-order Taylor series approximation expanded around $\boldsymbol{x}_0$, (3.45) is rewritten as

$$\hat{\boldsymbol{x}}_{MMSE} = \frac{\sum_{m=0}^{M-1} P[m] \int \boldsymbol{x} \mathcal{N}_{\boldsymbol{x}}\left(\boldsymbol{\mu}_{m,\boldsymbol{x}}, \Sigma_{m,\boldsymbol{x}}\right) \mathcal{N}_{\boldsymbol{x}}\left(\boldsymbol{x} + \mathbf{g}(\boldsymbol{x}_0, \boldsymbol{n}), \Psi\right) d\boldsymbol{x}}{p(\boldsymbol{y})}. \tag{3.46}$$

The closed form solution for (3.46) is

$$\hat{\boldsymbol{x}}_{MMSE} = \frac{\sum_{m=0}^{M-1} P[m] N_m(\boldsymbol{y}) \left[\mathbf{W}_1(m) \mu_{m,\boldsymbol{x}} + \mathbf{W}_2(m)\left(\boldsymbol{y} - \mathbf{g}(\boldsymbol{x}_0, \boldsymbol{n})\right)\right]}{\sum_{m=0}^{M-1} P[m] N_m(\boldsymbol{y})}, \tag{3.47}$$

in which

$$\mathbf{W}_1(m) = \left(\Sigma_{m,\boldsymbol{x}} + \Psi\right)^{-1} \Psi, \tag{3.48}$$

$$\mathbf{W}_2(m) = \left(\Sigma_{m,\boldsymbol{x}} + \Psi\right)^{-1} \Sigma_{m,\boldsymbol{x}}, \quad \text{and} \tag{3.49}$$

$$N_m(\boldsymbol{y}) = \mathcal{N}_y\left(\mu_{m,\boldsymbol{x}} + \mathbf{g}(\boldsymbol{x}_0, \boldsymbol{n}), \Sigma_{m,\boldsymbol{x}} + \Psi\right). \tag{3.50}$$

Because of the zeroth-order Taylor series expansion, the choice of the expansion point, $\boldsymbol{x}_0$, is important. To make the estimator less sensitive to poor choices of this point, the algorithm is made iterative by updating the expansion point with the newly estimated value from (3.47).

Deng et al. enhance this static estimator by modifying it to include a prior model of speech with dynamic (first-difference) features. This enhancement leads to further improvement of the feature estimation [17].

While each of the feature compensators presented here have solid theoretical foundations and work reasonably well in practice, each has its shortcomings. The possibilities for improvement upon these methods are discussed in the final section of this chapter.

3.5 Summary

In this chapter, a review of previous work in the development of robust ASR systems through enhancement of speech features is presented. The procedures used for noise estimation and tracking are summarized, as well as the basic speech enhancement techniques used in practice. Many of the most popular feature compensation approaches have also been discussed. While these approaches produce large improvements over the baseline with no noise removal, the recognition accuracy obtained with these methods remains inferior to that of systems operating in clean conditions.

A common thread throughout the feature compensation algorithms is the use of a prior model of clean speech as a GMM. The use of a prior model improves the likelihood that the estimated features look like actual human speech. With a prior model in the estimation formulation, the particular compensation algorithm produces either an MAP or MMSE estimate of the speech features. However, because of the log transformation in the MFCC computation, closed-form solutions are not easy to develop in either the MAP or MMSE case. To deal with this, the approaches presented in this chapter must use approximations. One of these approximations is the use of a Taylor series expansion. To ensure the fidelity of the approximation, accurate expansion points must be chosen. Because this is not easily done, these algorithms are often made iterative, causing an increase in the necessary processing.

Additionally, though the prior models are useful, they remain quite general. The models are trained over all speech sounds in all possible contexts and over all speakers. The prior distribution does not contribute any extra information about the context of the

speech signal in question. If more information were available, more reliable estimates of the features may be obtained.

These issues are addressed by the work in the remainder of this dissertation. In the next chapter, a distortion model that differs from that used in many of the feature compensators in this chapter (equation (3.7)) is introduced and used to derive an MMSE estimator of MFCC's. This estimator does not make use of the Taylor series expansion and is non-iterative. In Chapter 5, an alternative prior model is proposed and studied. This "advanced prior model" uses transcription hypotheses from a full recognition system to obtain more informative prior distributions of speech features.

Chapter 4 MMSE Estimator of Features with a Novel Distortion Model

In the previous chapter, several approaches to combating performance degradation of automatic speech recognition (ASR) systems due to ambient noise are discussed. Many of the more successful approaches attempt to estimate true clean speech features given a noisy speech signal, often in the log spectral domain [15, 16, 18]. In the log spectral domain, the interaction between the speech and noise signals is nonlinear, resulting in high complexity of compensation models even when the speech and noise signals are assumed independent. A common method for dealing with this issue uses a Taylor series expansion to make the compensation algorithm tractable [15, 17]. As a result, the reliability of the estimator depends on the choice of an expansion point. Because the optimal expansion point is not known *a priori*, the algorithm may become iterative, resulting in increased computation time. Additionally, a method for finding a reasonable initial expansion point is required.

This chapter introduces a new method for enhancing speech features, based on a minimum mean-squared error (MMSE) estimator of mel-frequency cepstral coefficients (MFCC's). This method addresses many of the issues explained above. The estimation procedure is non-iterative and requires no Taylor series approximation. Additionally, the estimator works entirely in the cepstral domain, alleviating the need for inversion of the discrete cosine transform (DCT) matrix, as is seen in several methods developed previously. Because the number of cepstral coefficients typically is less than the number of filter bank coefficients, the DCT matrix has no true inverse. Thus, a pseudoinverse is needed, introducing possible inaccuracies in the estimator.

The new estimator is developed using a novel approach, which treats speech and noise signal estimates as random variables, to modeling the interaction between speech and noise. As a result, the new method models the noise distortion as additive in the cepstral domain, leading to a closed-form solution to the estimation problem. The model is developed using filter bank energy coefficients of the speech and noise signals to match the computation of MFCC features. The first assumption is that the filter bank energy coefficients are Gamma distributed. The second is that the distortion of cepstral coefficients has a Gaussian distribution. These two key assumptions, which are justified in the following section, lead to a tractable solution for the estimator.

In this chapter, the new additive cepstral distortion model (ACDM) is presented, and the MMSE estimator for the MFCC's is derived. The practical issues of the algorithm in the context of a robust ASR system are discussed as well. The ACDM-MMSE estimator is compared to traditional baselines, in which no noise removal is implemented, and to the Vector Taylor Series (VTS) algorithm [15]. Also, a theoretical comparison of the ACDM-MMSE estimator and the VTS algorithm is given in the Appendix, highlighting the differences between the two methods.

4.1 MMSE Estimation of MFCC features

This section develops the new ACDM-MMSE estimator, briefly reviewing parameter estimation and relating it to the task of estimating MFCC features. The distortion model is presented and used for derivation of the final ACDM-MMSE estimator.

4.1.1 Statistical parameter estimation

As discussed in the previous chapter, several approaches for the statistical parameter estimation given a set of data are available, depending on the error measure used. The most common types of estimators are maximum likelihood (ML), maximum *a posteriori* (MAP), minimum mean-squared error (MMSE), and minimum absolute error (MAE). Previous research has shown that MMSE estimators tend to result in superior enhancement over other estimators, when used for estimation of speech features [56]. Consequently, the proposed method uses the MMSE estimator, which minimizes the expected mean-squared error between the estimate of a parameter, $\hat{z}$ and its true value, z, expressed as

$$\hat{z} = \arg\min_{\hat{z}} E\left[\left(z - \hat{z}\right)^2\right]. \tag{4.1}$$

Recall from Section 3.4.2.2 that the value that realizes this minimum is the mean of the conditional distribution of the parameter,

$$\hat{z} = E\left[z \mid D\right], \tag{4.2}$$

where D is the given data. Using the definition of the expected value, equation (4.2) is

$$\hat{z} = \int_{-\infty}^{\infty} z p\left(z \mid D\right) dz. \tag{4.3}$$

The distribution $p\left(z \mid D\right)$ typically is not directly available. Thus, Bayes' theorem is invoked, yielding

$$\hat{z} = \frac{\int_{-\infty}^{\infty} z p\left(D \mid z\right) p\left(z\right) dz}{p\left(D\right)} = \frac{\int_{-\infty}^{\infty} z p\left(D \mid z\right) p\left(z\right) dz}{\int_{-\infty}^{\infty} p\left(D \mid z\right) p\left(z\right) dz}. \tag{4.4}$$

In equation (4.4), the conditional, $p(D \mid z)$, describes the distribution of the data given a particular value of z. If the system that generates the data is known and is a function of z, a mathematical model for this distribution typically can be developed. The prior, $p(z)$, describes the distribution of z given no other information. If the data distribution, $p(D)$, is not available, the total probability theorem [38] is used to replace $p(D)$ with

$$\int_{-\infty}^{\infty} p(D \mid z)\, p(z)\, dz.$$

4.1.2 Estimation of MFCC features

The estimator described above in equation (4.4) is applied to the estimation of speech features. The parameter to be estimated is **c**, the vector of clean (desired) MFCC's, and the given data is the vector of distorted MFCC's, **d**. Thus, the expectation equation of (4.2) becomes

$$\hat{\mathbf{c}} = E[\mathbf{c} \mid \mathbf{d}], \tag{4.5}$$

where **c** is the clean cepstral coefficient vector, and **d** is the distorted vector. Following the derivation of the previous section, (4.5) is transformed into

$$\hat{\mathbf{c}} = \frac{\int_{-\infty}^{\infty} \mathbf{c} p(\mathbf{d} \mid \mathbf{c})\, p(\mathbf{c})\, d\mathbf{c}}{p(\mathbf{d})}. \tag{4.6}$$

Previous research has used a Gaussian mixture model (GMM) to represent the prior distribution, $p(\mathbf{c})$, [15, 17] and that approach is used in this work as well. This GMM is built by training over a large set of clean speech using the expectation maximization algorithm [19]. This prior is a model that represents the expected distribution of features

computed from human speech signals. The prior model reduces undesired distortion to the features caused by poor estimates of the noise signal by forcing the estimated features to more closely conform to feature patterns of real speech.

The conditional represents the distribution of the distorted features given the true features. A model that relates the distorted feature vector to the clean feature vector through an estimate of the noise is necessary for the development of the estimator. In the next section, a distortion model used to represent the conditional distribution is proposed.

4.1.3 Novel statistical distortion model

The ACDM is derived by representing the true speech filter bank coefficients as a function of the distorted filter bank coefficients and a gain vector,

$$\mathbf{x} = \mathbf{g} \bullet \mathbf{y}, \tag{4.7}$$

where **x** and **y** are the clean and distorted speech filter bank energy coefficient vectors for a frame of speech, respectively, **g** is the appropriate gain vector, and $\bullet$ represents element-wise multiplication. In the log domain, the relationship becomes

$$\ln(\mathbf{x}) = \ln(\mathbf{g}) + \ln(\mathbf{y}), \tag{4.8}$$

where the log operation of a vector is

$$\ln(\mathbf{z}) = \begin{bmatrix} \ln(z_0) \\ \ln(z_1) \\ \vdots \\ \ln(z_n) \end{bmatrix}. \tag{4.9}$$

Multiplication of both sides of (4.8) by a discrete cosine transform matrix, $\mathbf{\Lambda}$, yields

$$\mathbf{\Lambda}\ln(\mathbf{x}) = \mathbf{\Lambda}\ln(\mathbf{g}) + \mathbf{\Lambda}\ln(\mathbf{y}). \tag{4.10}$$

Since $\mathbf{\Lambda}\ln(\mathbf{x})$ and $\mathbf{\Lambda}\ln(\mathbf{y})$ are, by definition, equivalent to **c** and **d**, respectively, substitution of these terms and rearrangement gives

$$\mathbf{d} = \mathbf{c} - \mathbf{\Lambda}\ln(\mathbf{g}). \tag{4.11}$$

The term $\mathbf{\Lambda}\ln(\mathbf{g})$ represents the additive distortion in the cepstral domain. The gain vector, **g**, is treated as a random variable vector, allowing the form for the conditional distribution $p(\mathbf{d}|\mathbf{c})$ to be found, provided the distribution of **g** is known. The conditional $p(\mathbf{d}|\mathbf{c})$ is assumed to be Gaussian, ensuring that the MMSE estimator of (4.6) has a closed-form solution. This distributional assumption is justified by the central limit theorem [38], as $p(\mathbf{d}|\mathbf{c})$ is formed as a linear combination of random variables. The number of variables in the summation that produces the conditional distribution is 23, the size of the filter bank, which is a value that generally is sufficient to produce distributions that are very close to Gaussian [38]. The mean and variance of the conditional are

$$\boldsymbol{\mu}_{\mathbf{d}|\mathbf{c}} = \mathbf{c} - E\left[\mathbf{\Lambda}\ln(\mathbf{g})\right], \tag{4.12}$$

$$\mathbf{\Sigma}_{\mathbf{d}|\mathbf{c}} = E\left[\mathbf{\Lambda}\ln(\mathbf{g})^2\right] - E\left[\mathbf{\Lambda}\ln(\mathbf{g})\right]^2. \tag{4.13}$$

