Robust Speech Recognition in the Presence of Additive Noise

Svein Gunnar Storebakken Pettersen

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of

PHILOSOPHIAE DOCTOR

Department of Electronics and Telecommunications
Norwegian University of Science and Technology

2008

Abstract

It is well known that additive noise can cause a significant decrease in performance for an automatic speech recognition (ASR) system. For an ASR system to maintain an acceptable level of performance in noisy conditions, measures must be taken to make it robust. Since prior information about the noise is usually not available, this information typically has to be obtained from the observed noisy utterance that is to be recognized.

Model compensation is one way of achieving robustness towards noise. One of the main problems with model compensation is how to approximate the non-linear relationship between speech, noise, and noisy speech in the log-spectral domain. In an effort to investigate the effects of approximation accuracy, a comparative study of two existing and one new method for approximating this relationship is presented. The study shows that, although the approximation methods differ in accuracy on a one-dimensional example, the recognition results on Aurora2 are almost equal in practice.

Due to several factors, the noisy speech parameter estimates obtained when performing model compensation will normally be uncertain, limiting the attainable performance. We propose a new model compensation approach, in which a robust decision rule is combined with traditional parallel model combination (PMC) to compensate for uncertainty. Experiments show that the proposed approach is effective in increasing performance at low signal-to-noise ratios (SNRs) for most noise types compared to PMC.

Another way of improving ASR performance in noisy conditions is by applying a feature enhancement algorithm prior to recognition. Many existing feature enhancement techniques rely on probabilistic models of speech and noise. Thus, the performance is influenced by the quality of these models. Traditionally, the probabilistic models have been trained using maximum likelihood estimation. This dissertation investigates the use of an alternative estimation method for prior speech models, namely Bayesian learning. It is shown that, within the chosen experimental setup, Bayesian learning can be used for model selection, and that the recognition performance is comparable to the performance obtained with maximum likelihood in most cases.

A good probabilistic model for the noise can be difficult to obtain, since it usually has to be estimated directly from the utterance at hand. In order to improve the quality of the noise model used by the feature enhancement algorithm, we investigate the use of voice activity detection (VAD) to obtain information about the noise. An advantage of the proposed VAD approach is that it works in the same domain as the speech recognizer. Experiments show that the VAD approach on average obtains a 10.8% error rate reduction compared to simply using a speech-free segment from the beginning of the utterance for noise modeling.

Preface

This dissertation is submitted in partial fulfillment of the requirements for the degree of *philosophiae doctor* (PhD) at the Norwegian University of Science and Technology (NTNU). My supervisor has been Associate Professor Magne Hallstein Johnsen at the Department of Electronics and Telecommunications, NTNU. In addition, Dr. Tor André Myrvoll has been my co-supervisor. He was affiliated with the Department of Electronics and Telecommunications, NTNU until January 2007, and is currently with SINTEF, Trondheim.

The work was conducted in the period from July 2004 to September 2008. In addition to research activity, the work included the equivalent of one and a half semester of course studies, and one year of teaching assistant duties. The work also included a research visit to Eurecom in Sophia-Antipolis, France, under the supervision of Professor Christian Wellekens. The duration of this stay was approximately 6 months.

The research period of this work was funded by the Norwegian Research Council, through the BRAGE project, which is organized under the language technology program KUNSTI. The period with assistant duties was funded by the Department of Electronics and Telecommunications, NTNU.

Acknowledgments

First of all, I would like to thank my supervisor, Associate Professor Magne Hallstein Johnsen, for his guidance and support throughout this work. Moreover, I would like to thank Dr. Tor André Myrvoll for helpful ideas and suggestions.

I am also grateful to Professor Christian Wellekens for making my visit to Eurecom possible, and for being very helpful and supportive during my stay.

All my friends and colleagues at the Signal Processing Group at NTNU contributed to an enjoyable working environment during my time in Trondheim. In particular, I would like to thank Dr. Øystein Birkenes and Trond Skogstad for reviewing my thesis.

Finally, I am grateful to my family for their support. And especially, I would like to thank my wife Renate for her unceasing love and encouragement.

Drammen, September 2008
Svein Gunnar Pettersen

Contents

Notation and Symbols

$\mathbf{a}$	Bold lowercase letters are used for vectors
$\mathbf{A}$	Bold uppercase letters are used for matrices and sequences of vectors
$\mathbf{A}^T$	The transpose of $\mathbf{A}$
$\mathbf{A}^{-1}$	The inverse of $\mathbf{A}$
$\lvert\mathbf{A}\rvert$	The determinant of $\mathbf{A}$
$\mathrm{tr}(\mathbf{A})$	The trace of $\mathbf{A}$ (sum of the diagonal elements)
$\mathrm{diag}(\mathbf{a})$	A diagonal matrix containing the elements of $\mathbf{a}$
$\mathrm{vec}(\mathbf{A})$	The columns of $\mathbf{A}$ stacked into a column vector
$\lVert\mathbf{a}\rVert$	The Euclidean norm of $\mathbf{a}$
$\langle\mathbf{a},\mathbf{b}\rangle$	The inner product of $\mathbf{a}$ and $\mathbf{b}$
$\mathrm{E}[\cdot]$	Expectation
$\mathrm{Var}[\cdot]$	Variance
$\circ$	Element-wise matrix product
$\propto$	Proportional to

List of Abbreviations

ASR	automatic speech recognition
BP-MC	Bayesian predictive density based model compensation
BPC	Bayesian predictive classification
BPC-PMC	joint BPC and PMC
DCT	discrete cosine transform
DFT	discrete Fourier transform
EM	expectation-maximization
GMM	Gaussian mixture model
HMM	hidden Markov model
HTK	hidden Markov model toolkit
KL	Kullback-Leibler
LLR	log likelihood ratio
LRT	likelihood ratio test
MAP	maximum a posteriori
MBFE	model-based feature enhancement
MFCC	mel-frequency cepstrum coefficient
ML	maximum likelihood
MMSE	minimum mean-square error
PMC	parallel model combination
PSD	power spectral density
SNR	signal-to-noise ratio
VAD	voice activity detection
VB	variational Bayes
VTS	vector Taylor series

Chapter 1

Introduction

Automatic speech recognition (ASR) is the process of automatically converting an acoustic speech signal into text using a computer. ASR performance comparable to that of a human being has proved difficult to achieve. The vast amount of research publications in this field during the past few decades is a testament to the complexity of the problem. What makes the problem so difficult is the great variability in speech signals, channel, and environment conditions. The acoustic realization of an utterance depends on speaker characteristics such as age, sex, and dialect. In addition, the acoustic signal is affected by noise from the environment and distortions from the channel, microphone, and analog-to-digital converter before the signal can be processed in the computer.

Despite these challenges, ASR technology has already been put to use in a number of real-world applications such as dictation software and automatic telephone services. Many typical ASR applications require the ASR system to be *robust* in order to be useful. For an ASR system to be robust it has to be able to maintain an acceptable level of performance when exposed to a range of different conditions. This means that the system has to handle both speaker variabilities and acoustic variabilities.

The focus of this thesis will be on robustness towards noise. Many potential applications for ASR rely on the ability of ASR systems to work well in noisy environments. Consequently, a lot of research has been done on noise robustness. Speech signals can be influenced by both additive background noise and convolutional distortions resulting from microphones, room acoustics, and transmission channels. A lot of the research that has been done on noise robust ASR considers both additive and convolutional noise (see e.g. [44, 66, 46]). However, this thesis will focus on additive background noise.

The rest of this chapter is organized as follows. In Section 1.1, a brief description of the noise robustness problem as well as an overview of different approaches for dealing with the problem are given. Then, Section 1.2 will give an overview of the contributions of this thesis.

1.1 The Noise Robustness Problem

In many practical applications, ASR systems are trained without knowing exactly under which environmental conditions they will be used. As an example, consider a telephone based dialog system that lets users call from any location. Then, the system has to be able to deal with many different types of background noise. In practice, it will be impossible to collect training data that represent all possible environmental conditions for this system. The main problem we face then is that the observed speech will have statistics different from the statistics of the training data. In order to solve this problem there are generally four different approaches:

1. Use features that are robust towards noise, so that the changes in statistics will be as small as possible.

2. Find a transformation of the feature vectors that will reduce the effect of the noise on the feature vectors.

3. Transform the ASR model parameters to match the observed environmental conditions.

4. Use robust decision rules that compensate for uncertainty in parameter estimates.

In this thesis we will focus on methods that use one or more of the last three approaches, i.e., *robust feature transformations*, *model-based techniques*, and *robust decision rules*. It will be assumed that the ASR system has been trained using only clean speech data, i.e., with no additive background noise present.

Most of the techniques described in this thesis require a statistical model of the noise. In order to estimate the parameters of this model, a small amount of noise data is needed. Since we rarely have access to relevant noise data for all possible situations when training the recognizer, this data usually has to be obtained from the observed noisy utterance that is to be recognized. There are two fundamentally different approaches for obtaining noise data from a noisy speech signal. One can either try to detect segments of the signal that consist only of noise, or one can try to track the noise event through periods of speech activity. A simplified version of the former approach that is commonly used is to assume that the observed signal always contains a short noise-only segment before the speech begins. This segment is then either used to estimate noise parameters directly or to estimate an initial noise model for extracting more noise data from the rest of the signal. Throughout this thesis we will also make the assumption that the first portion of each observed signal is speech-free.

The database that will be used for testing noise robustness algorithms in this thesis is the Aurora2 database [32], which consists of spoken digit strings with artificially added noise. A description of this database can be found in Appendix A.

1.2 Contributions of This Thesis

This thesis provides a study of methods for robust automatic speech recognition in the presence of additive noise. In order to improve on some of the weaknesses of existing approaches, we present contributions for both model-based methods and robust feature transformations.

A Comparative Study of Approximations for Model Compensation

One of the main problems with model compensation is how to approximate the non-linear relationship between speech, noise, and noisy speech in the log-spectral domain. In an effort to investigate the effects of approximation accuracy, Chapter 4 presents a comparative study of methods for approximating this relationship. Two previously proposed techniques as well as one method that is new in this context are examined. A detailed analysis of experimental results is then carried out to investigate to which extent methods of different approximation accuracy will result in different estimates when used in practice.

Combining Model Compensation and a Robust Decision Rule

Due to several factors, the noisy speech parameter estimates obtained when performing model compensation will normally be uncertain. This uncertainty causes a limitation of attainable performance, which becomes more severe as the SNR decreases. Robust decision rules have been proposed in the context of ASR with the purpose of compensating for uncertainty in parameter estimates. This motivates a combined approach which is presented in Chapter 5. This approach to model compensation incorporates a robust decision rule called Bayesian predictive classification (BPC). The combination is achieved by using the noisy speech mean from model compensation in the prior distribution of BPC. Experimental results of the joint approach are then analyzed. The analysis reveals a problem with the joint approach when the noise consists mainly of background speech. In order to alleviate this problem a prior scaling technique is introduced.

Bayesian Learning of Speech Models for MMSE Feature Enhancement

Methods for minimum mean-square error (MMSE) feature enhancement of noisy speech rely on probabilistic models of speech and noise in order to obtain good estimates of clean speech. The quality of these probabilistic models will therefore influence the quality of the resulting clean speech estimates. Traditionally, the probabilistic models have been trained using maximum likelihood (ML) estimation. Chapter 6 starts by reviewing some of the theory behind Bayesian learning, which in many cases has advantages compared to ML learning. Then, a study on application of this theory to front-end models for MMSE feature enhancement is presented. Experiments with various model sizes are performed and compared to ML. The ability of Bayesian learning to do model selection is also investigated.

Improved Noise Modeling for MMSE Feature Enhancement Using Voice Activity Detection

One of the main challenges with most noise robustness techniques is to obtain reliable estimates of the statistical parameters of the noise. Extracting this information from only the current noisy utterance is far from trivial. In Chapter 7 we investigate one of the possible approaches for noise parameter estimation, namely voice activity detection (VAD). We propose a novel VAD method that is used for obtaining improved noise models for MMSE feature enhancement of noisy speech. One advantage of the proposed VAD approach is that it works in the same domain as the speech recognizer. The method is compared to only using the first portion of each utterance for noise modeling, as well as another well-known VAD method working in the discrete Fourier transform domain. The results are also compared to an approximate upper bound VAD. This upper bound VAD was obtained by running forced alignment on clean speech data to find silence segments that correspond to noise-only segments in the noisy speech data.

Chapter 2

Speech Recognition Based on the Hidden Markov Model

In this chapter we will give an overview of speech recognition based on the hidden Markov model (HMM). We will begin by looking at how the problem of speech recognition can be related to statistical decision theory, which is the basis of the probabilistic approach to ASR. Then, we will move on to describe the HMM, before we give an overview of a typical HMM-based ASR system. Finally, we explain how to calculate mel-frequency cepstrum coefficients, which have become the most popular features for ASR during the last couple of decades.

2.1 Statistical Decision Theory

As mentioned in Chapter 1, the objective of a speech recognizer is to convert a speech signal into the corresponding sequence of words. A speech recognizer can be represented mathematically as a function d that maps speech signals into word strings, i.e.,

$$d : \mathcal{Y} \rightarrow \mathcal{W}, \tag{2.1}$$

where $\mathcal{Y}$ is the set of all possible speech signals, and $\mathcal{W}$ is the set of all possible word strings. In statistical theory, such a function is called a *decision rule*. When using a speech recognizer in practice, the observed speech signal $\mathbf{Y} \in \mathcal{Y}$ is used to generate an estimate $\hat{W} \in \mathcal{W}$ of the correct word string $W \in \mathcal{W}$, i.e.,

$$\hat{W} = d(\mathbf{Y}). \tag{2.2}$$

The statistical way of measuring the quality of such an estimate is by defining a *loss function*. A loss function l maps a pair $(W, \hat{W})$ to a non-negative real number. For speech recognition a common choice is the (0,1)-loss function, which is defined as

$$l(W, \hat{W}) = \begin{cases} 0 & \text{if } \hat{W} = W \\ 1 & \text{if } \hat{W} \neq W. \end{cases} \tag{2.3}$$

This means that all recognition errors are assigned equal loss, while a correct recognition result gives zero loss. Having defined a loss function it would be useful to predict the expected performance of different speech recognizers. This can be achieved by viewing $(W, \mathbf{Y})$ as a jointly distributed random pair and looking at the expected value of the loss function. If we assume that the true distribution $p(W, \mathbf{Y})$ is known, we can calculate this expectation as

$$\begin{aligned} r(d(\cdot)) &= \mathrm{E}_{W,\mathbf{Y}}[l(W, d(\mathbf{Y}))] && (2.4) \\ &= \sum_{W \in \mathcal{W}} \int_{\mathbf{Y} \in \mathcal{Y}} l(W, d(\mathbf{Y})) p(W, \mathbf{Y}) d\mathbf{Y}. && (2.5) \end{aligned}$$

The quantity $r(d(\cdot))$ is called *total risk*. Given the (0,1)-loss function it can be shown [34] that the decision rule that minimizes the total risk is *Bayes' decision rule*, also called the *maximum a posteriori* (MAP) decision rule, given by

$$\begin{aligned} d(\mathbf{Y}) &= \arg\max_W P(W|\mathbf{Y}) && (2.6) \\ &= \arg\max_W p(\mathbf{Y}|W)P(W). && (2.7) \end{aligned}$$

In ASR the probability distribution of the pair $(W, \mathbf{Y})$ is usually modeled as

$$p(W, \mathbf{Y}) = p_{\mathbf{\Lambda},\mathbf{\Gamma}}(W, \mathbf{Y}) = p_{\mathbf{\Lambda}}(\mathbf{Y}|W) P_{\mathbf{\Gamma}}(W), \tag{2.8}$$

where $p_{\mathbf{\Lambda}}(\mathbf{Y}|W)$ is called the acoustic model and $P_{\mathbf{\Gamma}}(W)$ is called the language model. We have assumed that the acoustic model is a member of a parametric family with parameters $\mathbf{\Lambda}$. The language model probabilities are denoted by $\mathbf{\Gamma}$. Based on these distributions, the MAP decision rule in (2.7) is given by

$$\hat{W} = \arg\max_W p_{\mathbf{\Lambda}}(\mathbf{Y}|W) P_{\mathbf{\Gamma}}(W). \tag{2.9}$$

If the assumed model structures of $p_{\mathbf{\Lambda}}(\mathbf{Y}|W)$ and $P_{\mathbf{\Gamma}}(W)$ were correct and the true values of $(\mathbf{\Lambda}, \mathbf{\Gamma})$ known, the decision rule in (2.9) would be optimal

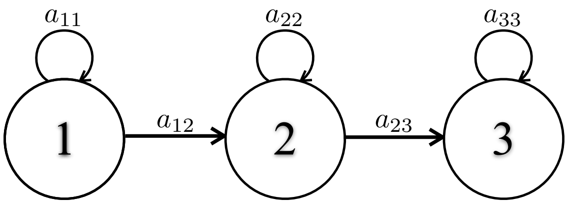

Figure 2.1: An example HMM

in the sense that it minimizes the expected sentence error rate. In practice, however, we have to select model structures in order to approximate the true (unknown) distribution $p(W, \mathbf{Y})$. Moreover, estimated values $(\hat{\mathbf{\Lambda}}, \hat{\mathbf{\Gamma}})$ have to be found from training data. These estimates are then plugged into (2.9), resulting in the *plug-in MAP* decision rule:

$$\hat{W} = \arg\max_{W} p_{\hat{\mathbf{\Lambda}}}(\mathbf{Y}|W) P_{\hat{\mathbf{\Gamma}}}(W). \tag{2.10}$$

The remaining problem is to find suitable probabilistic models for speech and language. In ASR, the most common choice for $p_{\mathbf{\Lambda}}(\mathbf{Y}|W)$ is the hidden Markov model, whereas the most common choice for $P_{\mathbf{\Gamma}}(W)$ is the N-gram language model.

2.2 The Hidden Markov Model

The hidden Markov model (HMM) is a powerful statistical model which is well suited for characterizing discrete time series of observations. The HMM can be viewed as an extension of the Markov chain. In a Markov chain there is a set of states with a corresponding set of probabilities for jumping from one state to another. Observations generated by each state are deterministic, i.e., a given state generates the same observation every time. In the HMM this model is extended by introducing a probability distribution in each state. Consequently, the same state can generate different observations. This means that, given a sequence of observations, it is usually not possible to know which state generated each observation. Thus, the underlying state sequence is *hidden*.

During the past decades the HMM has become the most popular choice for acoustic modeling in ASR. When applying this model to speech, one assumes that the speech signal is *short-time stationary*. This means that within a small time frame, e.g. 20-30 ms, we assume that the speech signal behaves like a stationary random process.

A HMM is defined by the following parameters:

- A set of states $\mathcal{S} = \{1, \ldots, N\}$.
- A transition probability matrix $\mathbf{A} = [a_{ij}]$.
- A set of state-dependent observation densities $\{p_i(\cdot)\}$.
- An initial state distribution $\boldsymbol{\pi} = \{\pi_i\}$.

Let us denote the state at time t by q_t, and the state sequence by $Q = (q_t)$. The HMM is based on the same assumption as a first order Markov chain, the *Markov assumption*:

$$P(q_t|q_{t-1}, \ldots, q_0) = P(q_t|q_{t-1}), \text{ for all } t. \tag{2.11}$$

In addition, the HMM assumes that the observation generated at time t only depends on the current state. An element of the transition probability matrix a_{ij} specifies the probability for a transition from state i to state j, i.e.,

$$a_{ij} = P(q_t = j|q_{t-1} = i). \tag{2.12}$$

An element of the initial state distribution specifies the probability for starting in state i, i.e.,

$$\pi_i = P(q_0 = i). \tag{2.13}$$

An example of a HMM with a topology that is commonly used for modeling a phoneme is shown in Figure 2.1. As we can see, this HMM has three states. With the allowed transitions given in the figure, the transition matrix is given by

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & 0 \\ 0 & a_{22} & a_{23} \\ 0 & 0 & a_{33} \end{bmatrix}. \tag{2.14}$$

The idea behind modeling phonemes with three-state HMMs is that the states should model the first, middle, and last parts of a phone respectively. Thus, one usually requires that the initial state must be the first state, which means that $\boldsymbol{\pi} = \{\pi_1 = 1, \pi_2 = 0, \pi_3 = 0\}$. For the HMM in Figure 2.1, the observation densities $p_1(\cdot)$, $p_2(\cdot)$, and $p_3(\cdot)$ also need to be specified. The most common choice of observation density is the Gaussian mixture model (GMM), and this is also what will be used throughout this thesis. Assuming that the GMMs of all states consist of M mixture components, the observation density for state i can be written as

$$p_i(\mathbf{y}_t) = \sum_{m=1}^{M} w_{im} \mathcal{N}(\mathbf{y}_t; \boldsymbol{\mu}_{im}, \boldsymbol{\Sigma}_{im}) \tag{2.15}$$

where $\mathbf{y}_t$ denotes the observation at frame t, and $\mathcal{N}$ denotes a multivariate normal distribution with mean vector $\boldsymbol{\mu}_{im}$ and covariance matrix $\boldsymbol{\Sigma}_{im}$, i.e.,

$$\mathcal{N}(\mathbf{y}_t; \boldsymbol{\mu}_{im}, \boldsymbol{\Sigma}_{im}) = \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}_{im}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{y}_t - \boldsymbol{\mu}_{im})^T \boldsymbol{\Sigma}_{im}^{-1}(\mathbf{y}_t - \boldsymbol{\mu}_{im})\right). \tag{2.16}$$

In (2.16) the dimension of the vector $\mathbf{y}_t$ is denoted by D. It is also common to apply a decorrelating transform during the computation of the feature vectors $\{\mathbf{y}_t\}$. Then, by assuming that there is no correlation between the elements of the feature vector, we can use diagonal covariance matrices instead of full. The expression in (2.16) is then reduced to

$$\mathcal{N}(\mathbf{y}_t; \boldsymbol{\mu}_{im}, \boldsymbol{\Sigma}_{im}) = \frac{1}{(2\pi)^{D/2}} \prod_{d=1}^{D} \frac{1}{\sigma_{imd}} \exp\left(-\frac{(y_{td} - \mu_{imd})^2}{2\sigma_{imd}^2}\right). \tag{2.17}$$

where

$$\mathbf{y}_t = [y_{t1}, \ldots, y_{tD}]^T \tag{2.18}$$

$$\boldsymbol{\mu}_{im} = [\mu_{im1}, \ldots, \mu_{imD}]^T \tag{2.19}$$

$$\boldsymbol{\Sigma}_{im} = \operatorname{diag}\left([\sigma_{im1}^2, \ldots, \sigma_{imD}^2]^T\right). \tag{2.20}$$

The use of diagonal covariance matrices reduces the computational complexity.

Before we can use the HMM for speech recognition, we need to estimate its parameters. Since the state sequence is hidden it is necessary to use an iterative procedure for maximum likelihood estimation of the parameters. Fortunately, an elegant solution is provided by the Baum-Welch algorithm [4, 59], which is a special case of the expectation-maximization (EM) algorithm [11].

When using the HMM for ASR, we are making some incorrect assumptions about speech:

- As a consequence of the Markov assumption, the modeling of state duration is constrained to a geometric distribution. This means that the state duration probability decreases exponentially as a function of time, and such a model is inappropriate for almost any speech event [41].

- The HMM assumes that a speech signal consists of a series of stationary segments. This is only an approximation, since an utterance usually contains several non-stationary segments as well.

- The assumption that observations are only dependent on the current state, and thus independent of neighboring observations, is obviously wrong.

Despite these limitations, the HMM has been shown to work well for modeling speech, and it is still the most popular model for speech recognition.

2.3 Overview of an HMM-Based ASR System

Most HMM-based speech recognizers can be described by the block diagram shown in Figure 2.2. The sampled speech waveform y is passed to a block that performs preprocessing. The output from preprocessing is a sequence of vectors $\mathbf{Y}$ which constitute an alternative representation of the speech signal that is more suitable for statistical pattern recognition. These vectors are then passed to the decoder which produces a recognition hypothesis $\hat{W}$. The decoder makes use of three additional modules: acoustic models, lexicon, and language model. We will soon give a brief description of each block in Figure 2.2, but first we will introduce the concept of sub-word units, which is important for many ASR systems.

2.3.1 Sub-Word Units

In principle one would need to train an acoustic model for every possible word string W. For many ASR applications this is impractical, since the number of possible sentences usually will be very large. When using the HMM it is possible to define a relatively small set of sub-word units that can be combined to generate all possible word strings. Then, one only has to train an HMM for each sub-word unit, and these can be concatenated to form all possible word strings. Examples of sub-word units are phonemes and syllables.

2.3.2 Preprocessing

The objective of the preprocessor block is to extract information from the sampled speech waveform. This is done by generating a sequence of feature vectors, which provide a more compact representation of the speech signal. Ideally, the feature vectors should discard information about factors such as speaker and environmental characteristics while retaining information about the linguistic content of the utterance.

The most common approach to preprocessing is to decompose the speech waveform into a series of short segments, and generate a feature vector

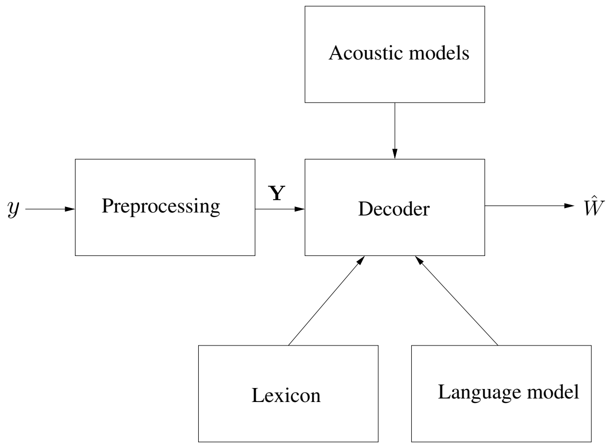

Figure 2.2: Block diagram of a HMM-based speech recognizer.

based on each segment. One such segment will be referred to as a *frame*. The duration of a frame is typically about 25 milliseconds. In order to have some overlap between consecutive frames, the frame shift is typically about 10 milliseconds. The assumption behind this processing scheme is that within a frame, the speech signal can be modeled as a stationary random process.

Each frame is processed by a feature extraction algorithm that results in a feature vector with a significantly lower dimension than the signal frame. Examples of different types of features are linear predictive coding (LPC) features (see e.g. [47]), perceptual linear predictive (PLP) features [31], and subband spectral centroid histogram (SSCH) features [24]. In this thesis, however, we will focus on mel-frequency cepstrum coefficients (MFCCs), which will be described in more detail in Section 2.4.

In order to capture temporal changes in the signal spectra, the feature vectors are usually augmented by time derivatives of the basic static parameters. It is common to use both delta and acceleration (delta-delta) coefficients, resulting in a feature vector of size three times the number of static parameters.

2.3.3 Pronunciation Lexicon

A pronunciation lexicon is used in ASR systems that are based on subword units, typically phonemes. The lexicon contains a list of words that constitute the vocabulary of the ASR system. For each vocabulary word the lexicon also contains one or more phonetic transcriptions. Thus, the lexicon tells the recognizer how to create a model for each word by concatenating sub-word models.

2.3.4 Acoustic Models

The acoustic models are the HMMs that we described in Section 2.2. They are used by the decoder for finding the likelihood that a given word string W resulted in a sequence of observed feature vectors. If the ASR system uses sub-word units, the likelihood $p_{\mathbf{\Lambda}}(\mathbf{Y}|W)$ will be found by concatenating the acoustic models of several sub-word units.

2.3.5 Language Model

In general the language model has two objectives. The first objective is to put constraints on how word strings can be constructed. This is done in order to limit the number of possible word strings to a set containing sentences that are meaningful for a given application. In addition, a good language model should be able to calculate prior probabilities of different word strings. As we have already mentioned, the most common approach for this purpose is the N-gram language model. An N-gram is a statistical language model that assigns a probability to a word by taking into account the $N-1$ previous words in the word string, in addition to the current word. The probabilities of the N-gram have to be estimated from language data. The amount of probabilities that need to be estimated increases rapidly as we increase N. Consequently, N is usually restricted to 3 or less.

2.3.6 Decoder

The objective of the decoder is to produce a recognition hypothesis according to a given decision rule, typically the plug-in MAP rule given by (2.10). Ideally, we should calculate $p_{\hat{\mathbf{\Lambda}}}(\mathbf{Y}|W)P_{\hat{\mathbf{\Gamma}}}(W)$ for every possible hypothesis W, and select the one with the greatest posterior probability. However, this would require that we sum over every possible sequence of HMM states in order to find the likelihood of each hypothesis. For continuous speech recognition, this approach is usually too computationally complex in practice. Therefore, the sum over all possible state sequences is usually approximated by the likelihood of the most likely state sequence. The decoder uses acoustic models, lexicon, and language model to create a network of HMM states. The Viterbi algorithm, which is based on dynamic programming, can then be used to search through the network, and this makes the computation feasible. After having found the most likely state sequence in

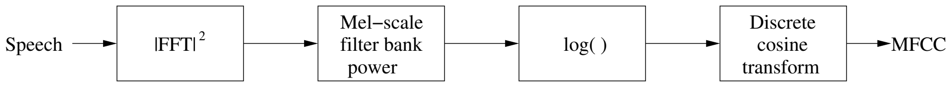

Figure 2.3: Block diagram for calculation of MFCC features.

the HMM-network, we can backtrack the result and find the corresponding sequence of words.

2.4 Mel-Frequency Cepstrum Coefficients

In [10] MFCC features were found to work well for ASR. Factors that have contributed to their popularity are low computational complexity and high recognition performance in clean conditions. The calculation of MFCC features contains elements motivated by human perception, such as a filter bank based on the perceptually motivated mel-scale (see e.g. [33, p. 34]), and approximation of perceived loudness by the logarithm of filter bank power. After decomposing the speech waveform into frames, MFCC features are calculated by passing the frames through the block diagram shown in Figure 2.3.

In the first block a discrete Fourier transform (DFT) is calculated by applying a fast Fourier transform (FFT) to the input signal. Then, the magnitude or squared magnitude is calculated for each of the DFT coefficients. In the next block the signal is passed through a filter bank where the filter center frequencies are uniformly spaced on the mel-scale, and the filter bandwidths are constant on the mel-scale. The result is a power estimate for each subband. In the next block we take the $\log(\cdot)$ of each subband power value. Finally, in the last block we apply a discrete cosine transform (DCT) to the vector of log power estimates. The main purpose of the DCT is decorrelation of the log filter bank powers. As we saw in Section 2.2, decorrelation of the feature vectors gives reduced computational complexity in the decoder due to the use of acoustic models with diagonal covariance matrices.

Since the DCT is a linear transform we can find the MFCC vector for a given frame by multiplying the corresponding log filter bank vector by a matrix $\mathbf{C}$. Whereas the number of filters in the filter bank is usually more than twenty, it is common to use only the first 13 coefficients resulting from the DCT. This means that the matrix $\mathbf{C}$ is rectangular. As we shall see in Chapter 3, several methods for robust ASR rely on the ability to do the inverse DCT in order to go back to the log-spectral domain. If the dimension has been reduced by the use of a rectangular DCT matrix, the inverse mapping will not be exact. However, since only the highest order coefficients have been omitted, a smoothed version of the original log-spectral domain vector can be obtained by using a pseudo-inverse mapping. Preliminary experiments performed during the work on this thesis indicated that using a pseudo-inverse mapping instead of the exact $\mathbf{C}^{-1}$ yields an approximation that is good enough for practical use in robust ASR.

2.5 Summary

In this chapter we have given a brief introduction to HMM-based speech recognition, and described the basic principles of the HMM. We also considered how HMM-based speech recognition fits into statistical decision theory, which is the basis of the statistical approach to pattern recognition. Finally, we described how to compute the most common type of feature vectors for speech recognition, namely MFCC.

Chapter 3

Approaches to Noise Robustness

We will begin this chapter by considering the effect additive noise has on MFCCs in Section 3.1. Then, we will give an overview of some well-known methods for robustness against additive noise. Model-based methods are described in Section 3.2, robust feature transformations are described in Section 3.3, and robust decision rules are described in Section 3.4. Most of the techniques presented in Section 3.2 and Section 3.3 require a statistical model for the noise, which usually has to be found from the given utterance. Therefore, we will give a brief overview of methods for noise estimation in Section 3.5.

3.1 The Influence of Noise on MFCCs

In this section, we will describe how MFCC features are affected by additive noise. Assume that we have a speech signal $s[m]$ and a noise signal $n[m]$. The observed signal in the time domain is given by

$$y[m] = s[m] + n[m]. \tag{3.1}$$

Now, let $Y[k]$, $S[k]$, and $N[k]$ denote the DFTs of $y[m]$, $s[m]$, and $n[m]$ respectively. In the frequency domain, the noise is still additive:

$$Y[k] = S[k] + N[k]. \tag{3.2}$$

Then, taking the magnitude square results in

$$|Y[k]|^2 = |S[k]|^2 + |N[k]|^2 + 2|S[k]||N[k]|\cos(\theta[k]), \tag{3.3}$$

where $\theta[k]$ is the angle between $S[k]$ and $N[k]$.

The next step is to calculate the power for each filter output from the mel-scale filter bank. This is done by applying a series of positive weights to $|Y[k]|^2$. Let W_k^j denote the weight for DFT index k in filter bank channel j. Then, the filter bank powers are given by

$$\begin{aligned}\sum_k W_k^j |Y[k]|^2 = \sum_k W_k^j |S[k]|^2 + \sum_k W_k^j |N[k]|^2 \\ + 2\sum_k W_k^j |S[k]||N[k]|\cos(\theta[k])\end{aligned} \tag{3.4}$$

for $j = 1, \ldots, J$.

Now, in order to simplify the notation of (3.4) we take a similar approach to that in [44] and define

$$Y_j^2 = \sum_k W_k^j |Y[k]|^2 \tag{3.5}$$

$$S_j^2 = \sum_k W_k^j |S[k]|^2 \tag{3.6}$$

$$N_j^2 = \sum_k W_k^j |N[k]|^2. \tag{3.7}$$

Then, we can write (3.4) as

$$Y_j^2 = S_j^2 + N_j^2 + 2\alpha_j S_j N_j \tag{3.8}$$

where we have also defined

$$\alpha_j = \frac{\sum_k W_k^j |S[k]||N[k]|\cos(\theta[k])}{S_j N_j}. \tag{3.9}$$

Now, we will try to obtain a relation corresponding to (3.8) in the log-spectral domain. By taking the log of each filter bank channel power we define the following log-spectral domain vectors:

$$\mathbf{y} = \begin{bmatrix} \log Y_1^2 \\ \log Y_2^2 \\ \vdots \\ \log Y_J^2 \end{bmatrix}, \quad \mathbf{s} = \begin{bmatrix} \log S_1^2 \\ \log S_2^2 \\ \vdots \\ \log S_J^2 \end{bmatrix}, \quad \mathbf{n} = \begin{bmatrix} \log N_1^2 \\ \log N_2^2 \\ \vdots \\ \log N_J^2 \end{bmatrix}. \tag{3.10}$$

We also define

$$\boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_J \end{bmatrix}. \tag{3.11}$$

With these definitions we can write (3.8) as

$$\exp \mathbf{y} = \exp \mathbf{s} + \exp \mathbf{n} + 2\boldsymbol{\alpha} \circ \exp \frac{\mathbf{s}}{2} \circ \exp \frac{\mathbf{n}}{2} \tag{3.12}$$

where the exp(·) function operates element-wise on vectors. Then, by taking the log on both sides we obtain

$$\mathbf{y} = \mathbf{s} + \log\left(\mathbf{1} + \exp(\mathbf{n}-\mathbf{s}) + 2\boldsymbol{\alpha} \circ \exp\left(\frac{\mathbf{n}-\mathbf{s}}{2}\right)\right), \tag{3.13}$$

where $\mathbf{1}$ denotes a vector with all elements equal to 1, and the log(·) function operates element-wise on vectors. We would now like to simplify (3.13) by removing the last term inside the log(·) and replacing it by a corresponding additive term on the outside. This can be done as follows:

$$\begin{aligned} \mathbf{y} &= \mathbf{s} + \log\left([\mathbf{1} + \exp(\mathbf{n}-\mathbf{s})] \circ \left[\mathbf{1} + \frac{2\boldsymbol{\alpha} \circ \exp\left(\frac{\mathbf{n}-\mathbf{s}}{2}\right)}{\mathbf{1} + \exp(\mathbf{n}-\mathbf{s})}\right]\right) \\ &= \mathbf{s} + \log[\mathbf{1} + \exp(\mathbf{n}-\mathbf{s})] + \mathbf{e} \end{aligned} \tag{3.14}$$

where the division $(2\boldsymbol{\alpha} \circ \exp\left(\frac{\mathbf{n}-\mathbf{s}}{2}\right))/(\mathbf{1} + \exp(\mathbf{n}-\mathbf{s}))$ is performed element-wise, and the term $\mathbf{e}$ is defined as

$$\mathbf{e} = \log\left[\mathbf{1} + \frac{2\boldsymbol{\alpha} \circ \exp\left(\frac{\mathbf{n}-\mathbf{s}}{2}\right)}{\mathbf{1} + \exp(\mathbf{n}-\mathbf{s})}\right]. \tag{3.15}$$

The influence of this additive term will be discussed in more detail in Section 3.1.1.

The final step is the DCT. Letting $\mathbf{C}$ denote the DCT matrix, we define $\mathbf{y}_C = \mathbf{C}\mathbf{y}$, $\mathbf{s}_C = \mathbf{C}\mathbf{s}$, and $\mathbf{n}_C = \mathbf{C}\mathbf{n}$. Then, we get

$$\mathbf{y}_C = \mathbf{s}_C + \mathbf{C}\log\left[\mathbf{1} + \exp(\mathbf{C}^{-1}[\mathbf{n}_C - \mathbf{s}_C])\right] + \mathbf{C}\mathbf{e}. \tag{3.16}$$

Equations (3.14) and (3.16) give the relationships between speech, noise and noisy speech in the log-spectral and cepstral domains respectively. These relationships are non-linear. As we will see in Chapter 3 this causes problems when we want to calculate unknown statistical parameters for the distribution of noisy speech given the distributions for clean speech and noise.

Note also that for the relationship in (3.16) to be exact, a square DCT matrix $\mathbf{C}$ must be used. However, as mentioned in Section 2.4, the matrix $\mathbf{C}$ is usually rectangular in practical ASR systems.

3.1.1 Discussion

In this section we will discuss the influence of the term $\mathbf{e}$ on (3.14). We will begin by considering the magnitude of α_j as defined in (3.9). Since $\cos(\theta[k]) \leq 1$, we have

$$|\alpha_j| \leq \frac{\sum_k W_k^j |S[k]||N[k]|}{S_j N_j}. \tag{3.17}$$

As was noted in the appendix of [14], the right-hand side of (3.17) is the normalized inner product of two vectors defined as

$$\mathbf{v}_S = \begin{bmatrix} \sqrt{W_1^j}|S[1]| \\ \sqrt{W_2^j}|S[2]| \\ \vdots \\ \sqrt{W_K^j}|S[K]| \end{bmatrix}, \quad \mathbf{v}_N = \begin{bmatrix} \sqrt{W_1^j}|N[1]| \\ \sqrt{W_2^j}|N[2]| \\ \vdots \\ \sqrt{W_K^j}|N[K]| \end{bmatrix}, \tag{3.18}$$

where K denotes the dimension of the DFTs. Hence, we have

$$|\alpha_j| \leq \frac{\langle \mathbf{v}_S, \mathbf{v}_N \rangle}{\|\mathbf{v}_S\|\|\mathbf{v}_N\|} \leq 1. \tag{3.19}$$

Since the different vector elements in (3.14) do not interact, we can look at each vector element of $\mathbf{y}$ independently when considering the relative influence of the term $\mathbf{e}$. Therefore, we will now work with a scalar version of (3.14), which can be written as

$$y = s + \log[1 + \exp(n - s)] + \log\left[1 + \frac{2\alpha \exp\left(\frac{n-s}{2}\right)}{1 + \exp(n - s)}\right]. \tag{3.20}$$

First, we will show that the last term can be ignored in cases where either speech or noise dominates.

If $s \gg n$ then $\exp(n - s) \approx 0$. This gives

$$y \approx s + \log[1 + 0] + \log\left[1 + \frac{0}{1}\right] = s \quad \text{if } s \gg n. \tag{3.21}$$

Thus, y will be dominated by speech. On the other hand, if $s \ll n$ then $\exp(n - s) \gg 1$. This results in

$$\begin{aligned} y &\approx s + \log(\exp(n - s)) + \log\left[1 + \frac{2\alpha \exp\left(\frac{n-s}{2}\right)}{\exp(n - s)}\right] \\ &= s + n - s + \log\left[1 + 2\alpha \exp\left(-\frac{n-s}{2}\right)\right] \quad \text{if } s \ll n. \end{aligned} \tag{3.22}$$

Since $\exp(-(n-s)/2) \approx 0$ in this case, and $|\alpha| \leq 1$, we end up with

$$y \approx s + n - s + \log[1+0] = n \quad \text{if} \quad s \ll n. \tag{3.23}$$

Thus, y will be dominated by noise.

In the case that neither s nor n dominates, the term $\mathbf{e}$ cannot be ignored. As an example consider the case $n = s$ and $\alpha = -1$. Then, we get

$$\begin{aligned} y &= s + \log 2 + \log\left[1 - \frac{2}{2}\right] \\ &= s + \log 2 + \log 0 \\ &= -\infty. \end{aligned} \tag{3.24}$$

A value of $-\infty$ here corresponds to a value of zero in the linear domain. The reason for this result is that a value of $\alpha = -1$ means that the DFT coefficients of the signal and the DFT coefficients of the noise in this subband are out of phase by an angle $\theta = \pi$, giving a sum of 0.

By assuming that the linear domain magnitude spectra $|S[k]|$ and $|N[k]|$ are constant over the bandwidth of each filter in the filter bank, each α_j can be modeled as a zero mean Gaussian random variable [17, 44]. Since α_j is then zero in an expected sense, the error term $\mathbf{e}$ is often ignored in (3.14). Some methods use the zero mean Gaussian model $p(\boldsymbol{\alpha})$ and account for uncertainty by modeling the relationship between $\mathbf{y}$, $\mathbf{s}$, and $\mathbf{n}$ as a Gaussian with the same variance as $p(\boldsymbol{\alpha})$ (see e.g. [17, 44, 14]). Note, however, that even if α_j is zero in an expected sense, the expected value of the error term $\mathbf{e}$ is not necessarily zero since the log operation is non-linear. This fact motivated Faubel *et al.* to develop a phase-averaged model in [20].