Since the gain variables are assumed to be independent across frequency bins, $\mathbf{\Sigma}_{\mathbf{d}|\mathbf{c}}$ is a diagonal matrix. The conditional distribution of the gain variables is determined by using a linear MMSE estimator, the Wiener filter, to represent the gain

$$g_k = \frac{x_k}{x_k + n_k}, \tag{4.14}$$

where x and n are the k^{th} filter bank energy coefficients of the speech and noise signals, respectively. Ideally, x_k is a known quantity, since **c** is given. However, the values for **x** cannot be fully recovered from **c**, as the discrete cosine transform used is not necessarily

invertible. A least-squares fit transform of the coefficients back into the log spectral domain is possible, but inclusion of that transform into the estimator would require that the algorithm become iterative. Therefore, as an approximation, x_k and n_k are both treated as random variables, and are assumed to be gamma distributed. The probability density function for a gamma distribution is given by

$$f(x) = \frac{x^{\alpha-1} e^{-x/\beta}}{\Gamma(\alpha)\beta^{\alpha}}, \quad x \geq 0, \tag{4.15}$$

where α and β are the two distribution parameters, and $\Gamma(z)$ is the complete gamma function [59]. The distribution is defined only for nonnegative values. The mean and variance of a gamma distribution are

$$\mu = \alpha\beta, \tag{4.16}$$

$$\sigma^2 = \alpha\beta^2. \tag{4.17}$$

Gamma distributions often have been used to model speech time samples and spectra in prior work [60-62]. An empirical goodness-of-fit test over clean condition training data confirms that the gamma distribution has a chi-squared statistic an order of magnitude better than a normal, uniform, exponential, or Rayleigh distribution. If x_k and n_k are assumed to have independent gamma distributions with parameters ($\alpha_{x,k}$, β) and ($\alpha_{n,k}$, β), respectively, g_k can be shown to be beta distributed, with

$$p(g_k) = \frac{\Gamma(\alpha_{x,k} + \alpha_{n,k})}{\Gamma(\alpha_{x,k})\Gamma(\alpha_{x,k})} (1 - g_k)^{\alpha_{n,k}-1} g_k^{\alpha_{x,k}-1}, \tag{4.18}$$

where $\alpha_{x,k}$ and $\alpha_{n,k}$ are parameters derived from the distributions of x and n. The beta value must be equivalent for the two gamma distributions. The distribution for $\ln(g_k)$ is

known as the exponential beta distribution, and the mean and variance are computed by [63]

$$\mu_{\ln g_k} = \psi_0(\alpha_{x,k}) - \psi_0(\alpha_{x,k} + \alpha_{n,k}), \tag{4.19}$$

$$\sigma^2_{\ln g_k} = \psi_1(\alpha_{x,k}) - \psi_1(\alpha_{x,k} + \alpha_{n,k}), \tag{4.20}$$

where ψ_0 and ψ_1 are the digamma and trigamma functions, respectively [64]. The final form of $p(\mathbf{d}\,|\,\mathbf{c})$ is

$$p(\mathbf{d}\,|\,\mathbf{c}) = \mathcal{N}\left(\mathbf{d}; \mathbf{c} - \boldsymbol{\Lambda}\boldsymbol{\mu}^g, \left|\boldsymbol{\Lambda}\right|\Sigma^g\right), \tag{4.21}$$

where $\boldsymbol{\mu}^g$ and Σ^g are the mean and variance vectors computed using (4.19) and (4.20). Inserting (4.21) into (4.6), and using the same procedure found in [17], the MMSE estimator is fit into a standard quadratic form, and a closed form solution for the estimator is

$$\hat{\mathbf{c}} = \sum_{m=1}^{M} \gamma_m \left[\mathbf{W}_1(m)\boldsymbol{\mu}_m^c + \mathbf{W}_2(m)\left(\mathbf{d} + \boldsymbol{\Lambda}\boldsymbol{\mu}^g\right)\right] \tag{4.22}$$

$$\mathbf{W}_1(m) = \left|\boldsymbol{\Lambda}\right|\Sigma^g \left(\Sigma_m^c + \left|\boldsymbol{\Lambda}\right|\Sigma^g\right)^{-1}, \tag{4.23}$$

$$\mathbf{W}_2(m) = \Sigma_m^c \left(\Sigma_m^c + \left|\boldsymbol{\Lambda}\right|\Sigma^g\right)^{-1}. \tag{4.24}$$

where $\boldsymbol{\mu}_m^c$ and Σ_m^c are the mean vector and covariance matrix of the GMM used for the prior model $p(\mathbf{c})$, and

$$\gamma_m = \frac{w_m p(\mathbf{d}\,|\,m)}{\sum_{m=1}^{M} w_m p(\mathbf{d}\,|\,m)}. \tag{4.25}$$

As in [17],

$$p(\mathbf{d}\,|\,m) = \mathcal{N}\left(\mathbf{d}; \boldsymbol{\mu}_m^c + \boldsymbol{\Lambda}\boldsymbol{\mu}^g, \Sigma_m^c + \left|\boldsymbol{\Lambda}\right|\Sigma^g\right). \tag{4.26}$$

Examination of (4.22) shows that the final ACDM-MMSE estimator is essentially a weighted average between the mean of the distortion compensation factor $\mathbf{d} + \boldsymbol{\Lambda}\boldsymbol{\mu}^g$ and the mean of each component in the prior model, where the weighting is determined by the ratio of variances of the conditional and prior distributions. If the prior model is assumed to be uniform, the estimator is essentially equivalent to applying a Wiener filter, although the computation is performed in the cepstral domain. The inclusion of the prior in the estimator forces the estimate to match more closely a pattern of actual speech.

Because the trigamma function is monotonically decreasing for positive numbers, for a given α_x, the variance computed in (4.20) increases along with α_n. Thus, as the estimated signal-to-noise ratio decreases, this variance increases, and more weight is given to the prior model in the estimation of the features. This is desirable, as it is expected that the accuracy of the distortion compensation factor will be worse for lower SNR values. The use of the prior model then becomes especially important, so that the negative effects caused by the inaccuracy of the distortion factor are not as detrimental to the estimate of the features, and subsequently the robustness of the recognition system.

4.2 Algorithm implementation

The ACDM-MMSE estimator derived in the previous section requires knowledge of the parameters of the distributions of the speech and noise filter bank energy coefficients to compute the $\boldsymbol{\mu}^g$ and Σ^g (mean and variance) terms of (4.22). *A priori* estimates for the speech and noise power are generated using noise estimation and spectral estimation algorithms. The improved minima-controlled recursive averaging (IMCRA) method [48] described in the previous chapter is used for estimation of the noise power.

For the spectral estimate, a decision-directed generalized Wiener filter [65] is implemented. The form of the filter is

$$H(\omega) = \frac{S_x(\omega)}{S_x(\omega) + \rho S_n(\omega)}, \tag{4.27}$$

where ρ is a multiplier that controls additional noise suppression. Because of the increased noise suppression, the filter in (4.27) sometimes significantly underestimates the speech power. Consequently, the filter in (4.27) is bounded, resulting in the modified form

$$H(\omega) = \frac{S_x(\omega)}{S_x(\omega) + \min\{\rho S_n(\omega), S_y(\omega)\}}, \tag{4.28}$$

where $S_y(\omega)$ is the spectral power value computed from the distorted speech signal. Additionally, the spectral estimate obtained using this filter is smoothed in frequency using a normalized window. The Wiener filtering and smoothing process is defined by

$$\hat{X}(\omega) = \sum_{i=-l}^{l} b(i)\left[H(\omega) S_y(\omega)\right], \tag{4.29}$$

where the coefficients $b(i)$ must sum to unity. The value used for l is one.

Once the *a priori* speech and noise estimates are generated, the parameters α_x and α_n are computed. The values for the noise and speech estimates, which are obtained by application of the IMCRA algorithm and the modified Wiener filter, respectively, are converted from spectral coefficients to filter bank energy coefficients by applying a Mel-spaced triangular filter bank. The resulting values for the k^{th} filter banks of the speech and noise are treated as the means of the gamma distributions for x_k and n_k. Using the definitions of the mean and variance for a gamma distribution from (4.16), the alpha parameters are

$$\alpha_{x,k} = \frac{\hat{x}_k}{\beta}, \tag{4.30}$$

$$\alpha_{n,k} = \frac{\hat{n}_k}{\beta}, \tag{4.31}$$

where $\hat{x}_k$ and $\hat{n}_k$ are the *a priori* estimates of the speech and noise filter bank energy coefficients, and β is treated as a free parameter. Once the values of alpha are computed, the values for the mean vector, $\boldsymbol{\mu}^g$, and the variance matrix, Σ^g, are then found using (4.19) and (4.20). These mean and variance measures, and subsequently the estimated values for the MFCC features, are affected by the choice of β. It has been observed that the choice of an appropriate β is important for success of the estimation algorithm, and that the computation of the mean is affected adversely by a poor choice of β more than the variance. Because of this sensitivity, in the implementation of the algorithm the computation of the mean from (4.19) is replaced by

$$\mu_{g,k} = \frac{\hat{x}_k}{\hat{x}_k + \hat{n}_k}. \tag{4.32}$$

This approach can be viewed as treating the speech and noise *a priori* estimates as deterministic instead of stochastic for the purpose of estimation of the mean of the conditional distribution. However, the variance of the conditional is still derived using the statistical assumptions developed in the previous sections. Empirical observations have indicated that it is beneficial to bound the variance computed in (4.22) to reduce the impact of outliers. The choice of the parameter β and the bounds are discussed in the next section.

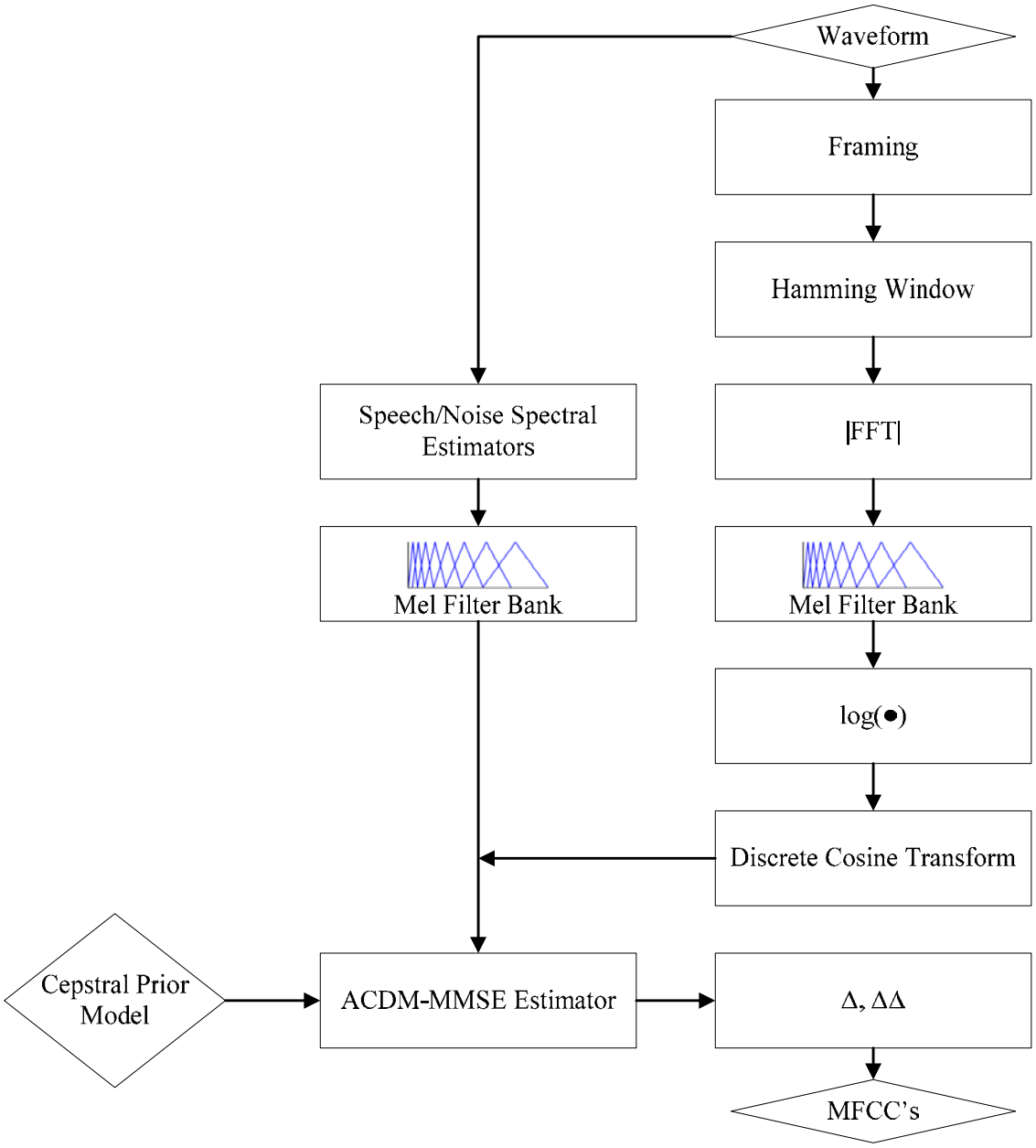

Figure 4-1. Block diagram for ACDM-MMSE estimation front-end.

The new estimation front-end, as depicted in Figure 4-1, is implemented to estimate the static cepstral coefficients, including c_0. The first and second derivative coefficients are then computed from the estimated features. While it is possible to compute all parameters for (4.22), including first and second derivatives, it has been

observed that doing so provides recognition accuracy improvement over the approach of estimating static coefficients only.

4.3 Experiments

To evaluate the ACDM-MMSE estimator, speech recognition experiments are conducted on data that is artificially corrupted by several different types of noises at varying signal-to-noise ratios (SNR's). The next section describes Aurora2 [46], the dataset used for experimentation, while the following section describes the ASR system configuration and algorithm parameters. The results, which show significant improvement in recognition accuracy over multiple baselines, are given after that.

4.3.1 Aurora2

Aurora2, which is derived from the TIDIGITS database [66], contains a large set of connected digit utterances from a number of American speakers, both male and female. Each utterance is a string of spoken numbers from zero to nine, plus "oh." The original signals were recorded with a close-talking Sennheiser microphone in a clean environment to ensure high quality signals. The data is divided into training and testing sets. The signals are sampled at 8 kHz, corresponding to a typical telephone bandwidth. Two training sets exist: the clean-condition set and the multi-condition set. The test data is split into sets A, B, and C. No speaker has utterances in both training and testing sets, making the task speaker-independent.

The clean-condition training set consists of 8,440 utterances of high quality speech data. The multi-condition set was created by artificially add to each signal (except

a portion that are left uncorrupted) in the clean-condition set one of four noises (subway, babble, car, and exhibition hall) at an SNR level between +5 and +20 dB.

Eight different noise types are found in the test sets: those present in the multi-condition training set and restaurant, street, airport, and train station noises. Test set A contains the first four, test set B contains the next four, and test set C contains both subway and street noises. The noise corruption in sets A and B is additive only. In test set C, the signals are filtered with the MIRS characteristic [46] to introduce convolutional distortion. For each noise type in each test set, 1,001 unique utterances are present at SNR levels of -5, 0, +5, +10, +15, and +20 dB. Clean versions of the signals are available as well. It is standard in the literature to report results only for SNR levels of 0 dB and up, so the -5 dB data is not used here.