3.2 Model-Based Techniques

The objective of model-based compensation techniques is to adapt the acoustic models to the observed noisy environment. Some of the model-based compensation methods that have been proposed require training on noisy speech data before they can be applied to speech recognition. Examples of such methods are neural-network-based HMM adaptation [23] and Jacobian model adaptation [63]. As these methods do not fit into our problem formulation, they will not be considered in this thesis.

Parallel model combination (PMC) is probably the most well-known model-compensation method that does not require noisy speech training data before use.[1] As we shall see in Section 3.2.1, the key point of this

[1]Again, as mentioned in Section 1.1, we are assuming that a noise model can be estimated from the utterance we are about to recognize.

method is how to approximate the non-linear relationship between speech, noise and noisy speech. Several different approximations have been proposed, and when we later in this thesis consider different methods for model-compensation, they are all using the PMC framework but differ in approximation technique.

3.2.1 Parallel Model Combination

Parallel model combination [25, 26, 29, 27, 28] is a technique that combines an HMM for clean speech with an HMM for noise resulting in an HMM that models noisy speech. Here we will assume that the noise HMM only consists of one state, although the method can also be used for noise models with more than one state [26]. We will also assume that the noise model consists of a single Gaussian mixture component.

In PMC, the parameters of an adapted acoustic model are found by combining each of the states in the clean speech HMMs with the parameters of the noise model. For each speech state and mixture component, cepstral domain model parameters for both speech and noise are mapped back to either the linear-spectral domain or the log-spectral domain and combined to find parameters for noisy speech. Then, the resulting parameters are transformed back to the cepstral domain.

Let us begin by considering the first step: to transform cepstral domain parameters for speech and noise back to the log-spectral domain. Assuming a distribution with cepstral domain mean $\boldsymbol{\mu}^C$ and covariance matrix $\boldsymbol{\Sigma}^C$, we get the corresponding log-spectral domain parameters as

$$\boldsymbol{\mu} = \mathbf{C}^{-1}\boldsymbol{\mu}^C \tag{3.25}$$

$$\boldsymbol{\Sigma} = \mathbf{C}^{-1}\boldsymbol{\Sigma}^C(\mathbf{C}^{-1})^T. \tag{3.26}$$

We can now either do the model combination in this domain, or we can do another transformation step to obtain parameters in the linear-spectral domain and then do the combination. Combination of parameters in the log-spectral domain is done as follows:

$$\boldsymbol{\mu}_{\mathbf{y}} \approx \log[\exp(\boldsymbol{\mu}_{\mathbf{s}}) + \exp(\boldsymbol{\mu}_{\mathbf{n}})]. \tag{3.27}$$

This method is called the *log-add* approximation. When using this approximation, the compensated covariance matrix equals the clean speech covariance matrix.

For the linear-domain version, the parameters obtained in (3.25) and (3.26) are transformed once more to obtain linear-spectral domain parameter values. These are calculated as

$$\mu_i^{\text{lin}} = \exp(\mu_i + \Sigma_{ii}/2) \tag{3.28}$$

$$\Sigma_{ij}^{\text{lin}} = \mu_i^{\text{lin}} \mu_j^{\text{lin}} \left[\exp(\Sigma_{ij}) - 1\right] \tag{3.29}$$

where the subscripts i and j are used to indicate row and column numbers in the mean vectors and covariance matrices. Since speech and noise distributions are usually modeled as Gaussians in the cepstral and log-spectral domains, the corresponding distributions in the linear-spectral domain are log-normal. Then, assuming that the distribution of the sum of two log-normally distributed variables is approximately log-normal, we only need to calculate the mean and covariance matrix of the noisy speech distribution, i.e.,

$$\boldsymbol{\mu}_{\mathbf{y}}^{\text{lin}} = \boldsymbol{\mu}_{\mathbf{s}}^{\text{lin}} + \boldsymbol{\mu}_{\mathbf{n}}^{\text{lin}} \tag{3.30}$$

$$\boldsymbol{\Sigma}_{\mathbf{y}}^{\text{lin}} = \boldsymbol{\Sigma}_{\mathbf{s}}^{\text{lin}} + \boldsymbol{\Sigma}_{\mathbf{n}}^{\text{lin}}. \tag{3.31}$$

This method is called the *log-normal* approximation, since it uses the assumption that the resulting distribution for noisy speech is log-normal. Then, in order to transform the noisy speech parameters back to the log-spectral domain, the inverses of (3.28) and (3.29) are used:

$$\mu_i = \log(\mu_i^{\text{lin}}) - \frac{1}{2}\log\left(\frac{\Sigma_{ii}^{\text{lin}}}{(\mu_i^{\text{lin}})^2} + 1\right) \tag{3.32}$$

$$\Sigma_{ij} = \log\left(\frac{\Sigma_{ij}^{\text{lin}}}{\mu_i^{\text{lin}}\mu_j^{\text{lin}}} + 1\right). \tag{3.33}$$

Finally, the cepstrum domain noisy speech parameters are obtained by taking the DCT:

$$\boldsymbol{\mu}_{\mathbf{y}}^{C} = \mathbf{C}\boldsymbol{\mu}_{\mathbf{y}} \tag{3.34}$$

$$\boldsymbol{\Sigma}_{\mathbf{y}}^{C} = \mathbf{C}\boldsymbol{\Sigma}_{\mathbf{y}}\mathbf{C}^T. \tag{3.35}$$

For the log-normal approximation, a compensation scheme for delta parameters was proposed in [27]. First, the delta parameters for both speech and noise are transformed back to the linear-spectral domain in the same way as the static parameters. To simplify notation, we will now assume that all parameters are in the linear-spectral domain, without using the superscript “lin” that was used above. Then, the corrupted delta parameters are calculated as weighted sums of the clean speech and noise delta parameters

$$\Delta\mu_{y,i} = \bar{\gamma}_i \Delta\mu_{s,i} + \bar{\eta}_i \Delta\mu_{n,i} \tag{3.36}$$

where $\Delta\mu_{s,i}$ and $\Delta\mu_{n,i}$ are elements of the clean speech and noise delta mean vectors respectively. The weights $\bar{\gamma}_i$ and $\bar{\eta}_i$ are given by

$$\bar{\gamma}_i = \frac{\frac{\mu_{s,i}}{\mu_{n,i}}}{\frac{\mu_{s,i}}{\mu_{n,i}} + 1} \tag{3.37}$$

$$\bar{\eta}_i = \frac{1}{\frac{\mu_{s,i}}{\mu_{n,i}} + 1}. \tag{3.38}$$

Each element of the corrupted covariance matrix is calculated as

$$\Delta\Sigma_{y,ij} = \bar{\gamma}_i\bar{\gamma}_j\Delta\Sigma_{s,ij} + \bar{\eta}_i\bar{\eta}_j\Delta\Sigma_{n,ij}, \tag{3.39}$$

where $\Delta\Sigma_{s,ij}$ and $\Delta\Sigma_{n,ij}$ are elements of the clean speech and noise delta covariance matrices respectively.

3.2.2 Model-Based Vector Taylor Series

Recall from (3.14) in Chapter 2 that the relationship between clean speech, noise, and noisy speech in the log-spectral domain is non-linear. Vector Taylor series (VTS) is a technique that uses a Taylor series expansion to obtain a linear approximation to this relationship. This approximation has been used as a part of many different robustness algorithms since it provides a way of obtaining closed form solutions to equations that would require numerical methods to be solved when using the exact relationship.

The VTS approximation was proposed by Moreno [50, 51], who mainly focused on a feature based minimum mean-square error (MMSE) approach. Several model-domain methods based on VTS have also been proposed [43, 1, 46]. This section will consider the model-based VTS approach, while the feature-based approach will be described in Section 3.3.2.

The VTS approximation is typically performed in the log-spectral domain. It can also be applied in the cepstral domain by introducing a few simple modifications. In this section, however, we will only consider the log-spectral domain version.

The goal of model-based VTS is the same as for PMC, i.e., to find HMMs that model noisy speech, and then perform recognition with these models. For a given HMM state the clean speech is assumed to be modeled as a GMM. Assuming that the noisy speech is also distributed as a GMM, and that each mixture component can be considered separately, we are again left with the problem of determining the mean and covariance matrix of the noisy speech given the parameters of clean speech and some knowledge about the noise.

Assuming that the error term $\mathbf{e}$ in (3.14) can be ignored, we begin by writing noisy speech as

$$\mathbf{y} = \mathbf{s} + \mathbf{g}(\mathbf{s}, \mathbf{n}) \tag{3.40}$$

where we have defined

$$\mathbf{g}(\mathbf{s}, \mathbf{n}) = \log\left[\mathbf{1} + \exp(\mathbf{n} - \mathbf{s})\right]. \tag{3.41}$$

In order to determine the mean of noisy speech, we take the expectation:

$$\boldsymbol{\mu}_{\mathbf{y}} = \mathrm{E}[\mathbf{s} + \mathbf{g}(\mathbf{s}, \mathbf{n})] = \mathrm{E}[\mathbf{s}] + \mathrm{E}[\mathbf{g}(\mathbf{s}, \mathbf{n})]. \tag{3.42}$$

For simplicity, we consider a single Gaussian mixture component, and assume that the noise is known. Then, we get

$$\begin{aligned} \boldsymbol{\mu}_{\mathbf{y}} &= \boldsymbol{\mu}_{\mathbf{s}} + \int_{\mathcal{S}} \mathbf{g}(\mathbf{s}, \mathbf{n}) p(\mathbf{s}) d\mathbf{s} \\ &= \boldsymbol{\mu}_{\mathbf{s}} + \int_{\mathcal{S}} \log\left[\mathbf{1} + \exp(\mathbf{n} - \mathbf{s})\right] \mathcal{N}(\mathbf{s}; \boldsymbol{\mu}_{\mathbf{s}}, \boldsymbol{\Sigma}_{\mathbf{s}}) d\mathbf{s}. \end{aligned} \tag{3.43}$$

Unfortunately, the integral in Equation (3.43) has no closed-form solution. In order to overcome this problem, the vector function $\mathbf{g}(\mathbf{s}, \mathbf{n})$ is approximated by a vector Taylor series. The accuracy of such an approximation will depend on the number of terms included in the Taylor series expansion. Here, we will only consider VTS of order zero and one. We will need the gradient of $\mathbf{g}$ with respect to $\mathbf{s}$. This is a diagonal matrix that depends on the values of $\mathbf{s}$ and $\mathbf{n}$:

$$\mathbf{G}_{\mathbf{s},\mathbf{n}} = \operatorname{diag}\left(\frac{-\exp(\mathbf{n} - \mathbf{s})}{\mathbf{1} + \exp(\mathbf{n} - \mathbf{s})}\right). \tag{3.44}$$

Approximations of order zero and one are then given by

$$\text{Order 0:} \quad \tilde{\mathbf{g}}(\mathbf{s}, \mathbf{n}) \approx \mathbf{g}(\mathbf{s}_0, \mathbf{n}) \tag{3.45}$$

$$\text{Order 1:} \quad \tilde{\mathbf{g}}(\mathbf{s}, \mathbf{n}) \approx \mathbf{g}(\mathbf{s}_0, \mathbf{n}) + \mathbf{G}_{\mathbf{s}_0,\mathbf{n}}(\mathbf{s} - \mathbf{s}_0), \tag{3.46}$$

where $\mathbf{s}_0$ is the operating point of the Taylor series. Thus, when using a VTS approximation of order zero, the expression for a noisy speech vector is given by

$$\mathbf{y} = \mathbf{s} + \mathbf{g}(\mathbf{s}_0, \mathbf{n}). \tag{3.47}$$

Using this approximation in (3.43) yields

$$\boldsymbol{\mu}_{\mathbf{y}} = \boldsymbol{\mu}_{\mathbf{s}} + \mathbf{g}(\mathbf{s}_0, \mathbf{n}). \tag{3.48}$$

The covariance matrix can be calculated in a similar way by noting that

$$\begin{aligned}\boldsymbol{\Sigma}_{\mathbf{y}} &= \mathrm{E}[(\mathbf{y}-\boldsymbol{\mu}_{\mathbf{y}})(\mathbf{y}-\boldsymbol{\mu}_{\mathbf{y}})^T] \\ &= \mathrm{E}[\mathbf{y}\mathbf{y}^T] - \boldsymbol{\mu}_{\mathbf{y}}\boldsymbol{\mu}_{\mathbf{y}}^T.\end{aligned} \tag{3.49}$$

When using VTS of order zero, the covariance matrix for noisy speech will be equal to the covariance matrix of clean speech:

$$\boldsymbol{\Sigma}_{\mathbf{y}} = \boldsymbol{\Sigma}_{\mathbf{s}}. \tag{3.50}$$

When using first order VTS, the expression for a noisy speech vector is given by

$$\mathbf{y} = \mathbf{s} + \mathbf{g}(\mathbf{s}_0, \mathbf{n}) + \mathbf{G}_{\mathbf{s}_0,\mathbf{n}}(\mathbf{s}-\mathbf{s}_0). \tag{3.51}$$

Using this approximation in (3.43) yields a mean of

$$\begin{aligned}\boldsymbol{\mu}_{\mathbf{y}} &= \boldsymbol{\mu}_{\mathbf{s}} + \mathbf{g}(\mathbf{s}_0, \mathbf{n}) + \mathbf{G}_{\mathbf{s}_0,\mathbf{n}}(\boldsymbol{\mu}_{\mathbf{s}}-\mathbf{s}_0) \\ &= (\mathbf{I}+\mathbf{G}_{\mathbf{s}_0,\mathbf{n}})\boldsymbol{\mu}_{\mathbf{s}} + \mathbf{g}(\mathbf{s}_0, \mathbf{n}) - \mathbf{G}_{\mathbf{s}_0,\mathbf{n}}\mathbf{s}_0.\end{aligned} \tag{3.52}$$

The covariance matrix when using first order VTS is given by

$$\boldsymbol{\Sigma}_{\mathbf{y}} = (\mathbf{I}+\mathbf{G}_{\mathbf{s}_0,\mathbf{n}})\boldsymbol{\Sigma}_{\mathbf{s}}(\mathbf{I}+\mathbf{G}_{\mathbf{s}_0,\mathbf{n}})^T. \tag{3.53}$$

So far we have assumed the noise to be known. However, the noise can also be modeled as a vector of random variables, with a multivariate Gaussian distribution having mean $\boldsymbol{\mu}_{\mathbf{n}}$ and covariance matrix $\boldsymbol{\Sigma}_{\mathbf{n}}$ [51]. Thus, $\mathbf{n}$ must be handled in the same way as $\mathbf{s}$ in the Taylor series. Then, the approximation of order one becomes

$$\mathbf{y} = \mathbf{s} + \mathbf{g}(\mathbf{s}_0, \mathbf{n}_0) + \mathbf{G}_{\mathbf{s}_0,\mathbf{n}_0}(\mathbf{s}-\mathbf{s}_0) + \mathbf{H}_{\mathbf{s}_0,\mathbf{n}_0}(\mathbf{n}-\mathbf{n}_0), \tag{3.54}$$

where the gradient of $\mathbf{g}$ with respect to the noise is given by

$$\mathbf{H}_{\mathbf{s},\mathbf{n}} = \mathrm{diag}\left(\frac{\exp(\mathbf{n}-\mathbf{s})}{\mathbf{1}+\exp(\mathbf{n}-\mathbf{s})}\right). \tag{3.55}$$

The mean and covariance matrix of the noisy speech are given by

$$\boldsymbol{\mu}_{\mathbf{y}} = \boldsymbol{\mu}_{\mathbf{s}} + \mathbf{g}(\mathbf{s}_0, \mathbf{n}_0) + \mathbf{G}_{\mathbf{s}_0,\mathbf{n}_0}(\boldsymbol{\mu}_{\mathbf{s}}-\mathbf{s}_0) + \mathbf{H}_{\mathbf{s}_0,\mathbf{n}_0}(\boldsymbol{\mu}_{\mathbf{n}}-\mathbf{n}_0) \tag{3.56}$$

$$\boldsymbol{\Sigma}_{\mathbf{y}} = (\mathbf{I}+\mathbf{G}_{\mathbf{s}_0,\mathbf{n}_0})\boldsymbol{\Sigma}_{\mathbf{s}}(\mathbf{I}+\mathbf{G}_{\mathbf{s}_0,\mathbf{n}_0})^T + \mathbf{H}_{\mathbf{s}_0,\mathbf{n}_0}\boldsymbol{\Sigma}_{\mathbf{n}}\mathbf{H}_{\mathbf{s}_0,\mathbf{n}_0}^T. \tag{3.57}$$

We still have to choose a value for the operating point $(\mathbf{s}_0, \mathbf{n}_0)$. A common choice is the pair of means $(\boldsymbol{\mu}_{\mathbf{s}}, \boldsymbol{\mu}_{\mathbf{n}})$, which results in the following expressions for the mean and covariance matrix:

$$\boldsymbol{\mu}_{\mathbf{y}} = \boldsymbol{\mu}_{\mathbf{s}} + \mathbf{g}(\boldsymbol{\mu}_{\mathbf{s}}, \boldsymbol{\mu}_{\mathbf{n}}) \tag{3.58}$$

$$\boldsymbol{\Sigma}_{\mathbf{y}} = (\mathbf{I}+\mathbf{G}_{\boldsymbol{\mu}_{\mathbf{s}},\boldsymbol{\mu}_{\mathbf{n}}})\boldsymbol{\Sigma}_{\mathbf{s}}(\mathbf{I}+\mathbf{G}_{\boldsymbol{\mu}_{\mathbf{s}},\boldsymbol{\mu}_{\mathbf{n}}})^T + \mathbf{H}_{\boldsymbol{\mu}_{\mathbf{s}},\boldsymbol{\mu}_{\mathbf{n}}}\boldsymbol{\Sigma}_{\mathbf{n}}\mathbf{H}_{\boldsymbol{\mu}_{\mathbf{s}},\boldsymbol{\mu}_{\mathbf{n}}}^T. \tag{3.59}$$

As discussed in Section 1.1, the noise parameters $\boldsymbol{\mu}_\mathbf{n}$ and $\boldsymbol{\Sigma}_\mathbf{n}$ usually have to be estimated from the given test utterance. In Section 3.5 we will consider different methods for noise parameter estimation.

Finally, we note that in the case of VTS order zero, with the same choice of operating point as before, the noisy speech mean is the same as in (3.58). The noisy speech covariance matrix is, as shown in (3.50), equal to $\boldsymbol{\Sigma}_\mathbf{s}$. These are the same parameters as are obtained when using PMC and the log-add approximation.

3.3 Robust Feature Transformations

Given that our ASR system has been trained in clean conditions, the goal of a robust feature transformation is to find an estimate of the underlying clean speech feature vector sequence $\mathbf{s}_1, \ldots, \mathbf{s}_T$ from the observed noisy feature vector sequence $\mathbf{y}_1, \ldots, \mathbf{y}_T$, i.e.,

$$\hat{\mathbf{s}}_t = \mathcal{F}(\mathbf{y}_t) \quad t = 1, \ldots, T. \tag{3.60}$$

Several of the robust feature transforms that will be described in this section are based on the *minimum mean-square error* (MMSE) criterion. This means that the target is to minimize the *mean-square error* (MSE) for each frame t. More specifically, we are seeking a transformation (3.60) that results in

$$\hat{\mathbf{s}}_t^{\text{MMSE}} = \arg\min_{\hat{\mathbf{s}}_t} \mathrm{E}[\|\mathbf{s}_t - \hat{\mathbf{s}}_t\|^2]. \tag{3.61}$$

It can be shown (see e.g. [71, pp. 312-313]) that the MMSE estimate is given by the mean of the conditional density when observing $\mathbf{y}_t$:

$$\hat{\mathbf{s}}_t^{\text{MMSE}} = \mathrm{E}[\mathbf{s}_t|\mathbf{y}_t]. \tag{3.62}$$

This means that if we are able to estimate the probability distribution $p(\mathbf{s}_t|\mathbf{y}_t)$, we can also calculate the estimate $\hat{\mathbf{s}}_t^{\text{MMSE}}$.

We will now move on to describe four robust feature transformation techniques. These are called spectral subtraction, feature based vector Taylor series, model-based feature enhancement, and Algonquin. The three latter techniques are all based on the MMSE estimate.

3.3.1 Spectral Subtraction

Spectral subtraction [6] is a simple feature transformation that has been widely used. Let $S_{ss}(f)$, $S_{nn}(f)$, and $S_{yy}(f)$ denote the power spectral densities (PSDs) of clean speech, noise, and noisy speech respectively. If we assume that clean speech and noise are uncorrelated, we have

$$S_{yy}(f) = S_{ss}(f) + S_{nn}(f). \tag{3.63}$$

Note that compared to (3.3), this expression is simpler since the last term on the right side of (3.3) is not included in (3.63). This is because the PSD is based on the expectation, and the last term in (3.3) is zero in an expected sense.

Then, given estimates $\hat{S}_{yy}(f)$ and $\hat{S}_{nn}(f)$ of the PSDs for noisy speech and noise, we can find an estimate for $S_{ss}(f)$ as

$$\hat{S}_{ss}(f) = \hat{S}_{yy}(f) - \hat{S}_{nn}(f). \tag{3.64}$$

Estimates for $S_{ss}(f)$ are found for each frame of the input signal. This means that estimates for the noise PSD are also needed for each frame. Usually they are found by averaging signal frames where there is no speech activity. Thus, the noise PSD is assumed not to change when speech is present.

One problem with this technique is that one can get negative values for $\hat{S}_{ss}(f)$. To avoid this a lower limit of zero can be used. Another problem is that the result can contain “musical noise”, which is a result of noise residuals.

Note that, unlike the other feature transformations we will consider in this chapter, the spectral subtraction technique does not assume that any prior information about clean speech is available.

3.3.2 Feature-Based Vector Taylor Series

In this section the basic MMSE filtering approach using VTS [50, 51] will be described. Unlike spectral subtraction, feature based VTS assumes that some prior information about speech is available in the form of a GMM trained on clean speech data, i.e.,

$$p(\mathbf{s}) = \sum_{m=1}^{M} P(m)p(\mathbf{s}|m) = \sum_{m=1}^{M} w_m \mathcal{N}(\mathbf{s}; \boldsymbol{\mu}_{\mathbf{s},m}, \boldsymbol{\Sigma}_{\mathbf{s},m}). \tag{3.65}$$

We begin by noting that, when ignoring the error term $\mathbf{e}$ in (3.14), the

MMSE estimate can be written as

$$\begin{aligned}\hat{\mathbf{s}} &= E[\mathbf{s}|\mathbf{y}] \\ &= \int_{\mathcal{S}} \mathbf{s}\, p(\mathbf{s}|\mathbf{y}) d\mathbf{s} \\ &= \mathbf{y} - \int_{\mathcal{S}} (\log[\mathbf{1} + \exp(\mathbf{n} - \mathbf{s})]) p(\mathbf{s}|\mathbf{y}) d\mathbf{s} \\ &= \mathbf{y} - \int_{\mathcal{S}} \mathbf{g}(\mathbf{s}, \mathbf{n}) p(\mathbf{s}|\mathbf{y}) d\mathbf{s}\end{aligned} \tag{3.66}$$

where $\mathcal{S}$ denotes the feature space and $\mathbf{g}(\mathbf{s}, \mathbf{n})$ is defined as in (3.41). Since the integral in (3.66) cannot be solved analytically, the function $\mathbf{g}$ is approximated using a vector Taylor series expansion. In this section we will only consider the zeroth order approximation, i.e., the approximation given in (3.45). Since we have assumed a mixture distribution as a prior for speech, the posterior $p(\mathbf{s}|\mathbf{y})$ will also be a mixture distribution. For each mixture component, the prior speech mean $\boldsymbol{\mu}_{\mathbf{s},m}$ is used as the operating point of the Taylor series. Thus, we get

$$\begin{aligned}\hat{\mathbf{s}} &= \mathbf{y} - \int_{\mathcal{S}} \sum_{m=1}^{M} P(m|\mathbf{y}) \mathbf{g}(\mathbf{s}, \mathbf{n}) p(\mathbf{s}|m, \mathbf{y}) d\mathbf{s} \\ &= \mathbf{y} - \sum_{m=1}^{M} P(m|\mathbf{y}) \int_{\mathcal{S}} \mathbf{g}(\mathbf{s}, \mathbf{n}) p(\mathbf{s}|m, \mathbf{y}) d\mathbf{s} \\ &\approx \mathbf{y} - \sum_{m=1}^{M} P(m|\mathbf{y}) \int_{\mathcal{S}} \mathbf{g}(\boldsymbol{\mu}_{\mathbf{s},m}, \mathbf{n}) p(\mathbf{s}|m, \mathbf{y}) d\mathbf{s} \\ &= \mathbf{y} - \sum_{m=1}^{M} P(m|\mathbf{y}) \mathbf{g}(\boldsymbol{\mu}_{\mathbf{s},m}, \mathbf{n}).\end{aligned} \tag{3.67}$$

The noise vector $\mathbf{n}$ is usually modeled as random vector with distribution $\mathcal{N}(\boldsymbol{\mu}_{\mathbf{n}}, \boldsymbol{\Sigma}_{\mathbf{n}})$. Then, using the noise mean as the VTS operating point, the MMSE estimate is given by

$$\hat{\mathbf{s}} = \mathbf{y} - \sum_{m=1}^{M} P(m|\mathbf{y}) \mathbf{g}(\boldsymbol{\mu}_{\mathbf{s},m}, \boldsymbol{\mu}_{\mathbf{n}}). \tag{3.68}$$

The posterior probability of each mixture component $P(m|\mathbf{y})$ can be found as

$$P(m|\mathbf{y}) = \frac{p(\mathbf{y}|m)P(m)}{p(\mathbf{y})} = \frac{p(\mathbf{y}|m)P(m)}{\sum_{j=1}^{M} p(\mathbf{y}|j)P(j)}. \tag{3.69}$$

Thus, we need to find the distribution $p(\mathbf{y}|m)$ given $p(\mathbf{s}|m)$ and $p(\mathbf{n})$. This is a simple case of model compensation, which was described in Section 3.2.

3.3.3 Model-Based Feature Enhancement

Model-based feature enhancement (MBFE) was originally proposed by Couvreur and Van hamme in [9], and further work was done by Stouten in [66] and Stouten *et al.* in [67, 68, 69]. The method operates in the MFCC domain.

The first step of the MBFE method is to find an HMM for noisy speech with parameters $\boldsymbol{\Lambda}^{\mathbf{y}}$. In order to do this, MBFE takes advantage of a prior HMM for speech with parameters $\boldsymbol{\Lambda}^{\mathbf{s}}$ and a prior HMM for noise with parameters $\boldsymbol{\Lambda}^{\mathbf{n}}$. If the HMM states in the priors have GMM probability density functions (pdfs), each state is decomposed into several parallel states with a single component pdf in each state. The parameters of the noisy speech model $\boldsymbol{\Lambda}^{\mathbf{y}}$ can then be found using different methods, e.g. PMC as described in Section 3.2.1 or VTS as described in Section 3.2.2. In this section we will follow the approach taken in [66], where VTS is used. Each state i in the decomposed clean speech HMM is then combined with a state j in the noise HMM using the cepstral equivalents of (3.58) and (3.59):

$$\boldsymbol{\mu}_{\mathbf{y}}^{(i,j)} = \boldsymbol{\mu}_{\mathbf{s},i} + \mathbf{C}\log\left[\mathbf{1} + \exp(\mathbf{C}^{-1}(\boldsymbol{\mu}_{\mathbf{n},j} - \boldsymbol{\mu}_{\mathbf{s},i}))\right] \tag{3.70}$$

$$\boldsymbol{\Sigma}_{\mathbf{y}}^{(i,j)} = \mathbf{A}_{(i,j)}\Sigma_{\mathbf{s},i}\mathbf{A}_{(i,j)}^{T} + \mathbf{B}_{(i,j)}\Sigma_{\mathbf{n},j}\mathbf{B}_{(i,j)}^{T} \tag{3.71}$$

where, in order to simplify notation, we have defined

$$\mathbf{A}_{(i,j)} = \mathbf{C}\ \operatorname{diag}\left(\frac{\mathbf{1}}{\mathbf{1} + \exp(\mathbf{C}^{-1}(\boldsymbol{\mu}_{\mathbf{n},j} - \boldsymbol{\mu}_{\mathbf{s},i}))}\right)\mathbf{C}^{-1} \tag{3.72}$$

$$\mathbf{B}_{(i,j)} = \mathbf{I} - \mathbf{A}_{(i,j)}. \tag{3.73}$$

For a given pair of states (i,j), the matrix $\mathbf{A}$ is the cepstral equivalent of the matrix $\mathbf{I} + \mathbf{G}_{\boldsymbol{\mu}_{\mathbf{s}},\boldsymbol{\mu}_{\mathbf{n}}}$, where $\mathbf{G}$ was defined in (3.44). Similarly, the matrix $\mathbf{B}$ is the cepstral equivalent of $\mathbf{H}_{\boldsymbol{\mu}_{\mathbf{s}},\boldsymbol{\mu}_{\mathbf{n}}}$, with $\mathbf{H}$ defined in (3.55). Note that the number of states in the resulting noisy speech HMM will be the number of states in the clean speech HMM times the number of states in the noise HMM.

Having obtained the approximate HMM for noisy speech, the forward-backward algorithm is used to obtain probabilities for being in certain states at certain times. More specifically, we calculate the probability of being in state (i, j) at frame t, i.e.,

$$\gamma_t^{(i,j)} = P(q_t^s = i, q_t^n = j|\mathbf{Y}) \tag{3.74}$$

where q_t^s and q_t^n denote the speech state and noise state at frame t respectively, and $\mathbf{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_T\}$ denotes the observed feature vectors.

In order to find a clean speech estimate, we note that the MMSE estimate can be written as

$$\hat{\mathbf{s}}_t = \sum_{i=1}^{M^s} \sum_{j=1}^{M^n} P(q_t^s = i, q_t^n = j|\mathbf{Y}) \, \mathrm{E}[\mathbf{s}_t | q_t^s = i, q_t^n = j, \mathbf{Y}] \tag{3.75}$$

where M^s denotes the number of states in the clean speech HMM and M^n denotes the number of states in the noisy speech HMM. Thus, what remains is to find state-conditional MMSE estimates $\mathrm{E}[\mathbf{s}_t | q_t^s = i, q_t^n = j, \mathbf{Y}]$. For a state (i, j) in the noisy speech HMM, the state-conditional estimate can be approximated by (see [66] for derivation)

$$\hat{\mathbf{s}}_t^{(i,j)} = \boldsymbol{\mu}_{\mathbf{s},i} + \boldsymbol{\Sigma}_{\mathbf{s},i} \mathbf{A}_{(i,j)}^T (\boldsymbol{\Sigma}_{\mathbf{y}}^{(i,j)})^{-1} (\mathbf{y}_t - \boldsymbol{\mu}_{\mathbf{y}}^{(i,j)}). \tag{3.76}$$

Finally, the resulting MMSE estimate is calculated as

$$\hat{\mathbf{s}}_t = \sum_{i=1}^{M^s} \sum_{j=1}^{M^n} \gamma_t^{(i,j)} \hat{\mathbf{s}}_t^{(i,j)}. \tag{3.77}$$

Although the original MBFE approach is based on the HMM, it was shown in [66] that good results can also be obtained by simply using a GMM. Later in this thesis we will use MBFE, and our implementation is based on the GMM. In addition, we will also limit the noise model to only consist of a single mixture component. This means that we can simplify the above formulation. The expressions in (3.70) and (3.71) can be simplified to

$$\boldsymbol{\mu}_{\mathbf{y},i} = \boldsymbol{\mu}_{\mathbf{s},i} + \mathbf{C} \log \left[\mathbf{1} + \exp(\mathbf{C}^{-1}(\boldsymbol{\mu}_{\mathbf{n}} - \boldsymbol{\mu}_{\mathbf{s},i})) \right] \tag{3.78}$$

$$\boldsymbol{\Sigma}_{\mathbf{y},i} = \mathbf{A}_i \Sigma_{\mathbf{s},i} \mathbf{A}_i^T + \mathbf{B}_i \Sigma_{\mathbf{n}} \mathbf{B}_i^T \tag{3.79}$$

where

$$\mathbf{A}_i = \mathbf{C} \ \mathrm{diag} \left(\frac{\mathbf{1}}{\mathbf{1} + \exp(\mathbf{C}^{-1}(\boldsymbol{\mu}_{\mathbf{n}} - \boldsymbol{\mu}_{\mathbf{s},i}))} \right) \mathbf{C}^{-1} \tag{3.80}$$

$$\mathbf{B}_i = \mathbf{I} - \mathbf{A}_i. \tag{3.81}$$

A component-conditional estimate for frame t can then be found as

$$\hat{\mathbf{s}}_t^i = \boldsymbol{\mu}_{\mathbf{s},i} + \boldsymbol{\Sigma}_{\mathbf{s},i}\mathbf{A}_i^T \boldsymbol{\Sigma}_{\mathbf{y},i}^{-1}(\mathbf{y}_t - \boldsymbol{\mu}_{\mathbf{y},i}). \tag{3.82}$$

The final MMSE estimate is then calculated as

$$\hat{\mathbf{s}}_t = \sum_{i=1}^{M} \gamma_t^i \hat{\mathbf{s}}_t^i \tag{3.83}$$

where M is the number of mixture components in the clean speech GMM and

$$\gamma_t^i = \frac{w_i \mathcal{N}(\mathbf{y}_t; \boldsymbol{\mu}_{\mathbf{y},i}, \boldsymbol{\Sigma}_{\mathbf{y},i})}{\sum_{j=1}^{M} w_j \mathcal{N}(\mathbf{y}_t; \boldsymbol{\mu}_{\mathbf{y},j}, \boldsymbol{\Sigma}_{\mathbf{y},j})}. \tag{3.84}$$

3.3.4 Algonquin

Algonquin [44, 21, 22] is a feature enhancement method that can operate either in the log-spectral domain or the cepstral domain. We will first consider the version that operates in the log-spectral domain before we consider the changes needed in order to work in the cepstral domain.

Algonquin uses a prior GMM for speech $p(\mathbf{s})$ and a prior GMM for noise $p(\mathbf{n})$ in order to approximate the posterior $p(\mathbf{s}|\mathbf{y})$, which is also modeled as a GMM. A variational algorithm is used to obtain this approximate posterior, which is denoted by $q_{\mathbf{y}}(\mathbf{s})$.

The relationship between clean speech, noise and noisy speech is modeled by a normal distribution, where variations due to the error term $\mathbf{e}$ in (3.14) are taken into account by the covariance matrix. Given $\mathbf{s}$ and $\mathbf{n}$, the distribution of $\mathbf{y}$ is given by

$$p(\mathbf{y}|\mathbf{s},\mathbf{n}) = \mathcal{N}(\mathbf{y}; \mathbf{s} + \log(1 + \exp(\mathbf{n} - \mathbf{s})), \boldsymbol{\Psi}) \tag{3.85}$$

where $\boldsymbol{\Psi}$ is the covariance matrix.

The joint distribution between $\mathbf{y}$, $\mathbf{s}$, $\mathbf{n}$, speech GMM component m^s and noise GMM component m^n is now given by

$$\begin{aligned} p(\mathbf{y},\mathbf{s},\mathbf{n},m^s,m^n) &= p(\mathbf{y}|\mathbf{s},\mathbf{n})p(m^s)p(\mathbf{s}|m^s)p(m^n)p(\mathbf{n}|m^n) \\ &= \mathcal{N}(\mathbf{y}; \mathbf{f}(\mathbf{s},\mathbf{n}), \boldsymbol{\Psi}) w_{m^s}^s \mathcal{N}(\mathbf{s}; \boldsymbol{\mu}_{\mathbf{s},m^s}, \boldsymbol{\Sigma}_{\mathbf{s},m^s}) \\ &\quad \cdot w_{m^n}^n \mathcal{N}(\mathbf{n}; \boldsymbol{\mu}_{\mathbf{n},m^n}, \boldsymbol{\Sigma}_{\mathbf{n},m^n}) \end{aligned} \tag{3.86}$$

where $w_{m^s}^s$ and $w_{m^n}^n$ denote the mixture component weights of speech component m^s and noise component m^n respectively. In addition, we have defined

$$\mathbf{f}(\mathbf{s},\mathbf{n}) = \mathbf{s} + \log[1 + \exp(\mathbf{n} - \mathbf{s})]. \tag{3.87}$$

For notational convenience we will now define the joint vector of clean speech and noise

$$\mathbf{z} = \begin{bmatrix} \mathbf{s} \\ \mathbf{n} \end{bmatrix}. \tag{3.88}$$

For each pair of prior speech and noise components (m^s, m^n), the joint prior $p(\mathbf{z}) = p(\mathbf{s})p(\mathbf{n})$ will have a component m with the following parameters

$$\boldsymbol{\mu}_m = \begin{bmatrix} \boldsymbol{\mu}_{\mathbf{s},m^s} \\ \boldsymbol{\mu}_{\mathbf{n},m^n} \end{bmatrix} \tag{3.89}$$

$$\boldsymbol{\Sigma}_m = \begin{bmatrix} \boldsymbol{\Sigma}_{\mathbf{s},m^s} & 0 \\ 0 & \boldsymbol{\Sigma}_{\mathbf{n},m^n} \end{bmatrix}. \tag{3.90}$$

Using the joint prior, we can write (3.86) as

$$p(\mathbf{y}, \mathbf{z}, m) = p(\mathbf{y}|\mathbf{z})p(m)p(\mathbf{z}|m). \tag{3.91}$$

In order to obtain tractable calculations, Algonquin proceeds by linearizing (3.87) using first order VTS as

$$\tilde{\mathbf{f}}(\mathbf{z}) = \mathbf{f}(\mathbf{z}_0) + \mathbf{F}(\mathbf{z} - \mathbf{z}_0) \tag{3.92}$$

where $\mathbf{z}_0$ is the operating point of the Taylor series, and $\mathbf{F}$ is the gradient matrix of the function $\mathbf{f}$. Denoting the dimension of a feature vector by D, the matrix $\mathbf{F}$ is $D \times 2D$, and it is given by

$$\mathbf{F} = [\mathbf{I} + \mathbf{G}_{\mathbf{z}_0}; \mathbf{H}_{\mathbf{z}_0}] \tag{3.93}$$

where $\mathbf{G}_{\mathbf{z}_0}$ and $\mathbf{H}_{\mathbf{z}_0}$ are as defined in (3.44) and (3.55) respectively. Note that the VTS operating point $\mathbf{z}_0$ corresponds to the pair $(\mathbf{s}_0, \mathbf{n}_0)$ which was used in the $\mathbf{G}$ and $\mathbf{H}$ matrices in Section 3.2.2. Using this approximation, we can rewrite (3.86) as

$$p(\mathbf{y}, \mathbf{z}, m) \approx \mathcal{N}(\mathbf{y}; \mathbf{f}(\mathbf{z}_0) + \mathbf{F}(\mathbf{z} - \mathbf{z}_0), \boldsymbol{\Psi}) w_m \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m). \tag{3.94}$$

Having obtained this linearized distribution, the goal is to use variational inference to find the parameters of the variational posterior $q_{\mathbf{y}}(\mathbf{z})$, which is defined to be a GMM. This GMM is given by

$$q_{\mathbf{y}}(\mathbf{z}) = \sum_m \rho_m \mathcal{N}(\mathbf{z}; \boldsymbol{\eta}_m, \boldsymbol{\Phi}_m). \tag{3.95}$$

where ρ_m, $\boldsymbol{\eta}_m$, and $\boldsymbol{\Phi}_m$ respectively denote the weight, mean, and covariance matrix of component m. The parameters of $q_{\mathbf{y}}(\mathbf{z}|m)$ are found by minimizing the Kullback-Leibler divergence between the true posterior $p(\mathbf{z}|\mathbf{y})$ and the variational posterior $q_{\mathbf{y}}(\mathbf{z})$, given by

$$\mathcal{K} = \sum_m \int_{\mathbf{z}} q_{\mathbf{y}}(\mathbf{z}, m) \log \frac{q_{\mathbf{y}}(\mathbf{z}, m)}{p(\mathbf{z}, m|\mathbf{y})} d\mathbf{z}. \tag{3.96}$$

However, since we do not have an expression for the true posterior, we can exploit the fact that minimizing (3.96) is equivalent to maximizing

$$\begin{aligned}\mathcal{F} &= \log p(\mathbf{y}) - \mathcal{K} \\ &= \sum_m \int_{\mathbf{z}} q_{\mathbf{y}}(\mathbf{z}, m) \log \frac{p(\mathbf{y}, \mathbf{z}, m)}{q_{\mathbf{y}}(\mathbf{z}, m)} d\mathbf{z}.\end{aligned} \tag{3.97}$$

Maximizing (3.97) by differentiating and equating to zero results in the following means and covariance matrices of $q_{\mathbf{y}}$ (see [44] for derivation)

$$\boldsymbol{\eta}_m = \boldsymbol{\Phi}_m[\boldsymbol{\Sigma}_m^{-1}\boldsymbol{\mu}_m + \mathbf{F}^T\boldsymbol{\Psi}^{-1}(\mathbf{y} - \mathbf{f} + \mathbf{F}\mathbf{z}_0)] \tag{3.98}$$

$$\boldsymbol{\Phi}_m = (\boldsymbol{\Sigma}_m^{-1} + \mathbf{F}^T\boldsymbol{\Psi}^{-1}\mathbf{F})^{-1}. \tag{3.99}$$

The mixture weights are given by

$$\rho_m = \frac{\gamma_m w_m}{\sum_i \gamma_i w_i}, \tag{3.100}$$

where

$$\begin{aligned}\gamma_m &= (2\pi)^{D/2}|\boldsymbol{\Sigma}_m|^{-1/2}|\boldsymbol{\Psi}|^{-1/2}|\boldsymbol{\Phi}_m|^{1/2} \\ &\cdot \exp\left[-\frac{1}{2}\Big(\boldsymbol{\mu}_m^T\boldsymbol{\Sigma}_m^{-1}\boldsymbol{\mu}_m + (\mathbf{y} - \mathbf{f} + \mathbf{F}\mathbf{z}_0)^T\boldsymbol{\Psi}^{-1}(\mathbf{y} - \mathbf{f} + \mathbf{F}\mathbf{z}_0) \\ &- \boldsymbol{\eta}_m^T\boldsymbol{\Phi}_m^{-1}\boldsymbol{\eta}_m\Big)\right].\end{aligned} \tag{3.101}$$

The final problem is to determine the operating point $\mathbf{z}_0$ of the vector Taylor series. Algonquin uses an iterative approach where each iteration uses the previously calculated posterior mean $\boldsymbol{\eta}_m$ as operating point. This will in most cases make $\boldsymbol{\eta}_m$ converge to a mode of the true posterior (see [44] for details). This leads to the following iterative algorithm.