4.3.2 ASR System Configuration

The software used for experimentation is the Sphinx-4 recognition system, a package developed entirely in Java at Carnegie Mellon University in conjunction with Sun Microsystems and the Mitsubishi Electric Research Laboratories [25]. In each system evaluated, the feature vector consists of 39 coefficients: 13 static MFCC's (including c_0), 13 delta coefficients, and 13 delta-delta coefficients. Cepstral mean subtraction (CMS) is applied as a post-processing step to all front-ends except the first baseline.

Word-level acoustic models are used, meaning that each word, plus silence, is represented by a single, context-independent Hidden Markov Model (HMM). The model for each word is a 16 state left-to-right model, while silence is represented by a 3 state model with a backward transition from the third to the first state. The GMM for each

state in the word models have 3 mixtures, while the silence model GMM's have 6 mixtures. The training procedure uses the standard script available with the Aurora2 data.

The algorithm parameters that must be set prior to running experiments with the ACDM-MMSE estimator are determined through experimentation on a development set. The development set is constructed as a subset of the multi-condition training set. The model parameters used for experimentation during parameter tuning are learned over the clean-condition training set exclusively.

The parameters tuned using the development set include the number of mixtures in the prior model, the value for β, the bounds on the variance of the conditional described in section 4.2, and the value for ρ in equation (4.27). The optimal number of mixtures (that which gives the greatest word accuracy) is 16. The value for β is 9000, the lower and upper bounds are 1.1 and 4.5, respectively, and the value for ρ is 4.

For the VTS algorithm, two configurations are used, one with 16 mixtures to match the ACDM-MMSE estimator and one with 256 mixtures to match the setup in [15]. While the VTS approach includes a mechanism for estimation of the noise power spectrum, the IMCRA algorithm is used instead to make the setup for the VTS approach more comparable to the ACDM-MMSE system.

4.3.3 Results

Experimental results are reported for both clean-condition and multi-condition trained models. For the multi-condition experiments, the prior models are first trained on the clean-condition data. The multi-condition data is then enhanced with either the ACDM-MMSE or VTS estimator. This data is used to train the acoustic models that are used in the recognition of the test data.

Because of the difference in number of mixtures, the ACDM-MMSE estimation algorithm is significantly faster than the 256 mixture VTS algorithm. Recognition experiments for the 256 mixture VTS method run in approximately 2.7x real time, compared to around 0.5x real time for the proposed estimation method. The experimental time for the 16 mixture VTS system is comparable to the ACDM-MMSE system.

Table 4-1 shows the results in word accuracy for each system evaluated using clean-condition trained models. The values presented are averaged over all noise types and SNR levels from 0 to 20 dB in each individual test set. The first baseline is a simple MFCC front-end system with no CMS. The second baseline is similar, though CMS is applied. Each value reported for the ACDM-MMSE system is significantly greater than the values of all other systems in a statistical sense using a type I error of 0.05.

Front-end	Set A	Set B	Set C	Overall
ACDM-MMSE	81.64%	83.05%	82.03%	82.24%
No enhancement (baseline 1)	56.56%	52.98%	66.77%	58.77%
CMS enhancement only (baseline 2)	65.17%	70.67%	66.39%	67.41%
VTS (256 mixture prior)	78.88%	80.11%	78.68%	79.22%
VTS (16 mixture prior)	67.00%	70.26%	68.03%	68.43%

Table 4-1. Average word accuracies for proposed estimator and baseline front-ends using clean-condition trained acoustic models on Aurora2.

The proposed estimator outperforms all baselines, including both versions of VTS. Inspection of the results for each of the test subsets (by noise type and SNR) for clean-condition training indicates that the proposed system gives superior performance over the both the standard feature set and VTS estimated features in all SNR levels 15 dB or lower and nearly equal numbers at the 20 dB SNR level. To see the effect of the prior

distribution, recognition also is run using the modified Wiener filter described in (4.28) as the front-end. The overall accuracy for Wiener filter front-end is 77.33%, compared to 82.24% accuracy for the ACDM-MMSE system, showing that inclusion of the prior model results in an absolute error reduction of 4.91%.

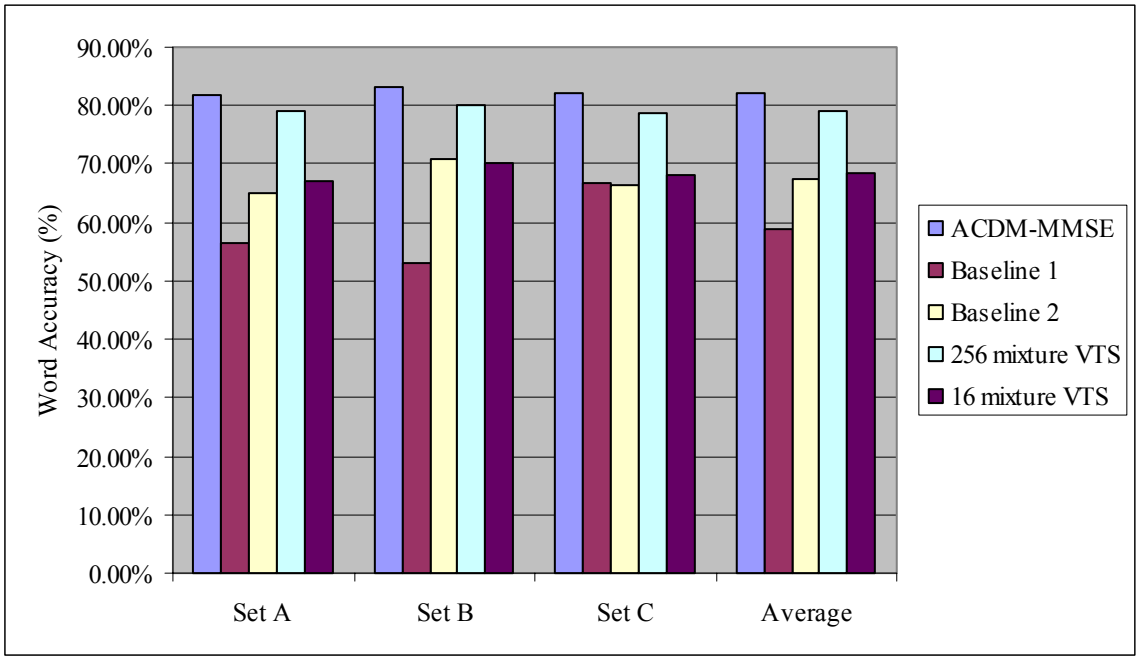

Figure 4-2. Bar chart of average word accuracies for proposed estimator and baseline front-ends using clean-condition trained acoustic models on Aurora2.

Each algorithm is also tested on clean data. The accuracy for the ACDM-MMSE estimation method is 98.50%, compared to 98.97% for the VTS and a baseline of 99.12%. While the proposed method causes some degradation in accuracy on clean data, the amount is relatively small.

Results for the multi-condition experiments are presented in Table 4-2. The relative improvement seen in these experiments is smaller than that of the clean-condition experiments, but the improvement seen is still consistent. All values for the ACDM-MMSE system are significantly greater than the values for all other systems in a

statistical sense using a type I error of 0.05. A modified Wiener front-end system gives an overall accuracy of 89.27% here, showing that inclusion of the GMM prior model results in an absolute error reduction of 0.47%. The VTS algorithm does not perform well on this task, actually decreasing the word accuracy in comparison to the baseline.

Front-end	Set A	Set B	Set C	Overall
ACDM-MMSE	89.80%	89.54%	89.87%	89.74%
No enhancement (baseline 1)	87.28%	85.75%	84.79%	85.94%
CMS enhancement only (baseline 2)	88.63%	88.66%	89.24%	88.84%
VTS (256 mixture prior)	85.04%	84.73%	85.42%	85.06%
VTS (16 mixture prior)	81.33%	82.22%	81.99%	81.85%

Table 4-2. Average word accuracies for ACDM-MMSE estimator and baseline front-ends using multi-condition trained acoustic models on Aurora2.

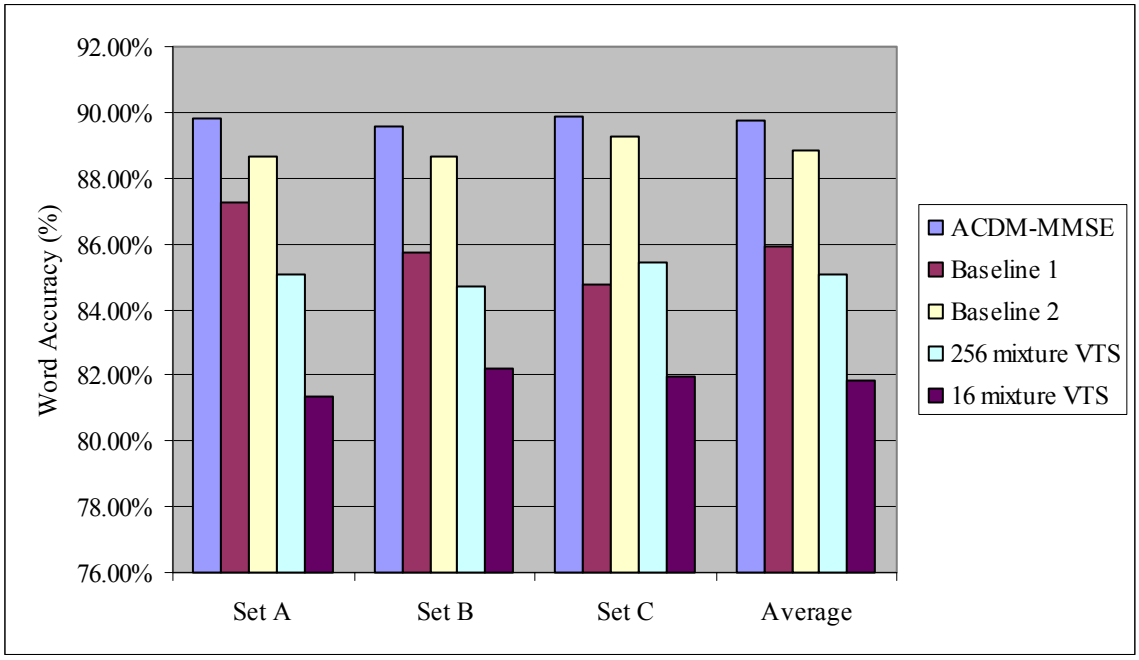

Figure 4-3. Bar chart of average word accuracies for ACDM-MMSE estimator and baseline front-ends using multi-condition trained acoustic models on Aurora2.

As stated Section 4.2, the ACDM-MMSE estimator has a free parameter, β, which controls the scaling of the conditional variance. To study the sensitivity of the algorithm to this parameter, a series of clean-condition recognition experiments are run, varying the value for β. A range of values from 10 to 50,000 is used, spaced logarithmically. The minimum and maximum accuracies, averaged over all test sets, are 79.91% and 82.80%. The lowest accuracy is a result of β = 50,000, and all other accuracies are within 0.4% of the maximum. This indicates that, provided the β value is not excessively large, the ACDM-MMSE estimator is robust to variances in the actual value of β.

4.4 Summary

The ACDM-MMSE estimator has been introduced and evaluated on the Aurora2 dataset. This approach models the noise distortion as additive in the cepstral domain and uses assumptions on the statistical distribution of the speech and noise in the spectral domain to derive the MMSE estimator. Unlike some previous approaches to estimation of speech features, the ACDM-MMSE algorithm is not iterative. Additionally, the estimation is performed entirely in the cepstral domain.

Experimental results show that the ACDM-MMSE estimator outperforms a baseline MFCC front-end and the well-known VTS algorithm on corrupted speech data. The new estimator also runs faster than the VTS algorithm with 256 mixtures, as used in [15].

The success of the estimation algorithm depends primarily on the quality of three components: the *a priori* noise power estimates, the *a priori* speech power estimates, and the cepstral prior model. The IMCRA algorithm is used for the noise estimate, and a generalized Wiener filter is used for estimation of speech.

The prior model used is a simple GMM trained over a large set of clean speech. The prior's major contribution is to ensure that the enhanced cepstral values are reasonable (that is, they resemble actual speech). However, the prior model does not differentiate between different classes of phonemes, such as vowels and fricatives. Instead, all frames of speech use the same prior model, which is a conglomeration of different classes of phonemes. If the prior model is made more specific for each individual frame of speech (i.e., a vowel model used for frames that are probably vowels, a fricative model for frames that are probably fricatives, etc.), it is probable that the estimator produces more accurate features. In the next chapter, a framework for the inclusion of more advanced prior modeling into the estimator is presented.

Chapter 5 Advanced Prior Modeling in the ACDM-MMSE Estimator

In the previous chapter, an automatic speech recognition (ASR) system front end that estimates the values of the true mel-frequency cepstral coefficients (MFCC's), given the corrupted MFCC features computed from a noisy speech signal, is developed. This front end, known as the additive cepstral distortion model minimum mean squared error (ACDM-MMSE) estimator, makes the assumption that the filter bank energy coefficients computed for each frame of speech are composed of uncorrelated speech and noise energies. The estimator also assumes that these energies can be modeled as gamma distributed random variables and that the distortion on the cepstral coefficients can be modeled as a multivariate Gaussian distribution with a diagonal covariance. These assumptions lead to a closed form solution for the estimator, given that *a priori* estimates of the speech and noise signals are available.

A significant component of this estimator is the prior distribution model. The form of the prior used so far is a single Gaussian Mixture Model (GMM) trained over a large set of clean speech features, following the approach adopted in [15-17]. While reasonable, this form of prior is limited; it contains no phoneme-specific or language model information regarding the likely distribution of the features. In this chapter, a more advanced prior model is implemented by generating GMM's from the acoustic models used by the recognizer. The model is built by running a full recognition pass, followed by regeneration of the prior using the transcription hypotheses.

In the next section, the potential for improvement over the base estimator presented in Chapter 4 is studied through the generation of "ideal" priors the using the

correct transcription for each utterance. Section 5.2 develops an iterative algorithm for re-composition of the priors with the Gaussian components of the acoustic model states extracted from the transcription hypotheses. Section 5.3 introduces an alternative approach to generating advanced priors, in which a class-based approach to advanced prior modeling is used. Finally, a summary of this chapter is given in Section 5.4.