1. For each mixture component m:

   (a) $\mathbf{z}_0^{(0)} \leftarrow \boldsymbol{\mu}_m$, $i \leftarrow 0$

   (b) Calculate $\mathbf{f} = \mathbf{f}(\mathbf{z}_0^{(i)})$ using (3.87)

   (c) Calculate $\mathbf{F} = \mathbf{F}(\mathbf{z}_0^{(i)})$ using (3.93)

(d) Calculate $\mathbf{\Phi}_m^{(i)}$ and $\boldsymbol{\eta}_m^{(i)}$ using (3.99) and (3.98)

(e) $\mathbf{z}_0^{(i+1)} \leftarrow \boldsymbol{\eta}_m^{(i)}$, $i \leftarrow i+1$

(f) If not converged, go to (b)

2. Calculate mixture weights ρ_m

After having found the posterior parameters, the MMSE estimate of the clean speech feature vector is found as the upper half of

$$\begin{aligned}\hat{\mathbf{z}} &= \int \mathbf{z} p(\mathbf{z}|\mathbf{y}) d\mathbf{z} \\ &\approx \int \mathbf{z} \sum_m q_{\mathbf{y}}(m) q_{\mathbf{y}}(\mathbf{z}|m) d\mathbf{z} \\ &= \sum_m \rho_m \boldsymbol{\eta}_m.\end{aligned} \tag{3.102}$$

In order to make Algonquin work in the cepstral domain, the function $\mathbf{f}$ defined in (3.87) and the matrix $\mathbf{F}$ defined in (3.93) must be changed. The cepstral domain version of $\mathbf{f}$ is given by

$$\mathbf{f}(\mathbf{z}) = \mathbf{s} + \mathbf{C} \log \left[\mathbf{1} + \exp(\mathbf{C}^{-1}(\mathbf{n} - \mathbf{s})) \right]. \tag{3.103}$$

For the cepstral domain version of the gradient matrix we first define

$$\mathbf{D} = \mathbf{C}\ \operatorname{diag}\left(\frac{\mathbf{1}}{\mathbf{1} + \exp(\mathbf{C}^{-1}(\mathbf{n}_0 - \mathbf{s}_0))} \right) \mathbf{C}^{-1}. \tag{3.104}$$

Then, the new matrix $\mathbf{F}$ can be written as

$$\mathbf{F} = [\mathbf{D}; \mathbf{I} - \mathbf{D}]. \tag{3.105}$$

3.4 Robust Decision Rules for ASR

Recall from Section 2.1 that most speech recognizers are based on the plug-in MAP decision rule

$$\hat{W} = \arg\max_W p_{\hat{\mathbf{\Lambda}}}(\mathbf{Y}|W) P_{\hat{\mathbf{\Gamma}}}(W), \tag{3.106}$$

where $(\hat{\mathbf{\Lambda}}, \hat{\mathbf{\Gamma}})$ are estimates that have been found from training data. As mentioned in Section 2.1, this decision rule is suboptimal in practice. There are several reasons for this. First of all, by using model structures such as HMMs and N-grams to model the unknown distributions $p(\mathbf{Y}|W)$ and

$P(W)$, we are making approximations. In addition, the estimates $(\hat{\mathbf{\Lambda}}, \hat{\mathbf{\Gamma}})$ are affected by the choice of estimation method, and the amount of available training data. Finally, in most cases there is some degree of mismatch between training and testing conditions. In order to address these issues, several approaches using *robust decision rules* have been proposed. The following sections will describe two such approaches: the minimax rule [49] and the Bayesian predictive classification rule [35]. Both these approaches are aimed at compensating for the uncertainty of parameter estimates obtained from training data. Note that in the following, we will only consider uncertainty in the parameters of the acoustic models.

3.4.1 The Minimax Decision Rule

The minimax decision rule for ASR was first studied by Merhav and Lee in [49], where minimax classification was applied to isolated word speech recognition. Suppose that there is some degree of mismatch between training and testing conditions, and that we have estimated a set of parameters $\mathbf{\Lambda}_0$ from our training data. We assume that the true parameters lie in a neighborhood region around the estimated $\mathbf{\Lambda}_0$, i.e.,

$$\mathbf{\Lambda} \in \eta(\mathbf{\Lambda}_0) \tag{3.107}$$

where $\eta(\mathbf{\Lambda}_0)$ is referred to as the *mismatch neighborhood.* The goal of the minimax decision rule is to minimize the worst-case probability of error given that the parameters do not depart from the allowed neighborhood. The resulting decision rule is given by

$$\hat{W} = \arg\max_{W} \left[\max_{\mathbf{\Lambda} \in \eta(\mathbf{\Lambda}_0)} p_{\mathbf{\Lambda}}(\mathbf{Y}|W)P(W) \right]. \tag{3.108}$$

As we can see from (3.108), the minimax approach allows the parameters to obtain the values inside the mismatch neighborhood which result in the highest likelihood from the acoustic model. Given these parameter values, the classification result is obtained in the same way as in the plug-in MAP decision rule. Merhav and Lee also proposed a mismatch neighborhood for the means of cepstral coefficients [49]. For a given HMM state with a pre-trained mean vector $\boldsymbol{\mu}^*$, the neighborhood is defined as

$$\eta(\boldsymbol{\mu}) = \left\{ \boldsymbol{\mu} : |\mu_d - \mu_d^*| \leq C d^{-1} \rho^d, d = 1, 2, \ldots, D \right\} \tag{3.109}$$

where d denotes the cepstral coefficient element number, D denotes the dimension of the cepstral vector, and C and ρ are constants such that $C > 0$

and $0 \leq \rho < 1$. These constant have to be tuned. Alternative neighborhoods for minimax classification which require less tuning were presented in [2]. A Viterbi-based minimax search algorithm was proposed in [40], allowing minimax classification for continuous speech recognition.

3.4.2 Bayesian Predictive Classification

The Bayesian predictive classification (BPC) approach for speech recognition was first introduced by Huo *et al.* in [35]. As we have already discussed, the minimax decision rule minimizes the worst-case probability of error. This is a very conservative strategy. The objective of Bayesian predictive classification (BPC) is to minimize the overall recognition error rate, when marginalizing both with respect to the input observation and with respect to model parameters. The parameters of the acoustic models are viewed as random variables, distributed according to a prior distribution $p(\mathbf{\Lambda}|\boldsymbol{\phi})$, where $\boldsymbol{\phi}$ denotes the set of hyperparameters. More specifically, the BPC rule is obtained by defining an overall risk, which is the risk averaged over all admissible distorted data models. Minimizing the overall risk under a (0,1)-loss function yields the BPC rule, which is given by (see e.g. [34] for derivation)

$$\hat{W} = \arg\max_{W} \tilde{p}(\mathbf{Y}|W)P(W) \tag{3.110}$$

where

$$\tilde{p}(\mathbf{Y}|W) = \int p(\mathbf{Y}|\mathbf{\Lambda}, W)p(\mathbf{\Lambda}|\boldsymbol{\phi}, W)d\mathbf{\Lambda}. \tag{3.111}$$

The pdf in (3.111) is called the predictive density. Exact implementation of the BPC rule is not easy. Consequently, different approximations have been proposed. The approach in [35, 38] uses a Laplace approximation to calculate the predictive pdf and a quasi-Bayes [36] algorithm for computation of posterior pdfs. The resulting BPC rule is called the quasi-Bayes predictive classification (QBPC) rule. In [39] two alternative approaches are presented. One of these is based on the Viterbi algorithm, and the resulting rule is called Viterbi Bayesian predictive classification (VBPC). The other approach calculates the Bayesian predictive density of each Gaussian mixture component, which is then used as a compensated distribution. The resulting method is called Bayesian predictive density based model compensation (BP-MC). This method will now be described in more detail, as it is used in the approach that will be presented in Chapter 5.

BP-MC is a straightforward way of applying Bayesian prediction to HMM-based speech recognition. Instead of directly modifying the decision rule, the modification is applied to each state density of the acoustic models. The parameters of each Gaussian mixture component are assumed to be uncertain, and the Bayesian predictive density replaces the standard pdf in order to compensate for this. This results in modified parameters for each state and mixture component which are then plugged into the standard MAP decision rule in (2.10). Recall from Section 2.2 that when diagonal covariance matrices are used, the pdf of HMM state i is given by

$$\begin{aligned} p_i(\mathbf{y}|\boldsymbol{\theta}_i) &= \sum_{m=1}^{M} w_{im}\mathcal{N}(\mathbf{y};\boldsymbol{\mu}_{im},\boldsymbol{\Sigma}_{im}) \\ &= \sum_{m=1}^{M} w_{im}\prod_{d=1}^{D}\frac{1}{\sqrt{2\pi}\sigma_{imd}}\exp\left(-\frac{(y_d-\mu_{imd})^2}{2\sigma_{imd}^2}\right) \end{aligned} \tag{3.112}$$

where $\boldsymbol{\theta}_i = \{w_{im},\boldsymbol{\mu}_{im},\boldsymbol{\Sigma}_{im} : m = 1,\ldots,M\}$ are the parameters of this state. We will also use the notation $\boldsymbol{\theta}_{im} = \{\boldsymbol{\theta}_{imd} : d = 1,\ldots,D\}$ for the parameters of mixture component m in state i, where $\boldsymbol{\theta}_{imd} = \{\mu_{imd},\sigma_{imd}\}$. For each state and mixture component we assume that a prior distribution $p(\boldsymbol{\theta}_{im}|\boldsymbol{\phi}_{im})$ is available, where $\boldsymbol{\phi}_{im}$ denotes the hyperparameters. Then, we can calculate the predictive density for each state as

$$\begin{aligned} \tilde{p}_i(\mathbf{y}) &= \sum_{m=1}^{M} w_{im}\int p_{im}(\mathbf{y}|\boldsymbol{\theta}_{im})p(\boldsymbol{\theta}_{im}|\boldsymbol{\phi}_{im})d\boldsymbol{\theta}_{im} \\ &= \sum_{m=1}^{M} w_{im}\prod_{d=1}^{D}\int p_{imd}(y_d|\boldsymbol{\theta}_{imd})p(\boldsymbol{\theta}_{imd}|\boldsymbol{\phi}_{imd})d\boldsymbol{\theta}_{imd}. \end{aligned} \tag{3.113}$$

In [39] only the uncertainty in the mean is compensated. A uniform prior is defined based on the uncertainty neighborhood that was used for minimax in [49], i.e., the neighborhood in (3.109). For a given state, mixture component, and feature vector element, the prior is then given by

$$p(\mu_{imd}|\boldsymbol{\phi}_{imd}) = \begin{cases} \frac{1}{2Cd^{-1}\rho^d} & \text{for } \mu^*_{imd} - Cd^{-1}\rho^d \le \mu_{imd} \le \mu^*_{imd} + Cd^{-1}\rho^d \\ 0 & \text{otherwise.} \end{cases} \tag{3.114}$$

Let us define the predictive distribution for feature vector element d as

$$\tilde{p}_{imd}(y_d) = \int p_{imd}(y_d|\boldsymbol{\theta}_{imd})p(\boldsymbol{\theta}_{imd}|\boldsymbol{\phi}_{imd})d\boldsymbol{\theta}_{imd}. \tag{3.115}$$

Using the prior in (3.114), and defining the constant $U_d = Cd^{-1}\rho^d$, we get

$$\begin{aligned}\tilde{p}_{imd}(y_d) &= \frac{1}{\sqrt{2\pi}\sigma_{imd}} \frac{1}{2U_d} \int\limits_{\mu^*_{imd}-U_d}^{\mu^*_{imd}+U_d} e^{-(y_d-\mu_{imd})^2/2\sigma^2_{imd}} d\mu_{imd} \\ &= \frac{1}{2U_d}\left[\Phi\left(\frac{\mu^*_{imd}-y_d+U_d}{\sigma_{imd}}\right) - \Phi\left(\frac{\mu^*_{imd}-y_d-U_d}{\sigma_{imd}}\right)\right]\end{aligned} \tag{3.116}$$

where

$$\Phi(z) = \frac{1}{\sqrt{2\pi}} \int\limits_{-\infty}^{z} e^{-x^2/2} dx. \tag{3.117}$$

3.5 Noise Estimation

Most of the techniques that have been described in this chapter require a statistical model of the noise, and as described in Section 1.1 this model is usually estimated from noise data which is extracted from the utterance that is to be recognized. However, the problem of how to obtain noise data from a noisy speech utterance is non-trivial, and many different techniques have been proposed to solve this problem. In this section we will give an overview of different categories of such techniques and mention a few examples.

As we mentioned in Section 1.1, there are two fundamentally different categories of noise estimation methods. One is to try to extract noise parameters from the whole utterance by tracking the noise during speech activity, and the other is to perform voice activity detection (VAD) to identify speech-free regions that can be used for estimating noise parameters. The former is usually referred to as noise tracking.

Methods for noise tracking work in different domains. An example of an approach that works in the spectral domain is the minimum statistics technique [48]. This technique is based on the observation that the power of a noisy speech signal frequently decays to the power level of the disturbing noise. Thus, it is possible to estimate the noise power spectral density by tracking the minimum of the noisy signal power spectral density.

Deng *et al.* have proposed several different noise tracking algorithms working in the log-spectral and cepstral domains. In [13] a noise tracking algorithm working in the cepstral domain based on recursive expectation-maximization (EM) was presented. The recursive EM estimate was based on maximum likelihood (ML) estimation. This framework was later extended to maximum a posteriori (MAP) estimation in [14], by incorporating a Gaussian prior distribution for the noise. Another approach, based on incremental Bayes learning with prior evolution was presented in [12]. This algorithm recursively updates the mean and variance of the noise prior based on the approximate Gaussian posterior computed at the preceding time step. Two other approaches that are related to the ML and MAP approaches described above are an ML-based sequential noise estimation method proposed by Ding *et al.* in [16], and a MAP-based sequential noise estimation method proposed by Ding in [15].

Some noise estimation techniques try to find a single set of optimal noise model parameters for a given noisy utterance based on all the signal frames instead of tracking the noise from frame to frame. In the context of feature-based VTS, Moreno *et al.* [51] proposed an iterative EM approach that uses the noisy speech GMM to maximize the likelihood of the noisy observations. The algorithm seeks the noise parameters that, when combined with the clean speech GMM, result in the maximum value for the likelihood. In the context of model-based VTS, Kim *et al.* [43] proposed an approach in which the joint likelihood of the recognized word sequence and noise parameters is maximized using an iterative procedure. This procedure alternates between keeping noise parameters fixed while finding the optimal word sequence and keeping the word sequence fixed while finding the optimal noise parameters. This HMM-based procedure becomes more complicated than Moreno *et al.*'s GMM-based procedure since we need to estimate the likelihoods that observations were generated by given states and mixtures in the HMM.

Voice activity detection is the other approach for obtaining noise data from a noisy speech utterance. A large amount of techniques has been proposed to solve this problem. Here, we will only mention some of the more recent techniques that work in the discrete Fourier transform domain, and use the likelihood ratio test (LRT) for classifying frames as speech-free or not. A well-known method based on the LRT was proposed by Sohn *et al.* in [65]. This approach will be reviewed in Section 7.3, since it is used as a basis for comparison in Chapter 7. A similar approach, using an LRT based on multiple observations was presented by Ramirez *et al.* in [60]. Kim *et al.* redefined the LRT-based decision rule by employing a uniformly most powerful test in [42]. We will discuss the VAD approach to noise estimation more in Chapter 7, where we also propose a novel VAD method that works in the MFCC domain.

3.6 Summary and Discussion

This chapter started by describing how MFCCs are influenced by noise. Then, several well-known noise robustness algorithms were described. Finally, a brief overview of different methods for noise estimation was given.

Both the model-based techniques described in Section 3.2 and the feature-based techniques described in Section 3.3 have been shown to significantly improve speech recognition performance in the presence of noise. However, they do have shortcomings. First of all, neither the model-based nor feature-based techniques have an effective way of dealing with the error term $\mathbf{e}$ in (3.14). Whereas most methods simply ignore it, Algonquin takes it into account through the covariance matrix $\boldsymbol{\Psi}$ in (3.85). However, the findings of Droppo *et al.* in [18] indicate that the performance gain obtained by using this model is limited, since the so-called zero variance model performed better in their experiments. Another weakness of methods that work in the log-spectral or cepstral domain is that they have to make an approximation to the non-linear relationship between speech, noise, and noisy speech. The accuracy of this approximation is likely to influence recognition performance. Another aspect that also needs to be considered is that most of these methods rely on statistical models of speech and noise. Thus, another factor that influences performance is the quality of these models, which in turn depends on the choice of estimation method and the amount of available training data.

In the following chapters we will describe different methods that attempt to compensate for some of these weaknesses. Chapter 4 presents a comparative study that investigates the influence of different approximations to the non-linear relationship between speech, noise, and noisy speech. Then, Chapter 5 describes a method that compensates for uncertainty in parameter estimates of models for noisy speech. In Chapter 6 we investigate an alternative estimation method for probabilistic speech models used in MMSE feature enhancement, namely Bayesian learning. Finally, in Chapter 7 we describe a method for voice activity detection that is used for extracting noise-only frames from noisy utterances. These are in turn used as training data in order to improve the noise model for MMSE feature enhancement.

Another issue that needs consideration when working with noise robustness is the compensation of delta and acceleration coefficients. Different approaches deal with this issue in different ways, and some techniques simply avoid it by using static coefficients only.

For model-based methods it is quite important to have a delta compensation scheme since most recognizers use delta coefficients. For PMC we have already described a delta compensation scheme in Section 3.2.1. This technique is also used in Chapter 5. For more on compensation of acceleration coefficients with PMC, see e.g. [25]. Methods for delta and acceleration compensation in the context of model-based VTS can be found in [1, 46]. Note that in the comparative study on model compensation in Chapter 4, no delta or acceleration coefficients are used during recognition. This is done because the aim of this study is to compare approximation accuracy without any "disturbing" factors.

Feature-based methods can achieve good performance by using only static coefficients in the front-end, and then calculate delta and acceleration coefficients based on the estimated clean speech feature vectors containing the static coefficients. Speech recognition can then be performed with delta and acceleration coefficients in the back-end. We have therefore chosen to use static coefficients only when performing feature enhancement in Chapter 6 and Chapter 7. In some of the experiments with MBFE in [66], delta parameters were used in the front-end, and found to improve performance. However, when estimating the noise model online on Aurora2, the inclusion of delta parameters degraded the performance instead of improving it. This was because the online estimates of the delta parameters for the noise were not accurate enough to improve the performance.

Chapter 4

A Comparative Study of Approximations for Model Compensation

In this chapter we will consider different ways of approximating the non-linear relationship between the statistical parameters of speech, noise, and noisy speech. In addition to the well-known log-normal approximation used by PMC, we will investigate two alternative approximations. One of these was proposed by Raut *et al.* in [61] and is based on Lagrange polynomials. The other method is a general technique for evaluation of the mean and variance of power sums with log-normal components, proposed by Schwartz and Yeh in [64]. We will show how this method can be used for model compensation. The study presented in this chapter will focus on compensation of the mean. Parts of this work were also presented in [56].

4.1 Different Approximations of the Noisy Speech Mean

Recall from Chapter 2 that if we ignore the error term in (3.14), the relationship between speech, noise and noisy speech in the log-spectral domain is given by

$$\mathbf{y} = \mathbf{s} + \log\left(\mathbf{1} + \exp(\mathbf{n} - \mathbf{s})\right), \tag{4.1}$$

which yields a noisy speech mean given by

$$\boldsymbol{\mu}_{\mathbf{y}} = \boldsymbol{\mu}_{\mathbf{s}} + \mathrm{E}[\log\left(\mathbf{1} + \exp(\mathbf{n} - \mathbf{s})\right)]. \tag{4.2}$$

As we have seen in Chapter 3 the problem is that there is no closed form solution for the expectation on the right-hand side of (4.2). In Section 3.2.1 we saw that the PMC log-normal approximation takes the approach of transforming the model parameters back to the linear spectral domain, while we saw in Section 3.2.2 that VTS found an approximation directly in the log-spectral domain. Both the alternative approaches presented here also work directly in the log-spectral domain.

Since we will assume that the ASR system is working with mixtures of Gaussian distributions in the cepstral domain, the distributions of speech and noise mixture components in the log-spectral domain are also Gaussian. Consequently, the corresponding variables in the linear spectral domain are log-normally distributed. Since we have to end up with Gaussians in the log-spectral domain after compensation, we will be working under the assumption that the sum of two log-normally distributed random variables is also approximately log-normally distributed.

Further, we will assume that the feature vector elements are independent in the log-spectral domain. This means that we do not take into account the correlations that exist between feature vector elements in the log-spectral domain. Due to the use of overlapping filters in the mel-scale filter bank, this is incorrect. However, it is quite common to make this assumption when working with noise robust ASR. The advantage is that we can work with scalars instead of vectors and matrices.

4.1.1 Schwartz-Yeh Approximation

In [64] Schwartz and Yeh proposed a method for calculating the mean and variance of a sum of log-normally distributed random variables.

We will consider a single element of the log-spectral domain feature vector and denote speech and noise by s and n respectively, and their corresponding means and variances by μ_s, μ_n, σ_s^2 and σ_n^2. Define w to be a Gaussian random variable given by

$$w = n - s. \tag{4.3}$$

This yields the following statistical properties of w.

$$\mathrm{E}[w] = \mu_w = \mu_n - \mu_s \tag{4.4}$$

$$\mathrm{Var}[w] = \sigma_w^2 = \sigma_n^2 + \sigma_s^2 \tag{4.5}$$

Now, the scalar version of (4.2) can be written as

$$\mu_y = \mu_s + \mathrm{E}\left[\log(1 + e^w)\right]. \tag{4.6}$$

The second term can be written as

$$\mathrm{E}\left[\log(1+e^{w})\right] = \int_{-\infty}^{\infty} [\log(1+e^{w})]p(w)dw, \tag{4.7}$$

where $p(w)$ denotes the normal density of the random variable w. The function $\log(1+z)$ can be expanded in the following power series

$$\log(1+z) = \sum_{j=1}^{\infty} C_j z^j, \quad C_j = \frac{(-1)^{j+1}}{j}, \tag{4.8}$$

which is valid when $|z| < 1$. In order to use this power series, the integral in (4.7) must be split into two parts:

$$\mathrm{E}\left[\log(1+e^{w})\right] = \int_{-\infty}^{0} [\log(1+e^{w})]\, p(w)\, dw + \int_{0}^{\infty} [\log(1+e^{-w}) + w]\, p(w)\, dw. \tag{4.9}$$

Now, a result on the following form can be derived:

$$\begin{aligned}\mu_y = \mu_s &+ \frac{\sigma_w}{\sqrt{2\pi}} e^{-\mu_w^2/2\sigma_w^2} + \mu_w \Phi\left(\frac{\mu_w}{\sigma_w}\right) \\ &+ \sum_{k=1}^{\infty} C_k e^{k^2\sigma_w^2/2} \left[e^{k\mu_w} \Phi\left(\frac{-\mu_w - k\sigma_w^2}{\sigma_w}\right) + e^{-k\mu_w} \Phi\left(\frac{\mu_w - k\sigma_w^2}{\sigma_w}\right)\right].\end{aligned} \tag{4.10}$$

Here, the function $\Phi(z)$ is given by

$$\Phi(z) = \frac{1}{2} + \frac{1}{2}\operatorname{erf}\left(\frac{z}{\sqrt{2}}\right). \tag{4.11}$$

Thus, μ_y can be found without using numerical integration. However, the number of terms in the sum must be limited to a finite number. It was found in [64] that about 40 terms were required for the 4th significant digit. Increasingly accurate approximations can be obtained by including more terms, at the cost of higher computational complexity.

4.1.2 Lagrange Polynomial Approximation

In [61] Lagrange polynomials were used to approximate mean parameters for noisy speech. For this approach, w is again defined as in (4.3) and assumed to be normally distributed with mean and variance as given by (4.4) and (4.5). Then, a 2nd-order Lagrange interpolating polynomial $L_2(w)$ is used to approximate the function

$$g(w) = \log(1 + e^w). \tag{4.12}$$

The approximation is then given by

$$g(w) \approx L_2(w) = \sum_{k=0}^{2} g(w_k) \prod_{i=0, i \neq k}^{2} \frac{(w - w_i)}{(w_k - w_i)}. \tag{4.13}$$

For this approximation, three points must be chosen. The first point is placed at $w_0 = \mu_w$. The other two points should be selected such that $w_1 = w_0 - \alpha$ and $w_2 = w_0 + \alpha$. The value α should be chosen as a compromise between accuracy around the point w_0 and accuracy at points far away from w_0. Thus, the variance of w should be taken into account. The resulting polynomial is on the form $g(w) = aw^2 + bw + c$, and the mean estimate is therefore given by

$$\mu_y = \mu_s + \mathrm{E}[g(w)] = \mu_s + a(\sigma_w^2 + \mu_w^2) + b\mu_w + c. \tag{4.14}$$

This method gives accurate estimates at low computational complexity.

4.2 Analysis of Different Approximations

To analyze the accuracy of different approaches, a simulation was run on a one-dimensional example, where the noise parameters were set to constants $\mu_n = 10$ and $\sigma_n^2 = 0.1$. The clean speech variance was set to $\sigma_s^2 = 6$, while the clean speech mean μ_s varied from 3 to 12. Monte-Carlo simulation was used to generate a basis for comparison. (Note that this is essentially the same setup as used for a similar experiment in [61], but different methods are compared.) The result can be seen in Figure 4.1. The range for μ_s has been chosen as the region where the methods differ the most. For values $\mu_s < 3$ and $\mu_s > 12$, the estimates become more and more similar for all methods. From the plot it can be seen that both the Schwartz-Yeh approximation and the Lagrange polynomial approximation give estimates close to the Monte-Carlo simulation. The log-normal approximation is very inaccurate around $\mu_s = 7$, where the speech and noise means in the linear domain have values close to each other. Note also that in the region $3 < \mu_s < 7$, the log-normal estimate of the noisy speech decreases as the mean of clean speech increases.

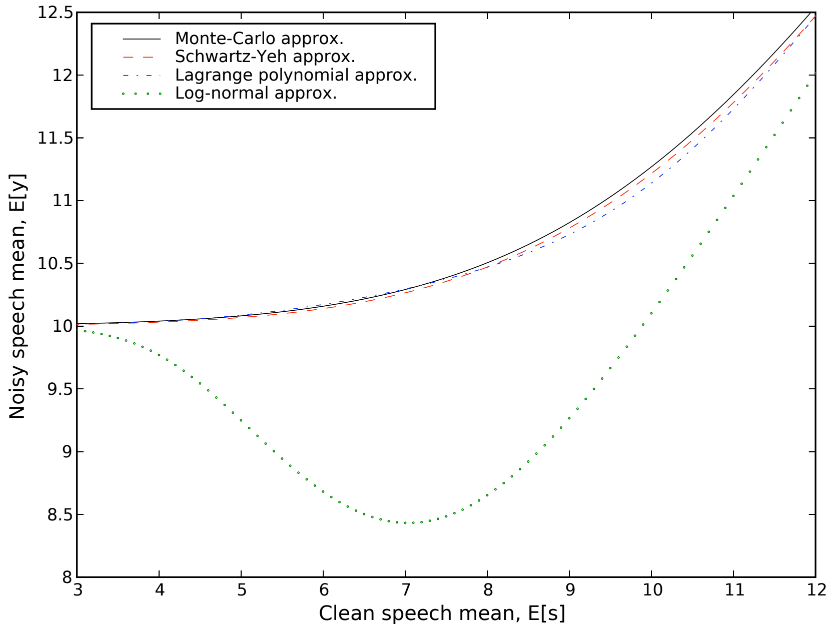

Figure 4.1: Corrupted speech mean estimates using different methods. Noise mean $\mu_n = 10$, noise variance $\sigma_n^2 = 0.1$, speech variance $\sigma_s^2 = 6$. Speech mean μ_s varying from 3 to 12.

4.3 Experiments and Results

The three methods described above were tested on the part of the Aurora2 database with exhibition noise. Exhibition noise was chosen since this type of noise was also used for experiments in [61]. The noise parameters were estimated from the 10 first frames of each test utterance. This could result in poor noise estimates in the case of non-stationarities in the noise. Therefore, we also added stationary white Gaussian noise at different SNRs and ran additional experiments. The noise power for a given SNR was set to the average power of the first 10 frames of all files in set A of Aurora2 with this SNR.

Recognition experiments were run using 13-dimensional MFCC features (i.e. only static parameters), and C_0 was used instead of log-energy. The rest of the setup was standard. For the Schwartz-Yeh approximation, 40 terms were used for the sum in (4.10). For the Lagrange polynomial approach,

Table 4.1: Results (word accuracy) for exhibition noise.

SNR	Baseline	Log-norm.	Schw.-Y.	Lagrange
20	69.70	93.49	93.43	93.46
15	52.08	90.10	90.19	90.19
10	34.68	85.07	85.41	85.47
5	20.46	73.43	73.62	73.65
0	12.03	51.62	51.87	51.90
-5	9.47	29.53	29.28	29.31
Avg. (0–20)	37.79	78.74	78.90	78.93

Table 4.2: Results (word accuracy) for white Gaussian noise.

SNR	Baseline	Log-norm.	Schw.-Y.	Lagrange
20	54.71	90.59	90.47	90.53
15	40.36	86.98	87.16	87.20
10	27.86	81.39	81.61	81.64
5	18.61	67.17	67.14	67.20
0	11.79	46.34	46.44	46.41
-5	8.55	23.42	23.67	23.63
Avg. (0–20)	30.67	74.49	74.56	74.60

the points w_1 and w_2 were placed two times σ_w away from the point w_0.

The results for experiments on exhibition noise are shown in Table 4.1, and results for experiments on additive white Gaussian noise are shown in Table 4.2. The baseline is the result when using clean speech models. For both noise types it can be observed that all the methods perform significantly better than the baseline. In addition, the results for the different approaches are very similar, and none of the methods are able to significantly outperform any of the others.[1]

4.4 Discussion

Figure 4.1 showed that the differences between the log-normal approximation and the two other approaches were quite significant when testing on

[1]Under a 5% confidence level, the only statistically significant differences are at 10dB. For 10dB exhibition noise both Schwartz-Yeh and Lagrange are significantly better than log-normal, whereas for 10dB white noise Lagrange is significantly better than log-normal.

Table 4.3: Comparison of estimates obtained using the log-normal and Schwartz-Yeh approximations. The table lists relative frequency for different relative deviations of estimated mean values for 25 files containing exhibition noise at SNRs of 20dB and -5dB.

$\frac{\lvert\mu_y^{\mathrm{LN}}-\mu_y^{\mathrm{SY}}\rvert}{\frac{1}{2}(\mu_y^{\mathrm{LN}}+\mu_y^{\mathrm{SY}})}$	Rel. freq., 20dB	Rel. freq., -5dB
0%-5%	98.65%	99.51%
5%-10%	0.85%	0.37%
10%-15%	0.42%	0.08%
15%-20%	0.06%	0.02%
20%-25%	0.01%	0.02%

Table 4.4: Comparison of estimates obtained using the log-normal and Lagrange polynomial approximations. The table lists relative frequency for different relative deviations of estimated mean values for 25 files containing exhibition noise at SNRs of 20dB and -5dB.

$\frac{\lvert\mu_y^{\mathrm{LN}}-\mu_y^{\mathrm{LP}}\rvert}{\frac{1}{2}(\mu_y^{\mathrm{LN}}+\mu_y^{\mathrm{LP}})}$	Rel. freq., 20dB	Rel. freq., -5dB
0%-5%	98.74%	99.53%
5%-10%	0.78%	0.35%
10%-15%	0.42%	0.08%
15%-20%	0.05%	0.02%
20%-25%	0.01%	0.02%

a one-dimensional example. However, after running speech recognition experiments, these differences are not reflected in the recognition results. In order to get an indication of how often methods of different approximation accuracy will result in significantly different estimates of the noisy speech mean, we calculated the relative deviations between all estimated means on 25 files with exhibition noise when using the different techniques. The estimates calculated using the log-normal approximation, μ_y^{LN}, were compared to:

1. Schwartz-Yeh estimates, μ_y^{SY}.
2. Lagrange polynomial estimates, μ_y^{LP}.

The comparisons were performed for different SNRs. The results for the two cases are shown in Table 4.3 and Table 4.4. In these tables the first column

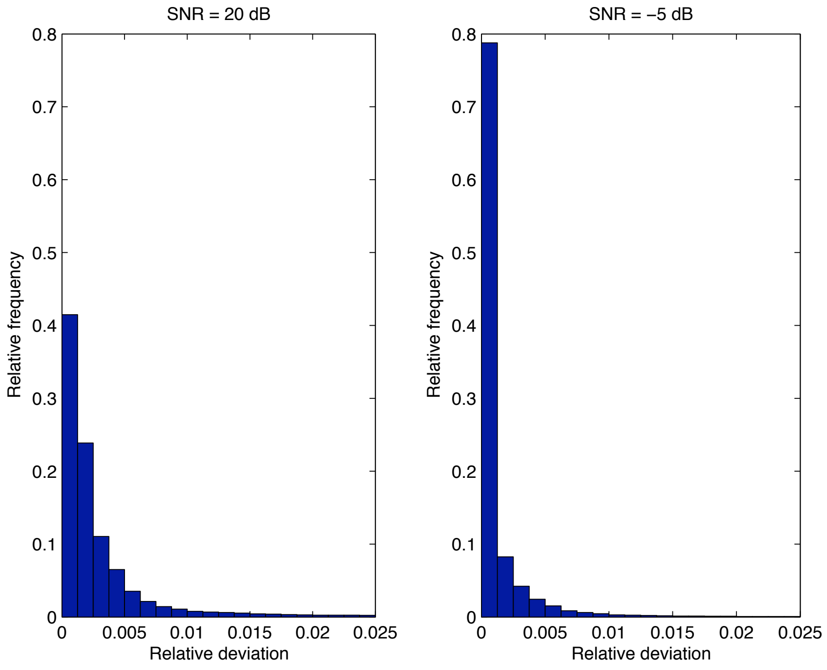

Figure 4.2: Distribution of relative deviations when comparing log-normal estimates versus Schwartz-Yeh estimates.

lists different ranges of relative deviations between the mean estimate μ_y^{LN} and the estimate using the other method (i.e. Schwartz-Yeh estimate in Table 4.3 and Lagrange polynomial estimate in Table 4.4). The following columns show the percentage of all mean estimates for which the relative deviation was in the specified range. The first row of Table 4.3 tells us that for an SNR of 20 dB the estimates made by the two methods differ by 5% or less in 98.65% of the cases. For an SNR of -5 dB, 99.51% of the estimates differ by 5% or less. Table 4.4 shows similar results when comparing the log-normal approximation to the Lagrange polynomial approximation.

In order to look closer at the distribution of these small differences, histograms were plotted for relative deviations up to 2.5%. These histograms are given in Figure 4.2 and Figure 4.3. From the plots it can be observed that in both cases the relative frequency is greatest for small relative deviations, and it decreases rapidly for increasing relative deviations. This tendency is more pronounced for an SNR of -5 dB than for 20 dB. Experiments on the other SNRs showed that the distribution varies gradually from

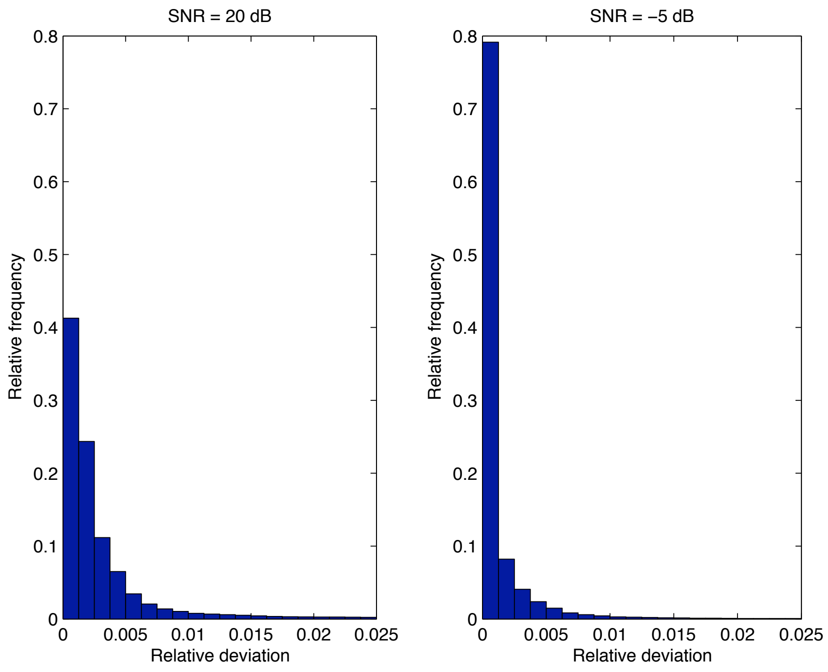

Figure 4.3: Distribution of relative deviations when comparing log-normal estimates versus Lagrange estimates.

that of 20 dB to that of -5 dB as the SNR decreases.

The conclusion from this investigation is that the methods rarely result in significantly different estimates, and this is probably part of the reason why there was so little difference of performance between the approaches.

Another possible reason is that there are weaknesses in the assumption that the resulting noisy distribution is Gaussian. It has been shown [25, 1] that for some combinations of speech and noise parameters, the true noisy distribution is bimodal and hence quite different from a Gaussian. If there are weaknesses in the underlying assumptions, it might not help to improve the accuracy of the approximation.

The results obtained here are quite different from what was reported in [61], but this could be due to differences in database, noise characteristics, and recognizer setup.

4.5 Conclusion

This chapter presented a comparison between three approximations to the non-linear relationship between noise, speech, and noisy speech applied for model compensation. The three approaches differ in theoretical accuracy and computational complexity. However, recognition experiments on a digit recognition task with artificially added noise resulted in very similar performance for the three methods.

Chapter 5

Combining Model Compensation and a Robust Decision Rule

In this chapter we will propose an approach that makes use of both BPC and PMC to achieve increased robustness towards noise. As described in Chapter 3, PMC is a method for estimating the parameters of speech corrupted by noise. In practice there will always be some uncertainty in these estimates. The BPC technique provides a way of compensating for uncertainty in parameter estimates. By combining these two approaches, we can obtain information about the mismatch situation and simultaneously account for uncertainty in this information. Parts of this work were also presented in [57] and [54].

5.1 Joint Bayesian Predictive Classification and Parallel Model Combination

Parallel model combination is a method that allows us to increase noise robustness by replacing the clean speech HMM parameters with estimates of the noisy speech HMM parameters. If we assume that no prior information about the noise is available, the noise parameters have to estimated from the current utterance. A common approach is to assume that the first few frames of the utterance are speech-free and estimate the noise parameters using these frames. This can result in uncertain estimates. In addition, as we have seen in Chapter 4, PMC only provides an approximation to the non-linear relationship between the parameters of clean speech, noise, and noisy speech. Due to these factors, we have to expect that there will be some degree of uncertainty in the noisy speech parameter estimates obtained by PMC.

As described in Section 3.4.2, BPC can be used to compensate for uncertainty in parameter estimates. To this end, we will use the Bayesian predictive density based model compensation approach to compensate for the uncertainty in the PMC parameter estimates. In this combined approach, which we will call BPC-PMC, only the uncertainty in the mean parameters will be considered. The BPC approach relies on information embedded in prior distributions, and the proposed approach takes advantage of knowledge about the mismatch situation obtained by PMC when specifying the priors. More specifically, the mean parameters estimated by PMC are used as prior means. The width of the prior is set based on the neighborhood given by (3.109).

In the Bayesian predictive density based model compensation approach described in Section 3.4.2 a uniform prior distribution was used, and the resulting density in (3.116) is clearly different from a Gaussian pdf. Using a Gaussian pdf as a prior instead of a uniform pdf results in a predictive distribution that is also Gaussian. Since this reduces the computational complexity, we chose a Gaussian prior for our experiments. Gaussian priors were also used in [37].

5.1.1 BPC with a Gaussian Prior

Let us first consider the case of a Gaussian prior with mean α_{imd} and variance ν_d^2 for HMM state i, mixture component m, and feature vector element d. (Note that the variance is independent of state and mixture component number.) Then, we can calculate (3.115) as

$$\begin{aligned}\tilde{p}_{imd}(y_d) &= \int \mathcal{N}(y_d; \mu_{imd}, \sigma_{imd}^2)\mathcal{N}(\mu_{imd}; \alpha_{imd}, \nu_d^2) d\mu_{imd} \\ &= \mathcal{N}(y_d; \alpha_{imd}, \sigma_{imd}^2 + \nu_d^2). \end{aligned} \tag{5.1}$$

Plugging this into (3.113) results in the following state density for HMM state i:

$$\tilde{p}_i(\mathbf{y}) = \sum_{m=1}^{M} w_{im} \prod_{d=1}^{D} \mathcal{N}(y_d; \alpha_{imd}, \sigma_{imd}^2 + \nu_d^2). \tag{5.2}$$

5.1.2 Setting the Hyperparameters

In the joint BPC-PMC approach, the prior mean is set to the noisy speech mean estimated by PMC, i.e.,

$$\alpha_{imd} = \mu^{y}_{imd} \tag{5.3}$$

where μ^{y}_{imd} is the result of (3.34). The variance is set as follows. In order to avoid having to deal with the placement of different cepstral coefficients and their delta and acceleration coefficients in the feature vector, we define $q(d)$ to be a function that maps feature vector element d to the order of the corresponding cepstral coefficient.[1] The prior variance is set to

$$\nu_d^2 = \begin{cases} \frac{1}{3}C^2 q(d)^{-2}\rho^{2q(d)} & \text{if } q(d) > 0 \\ k_d \bar{\sigma}_d & \text{if } q(d) = 0, \end{cases} \tag{5.4}$$

where $C > 0$ and $\rho \in (0, 1]$ are constants. The variance $\frac{1}{3}C^2 q(d)^{-2}\rho^{2q(d)}$ is a result of minimizing the Kullback-Leibler divergence between our Gaussian and a uniform distribution of width equal to $2Cq(d)^{-1}\rho^{q(d)}$ [37]. For the zeroth order cepstral coefficient, the prior variance has been set to a constant k_d multiplied by the global standard deviation $\bar{\sigma}_d$ from the clean speech training set for the corresponding coefficient (static, delta, or delta-delta).