5.1 Ideal prior model generation

To examine the potential improvement that can be achieved by replacing the global GMM prior model with a more informative prior, an upper bound is sought. This is done by using the correct transcription of the utterance to generate the true state occupancy likelihoods for each individual frame. Given the trained acoustic models, the forward-backward algorithm described in Chapter 2 is used to determine these likelihoods.

The state occupancy likelihoods are used in one of two ways to generate the ideal prior models. In the first method, referred to as re-composition, the likelihoods are used as multiplicative weighting factors on the corresponding state distributions (from the acoustic models), and all possible state distributions are combined to form a single GMM for each frame. In the second method, a form of winner-take-all, the GMM corresponding to the most likely state for each frame is chosen as the new prior model. The net effect in each approach is to create an optimized GMM for each frame based on knowledge of the true transcription. The Hidden Markov Models (HMM's) used for this purpose are trained over clean speech data only.

Figure 5-1 depicts the process of re-composition of the prior models from the set of acoustic models for a particular utterance. Every frame in the utterance has an

associated set of ordered HMM states (each state occupies one or more contiguous frames), each with a state occupancy likelihood, $p\left(s_t^i\right)$, (which determines the ordering) computed through the forward-backward algorithm. A GMM is associated with each state, learned with the Baum-Welch training algorithm discussed in Chapter 2. For each frame, a new GMM is re-composed by collecting and re-weighting the entire set of Gaussians from all GMM's in that frame. The new weight for the m^{th} Gaussian in the re-composed GMM is

$$w'_m(t) = p\left(s_t^i\right) w_l^i(t), i = 0, \ldots, N,\ m = 0, \ldots, M,\ l = 0, \ldots, L, \tag{5.1}$$

where i is the state index, t is the frame index, N is the number of total states in the acoustic model, L is the number of mixtures in each original state, and M is the number of mixtures in the re-composed GMM. Note that $m = l * i$.

The second method of ideal prior generation can be viewed as a degenerate form of the first method. In this approach, only the most likely state in each frame is used for re-composition. Consequently, the new mixture weights are simply equivalent to the old weights that appear in the acoustic model.

Given the new GMM prior for each frame, equation (4.22) is used to produce enhanced speech features. A full recognition pass is run using the enhanced features. Of course, this method of prior generation is unusable in practice, since the true word sequence for the utterance must be known, but it can be used to demonstrate the value of a more informative prior model in a statistical speech feature estimator.

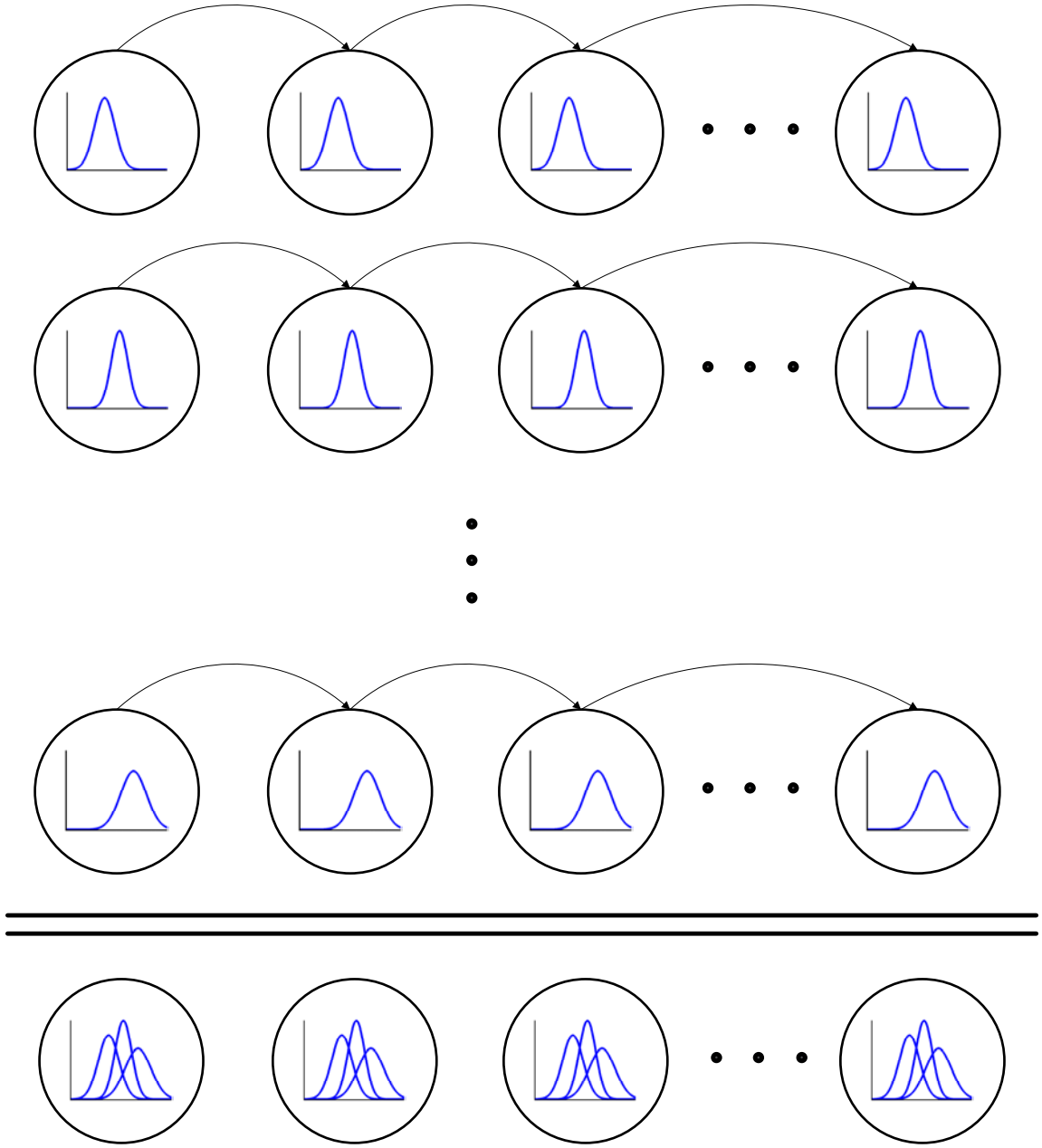

Figure 5-1. Graphical depiction of GMM re-composition method.

5.1.1 Evaluation

A set of ASR experiments test the possibility for improvement that can be gained using an advanced prior model over the base estimator introduced in chapter 4. These evaluation experiments are conducted on the standard speech recognition dataset Aurora4

[67], which is derived from another standard dataset, the Wall Street Journal 5000 word task, or WSJ-5k [68]. In the development WSJ-5k, a number of speakers, both male and female, were asked to read selected sentences from various copies of the Wall Street Journal. The utterances were recorded with a close-talking Sennheiser microphone to ensure high quality of signals. The data is partitioned into training and test sets in a manner such that no speaker has utterances in both partitions. The vocabulary for the test set is restricted to a predefined dictionary of 5000 words.

Aurora4 was built by artificially degrading the test set in WSJ-5k (after duplication) by six different recorded noises. The level of degradation, which is purely additive, varies by utterance. The noise power for each utterance was chosen randomly such that the signal-to-noise ratio of the test signals follows a uniform distribution with bounds at +5 and +15 dB. The six noises used are car, babble (cocktail party), street, restaurant, airport, and train.

Table 5-1 shows the results of the experiments conducted to evaluate the potential of incorporating advanced priors into the ACDM-MMSE estimator. Four systems are tested on the six different test sets in Aurora4, each of which consists of 166 utterances. Each system uses the same back-end, which consists of the acoustic and language modeling units. Each acoustic model is a context-dependent (triphone) three-state left-to-right HMM. There are a total of 4,402 unique states, tied using a phonetic decision tree. A 16-mixture GMM is associated with each state, trained over clean speech data only. The language model used is a publicly available trigram model [69], intended specifically for the WSJ-5k task.

The signals are sampled at 8 kHz, corresponding to a typical telephone bandwidth. In each system, the feature vector consists of 39 coefficients: 13 static MFCC's (including c_0), 13 delta coefficients, and 13 delta-delta coefficients. Cepstral mean subtraction (CMS) is applied as a post-processing step to all front-ends. All experiments are run with the Sphinx-4 recognition system, a software package developed entirely in Java at Carnegie Mellon University, in conjunction with Sun Microsystems and the Mitsubishi Electric Research Laboratories [25].

The four different results, given in word accuracy (%), correspond to four different front-ends: A) a baseline system with no feature compensation, B) the base ACDM-MMSE estimator with a global prior model, C) an ACDM-MMSE estimator with frame-dependent prior models re-composed from the acoustic models using the state occupancy likelihoods obtained with the forward-backward algorithm, and D) an ACDM-MMSE estimator with prior models taken directly from the most likely acoustic model state for each frame as determined with forward-backward.

Front-end	Car	Babble	Street	Restaurant	Airport	Train
A	45.6%	32.2%	29.3%	33.7%	33.7%	30.2%
B	50.4%	50.5%	48.5%	45.6%	44.8%	50.7%
C	62.8%	64.7%	61.9%	62.8%	62.0%	60.6%
D	66.5%	65.5%	63.2%	65.4%	64.6%	62.8%

Table 5-1. Word accuracy over the six noise test sets for Aurora4 for A) a baseline system, B) the ACDM-MMSE system with a global prior, C) the ACDM-MMSE system with re-composed advanced priors, and D) the ACDM-MMSE system with advanced priors taken directly from the top state.

As can be seen from Table 5-1, the ACDM-MMSE estimator with the global GMM prior consistently outperforms the baseline system across all noise types. The

ACDM-MMSE front-end with the re-composed GMM prior offers further improvement, with word accuracies of approximately 10-17% increase in absolute accuracy. When only the most likely state as computed from the forward-backward algorithm is used as the prior, another 1-3% accuracy is gained. These results indicate that the value of a more informative prior (as compared to the global prior) is substantial.

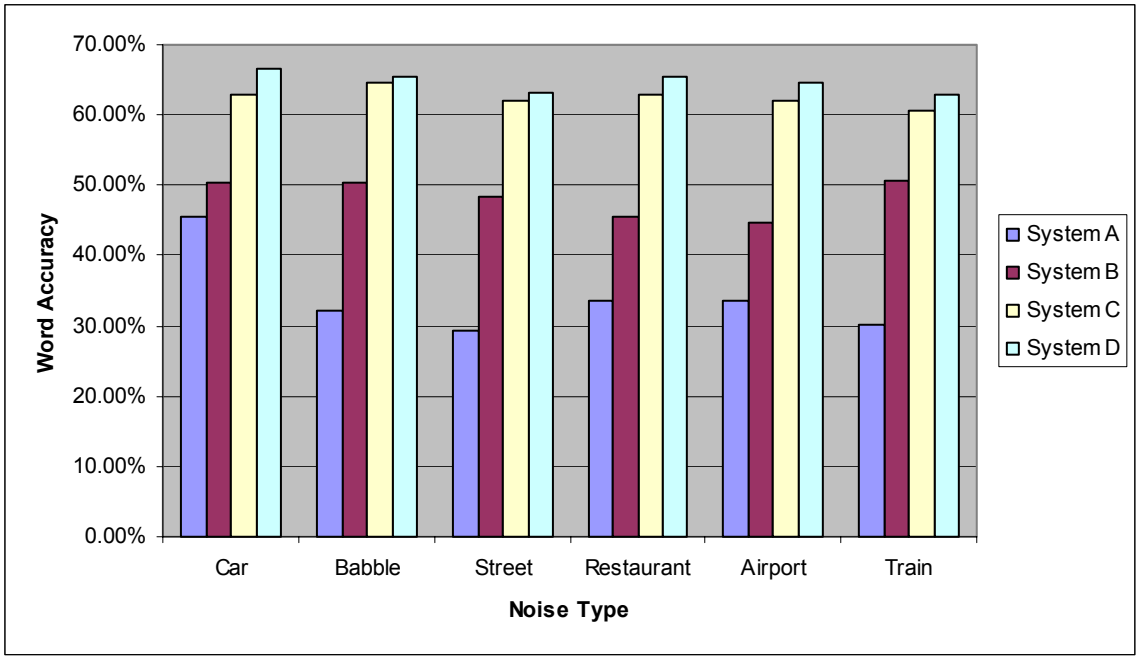

Figure 5-2. Word accuracy over the six noise test sets for Aurora4 for A) a baseline system, B) the ACDM-MMSE system with a global prior, C) the ACDM-MMSE system with re-composed advanced priors, and D) the ACDM-MMSE system with advanced priors taken directly from the top state.

Of course, the accuracies given for systems C and D in Table 5-1 are not achievable in a real system, as these systems use the correct transcription for each utterance to build the priors, a piece of information that is not known *a priori* in a real task. In the next section, two methods for obtaining advanced priors are proposed, based on the information contained in an N-best list that is produced by a recognition system.

5.2 Prior Generation from an N-best List

To take advantage of the potential for performance improvement in an ASR system that uses the ACDM-MMSE front-end with advanced prior models, an iterative algorithm is proposed. This algorithm runs as follows:

1. Parameterize corrupted speech signal with ACDM-MMSE estimator using global GMM prior
2. Run a recognition pass, generating an N-best list with the token-passing algorithm
3. Rebuild frame-by-frame priors from N-best list
4. Re-parameterize speech signal with ACDM-MMSE estimator with advanced prior models
5. Return to step 1 until J iterations have been complete.

If the recognition pass in step 2 can generate an *N*-best list with reliable state likelihoods, one would expect (based on the evidence presented in the previous section) that the re-parameterization in step 4 will produce better estimates than those obtained in step 1, resulting in higher word accuracy. Subsequently, as a more reliable *N*-best list is generated in the second recognition pass, an iterated rebuilding of the advanced priors may lead to incremental improvements in the performance of the recognizer up to a certain point of saturation. Figure 5-2 shows the block diagram for the front-end estimator using the iterative method for updating prior models. Unlike the system diagram depicted in Figure 2-1, this system uses a feedback mechanism to enhance the parameterization system.

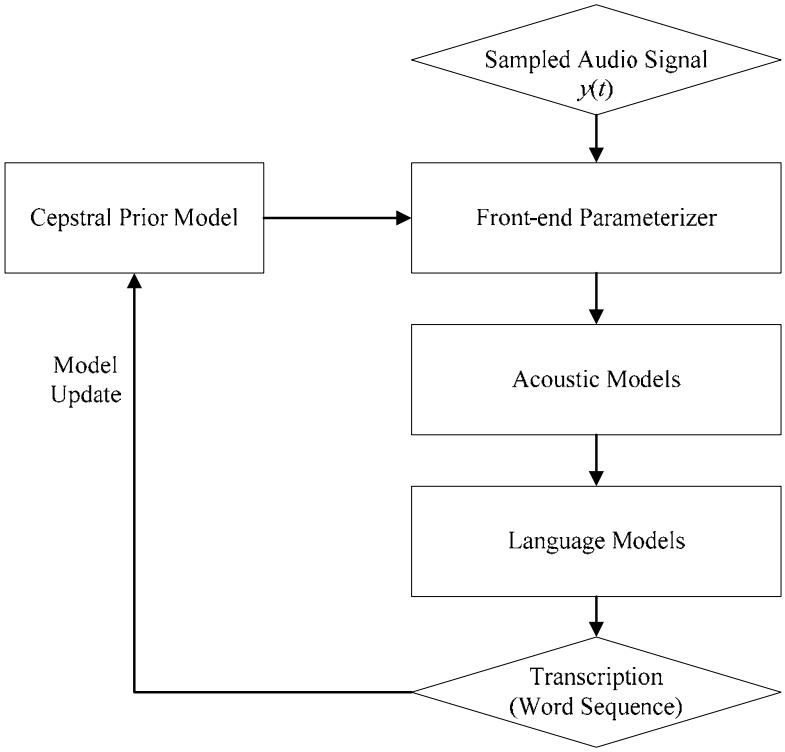

Figure 5-3. Block diagram of an automatic speech recognition system using advanced priors.

To achieve the desired effect of incrementally increasing accuracy, two components are necessary. First, the initial recognition pass must generate state likelihoods of sufficient quality for regeneration of the advanced priors. Second, an efficient and reliable method for rebuilding these priors from the *N*-best list must be developed.

Here, two methods for rebuilding the frame-by-frame advanced priors are proposed. In the first method, GMM's are re-composed from the top *N* states in each frame, in a way similar to the approach presented in Section 5.1. In the other method, a set of phonological classes are defined, each one associated with a unique prior model

that is learned offline (before testing). Following a recognition pass, the *N*-best list is used to choose one of the class-based priors for each frame.

5.2.1 Re-composition of Prior GMM's from the N-best List

The N-best list generated by a recognition pass contains a list of hypotheses $h_0, h_1, \ldots, h_{N-1}$. Each hypothesis, h_i, contains a full state-level sequence, $s_0^i,\ s_1^i,\ \ldots,\ s_{L-1}^i$, where L is the number of frames in the utterance. Each state has an associated log likelihood $\ln p\left(s_l^i\right)$, which is obtained from the token-passing algorithm described in Chapter 2. As in the approach described in Section 5.1 and depicted in Figure 5-1, the new prior model for each frame is built through composition of the Gaussians in the top N states for frame l, as

$$w'_m(t) = p\left(s_l^i\right) w_r^i(l), i = 0, \ldots, N,\ r = 0, \ldots, R,\ m = i * R, \tag{5.2}$$

where each likelihood $p\left(s_l^i\right)$ is derived from token-passing search algorithm, and R is the number of mixtures in each original state GMM. One issue here involves the nature of the log likelihoods computed from the search. If the set of log likelihoods for a particular frame are exponentiated and normalized to sum to unity, the likeliest state often has probability nearly equal to one. Because of this, the prior models generated are likely to simply reinforce the top hypothesis, and the output from the recognizer will not change. Consequently, a smoothing is performed across the frames of a hypothesis according to

$$\ln p^*\left(s_t^i\right) = \sum_{k=-W}^{W} \alpha_k \ln p\left(s_l^i\right), \tag{5.3}$$

where the α_k coefficients are computed as a normalized Hamming window, and W determines the window size. Equation (5.2) then becomes

$$w'_m(t) = p*\left(s_t^i\right) w_r^i(l), i = 0..N, r = 0..R, m = i*R. \quad (5.4)$$

A limit on the number of mixtures, M, in the updated prior may be instituted. For a sufficiently large M, the effect of this limit on the estimation is expected to be negligible, as the weights for the smaller mixtures are likely to be close to 0. Limiting the number of mixtures reduces the calculations necessary in the computation of the estimate.

While this method for rebuilding the prior models is very specific, it is sensitive to inaccuracies in the likelihoods computed from the token-passing algorithm. If the likelihoods are heavily distorted by the corrupting noise, the rebuilt prior models may cause the estimate to be worse. Thus, a second approach is proposed, one that is intended to be more robust to inaccurate likelihood computations in the recognition pass.

5.2.2 Class-based Prior Models

The prior models described in the previous section are very specific, yet choosing the correct prior for each frame is difficult. With class-based prior models, the number of possible priors is reduced. This has the effect of making the priors more general, reducing to a degree the possibility for improvement over the global prior. However, the problem of choosing the correct model for each frame becomes an easier one, especially if the classes are well separated.

The classes introduced for this approach are based on the major phonological classes: vowels, fricatives, glides (semivowels), nasals, and plosives (stops). Appendix A describes which phonemes fit into which classes. Those phonemes that do not fit in one of these classes (e.g., silence) are associated with the global prior model, as used for the front-end in Chapter 4.

To implement a system with class-based advanced prior models, two tasks must be accomplished. The class-based models must be learned previous to recognition (that is, offline). The method for training of the models involves re-weighting of the mixtures derived from the global prior. This process is described in section 5.2.2.1. During recognition, one of the classes must be chosen for each frame. The approach adopted is a simple one: the class associated with the most likely state for each frame is chosen. Alternatively, a voting scheme, possibly weighted by likelihoods, could be used. While that method is not studied in this dissertation, that could be an avenue for further research.

5.2.2.1 Class-based Model Generation

The method chosen for learning the parameters in the class-based prior GMM's begins by building a prototype model in the same way that the global GMM prior used in Chapter 4 is built. A number of mixtures is chosen, and the means, variances, and mixture weights are learned through an iterative algorithm that combines mixture-splitting and expectation maximization.

Once the prototype model is created, each class-based model is trained by re-weighting each mixture with labeled training data. This is done by first duplicating the prototype model, so that a unique model for each phonological class exists. Then, the forward-backward algorithm is run on each utterance in a subset of the clean-condition training set, using acoustic models trained on the entire training set. Only a portion of the training set is used because of the time required to run a number of experiments. Given the state occupancy likelihoods obtained with the forward-backward algorithm, each frame is labeled as one of the phonological classes (or "default" if the frame does not

belong to one of the classes, as is the case for silence frames), using the most likely state. For example, if the acoustic model state with the highest likelihood for frame *l* belongs the phoneme model "aa," frame *l* is labeled as a vowel. Each mixture likelihood in the vowel prior is then computed according to the feature vector for frame *l* and added to an accumulator.

Once all the training utterances have been processed, mixture *m* in the GMM for class *j* is re-weighted according to

$$w_m^j * = \frac{\sum_{l=0}^{L} p_m^j(c_l)}{\sum_{m=0}^{M-1}\sum_{l=0}^{L-1} p_m^j(c_l)} w_m^j, \tag{5.5}$$

where w_m^j is the original mixture weight, $w_m^j *$ is the new mixture weight, and $p_m^j(c_l)$ is the likelihood for mixture *m* in class *j* computed over the cepstral vector for frame *l*. This quantity is only nonzero if the label for frame *l* matches class *j*.

With this algorithm, a class prior for each of the five phonological classes is built using a global prior as a "bootstrap" model. A possible alternative approach would to label each frame as one of the classes using the forward-backward algorithm as described above, then group the data into the five classes and retrain each model from scratch. Empirical observation of this approach, however, shows it to result in garbage recognition transcriptions and word error rates that sometimes exceed 100%, so this method is not employed in this dissertation.

In the following sections, the described methods for implementation of advanced priors are evaluated on the Aurora4 dataset. Results demonstrate small improvement with the iterative approach over the ACDM-MMSE estimator with a global prior model.

5.3 Advanced Prior Modeling ASR Experiments

ASR experiments are run over the Aurora4 dataset with the same configuration as described in Section 5.1.1. The iterative algorithm described in section 5.2 is used in conjunction with the ACDM-MMSE estimator for parameterization of corrupted speech signals. Step 3 is performed with the two different methods for rebuilding prior models: re-composition of GMM's and class-based priors. The configuration parameters for the iterative prior rebuilding algorithm include N, the number of hypotheses to generate, and W, the size of the smoothing window used in the re-composition method. Based on development experiments, the chosen values are $N = 10$ and $W = 11$.

As described in section 5.1.1, the six Aurora4 (core) test sets consist of 166 utterances with SNR's ranging from +5 to +15 dB in a uniform distribution. Because of the iterative nature of the advanced prior modeling algorithm, it is expected that more accurate initial recognition hypotheses will result in larger improvements in feature estimation with the advanced priors. To study this effect, the noisy test sets are grouped into four quartiles by sorting the signals according to SNR, then partitioning the set. Quartile 1 has utterances with the lowest SNR, while quartile 4 has those with the highest.

5.3.1 Re-composition Experiments

Table 5-2 shows the word accuracy results by iteration for the ACDM-MMSE estimator with advanced priors using the GMM re-composition method. The values are averaged over the six different noise cases, and displayed by SNR Quartile. The first row shows the base accuracy values (using the global prior model), while the following rows

show the accuracy values for each iteration, plus the absolute difference in accuracy between each iteration and the base accuracy in parentheses.

The estimator is developed using no CMS operation in the derivation. Consequently, the prior models cannot be re-composed from acoustic models that have been trained on data that has been post-processed with CMS. Therefore, unlike the previous experiments, CMS is not used for the experiments that produce the results of Table 5-2. Those experiment results marked with an asterisk (*) prove to be greater than the base accuracies in a statistically significant sense, using a type I error of 0.05.

Iteration Number	SNR Quartile			
	# 1 (Noisiest)	#2	#3	#4 (Cleanest)
Base	33.0%	42.8%	53.9%	61.8%
# 1	32.2% (–0.8%)	42.5% (–0.3%)	54.1% (+0.2%)	63.3% (+1.5%)*
# 2	31.2% (–1.8%)	43.1% (+0.3%)	54.6% (+0.7%)	63.2% (+1.4%)*
# 3	30.7% (–2.3 %)	43.3% (+0.5%)	54.3% (+0.4%)	63.2% (+1.4%)*

Table 5-2. Word accuracy for base ACDM-MMSE estimator and absolute accuracy differences for successive iterations using re-composed advanced priors, by SNR Quartile in Aurora4 core test sets.

As expected, the word accuracies for the base ACDM-MMSE estimator increase monotonically as the SNR increases. Examination of the absolute differences between the accuracies for the recognition passes with advanced priors shows that improvement in performance is gained only for the higher SNR utterances. For the noisier signals, a decrease in performance is seen. Unfortunately, when averaged over all utterances in the test set, the overall performance gained by using re-composed priors is negligible. The improvement gained for the higher SNR utterances is canceled out by the degradation seen in the lower SNR utterances.

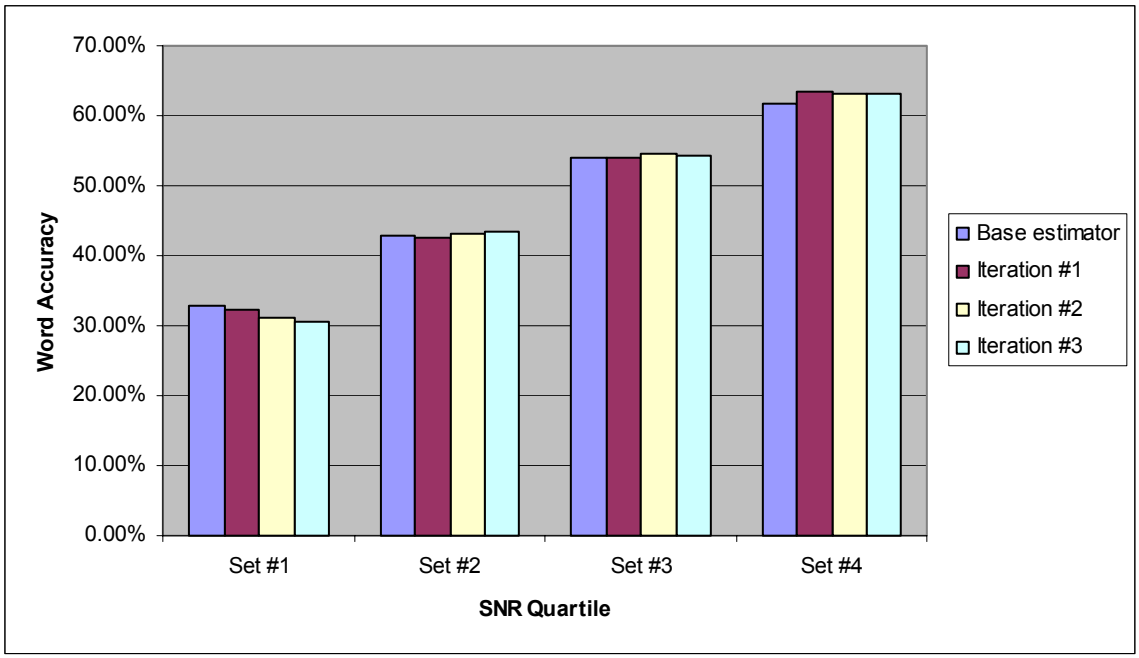

Figure 5-4. Word accuracy for base ACDM-MMSE estimator and absolute accuracy differences for successive iterations using re-composed advanced priors, by SNR Quartile in Aurora4 core test sets.

Based on these observations, it is reasonable to expect that higher initial performance would lead to better gains in accuracy through the use of advanced priors. One simple modification that can improve the accuracy to some degree is the use of CMS. However, because of the potential mismatch between the generated priors and the conditional, acoustic models trained on CMS processed features cannot be used to re-compose the priors directly. To solve this problem, two sets of acoustic models are trained, each with the same structure. One set, which is learned on CMS processed data, is used for recognition. The second set, learned on "raw" cepstral features, is used for the re-composition of priors. Of course, there is not a one-to-one correspondence of mixtures between the two model sets, but this is not an issue, since it is only the state-level likelihoods that are obtained through the recognition pass.