5.1.3 Summary of the Procedure

We will now sum up the steps needed to implement the proposed procedure. The following steps are performed for each file to be recognized:

1. Estimate noise parameters from the first N frames
2. Using PMC as described in Section 3.2.1, find estimates $\{\mu^{y}_{imd}\}$ of the noisy speech mean for all mixture components m in all HMM states i
3. For each HMM state i, mixture component m, and feature vector element d, set the parameters of the predictive distribution as follows:
   (a) The mean is set to $\alpha_{imd} = \mu^{y}_{imd}$
   (b) The variance is set to $\sigma^2_{imd} + \nu_d^2$, with ν_d^2 given in (5.4)

[1]For example, a d that corresponds to the delta coefficient of cepstral coefficient 5 will be mapped to 5.

5.1.4 Experiments and Results

Experiments are performed on set A of the Aurora2 database. Recognition was run using static, delta and delta-delta parameters. PMC was used to estimate new means for static and delta parameters, while delta-delta parameters were left unchanged. In some experiments, PMC was also used to estimate new variances for static parameters. Noise parameters were estimated using the first 10 frames of each test utterance. For the joint BPC-PMC approach, BPC was used to compensate for uncertainty in all mean parameters. For the variance of the BPC prior, we used values of $C = 2$ and $\rho = 0.95$ for the constants in the first case of (5.4). The constants k_d were set to 1.0, 1.5, and 2.0 for the static, delta, and delta-delta parameters of the zeroth cepstral coefficient respectively.

The resulting recognition performance for different techniques, including baseline, and different noise types are given in Table 5.1. In this table, recognition using only PMC with updated mean values has been given the label PMC. When variances are also compensated for, the label PMC-var is used. The joint BPC and PMC technique is labeled BPC-PMC. Note also that a comparison of PMC and BPC-PMC with statistical significance testing can be found in Table F.1 in Appendix F.

Table 5.1 shows that model compensation by PMC significantly improves performance compared to the baseline, which is based on clean speech models. Moreover, the results show that BPC-PMC obtains improved performance compared to PMC for subway, car, and exhibition noise at low SNRs. In addition, the performance at high SNRs is comparable to PMC. However, for babble noise performance drops when using BPC-PMC compared to PMC. PMC-var only seems to work better than PMC in a few cases.

For SNRs of 10dB and higher, the PMC technique obtains word accuracies of at least 80% for all noise types. This indicates that good estimates of noisy model parameters are obtained. However, for low SNRs there is a higher degree of uncertainty in the compensated models. By using the noisy model estimates in prior specification for BPC, we are, to some extent, able to compensate for this uncertainty. This, however, does not apply for babble noise, where a drop in performance is observed. By taking a closer look at the recognition results, one can see that the problem that arises in this case is a big increase in the amount of insertion errors. This can be explained as follows. The noise model for babble noise can be viewed as a model for "background speech". When applying PMC with this noise model, the compensated HMMs will model a mixture of speech from different sources. Using these models for recognition will result in more insertion errors compared to the original models, since the compensated models are

Table 5.1: Word accuracies (%) on Aurora2 using PMC and BPC-PMC

Noise	SNR	Baseline	PMC	PMC-var	BPC-PMC
subway	20	96.90	97.88	95.67	97.39
	15	92.08	96.53	91.10	96.01
	10	73.87	92.66	82.13	92.20
	5	45.26	80.35	63.95	82.13
	0	20.94	46.67	42.59	56.00
	-5	10.72	21.31	20.79	25.88
babble	20	91.81	96.83	88.94	87.48
	15	75.54	92.65	76.27	79.20
	10	48.76	85.04	59.43	63.27
	5	22.16	67.05	38.00	47.22
	0	9.52	39.57	17.38	24.12
	-5	5.77	17.11	3.57	8.16
car	20	96.57	98.30	98.12	97.64
	15	84.76	96.96	96.39	96.87
	10	57.35	91.50	91.05	92.48
	5	23.38	70.27	76.29	76.74
	0	8.65	31.05	49.00	45.75
	-5	6.53	15.72	19.89	18.25
exhibition	20	96.39	97.41	96.36	96.36
	15	89.36	95.83	92.75	94.35
	10	68.44	91.05	84.73	89.02
	5	35.11	75.75	65.47	75.13
	0	11.05	42.27	38.20	46.74
	-5	6.60	15.46	16.23	19.72
Avg. (0–20dB)		57.39	79.28	72.19	76.81

more likely to find a match in segments containing only babble noise. The use of BPC and the resulting increased variances will make this effect even more pronounced, and an even higher number of insertion errors will occur.

By looking at the average word accuracies in Table 5.1, one can observe that the performance for babble noise is so poor that the average word accuracy is lower for BPC-PMC than for PMC. In the next section we will see how one can improve the result by introducing prior scaling.

5.2 Joint BPC-PMC with Prior Scaling

Compared to traditional PMC, the BPC-PMC approach presented in Section 5.1 did not perform well in the presence of babble noise. As discussed in Section 5.1.4, the problem is caused by a lot of insertion errors when using BPC to increase variances in presence of noise which contains "background speech". In order to avoid this negative effect, some measures must be taken to reduce the number of insertion errors.

In this section we improve our approach by using a prior that is scaled on a frame-by-frame basis in order to avoid an increase of the variance in periods where we are relatively confident that no speech is present. This is done by calculating a confidence measure for each frame. More specifically, this confidence measure is calculated as a likelihood ratio based on a model for noisy speech and a noise model. Then, the width of the BPC prior is scaled according to the value of the likelihood ratio. The resulting approach can be viewed as compromise between using plain PMC and the joint BPC-PMC approach. As we shall see, the resulting performance is increased for noise types containing background speech at the cost of a performance reduction for other noise types.

5.2.1 Introducing the Scale Factor

As can be seen from (5.2), the variance is increased compared to the original state density. We would like to reduce the variance during periods when there is no speech activity by introducing a frame-dependent scale factor $\xi_t \in [0, 1]$ on the prior variance. The resulting prior for frame t is then

$$p_t(\mu_{imd}|\phi^t_{imd}) = \mathcal{N}(\mu_{imd}; \alpha_{imd}, \xi_t \nu_d^2). \tag{5.5}$$

This prior results in the following state density for frame t and HMM state i:

$$\tilde{p}_{ti}(\mathbf{y}_t) = \sum_{m=1}^{M} w_{im} \prod_{d=1}^{D} \mathcal{N}(y_{td}; \alpha_{imd}, \sigma^2_{imd} + \xi_t \nu_d^2). \tag{5.6}$$

The scale factor will be set to reflect our confidence that frame t does contain speech. A high confidence will result in a scale factor with value close to 1, while a low confidence will give a value close to 0.

5.2.2 Setting the Scale Factor

In order to determine whether a given frame t is likely to contain speech or not, we make use of the log likelihood ratio (LLR)

$$\mathrm{LLR}(\mathbf{y}) = \log \frac{p(\mathbf{y}|\Omega_0)}{p(\mathbf{y}|\Omega_1)} = \log p(\mathbf{y}|\Omega_0) - \log p(\mathbf{y}|\Omega_1), \tag{5.7}$$

where the classes Ω_0 and Ω_1 represent speech present and speech absent respectively. The method that is used to calculate this LLR will be described in detail in Chapter 7, where it is used for a different purpose.

Let η_t denote the LLR for frame t. The problem that remains is how to set the scale factor as a function of η_t. The most straightforward approach is to perform voice activity detection by simply setting a threshold τ on the value of η_t, i.e.,[2]

$$\xi_t = \begin{cases} 1 & \text{if } \eta_t > \tau \\ 0 & \text{if } \eta_t \leq \tau. \end{cases} \tag{5.8}$$

However, this means that we make a hard decision at each frame. An alternative solution is to use a continuous function $f : \mathbb{R} \to [0,1]$ for mapping the LLR-values to scale factors, i.e., $\xi_t = f(\eta_t)$. A natural choice for f is a sigmoid function. The parameters of the sigmoid have to be optimized in order to find a suitable slope and shift. In this chapter we use a sigmoid on the form

$$\xi_t = \frac{1}{1 + \exp(-\frac{9.2}{\psi}(\eta_t - \frac{3}{2}\psi))}, \tag{5.9}$$

where the parameter ψ determines the shift and plays a similar role as the threshold in (5.8).

5.2.3 Summary of the Procedure

We will now sum up the steps needed to implement the proposed procedure. The following steps are performed for each file to be recognized:

1. Estimate noise parameters from the first N frames
2. Using PMC as described in Section 3.2.1, find estimates $\{\mu^y_{imd}\}$ of the noisy speech mean for all mixture components m in all HMM states i
3. For each frame t:

[2]Note that setting $\xi_t = 0$ reduces the prior to a delta function: $p_t(\mu_{imd}|\phi^t_{imd}) = \delta(\mu_{imd} - \alpha_{imd})$.

(a) Calculate the LLR value η_t

(b) Calculate the scale factor ξ_t using either (5.8) or (5.9)

(c) For each HMM state i, mixture component m, and feature vector element d, set the parameters of the predictive distribution as follows:

   i. The mean is set to $\alpha_{imd} = \mu^{y}_{imd}$

   ii. The variance is set to $\sigma^2_{imd} + \xi_t \nu^2_d$, with ν^2_d given in (5.4)

5.2.4 Experiments and Results

Most of the experimental setup was the same as in Section 5.1.4. Note that for the calculation of the LLR in (5.7), only the static parameters were used.

The results for four different methods are shown in Table 5.2. The approach using hard decision prior scaling in (5.8) has been given the label BPC-vad, and the approach using the sigmoid in (5.9) has been given the label BPC-sig. For these approaches, the parameters τ and ψ have been optimized to give the best average performance. The result was $\tau = 26$ for BPC-vad, and $\psi = 21$ for BPC-sig. The performance of BPC-sig was almost the same as for BPC-vad.

When we look at the two approaches using prior scaling, we can see that the performance for babble is significantly increased compared to BPC-PMC. At the same time, the performance on subway and car is reduced. On exhibition we can observe an improvement (except at -5dB). This is because exhibition noise also contains some background speech, and we are able to remove insertion errors by using prior scaling. On average we are able to improve the performance compared to PMC.

Statistical significance tests comparing PMC, BPC-PMC, and BPC-vad can be found in Table F.2 and Table F.3 in Appendix F.

5.3 Conclusion

In this chapter we presented an approach that combined parallel model combination (PMC) and Bayesian predictive classification (BPC). The motivation for this joint approach was to compensate for uncertainty in the noisy speech model calculated by PMC. The BPC-PMC approach was realized by setting the BPC prior mean to the noisy speech mean calculated by PMC. While the BPC-PMC approach performed well for most noise types, it resulted in large amounts of insertion errors in the case of babble noise due to increased variance of the compensated models. In order to mitigate

Table 5.2: Word accuracies (%) on Aurora2 using PMC, BPC-PMC, BPC-vad, BPC-sig

Noise	SNR	PMC	BPC-PMC	BPC-vad	BPC-sig
subway	20	97.88	97.39	97.39	97.39
	15	96.53	96.01	95.95	95.89
	10	92.66	92.20	91.86	91.93
	5	80.35	82.13	81.58	81.67
	0	46.67	56.00	52.72	52.59
	-5	21.31	25.88	23.24	23.30
babble	20	96.83	87.48	94.50	94.50
	15	92.65	79.20	89.60	89.66
	10	85.04	63.27	79.50	79.56
	5	67.05	47.22	64.33	64.06
	0	39.57	24.12	38.15	38.12
	-5	17.11	8.16	16.60	16.48
car	20	98.30	97.64	97.55	97.52
	15	96.96	96.87	96.75	96.72
	10	91.50	92.48	91.92	91.92
	5	70.27	76.74	74.80	74.56
	0	31.05	45.75	38.56	38.41
	-5	15.72	18.25	16.19	16.10
exhib.	20	97.41	96.36	96.82	96.85
	15	95.83	94.35	95.09	95.09
	10	91.05	89.02	89.85	89.82
	5	75.75	75.13	77.45	77.23
	0	42.27	46.74	47.39	47.42
	-5	15.46	19.72	19.13	19.01
Avg. (0–20dB)		79.28	76.81	79.59	79.55

this effect, prior scaling was introduced. This approach scaled the prior variance such that the variance of the compensated models became large if the confidence in speech presence was high, and small if the confidence in speech presence was low. This increased the performance for babble noise at the cost of a reduction in performance for some of the other noise types, resulting in improved average word accuracy.

Chapter 6

Bayesian Learning of Speech Models for MMSE Feature Enhancement

Several methods for MMSE feature enhancement of noisy speech make use of probabilistic models of speech and noise. The traditional approach for training such models is maximum likelihood (ML) estimation. An alternative to ML learning is Bayesian learning. Bayesian learning has some advantages compared to ML learning, and in this chapter we investigate whether these advantages can contribute to improving the performance of MMSE feature enhancement. A part of this work was presented in [58].

In order to clarify what the novel contributions of this chapter are, we will start by giving a short outline. First, we begin by reviewing the general principles of Bayesian learning in Section 6.1. Then, in Section 6.2 we describe variational Bayesian (VB) learning, which is an approximation to exact Bayesian learning. Our presentation in this section is mainly based on the work of Attias [3]. Then, we describe how variational Bayesian learning can be used on a Gaussian mixture model in Section 6.3. This section is also based on [3], where the formulas for VB learning were given without derivation. To the best of our knowledge, the derivations behind these formulas have not been published anywhere. Therefore, we have included derivations of VB learning for the GMM in Appendix C. Then, in Section 6.4 we present the novel idea of this chapter, namely application of VB learning to front-end speech models for MMSE feature enhancement. Section 6.5 then describes different ways of initializing the VB learning algorithm, followed by MMSE feature enhancement experiments using Algonquin in Section 6.6 and MBFE in Section 6.7. Finally, conclusions are drawn in Section 6.8.

6.1 Bayesian Learning

The conventional method for training statistical models such as GMMs and HMMs is maximum likelihood (ML). However, there are a couple of drawbacks to ML learning. One problem is that ML has a tendency to overfit the model to the training data, causing a lack of generality in the resulting model. Another problem is that ML always prefers complex models over simple models. The reason for this is that the objective of ML learning is to maximize the likelihood of the training data. A more complex model has more parameters than a simple model, and using more parameters allows ML to obtain a better fit to the data. Thus, the criterion used for ML learning is not well suited for selecting an optimal model structure.[1]

The Bayesian approach to model learning tries to solve these problems. In general, the Bayesian approach has three advantages over ML learning [74]:

1. Incorporation of prior knowledge through the use of prior distributions
2. Model selection by maximization of the posterior over model structures
3. Robust classification through marginalization over model parameters

In ML learning, the parameters are viewed as fixed but unknown quantities, and the objective is to find the *point estimate* of the model parameters that results in the highest likelihood. In Bayesian learning, the parameters are considered to be random variables, distributed according to some probability distribution. The goal of Bayesian learning is to estimate the *posterior distribution* of the model parameters, instead of a point estimate.[2] Having obtained this posterior one can reduce effects of overfitting as well as increase classification robustness by integrating out the model parameters. In addition to obtaining the posterior of the model parameters, the Bayesian approach tries to estimate the posterior distribution of model structures. This distribution can be used for model selection.

For most statistical models Bayesian learning is far from trivial. To be able to implement Bayesian learning for models such as GMMs and HMMs, one has to resort to approximations. One such approximation is

[1]Note that it is possible to use additional criteria, such as the minimum description length (MDL) [62], in order to perform model selection based on ML learning.

[2]Note that maximum a posteriori (MAP) estimation, which does consider the parameters to be random variables, is not a completely Bayesian approach. This is because it only seeks the mode of the posterior, which may not be representative of the posterior distribution. See [5, p. 33] for more details on this.

the *variational Bayesian* approach [3, 74, 5, 72]. Examples of previous applications of VB learning to speech recognition are training of GMMs for recognition of confusable phones [73] and HMM model-selection and training for speech recognition [75, 76].

6.2 Variational Bayesian Learning

In [3] Attias proposed a variational approach for Bayesian learning of graphical models. Let $\mathbf{Y} = \{\mathbf{y}_1, \cdots, \mathbf{y}_N\}$ denote an observed dataset consisting of N independent and identically distributed items. Moreover, let $\mathbf{Z} = \{\mathbf{z}_1, \cdots, \mathbf{z}_N\}$ denote hidden variables and $\boldsymbol{\theta}$ denote the parameters. For a given model structure M, the goal is to compute the parameter posterior $p(\boldsymbol{\theta}|\mathbf{Y}, M)$. In addition, for the purpose of model selection, the posterior of model structures $p(M|\mathbf{Y})$ is of interest.

To make Bayesian computations tractable, the key point is to approximate the joint posterior $p(\mathbf{Z}, \boldsymbol{\theta}|\mathbf{Y}, M)$ by a variational posterior $q(\mathbf{Z}, \boldsymbol{\theta}|\mathbf{Y})$ which is restricted to a factorized form as

$$q(\mathbf{Z}, \boldsymbol{\theta}|\mathbf{Y}) = q(\mathbf{Z}|\mathbf{Y})q(\boldsymbol{\theta}|\mathbf{Y}). \tag{6.1}$$

Note that q should always be understood as conditioned on $\mathbf{Y}$, although it is common not to write this explicitly. We will also follow this convention.

If we knew the true posterior $p(\mathbf{Z}, \boldsymbol{\theta}|\mathbf{Y}, M)$ we would be able to calculate the marginal log likelihood $\log p(\mathbf{Y}|M)$, which could then be used for comparing different models. However, since we do not know $p(\mathbf{Z}, \boldsymbol{\theta}|\mathbf{Y}, M)$, we have to use the factorized form in (6.1) in order to find an approximate posterior. This can be done by seeking the posterior $q(\mathbf{Z}, \boldsymbol{\theta})$ that maximizes a lower bound for $\log p(\mathbf{Y}|M)$. This lower bound can be derived as follows:

$$\begin{aligned}\log p(\mathbf{Y}|M) &= \log \int p(\mathbf{Y}, \mathbf{Z}, \boldsymbol{\theta}|M) d\mathbf{Z} d\boldsymbol{\theta} \\ &= \log \int q(\mathbf{Z}, \boldsymbol{\theta}) \frac{p(\mathbf{Y}, \mathbf{Z}, \boldsymbol{\theta}|M)}{q(\mathbf{Z}, \boldsymbol{\theta})} d\mathbf{Z} d\boldsymbol{\theta}.\end{aligned} \tag{6.2}$$

Now, by applying Jensen's inequality, and the factorization in (6.1), we obtain

$$\begin{aligned}\log p(\mathbf{Y}|M) &\geq \int q(\mathbf{Z}, \boldsymbol{\theta}) \log \frac{p(\mathbf{Y}, \mathbf{Z}, \boldsymbol{\theta}|M)}{q(\mathbf{Z}, \boldsymbol{\theta})} d\mathbf{Z} d\boldsymbol{\theta} \\ &= \int q(\mathbf{Z})q(\boldsymbol{\theta}) \log \frac{p(\mathbf{Y}, \mathbf{Z}, \boldsymbol{\theta}|M)}{q(\mathbf{Z})q(\boldsymbol{\theta})} d\boldsymbol{\theta} d\mathbf{Z}.\end{aligned} \tag{6.3}$$

The lower bound in (6.3) is often referred to as *free energy*, and we will denote it by $\mathcal{F}_M(q(\mathbf{Z}), q(\boldsymbol{\theta}))$. We have now reformulated the problem of computing the posterior as an optimization problem, where the cost function is $\mathcal{F}_M$.

The function $\mathcal{F}_M$ can also be rewritten on a form that shows more clearly that it penalizes model complexity. This can be done as follows:

$$\begin{aligned}\mathcal{F}_M(q(\mathbf{Z}), q(\boldsymbol{\theta})) &= \int q(\mathbf{Z})q(\boldsymbol{\theta}) \left[\log \frac{p(\mathbf{Y}, \mathbf{Z}|\boldsymbol{\theta}, M)}{q(\mathbf{Z})} + \log \frac{p(\boldsymbol{\theta}|M)}{q(\boldsymbol{\theta})}\right] d\boldsymbol{\theta} d\mathbf{Z} \\ &= \int q(\mathbf{Z})q(\boldsymbol{\theta}) \log \frac{p(\mathbf{Y}, \mathbf{Z}|\boldsymbol{\theta}, M)}{q(\mathbf{Z})} d\boldsymbol{\theta} d\mathbf{Z} - \mathrm{KL}[q(\boldsymbol{\theta})\|p(\boldsymbol{\theta}|M)] \\ &= \mathrm{E}_{\mathbf{Z},\boldsymbol{\theta}}\left[\log \frac{p(\mathbf{Y}, \mathbf{Z}|\boldsymbol{\theta}, M)}{q(\mathbf{Z})}\right] - \mathrm{KL}[q(\boldsymbol{\theta})\|p(\boldsymbol{\theta}|M)].\end{aligned} \tag{6.4}$$

Here, $\mathrm{E}_{\mathbf{Z},\boldsymbol{\theta}}[\cdot]$ denotes the expectation with respect to $q(\mathbf{Z}, \boldsymbol{\theta})$ and KL denotes the Kullback-Leibler distance. Thus, the last term is now the KL distance between the parameter prior and the variational posterior $q(\boldsymbol{\theta})$. While the first term corresponds to the averaged likelihood, the second term can be interpreted as a penalty term for more complex models. As we increase the number of parameters in order to increase the average likelihood, the KL distance will also increase and thus reduce the total value of $\mathcal{F}_M$. Assuming equal prior probabilities for all model structures M, the model with the highest value of $\mathcal{F}_M$ corresponds to the model with the highest posterior probability.