Table 5-3 shows the results for the iterative prior re-composition front-end using CMS processed acoustic models for recognition. In this experiment, improvement is gained for each SNR Quartile, and the improvement increases for each successive iteration. After three iterations, the average absolute improvement is 1.6% over a baseline (global prior) of 51.3% accuracy. Those experiment results marked with an asterisk (*) prove to be greater than the base accuracies in a statistically significant sense, using a type I error of 0.05.

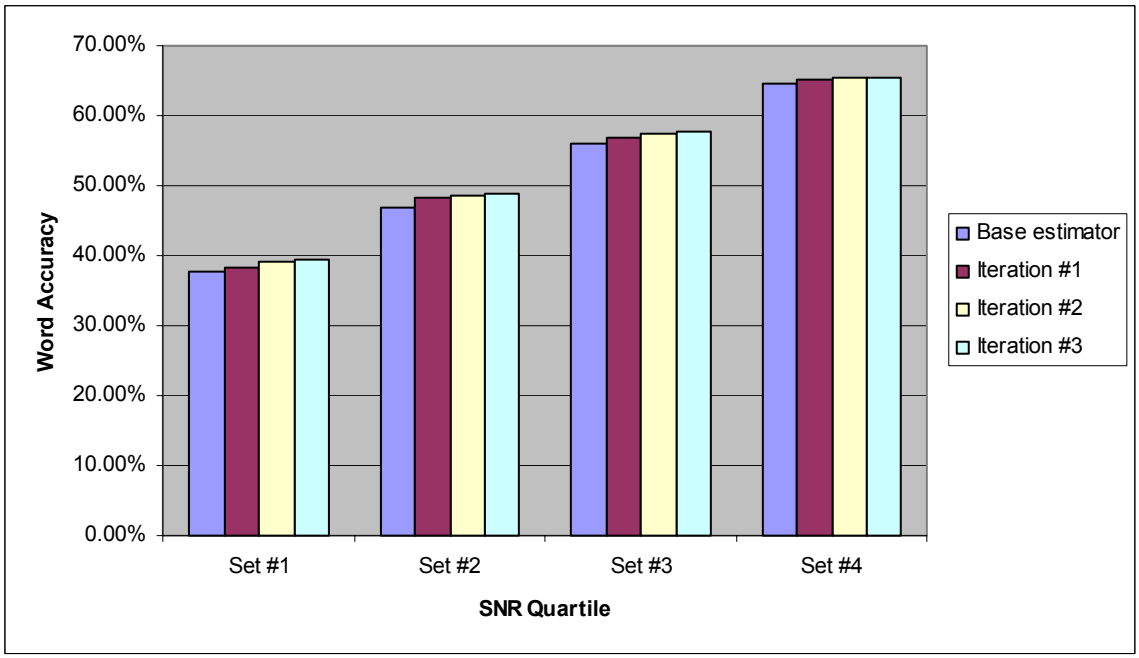

Figure 5-5. Word accuracy for base ACDM-MMSE estimator and absolute accuracy differences for successive iterations using re-composed advanced priors using CMS, by SNR Quartile in Aurora4 core test sets.

Iteration Number	SNR Quartile			
	# 1 (Noisiest)	#2	#3	#4 (Cleanest)
Base	37.8%	46.8%	56.1%	64.5%
# 1	38.2% (+0.4%)	48.3% (+1.5%)*	56.9% (+0.8%)	65.2% (+0.7%)
# 2	39.1% (+1.3%)	48.7% (+1.9%)*	57.4% (+1.3%)*	65.5% (+1.0%)*
# 3	39.3% (+1.5 %)	49.0% (+2.2%)*	57.8% (+1.7%)*	65.5% (+1.0%)*

Table 5-3. Word accuracy for base ACDM-MMSE estimator and absolute accuracy differences for successive iterations using re-composed advanced priors using CMS, by SNR Quartile in Aurora4 core test sets.

5.3.2 Analysis of Re-composition Experiments

While improvement is seen, the relative reduction in error is only 3.3%. This improvement comes at the cost of extra computational time introduced. Three more full recognition passes, combined with the re-composition process, causes the runtime of the overall recognition to increase by more than four-fold.

The iterative prior re-composition front-end fails to provide improvement on the order of magnitude seen with the ideal priors, with a relative word error rate reduction of 3.3% as compared to 31.4%. The lack of gain in performance may be due to locality of the errors in the feature estimates. The frames in which the *a priori* speech and noise estimates are poor are those that would benefit most from a more specific prior model, provided it is sufficiently accurate. However, when priors are re-composed for these frames, as the feature estimates have large errors, the state likelihoods used to generate the new prior models do not contain enough information to provide for reliable priors. One of the primary motivations for using a full recognizer to generate new prior models comes from the non-locality of the language model likelihoods. The total likelihood for a

state in a given frame is affected by the language model likelihood, which in turn is affected by other frames. Because of this, the errors caused by poor local acoustic likelihood estimates are sometimes negated through the token-passing algorithm. However, this effect, even when combined with the smoothing operation described in equation (5.3), does not appear to be enough to provide for reliable priors to be built from the state likelihoods of frames in which the initial feature estimates are poor.

The difficulty of solving this problem can be seen if the first stage of the prior re-composition process is viewed as a classification problem. Given S unique acoustic states, reliable priors are generated only if the correct state for a given frame (or one with a GMM that is very similar) can be identified. In a three state context-independent system with 40 phoneme models, this becomes a 120 class decision problem. For the context-dependent system used for the Aurora4 task, there are over 4,400 classes. If the single correct state out of 4,400 must be identified for a large percentage of the frames, the chance of significantly improvement in system performance through the use of re-composed priors appears low.

This problem motivates the use of class-based priors, as only five (plus a default global) models are available. Choosing the correct prior in this case is a much easier task. The experimental results for this approach are given in the next section.

5.3.3 Class-based Prior Experiments

The dataset and experimental setup for the class-based approach to advanced prior modeling matches that of the re-composition approach. CMS is used as a post-processing step for this set of experiments. Only one set of acoustic models is necessary, as the prior model parameters are not derived directly from the acoustic models. The results,

measured in word accuracy, for the base estimator and three iterations of class-based prior rebuilding are shown in Table 5-4, organized by SNR quartile. The values shown do not follow a discernable pattern as do those seen for the re-composition approach. Improvement over the estimator with the global model is seen only for the third quartile.

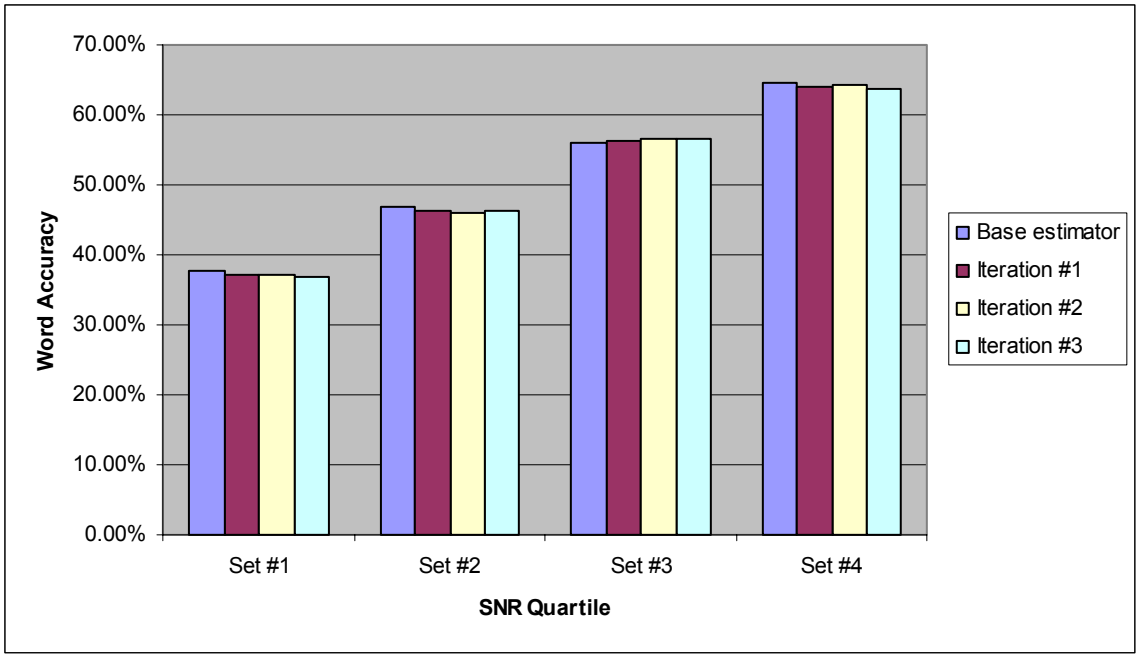

Figure 5-6. Word accuracy for base ACDM-MMSE estimator and absolute accuracy differences for successive iterations using class-based advanced priors, by SNR quartile in Aurora4 core test sets.

Iteration Number	SNR Quartile			
	# 1 (Noisiest)	#2	#3	#4 (Cleanest)
Base	37.8%	46.8%	56.1%	64.5%
# 1	37.2% (-0.8%)	46.4% (-0.4%)	56.3% (+0.2%)	64.1% (-0.4%)
# 2	37.2% (-0.8%)	45.9% (-0.9%)	56.7% (+0.6%)	64.4% (-0.1%)
# 3	37.0% (-1.0%)	46.2% (-0.6%)	56.7% (+0.6%)	63.6% (-0.9%)

Table 5-4. Word accuracy for base ACDM-MMSE estimator and absolute accuracy differences for successive iterations using class-based advanced priors, by SNR quartile in Aurora4 core test sets.

5.3.4 Analysis of Class-based Prior Experiments

Unlike the ASR experiments in which re-composition is used (with CMS), the class-based prior system fails to gain any improvement in accuracy over the ADCM-MMSE estimator with a global prior. Two possible reasons for this are that the prior models themselves are no better than the global prior, or even with a limited number of distinct priors, the system is still unable to choose correctly the appropriate prior for a sufficient number of frames.

To analyze the lack of improvement in accuracy, the ideal prior experimental approach as described in Section 5.1 is repeated using the class-based priors. For each frame, the state with the greatest state occupancy likelihood as computed with the forward-backward algorithm using the true transcription to choose the appropriate prior. Averaged over all noise types and utterances, the ACDM-MMSE estimator with ideal class-based priors produces a word accuracy of 52.9%, compared to 52.2% produced by the estimator with a global prior. This is a relative reduction in error of only about 0.6%, a value smaller than seen with the re-composition approach, even though in this case, the

true transcription is used to rebuild the priors. This suggests that the approach introduced for developing the different prior models is flawed. Adaptation of the means and variances of the global prior model in addition to the mixture weights may lead to better performance improvements.

5.4 Summary

In this chapter, the concept of replacing the global prior model commonly used in estimation of speech features with a context specific version has been introduced and explored. Experiments using the ACDM-MMSE estimator with ideal priors generated with the use of true utterance transcriptions over the Aurora4 dataset indicate that a large improvement in word accuracy can be achieved with more advanced priors, as an average relative word error rate reduction of 31.4% is achieved over the core test sets in Aurora4.

Two methods for implementing the advanced prior modeling scheme have been presented: re-composition of frame-by-frame priors from the acoustic models obtained from an *N*-best list and phonological class-based priors. The re-composition approach is able to provide a relative reduction in word error rate of 3.3%, though the class-based prior approach does not produce any error reduction.

Using the ideal prior experiments as an upper bound, it is clear that substantial improvement can be made over the prior updating methods introduced in this chapter. Given the cost of increased runtime introduced by the prior adaptation methods, the relatively small decrease in error does not justify the added computational time. Further improvements in the methods for prior rebuilding are necessary before advanced prior modeling can be considered a viable option for real systems.

Chapter 6 Discussion

This dissertation has introduced a new estimation method for feature compensation in automatic speech recognitions systems that must operate in noisy environments. The additive cepstral distortion model minimum mean-square error (ACDM-MMSE) estimator, developed in Chapter 4, models the effect of corrupting noise on speech signals as additive in the cepstral domain. A closed form solution was derived, using the assumptions that the distortion on the mel-frequency cepstral coefficients (MFCC's) is Gaussian and that the prior distribution of the MFCC's is represented by a Gaussian Mixture Model (GMM). A strategy for incorporating context specific prior models into the estimator was introduced in Chapter 5. While the improvement seen in speech recognition accuracy with the use of the updated priors as compared to a global prior model is small, the potential for large gains in accuracy was demonstrated.

6.1 Contributions

The ACDM-MMSE estimator represents a new estimation formulation for compensation of noise corrupted speech features, specifically MFCC's. Unlike many of the previous methods for estimation of features, the new approach is not iterative. Empirical results show that the ACDM-MMSE estimator requires fewer mixtures than the VTS estimation algorithm in the GMM prior distribution to achieve high word accuracies when used as a front-end to a robust automatic speech recognition (ASR) system. As a consequence of these two factors, the ACDM-MMSE estimator requires less computational time than many other approaches such as the Vector Taylor Series (VTS) estimator. On the Aurora2 connected digit recognition task, a system using the ACDM-

MMSE estimator as a front-end is shown to run more than five times faster than one that used the VTS estimator and produce better word accuracy.

The new estimator operates in the cepstral domain, without the need for a pseudo-inverse matrix computation. This feature allows for easy integration of an adaptive prior modeling scheme such as the approach introduced in Chapter 5.

The advanced prior modeling scheme represents a departure from the previous approaches to prior distribution modeling in the literature. A global prior distribution was used in each of these estimation methods, limiting the effectiveness of the prior to compensate for errors in the *a priori* estimates of the noise spectra. In chapter 5, the potential for improvement in ASR system performance with the use of a more specific and informative prior modeling scheme was demonstrated. A relative word error rate (WER) reduction as high as 36.4% over one of the Aurora4 core test sets was observed when ideal priors were used in place of a global prior in the ACDM-MMSE estimator.

Two methods for rebuilding prior models from recognition transcription hypotheses were introduced. The first approach re-composes GMM's from the acoustic models, using the state likelihoods derived from a full recognition pass. This method resulted in a relative WER reduction of as much as 3.3% when averaged over the different noises in Aurora4, provided cepstral mean subtraction was incorporated. The second method used a categorization approach to produce phonologically driven class-based priors. Unfortunately, this method did not produce improvement in performance overall.

6.2 Further Analysis of the ACDM-MMSE Estimator

The ACDM-MMSE estimator makes three primary assumptions and approximations. First, the filter bank energy coefficients for the speech and noise components are assumed to be gamma distributed random variables. Second, it is assumed that the conditional distribution of the distorted cepstral coefficient vector for a frame of speech, given the clean cepstral coefficient vector, when affected only by additive noise, is well represented by a multivariate Gaussian with a diagonal covariance matrix. Last, in the derivation of the estimator, while the clean speech cepstral vector is a given quantity, the clean speech filter bank energy vector is treated as a random vector. The distribution for this vector is determined by parameters that must be estimated *a priori*. This approximation is necessary to ensure a closed-form solution to the estimator can be found.

6.2.1 Distribution Assumptions

The central limit theorem provides a strong justification for the assumption that the conditional distribution is a multivariate Gaussian with diagonal covariance, as is described in Chapter 4.

The assumption that the speech and noise filter bank energies are gamma distributed does not have a theoretical justification, however. A statistical analysis shows that a gamma distribution fits more closely than four other common distributions: normal, uniform, Rayleigh, and exponential. However, when tested using a Chi-square distribution test, the null hypothesis that each filter bank energy coefficient follows a gamma distribution is rejected. Consequently, this assumption appears more tenuous. The densities appear to follow a rising and falling exponential curve in the manner of gamma

densities, yet appear to decline more rapidly in the right tail. If a more fitting distribution that still leads to a closed form solution could be identified, a more robust estimator may be developed.

6.2.2 *A Priori* Speech Approximation

Although the clean speech filter bank energy vector should be treated as a given in the derivation of the ACDM-MMSE estimator, it is treated as a random vector whose elements are independently gamma distributed random variables. This is done to ensure a closed-form solution for the estimator. A major consequence of this choice is that the speech energies must be estimated *a priori*. This is in contrast with most other feature compensation algorithms that use estimators, although some of these methods, primarily the approaches that use a Taylor series expansion, must have an initial guess. While this creates the need for another system component, more flexibility is introduced as a consequence. The derived estimator is independent of the choice of speech and noise estimation algorithms. Any method for *a priori* estimation can be used, and improvement in either component (speech or noise estimator) is likely to lead to an improved estimate of the MFCC values.

6.3 Further Analysis of Advanced Prior Modeling

The suggestion that a more specific and informative prior could lead to improved estimation of speech features was proved in Chapter 5 with the ideal prior modeling experiment. Significant improvement (better than 30% relative word error rate reduction) was observed when the true transcriptions were used to rebuild the updated priors.

However, when a realistic recognition system was used to generate the prior models, only a fraction of that improvement was seen.

It was suggested in Chapter 5 that the reason the re-composition approach failed to give better results is rooted in the locality of the *a priori* estimation errors. This leads to a cyclical situation, and hence a fundamental obstacle in the implementation of advanced prior modeling. If the *a priori* estimates of the speech and noise signals are perfect, there is no need for a prior model, as the clean feature vector could be computed directly; the primary motivation for the prior model is to ensure that poor speech and noise estimates do not lead to unreasonable feature estimates. Those frames that require a strong influence from the prior model are those in which large errors are present in the *a priori* estimates. Conversely, the frames for which accurate *a priori* estimates are available do not need such influence from the prior model. Unfortunately, this is the opposite of what is available through the use of re-composed priors as introduced in Chapter 5: poor feature estimates with the global model yield poor state likelihoods in the transcription hypotheses, and subsequently poor prior models, whereas quality feature estimates with the global model yield quality state likelihoods, and subsequently (unnecessarily) strong prior models. Hence, the frames that require the informative, reliable priors are just the frames that cannot obtain them.

For this problem to be overcome, a method for redistribution of information must be developed. An intelligent method for building priors for frames in which large errors in the *a priori* estimates are present from the information contained in frames in which quality *a priori* estimates are available is necessary.

In the class-based prior approach, empirical observation showed that it was not necessarily the poor state likelihoods from the transcription hypotheses that lead to a lack of performance improvement over the global prior experiments, but instead insufficiently informative prior models. This was seen through the ideal class-based prior model experiments, which yielded little or no improvement, even when the true transcriptions were available.

The problem could be that either the classes chosen are not sufficiently distinct, or that the method for building the priors is flawed. Given the clear differences between the different phonological classes, it seems more likely that the latter is the root cause for the lack of improvement. Suggestions for alternative approaches to class-based prior model building are given in the following section.

6.4 Suggestions for Future Research Directions

There are several avenues in which further progress can be made in the estimation approach introduced in this dissertation. These include adjusting or relaxing some of the assumptions in the distortion model, developing a better method for setting the parameter β discussed in Chapter 4, overcoming the problem with locality of errors discussed in Section 6.3, developing a method for building more useful class-based priors, and incorporating multi-channel signals into the estimator.

First, adjustments to the underlying assumptions and approximations described in Section 6.2 are possible. While the assumption that the speech and noise filter bank energy coefficients are random variables with gamma distributions appears to be a reasonable approximation, there may be some difference between the true underlying distributions and those used in the derivation of the estimator. If a more accurate

modeling assumption could be discovered, a more rigorous estimator may be derived. For practical reasons, it is still necessary to formulate a distribution assumption that leads to a closed-form solution.

A related point involves choice of the value for the parameter β. The gamma distribution has two free parameters. In the computation of the estimator, for each of the speech and noise signal estimates, only one value is available (for each filter). The assumption is made that the value for the filter estimate for each signal is the mean of a gamma distribution. To determine the parameter α for each gamma from the mean values, it is necessary to supply the value for β. Currently, this is done by declaring β to be a constant value, and tuning it as an algorithm parameter. Intuitively, this seems suboptimal. If a more rigorous way to discover the best value for β could be found, the performance of the estimator may improve. One method for approaching this task is replacement of the "point" estimates of the speech and noise signals (single values) with interval or distributional estimates. This would then give some information about both the mean and variance of the gamma distributions.

Regarding the advanced prior modeling, the problem remains that the information necessary for improving the prior models cannot be extracted easily from the transcription hypotheses. As discussed in Section 6.2, this appears to be because the initial estimates for the noisiest frames are not of sufficient quality to yield more specific yet reliable priors. Some method for using the higher SNR frames to rebuild the priors for the lower SNR frames must be developed. In any approach that accomplishes this, the first step is to identify which frames have quality estimates and which frames have poor estimates. There has been some work in confidence scoring [70, 71] that might aid in this

task. Overcoming this problem appears to be a major research undertaking, yet if the improvement performance gained through rebuilding priors from transcription hypotheses could approach that of the ideal priors, large gains would be achieved.

As shown in Chapter 5, the class-based prior modeling scheme did not yield positive results. This appears to be due to the lack of specificity improvement over the global model. Currently, the class priors are built by adapting the weights of the global model with the training data. Other methods for building class-based priors must be explored. One avenue to consider is adaptation of the means and variances of the global model on a class basis. Algorithms for this type of adaptation already exist [4], so this would not be a difficult task. Another possibility is to make the class priors state-dependent, meaning that instead of using a single prior for vowels, for example, a prior would exist for each state (i.e. 1, 2, and 3 in a 3-state HMM) in the vowel class. Finally, the phonological class categorization used in Chapter 5 could be replaced by a data-driven categorization. In this approach, a clustering mechanism would be used to separate the prior classes.

The ACDM-MMSE estimator introduced in this dissertation operates exclusively on single channel signals. The estimator requires *a priori* estimates of the speech and noise signals, yet these quantities are very difficult to estimate in a single channel system. Microphone arrays have been shown to allow for better signal separation than single microphone systems [10], and use of such a mechanism is likely to provide better *a priori* estimates of the speech and noise signals, and consequently better speech feature estimates, leading to better recognition performance.

Bibliography

[1] "ESE2 special sessions on noise robust recognition," *at European Conference on Speech Communication*, Aalborg, Denmark, 2001.
[2] *Speech Communication*, vol. 34, 2001.
[3] "Nuance - Dragon NaturallySpeaking Preferred 9," *Nuance Communications, Inc.* [Online].
[4] C. Leggetter and P. Woodland, "Speaker adaptation of continuous density HMMs using multivariate linear regression," *presented at ISCLP*, Yokohama, Japan, 1994.
[5] S. Furui, "Cepstral analysis technique for automatic speaker verification," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 29, pp. 254-272, 1981.
[6] Y. Ephraim and D. Malah, "Speech enhancement using a minimum mean-square error short-time spectral amplitude estimator," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, pp. 1109-1121, 1984.
[7] Y. Ephraim and D. Malah, "Speech enhancement using a minimum mean-square error log-spectral amplitude estimator," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 33, pp. 443-445, 1985.
[8] R. Martin, "Spectral subtraction based on minimum statistics," *presented at European Signal Processing Conference*, Edinburgh, Scotland, 1994.
[9] I. Cohen, "Optimal speech enhancement under signal presence uncertainty using log-spectal amplitude estimator," *IEEE Signal Processing Letters*, vol. 9, pp. 113-117, 2002.
[10] J. E. Adcock, "Optimal filtering and speech recognition with microphone arrays," Ph.D. Dissertation, Brown University, 2001.
[11] M. Brandstein and D. Ward, *Microphone Arrays*, New York: Springer-Verlag, 2001.
[12] I. A. McCowan and S. Sridharan, "Multi-channel sub-band speech recognition," *Eurasip Journal on Applied Signal Processing*, vol. 2001, pp. 45-52, 2001.
[13] H. Hermansky, "Perceptual linear predictive (PLP) analysis for speech recognition," *Journal of the Acoustical Society of America*, vol. 87, no. 4, pp. 1738-52, 1990.
[14] H. Hermansky, N. Morgan, and H. G. Hirsch, "Recognition of speech in additive and convolutional noise based on RASTA spectral processing," *presented at IEEE ICASSP*, Minneapolis, MN, 1993.
[15] P. J. Moreno, "Speech recognition in noisy environments," Ph.D. Dissertation, Carnegie Mellon University, 1996.
[16] A. Acero, "Acoustic and environmental robustness in automatic speech recognition," Ph.D. Dissertation, Carnegie Mellon University, 1990.
[17] L. Deng, J. Droppo, and A. Acero, "Estimating cepstrum of speech under the presence of noise using a joint prior of static and dynamic features," *IEEE Transactions on Speech and Audio Processing*, vol. 12, pp. 218-233, 2004.

[18] L. Deng, J. Droppo, and A. Acero, "Recursive estimation of nonstationary noise using iterative stochastic approximation for robust speech recognition," *IEEE Trans. on Speech and Audio Processing*, vol. 11, pp. 568-580, 2003.
[19] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society, Series B*, vol. 39, pp. 1-38, 1977.
[20] T. E. Quatieri, *Discrete-Time Speech Signal Processing*. Upper Saddle River, New Jersey: Prentice Hall, 2002.
[21] J. R. Deller, J. H. L. Hansen, and J. G. Proakis, *Discrete-Time Processing of Speech Signals*, Second ed. New York: IEEE Press, 2000.
[22] B. Gold and N. Morgan, *Speech and Audio Signal Processing*. New York: John Wiley and Sons, 2000.
[23] S. B. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Transactions on Acoustics Speech and Signal Processing*, vol. 28, pp. 357-366, 1980.
[24] Cambridge University, "HTK."
[25] Carnegie Mellon University, Sun Microsystems Laboratories, Mitsubishi Electric Research Labs, and Hewlett Packard, "Sphinx-4."
[26] A. V. Oppenheim, R. W. Schafer, and J. R. Buck, *Digital Signal Processing*, 2nd ed. Upper Saddle River, New Jersey: Prentice Hall, 1999.
[27] S. Stevens and J. Volkmann, "The relation of pitch to frequency: A revised Scale," *American Journal of Psychology*, vol. 53, pp. 329-353, 1940.
[28] K. M. Indrebo, R. J. Povinelli, and M. T. Johnson, "Third-order moments of filtered speech signals for robust speech recognition," *presented at International Conference on Non-linear Speech Processing (NOLISP2005)*, Barcelona, Spain, 2005.
[29] S. Ikbal, H. Misra, and H. Bourlard, "Phase autocorrelation (PAC) derived robust speech features," *presented at IEEE ICASSP*, Hong Kong, 2003.
[30] S. Haykin, *Adaptive Filter Theory*, 3rd ed. Upper Saddle River, New Jersey: Prentice Hall, 1996.
[31] B. S. Atal, "Effectiveness of linear prediction characteristics of the speech wave for automatic speaker identification and verification," *The Journal of the Acoustical Society of America*, vol. 55, pp. 1304-1322, 1974.
[32] H. Soltau, B. Kingsbury, L. Mangu, D. Povey, G. Saon, and G. Zweig, "The IBM 2004 conversational telephony system for rich transcription," *presented at IEEE ICASSP*, Philadelphia, PA, 2005.
[33] R. Schwartz, T. Colthurst, N. Duta, H. Gish, R. Iyer, C.-L. Kao, D. Liu, O. Kimball, J. Ma, J. Makhoul, S. Matsoukas, L. Nguyen, M. Noamany, R. Prasad, B. Xiang, D.-X. Xu, J.-L. Gauvain, L. Lamel, H. Schwenk, G. Adda, and L. Chen, "Speech recognition in multiple languages and domains: The 2003 BBN/LIMSI EARS system," *presented at IEEE ICASSP*, Montreal, Canada, 2004.
[34] M. J. Reyes-Gomez, B. Raj, and D. P. Ellis, "Multi-channel source separation by factorial HMMs," *presented at IEEE ICASSP*, Hong Kong, 2003.

[35] L. M. Arslan and D. Talkin, "Speaker transformation using sentence HMM based alignments and detailed prosody modification," *presented at IEEE ICASSP*, Seattle, WA 1998.

[36] A. Krogh, B. Larsson, G. v. Heijne, and E. L. L. Sonnhammer, "Predicting transmembrane protein topology with a hidden Markov model: application to complete genomes," *Journal of Molecular Biology*, vol. 305, pp. 567-580, 2001.

[37] D. Miller, T. Leek, and R. Schwartz, "A hidden Markov model information retrieval system," *presented at 22nd Annual ACM SIGIR Conference on Research and Development in Information Retrieval*, Berkely, California, 1999.

[38] A. Papoulis, *Probability, Random variables, and Stochastic Processes*, 3rd ed. New York: McGraw-Hill, 1991.

[39] K.-F. Lee, S. Hayamizu, H.-W. H. C. Huang, J. Swartz, and R. Weide, "Allophone clustering for continuous speech recognition," *presented at IEEE ICASSP*, Albuquerque, NM, 1990.

[40] S. Young, J. Odell, D. Ollason, V. Valtchev, and P. Woodland, "The HTK book," 1997.

[41] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. 13, pp. 260-269, 1967.

[42] S. Chen and J. Goodman, "An empirical study of smoothing techniques for language modeling," *presented at 34th Annual Meeting on Association for Computational Linguistics*, Santa Cruz, CA, 1996.

[43] T. C. Bell, J. G. Cleary, and I. H. Witten, *Text Compression*. Englewood Cliffs, New Jersey: Prentice Hall, 1990.

[44] H. Ney, U. Essen, and R. Kneser, "On structuring probabilistic dependencies in stochastic language modeling," *Computer Speech and Language*, vol. 8, pp. 1-38, 1994.

[45] T. Kamm, G. Andreou, and J. Cohen, "Vocal tract length normalization in speech recognition: compensating for systematic speaker variability," *presented at 15th Annual Speech Research Symposium*, Baltimore, MD, 1995.

[46] D. Pearce and H. Hirsch, "The AURORA experimental framework for the performance evaluation of speech recognition systems under noisy conditions," *presented at ISCA ITRW ASR2000*, Beijing, China, 2000.

[47] H. Bourlard and S. Dupont, "Subband-based speech recognition," *presented at IEEE ICASSP*, Munich, Germany, 1997.

[48] I. Cohen, "Noise spectrum estimation in adverse environments: improved minima controlled recursive averaging," *IEEE Trans. on Speech and Audio Processing*, vol. 11, pp. 466-475, 2004.

[49] L. R. Rabiner and M. Sambur, "Voice-unvoiced-silence detection using the Itakura LPC distance Measure," *presented at IEEE ICASSP*, Hartford, CT, 1977.

[50] R. J. McAulay and M. L. Malpass, "Speech enhancement using a soft-decision noise suppression filter," *IEEE Transactions on Acoustics Speech and Signal Processing*, vol. 28, pp. 137-145, 1980.

[51] J. Allen, "Short-term spectral analysis, and modification by discrete Fourier Transform," *IEEE Transactions on Acoustics Speech and Signal Processing*, vol. 25, pp. 235-238, 1977.

[52] J. Poruba, "Speech enhancement based on nonlinear spectral subtraction," *presented at the Fourth IEEE International Caracas Conference on Devices, Circuits, and Systems*, 2002.
[53] Y. Ren and M. T. Johnson, "An improved SNR estimator for speech enhancement," *presented at IEEE ICASSP*, Las Vegas, NV, 2008.
[54] C.-W. Hsu and L.-S. Lee, "Higher order cepstral moment normalization (HOCMN) for Robust Speech Recognition," *presented at IEEE ICASSP*, Montreal, Canada, 2004.
[55] H. L. Van Trees, *Detection, Estimation, and Modulation Theory, Volume I*. New York: Wiley, 1968.
[56] H. Sameti, H. Sheikhzadeh, L. Deng, and R. Brennan, "HMM-based strategies for enhancement of speech embedded in nonstationary noise," *IEEE Transactions on Speech and Audio Processing*, vol. 6, pp. 445-455, 1998.
[57] A. Acero and R. Stern, "Robust speech recognition by normalization of the acoustic space," *presented at IEEE ICASSP*, Toronto, Canada, 1991.
[58] P. Moreno, B. Raj, E. Gouvea, and R. Stern, "Multivariate-gaussian-based cesptral normalization for robust speech recognition," *presented at IEEE ICASSP*, Detroit, MI, 1995.
[59] E. W. Weisstien, "Gamma function," *Mathworld-A Wolfram Web Resource* [Online].
[60] R. Martin, "Speech enhancement using MMSE short time spectral estimation with gamma distributed speech priors," *presented at IEEE ICASSP*, Orlando, Florida, USA, 2002.
[61] S. Gazor, "Speech probability distribution," *IEEE Signal Processing Letters*, vol. 10, pp. 204-207, 2003.
[62] J. W. Shin, J.-H. Chang, and N. S. Kim, "Statistical modeling of speech signals based on generalized gamma distribution," *IEEE Signal Processing Letters*, vol. 12, pp. 258-261, 2005.
[63] A. K. Gupta and S. Nadarajah, *Handbook of Beta Distribution and its Applications*. New York: Marcel Dekker, 2004.
[64] J. L. Spouge, "Computation of the gamma, digamma, and trigamma functions," *SIAM Journal on Numerical Analysis*, vol. 31, pp. 931-944, 1994.
[65] L. Arslan, A. McCree, and V. Viswanathan, "New methods for adaptive noise suppression," *presented at IEEE ICASSP*, Detroit, MI, 1995.
[66] Linguistic Data Consortium, "TIDIGITS." [Online].
[67] G. Hirsch, "Experimental framework for the performance evaluation of speech recognition front-ends in a large vocabulary task," ETSI STQ-Aurora DSR Working Group December 2002.
[68] Linguistic Data Consortium, "CSR-I (WSJ0) Complete." [Online].
[69] "WSJ5K.DMP," *Sourceforge.net*. [Online].
[70] E. Lleida and R. C. Rose, "Likelihood Ratio decoding and confidence measures for continuous speech recognition," *presented at ICSLP*, Philadelphia, PA,1996.
[71] P. Fetter, F. Dandurand, and P. Regel-Brietzmann, "Word graph rescoring using confidence measures," *presented at ICSLP*, Philadelphia, PA,1996.

Appendix A Phoneme Set

The set of phonemes used in the automatic speech recognition (ASR) system in Chapter 5 is detailed in Table A-1, grouped by phonological class. For each phoneme, an example word is presented in which the bold letter(s) comprise the given phoneme.

Vowels		Fricatives	
aa	p**o**x	ch	**ch**ill
ae	b**a**t	dh	**th**ere
ah	**a**bode	f	**f**rog
ao	b**oa**rd	jh	**j**ail
aw	ab**ou**t	s	**s**ome
ay	fr**ie**d	sh	**sh**ine
eh	m**e**ss	th	wor**th**
ey	w**ai**t	z	**z**ebra
ih	h**i**d	zh	a**si**a
iy	h**ee**d	Nasals	
ow	b**oa**t	m	**m**onkey
oy	j**oy**	n	**n**ose
uh	w**oo**d	ng	ri**ng**
uw	f**oo**d	Plosives (Stops)	
Glides		b	**b**ud
er	batt**er**	d	**d**one
l	**l**oud	g	**g**old
r	**r**ide	k	fa**k**e
w	**w**ax	p	**p**et
y	**y**ard	t	**t**ea
hh	**h**ouse		

Table A-1. Phoneme Set used for experiments in Chapter 5.

Appendix B Comparison of ACDM-MMSE to VTS

In this appendix, a derivation of the VTS-1 estimation equation in the cepstral domain is presented, with the objective of deriving a result for comparison with the proposed ACDM-MMSE estimator. The derivation starts with the well-known nonlinear acoustic distortion model

$$\begin{aligned} &\boldsymbol{y} = \boldsymbol{x} + g(\boldsymbol{x}, \boldsymbol{n}), \\ &g\left(\boldsymbol{x}, \boldsymbol{n}\right) = \log\left(\mathbf{i} + e^{\boldsymbol{n}-\boldsymbol{x}}\right). \end{aligned} \tag{B.1}$$

Here, $\boldsymbol{x}$, $\boldsymbol{n}$, and $\boldsymbol{y}$ are the clean speech, noise, and corrupted speech log filter bank coefficient vectors, respectively, and **i** is the identity vector. Equation (B.1) is expanded around an initial point $\boldsymbol{x}_0$ with a first-order Taylor series expansion, using $\boldsymbol{n} = \boldsymbol{n}_0$, to give

$$\boldsymbol{y} = \left(I + \nabla_{\boldsymbol{x}} g\left(\boldsymbol{x}_0, \boldsymbol{n}_0\right)\right)\mathbf{x} + g\left(\boldsymbol{x}_0, \boldsymbol{n}_0\right) - \nabla_{\boldsymbol{x}} g\left(\boldsymbol{x}_0, \boldsymbol{n}_0\right)\boldsymbol{x}_0. \tag{B.2}$$

If both sides of (B.2) are multiplied by a DCT matrix, Λ, the result is (after splitting the first term)

$$\Lambda\boldsymbol{y} = \Lambda\left(I + \nabla_{\boldsymbol{x}} g\left(\boldsymbol{x}_0, \boldsymbol{n}_0\right) - I\right)\boldsymbol{x} + \Lambda\boldsymbol{x} + \Lambda g\left(\boldsymbol{x}_0, \boldsymbol{n}_0\right) - \Lambda\nabla_{\boldsymbol{x}} g\left(\boldsymbol{x}_0, \boldsymbol{n}_0\right)\boldsymbol{x}_0. \tag{B.3}$$

Which, using $\mathbf{d}=\Lambda\,\boldsymbol{y}$ and $\mathbf{c}=\Lambda\,\boldsymbol{x}$, can be rewritten as

$$\mathbf{d} = \mathbf{c} + \Lambda\left(\nabla_{\boldsymbol{x}} g\left(\boldsymbol{x}_0, \boldsymbol{n}_0\right)\right)\boldsymbol{x} + \Lambda\left(g\left(\boldsymbol{x}_0, \boldsymbol{n}_0\right) - \nabla_{\boldsymbol{x}} g\left(\boldsymbol{x}_0, \boldsymbol{n}_0\right)\boldsymbol{x}_0\right). \tag{B.4}$$

The second term on the right side of (B.4) can be rewritten as

$$\Lambda\left(\nabla_{\boldsymbol{x}} g\left(\boldsymbol{x}_0, \boldsymbol{n}_0\right)\right)\boldsymbol{x} = \Lambda\left(\nabla_{\boldsymbol{x}} g\left(\boldsymbol{x}_0, \boldsymbol{n}_0\right)\right)\left(\Lambda^{-1}\Lambda\right)\boldsymbol{x}. \tag{B.5}$$

(B.4) is then represented as

$$\begin{aligned} &\mathbf{c} = \mathbf{d} - \mathbf{A}\mathbf{c} + \mathbf{b}, \\ &\mathbf{A} = \Lambda\left(\nabla_{\boldsymbol{x}} g\left(\boldsymbol{x}_0, \boldsymbol{n}_0\right)\right)\Lambda^{-1}, \\ &\mathbf{b} = \Lambda\left(\nabla_{\boldsymbol{x}} g\left(\boldsymbol{x}_0, \boldsymbol{n}_0\right)\boldsymbol{x}_0 - g\left(\boldsymbol{x}_0, \boldsymbol{n}_0\right)\right). \end{aligned} \tag{B.6}$$

The MMSE estimator for **c** is found by

$$\hat{\mathbf{c}}_{MMSE} = \int_C \mathbf{c} p(\mathbf{c}\,|\,\mathbf{d})\, d\mathbf{c} = \int_C \mathbf{d} - \mathbf{A}\mathbf{c} + \mathbf{b} p(\mathbf{c}\,|\,\mathbf{d})\, d\mathbf{c} \tag{B.7}$$

$$= \sum_{m=0}^{M} p[m\,|\,\mathbf{d}] \int_C \mathbf{d} - \mathbf{A}\mathbf{c} + \mathbf{b} p(\mathbf{c}\,|\,\mathbf{d}, m)\, d\mathbf{c}, \tag{B.8}$$

where *m* is the index of a mixture in a GMM prior model of clean speech. The integral can be split and terms can be rearranged to give

$$\hat{\mathbf{c}} = \sum_{m=0}^{M} P[m\,|\,\mathbf{d}] \left\{ -\mathbf{A} \int_C \mathbf{c} p(\mathbf{c}\,|\,\mathbf{d}, m)\, d\mathbf{c} + \mathbf{d} \int_{\mathbf{C}} p(\mathbf{c}\,|\,\mathbf{d}, m) + \mathbf{b} \int_C p(\mathbf{c}\,|\,\mathbf{d}, m)\, d\mathbf{c} \right\}, \tag{B.9}$$

Substituting $\int_{\mathbf{C}} \mathbf{c} p(\mathbf{c}\,|\,\mathbf{d}, m)\, d\mathbf{c} = \boldsymbol{\mu}_{\mathbf{c},m}$ and $\int_{\mathbf{C}} p(\mathbf{c}\,|\,\mathbf{d}, m)\, d\mathbf{c} = 1$, equation (B.9) can be transformed into

$$\begin{aligned}
&\hat{\mathbf{c}} = \sum_{m=0}^{M} \gamma_m \left\{ \mathbf{W}_1 \boldsymbol{\mu}_{\mathbf{c},m} + [\mathbf{d} + \mathbf{f}_0] \right\}, \\
&\mathbf{W}_1 = -\mathbf{A} = \left(-\Lambda \nabla_{\boldsymbol{x}} g(\boldsymbol{x}_0, \boldsymbol{n}_0) \Lambda^{-1} \right), \\
&\mathbf{f}_0 = \mathbf{b} = \Lambda \left(g(\boldsymbol{x}_0, \boldsymbol{n}_0) - \nabla_{\boldsymbol{x}} g(\boldsymbol{x}_0, \boldsymbol{n}_0) \boldsymbol{x}_0 \right), \\
&\gamma_m = P[m\,|\,\mathbf{d}].
\end{aligned} \tag{B.10}$$

By comparing (4.22) and (B.10), it can be seen that, although the form is similar, the weights on the two components for each mixture *m*, the prior mean and the enhanced value, are not the same. In the ACDM-MMSE estimator, they will always sum to unity and are based on the relative variances of the two Gaussians (prior and conditional). In the VTS equation, the weight for the prior mean is not based on the variance of the prior or conditional Gaussian and the weights will never sum to unity, since the weight on the enhanced value is already 1. Also, the enhanced values $\mathbf{f}_0$ and $\boldsymbol{\mu}^{\mathbf{g}}$ are computed differently.