In order to optimize the free energy with respect to the posteriors $q(\mathbf{Z})$ and $q(\boldsymbol{\theta})$, an EM-like algorithm can be derived. See e.g. [5] for a derivation of this procedure. The E-step consists of computing the variational posterior over hidden variables as

$$q(\mathbf{Z}) \propto \exp\left\{\mathrm{E}_{\boldsymbol{\theta}}[\log p(\mathbf{Y}, \mathbf{Z}|\boldsymbol{\theta}, M)]\right\}. \tag{6.5}$$

The M-step is then to compute the variational parameter posterior as

$$q(\boldsymbol{\theta}) \propto \exp\{\mathrm{E}_{\mathbf{Z}}[\log p(\mathbf{Y}, \mathbf{Z}|\boldsymbol{\theta}, M)]\}p(\boldsymbol{\theta}|M). \tag{6.6}$$

6.3 Variational Bayesian Learning for the GMM

The application of VB learning as described in Section 6.2 for the GMM was presented in [3]. In the case of a GMM, the hidden variables are scalars $\mathbf{Z} = \{z_1, \ldots, z_N\}$ describing which mixture component that generated each observed vector. Using the hidden variables, the GMM can be written on the form

$$p(\mathbf{y}_n|\boldsymbol{\theta}, M) = \sum_{z=1}^{M} p(\mathbf{y}_n|z_n = z, \boldsymbol{\theta})p(z_n = z|\boldsymbol{\theta}). \tag{6.7}$$

In this case, the model structure is simply the number of components, denoted M. Each component has a Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z)$. Note that in this section we will make use of the inverse covariance matrix $\boldsymbol{\Gamma}_z = \boldsymbol{\Sigma}_z^{-1}$. The matrix $\boldsymbol{\Gamma}_z$ is called the precision matrix, and for the algorithm presented in this section it is convenient to use precision instead of covariance. Note that $p(z_n = z|\boldsymbol{\theta})$ in (6.7) is simply the mixture weight for component z, i.e., $p(z_n = z|\boldsymbol{\theta}) = w_z$.

It is useful to choose prior densities from conjugate families, since the posterior densities will then belong to the same families as the priors. As a consequence, the VB learning simply amounts to updating the hyperparameters of the posteriors. Thus, the following conjugate priors are defined for the parameters $\boldsymbol{\theta}$:

$$p(\{w_z\}) = \mathcal{D}(\lambda^0) \tag{6.8}$$
$$p(\boldsymbol{\mu}_z|\boldsymbol{\Gamma}_z) = \mathcal{N}(\boldsymbol{\rho}^0, \beta^0\boldsymbol{\Gamma}_z) \tag{6.9}$$
$$p(\boldsymbol{\Gamma}_z) = \mathcal{W}(\nu^0, \boldsymbol{\Phi}^0). \tag{6.10}$$

Here $\mathcal{D}$ and $\mathcal{W}$ denote Dirichlet and Wishart densities respectively. A short description of these distributions can be found in Section B.2 and Section B.3 in the appendix. The corresponding variational posteriors are given by:

$$q(\{w_z\}) = \mathcal{D}(\{\lambda_z\}) \tag{6.11}$$
$$q(\boldsymbol{\mu}_z|\boldsymbol{\Gamma}_z) = \mathcal{N}(\boldsymbol{\rho}_z, \beta_z\boldsymbol{\Gamma}_z) \tag{6.12}$$
$$q(\boldsymbol{\Gamma}_z) = \mathcal{W}(\nu_z, \boldsymbol{\Phi}_z). \tag{6.13}$$

The objective of the E-step is to compute the variational posterior over hidden variables $q(\mathbf{Z})$. Defining $\gamma_z^n = q(z_n = z|\mathbf{y}_n)$, this can be done as follows:

$$q(z_n = z) = \gamma_z^n \propto \tilde{w}_z\tilde{\Gamma}_z^{1/2} \exp\left\{-\frac{1}{2}(\mathbf{y}_n - \boldsymbol{\rho}_z)^T\bar{\boldsymbol{\Gamma}}_z(\mathbf{y}_n - \boldsymbol{\rho}_z) - \frac{D}{2\beta_z}\right\} \tag{6.14}$$

$$q(\mathbf{Z}) = \prod_{n=1}^{N} q(z_n), \tag{6.15}$$

where

$$\log \tilde{w}_z = \mathrm{E}_{\boldsymbol{\theta}}[\log w_z] = \psi(\lambda_z) - \psi\left(\sum_{z'} \lambda_{z'}\right) \tag{6.16}$$

$$\log \tilde{\Gamma}_z = \mathrm{E}_{\boldsymbol{\theta}}[\log |\mathbf{\Gamma}_z|] = \sum_{i=1}^{D} \psi\left(\frac{\nu_z + 1 - i}{2}\right) - \log |\mathbf{\Phi}_z| + D \log 2 \tag{6.17}$$

$$\bar{\mathbf{\Gamma}}_z = \mathrm{E}_{\boldsymbol{\theta}}[\mathbf{\Gamma}_z] = \nu_z \mathbf{\Phi}_z^{-1}. \tag{6.18}$$

In the above equations, ψ denotes the digamma function, which is briefly described in Section B.1 in the appendix. The normalization constant of γ_z^n can be found by using the constraint that $\sum_{z=1}^{M} \gamma_z^n = 1$ for all n.

The M-step can be divided into two stages. In the first stage, which is the same as in the ordinary EM algorithm, the following quantities are computed:

$$\bar{w}_z = \frac{1}{N} \sum_{n=1}^{N} \gamma_z^n \tag{6.19}$$

$$\bar{\boldsymbol{\mu}}_z = \frac{1}{\bar{N}_z} \sum_{n=1}^{N} \gamma_z^n \mathbf{y}_n \tag{6.20}$$

$$\bar{\mathbf{\Sigma}}_z = \frac{1}{\bar{N}_z} \sum_{n=1}^{N} \gamma_z^n \mathbf{C}_z^n. \tag{6.21}$$

In the above equations, we have defined

$$\mathbf{C}_z^n = (\mathbf{y}_n - \bar{\boldsymbol{\mu}}_z)(\mathbf{y}_n - \bar{\boldsymbol{\mu}}_z)^T \tag{6.22}$$

$$\bar{N}_z = N \bar{w}_z. \tag{6.23}$$

The hyperparameters of the posteriors are then updated in the second stage.

$$\lambda_z = \bar{N}_z + \lambda^0 \tag{6.24}$$

$$\nu_z = \bar{N}_z + \nu^0 \tag{6.25}$$

$$\beta_z = \bar{N}_z + \beta^0 \tag{6.26}$$

$$\boldsymbol{\rho}_z = \frac{\bar{N}_z \bar{\boldsymbol{\mu}}_z + \beta^0 \boldsymbol{\rho}^0}{\bar{N}_z + \beta^0} \tag{6.27}$$

$$\mathbf{\Phi}_z = \bar{N}_z \bar{\mathbf{\Sigma}}_z + \frac{\bar{N}_z \beta^0}{\bar{N}_z + \beta^0} (\bar{\boldsymbol{\mu}}_z - \boldsymbol{\rho}^0)(\bar{\boldsymbol{\mu}}_z - \boldsymbol{\rho}^0)^T + \mathbf{\Phi}^0 \tag{6.28}$$

Since posteriors are computed instead of parameters, the predictive density is used for unseen data. In this density the parameters $\boldsymbol{\theta}$ are integrated out. This gives us a mixture of multivariate t-distributions[3] on the form

$$p(\mathbf{y}|\mathbf{Y}) = \sum_{z=1}^{M} \bar{w}_z t_{\kappa_z}(\mathbf{y}; \boldsymbol{\rho}_z, \boldsymbol{\Omega}_z). \tag{6.29}$$

For component z, the degrees of freedom are $\kappa_z = \nu_z + 1 - D$, the mean is $\boldsymbol{\rho}_z$ and the covariance is $\boldsymbol{\Omega}_z = ((\beta_z + 1)/\beta_z \kappa_z)\boldsymbol{\Phi}_z$. The mixture weight is given by $\bar{w}_z = \lambda_z / \sum_{z'} \lambda_{z'}$.

In order to be able to compare different models, we still need to find an expression for the free energy of the GMM. A derivation of this expression can be found in Appendix D.

6.3.1 The Diagonal Covariance Case

If diagonal covariance matrices are used, a few changes are needed to the algorithm presented in Section 6.3. In this case we are making the assumption that the vector elements are statistically independent. When the vector elements of a D-dimensional multivariate normal distribution are independent, the distribution can be simplified to a product of D one-dimensional normal distributions. For a one-dimensional Gaussian with parameters mean and precision, the conjugate prior is the normal-gamma prior. This means that instead of using a Wishart density as a prior for the precision matrix, we now consider each element of the diagonal precision matrix independently, and use a gamma density as prior. Assuming that the precision matrix is diagonal, we can write

$$\boldsymbol{\Gamma}_z = \mathrm{diag}(\mathbf{r}_z), \quad \mathbf{r}_z = \begin{bmatrix} r_{z1} \\ \vdots \\ r_{zD} \end{bmatrix} \tag{6.30}$$

where $\mathbf{r}_z$ is the vector consisting of the diagonal precision elements. Denoting the gamma density by $\mathcal{G}$, the new prior is given by

$$p(\boldsymbol{\Gamma}_z) = \prod_{d=1}^{D} p(r_{zd}) = \prod_{d=1}^{D} \mathcal{G}(\nu^0, \phi_d^0). \tag{6.31}$$

A short description of the gamma distribution can be found in Section B.4 in the appendix. The expression for the E-step in (6.14) can be simplified

[3]See Section B.5 in the appendix for a short description.

to

$$\gamma_z^n \propto \tilde{w}_z \tilde{\Gamma}_z^{1/2} \exp\left\{-\frac{1}{2}\sum_{d=1}^{D} \bar{r}_{zd}(y_{nd}-\rho_{zd})^2 - \frac{D}{2\beta_z}\right\}, \tag{6.32}$$

where

$$\log \tilde{\Gamma}_z = \sum_{d=1}^{D} \mathrm{E}_{\boldsymbol{\theta}}[\log r_{zd}] = D\psi(\nu_z/2) - \sum_{d=1}^{D} \log(\phi_{zd}/2) \tag{6.33}$$

$$\bar{r}_{zd} = \frac{\nu_z}{\phi_{zd}}. \tag{6.34}$$

Moreover, (6.21) can be simplified to the following scalar version:

$$\bar{\sigma}_{zd}^2 = \frac{1}{\bar{N}_z}\sum_{n=1}^{N} \gamma_z^n (y_{nd}-\bar{\mu}_{zd})^2. \tag{6.35}$$

Consequently, a scalar version of (6.28) for the update of hyperparameter ϕ_{zd} is given by

$$\phi_{zd} = \bar{N}_z\bar{\sigma}_{zd}^2 + \frac{\bar{N}_z\beta^0}{\bar{N}_z+\beta^0}(\bar{\mu}_{zd}-\rho_d^0)^2 + \phi_d^0. \tag{6.36}$$

6.4 VB Trained Models used in MMSE Feature Enhancement

As described in Section 6.1, there are several advantages to the Bayesian learning approach compared to traditional ML training. When only a small amount of data is available, ML training suffers from overfitting problems if the chosen model structure is too complex. In addition, if a component is assigned very few observations during ML training, numerical problems can arise. Because of the regularization effects from the priors, the VB training has no such numerical problems. In addition, since the VB objective function contains a penalty term for complex models, the training has an ability to prune the trained model according to the amount of data available. Thus, even if the model structure is chosen too complex, the model will not have the same overfitting problems as ML. Moreover, the VB free energy can be used as a model selection criterion to choose the right model complexity.

We will apply the algorithm described in Section 6.3 to train the speech priors $p(\mathbf{s})$ used by Algonquin and MBFE for MMSE feature enhancement. The result of the VB training is posteriors for the parameters of $p(\mathbf{s})$. Ideally, we should use the predictive distribution given by (6.29) as a prior for MMSE feature enhancement. However, since Algonquin and MBFE are based on the assumption that the mixture components are Gaussian, we approximate each of the multivariate t-distributions with the multivariate Gaussian that was closest with respect to KL-distance. Given a component z, it can be shown that this is a Gaussian with mean and covariance equal to that of the multivariate t-distribution [7], i.e.,

$$p(\mathbf{s}) = \sum_{z=1}^{M} \bar{w}_z \mathcal{N}(\mathbf{s}; \boldsymbol{\rho}_z, \boldsymbol{\Omega}_z). \tag{6.37}$$

6.5 Initialization of the VB Algorithm

The VB algorithm for training a GMM uses an iterative EM approach. When using the EM algorithm, initial parameter values are needed as a starting point for the iterative procedure. There are several possible ways of generating initial parameter estimates.

One option is to use k-means clustering [70, p. 532]. In this case the model size will be predefined by the number of clusters k. The k-means algorithm partitions the data set into k sets, and using this partition one can set initial values of $\gamma_z^n = q(z_n = z|\mathbf{y}_n)$ which are used in the VB training algorithm. Given that vector $\mathbf{y}_n$ has been assigned to cluster z, we set $\gamma_z^n = 1$ and $\gamma_i^n = 0$ for all $i \neq z$. With these initial values one can run the M-step of the VB algorithm and obtain initial parameter values.

Another option is to use iterative mixture splitting, which is commonly used when training HMMs for ASR. This is done by starting off with only one mixture component and estimating its parameters. Then, one can split this component by generating a new component with almost the same parameters and start another EM cycle. When there are several mixture components one has to use some criterion for determining which component to split next. Note that it is possible to split several components at the same time. However, in this thesis we will only split one component between each EM cycle. More specifically, we use the following iterative mixture splitting for the VB algorithm:

1. Select the component $z = \arg\max_l \lambda_l$ for splitting.

2. Split component z into two components i and j.

3. Set the new values of λ, β and ν as follows:

$$\lambda_i = \lambda_j = \frac{\lambda_z - \lambda^0}{2} \tag{6.38}$$

$$\beta_i = \beta_j = \frac{\beta_z - \beta^0}{2} \tag{6.39}$$

$$\nu_i = \nu_j = \frac{\nu_z - \nu^0}{2}. \tag{6.40}$$

4. The value of $\mathbf{\Phi}$ is left unchanged:

$$\mathbf{\Phi}_i = \mathbf{\Phi}_j = \mathbf{\Phi}_z. \tag{6.41}$$

5. The value of $\boldsymbol{\rho}$ is perturbed by a perturbation vector $\mathbf{p}$ as follows:

   (a) Set the perturbation vector to a small amount ϵ of the (expected) standard deviation of each vector element in component z, i.e.,

$$\mathbf{p} = \epsilon \begin{bmatrix} \sqrt{\frac{\phi_{z1}}{\nu_z}} \\ \vdots \\ \sqrt{\frac{\phi_{zD}}{\nu_z}} \end{bmatrix} \tag{6.42}$$

   where ϕ_{zd} denotes element d on the diagonal of matrix $\mathbf{\Phi}_z$.

   (b) Set the new values of $\boldsymbol{\rho}$ as

$$\boldsymbol{\rho}_i = \boldsymbol{\rho}_z - \mathbf{p} \tag{6.43}$$

$$\boldsymbol{\rho}_j = \boldsymbol{\rho}_z + \mathbf{p}. \tag{6.44}$$

6.6 Experiments in the Log-Spectral Domain with Algonquin

6.6.1 Preliminary Experiments Using Subway Noise

The first experiments were run using the log-spectral domain version of Algonquin. Noisy speech files were first denoised in the log-spectral domain using different speech GMMs as the prior $p(\mathbf{s})$. Then, the resulting files were transformed to the MFCC domain. The log-spectral feature vectors used during enhancement were 23-dimensional, while the MFCCs were 39-dimensional including delta and acceleration parameters, with C_0 as energy. Using these feature vectors, recognition was then performed with models

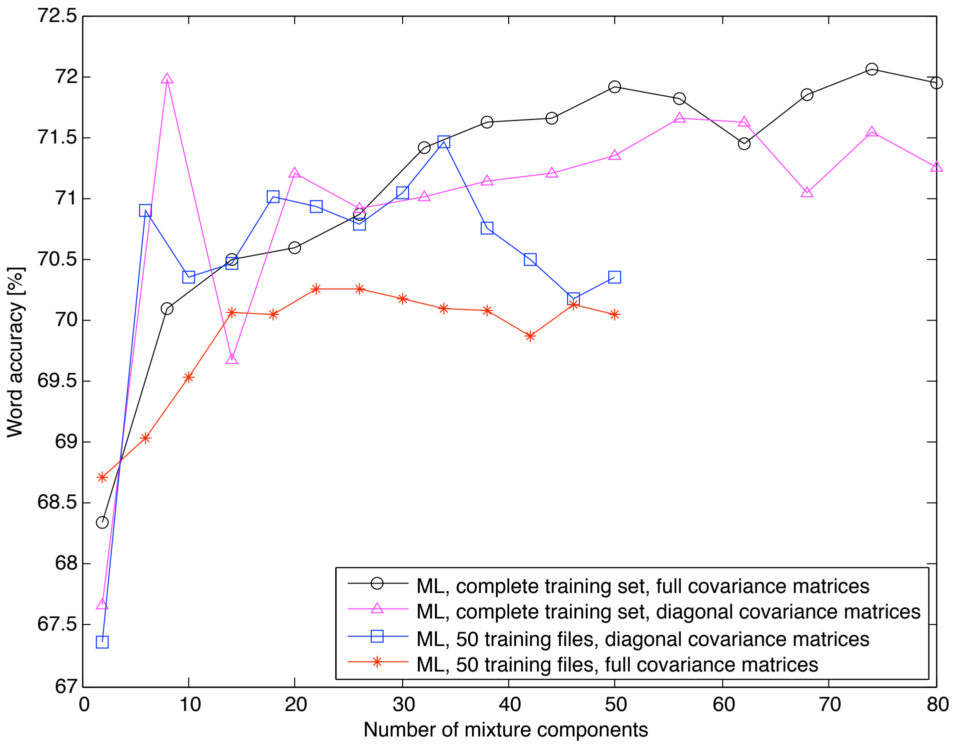

Figure 6.1: Feature enhancement in log-spectral domain with Algonquin. Recognition performance for ML when using different training sets and diagonal/full covariance matrices.

trained in clean condition. For the experiments in this section, we chose the subset of Aurora2 containing subway noise at 5 dB. The baseline recognition result, using no feature cleaning, was a word accuracy of 45.26%.

In the log-spectral domain there is a high degree of correlation between neighboring feature vector elements due to the use of overlapping filters in the mel-scale filter bank. Thus, it is beneficial to use full covariance matrices. It was observed experimentally that training full covariance matrices gave significantly higher performance than diagonal covariance matrices. However, when using the full covariance matrix in Algonquin, only a small advantage in performance was observed compared to simply using the diagonal elements of the fully trained covariance matrix. Due to the need for a lot of matrix inversion operations when using full covariance matrices during feature enhancement, the computational complexity becomes a lot higher than when diagonal covariance matrices are used. As a result, full covariance matrices were used during training, but only the diagonal elements were used during feature enhancement. Figure 6.1 shows experimental results obtained with diagonal and full covariance matrices during feature enhancement.

In order to further reduce the computational complexity, we selected only a subset of the complete Aurora2 training set for model estimation. We chose four different training sets, each consisting of 50 randomly selected files. Figure 6.1 shows the performance of ML-trained GMMs using full covariance matrices and diagonal covariance matrices, trained on both the complete training set and the four training sets which only contain 50 files each. The latter has been averaged over the four selected sets. Note that all the models used here were trained using iterative mixture splitting. The plot shows that when only using 50 files for model training, the performance obtained with diagonal covariance is better than with full covariance matrices. This is probably because the amount of training data is too small to obtain robust estimates of the full covariance matrices. In addition, we can see that the performance of the diagonal covariance matrices and a training set of 50 files is comparable to using the complete training set for models with less than 40 mixture components. Thus, in this section, we will keep the model sizes in this range, with a maximum of 50 mixture components.

In order to compare ML and VB training we tested both k-means initialization and iterative mixture splitting for both methods. For the VB training, the prior was set as follows:

$$\lambda^0 = 1, \quad \beta^0 = 1, \quad \nu^0 = D \tag{6.45}$$

$$\boldsymbol{\rho}^0 = \mathbf{0}, \quad \boldsymbol{\Phi}^0 = \tau\mathbf{I} \tag{6.46}$$

where D denotes the feature dimension and $\mathbf{I}$ denotes the $D \times D$ identity matrix.

The first experiment was done with k-means initialization for both ML and VB. The scaling factor for $\boldsymbol{\Phi}^0$ was set to $\tau = 10.0$. Recognition results for models with number of mixture components M varying from 10 to 50 can be seen in Table 6.1. This table shows results for each of the 4 training sets. The ML results stop at around 30 mixture components, where numerical problems arose due to lack of training data. However, the results show that for VB the results keep improving after that point. In addition, for a given model size VB is slightly better than ML in most cases.

In the next experiment we tested ML and VB using iterative mixture splitting. The best scaling factor for $\boldsymbol{\Phi}^0$ was found at $\tau = 3.0$. A plot showing recognition performance for all cases averaged over the 4 training

Table 6.1: Recognition performance (word accuracy) after denoising files containing subway noise at 5 dB, using models trained with four different training sets.

	Set 1		Set 2		Set 3		Set 4	
M	ML	VB	ML	VB	ML	VB	ML	VB
10	70.22	71.11	69.48	69.82	69.48	70.53	70.10	70.68
14	70.65	69.88	70.19	71.23	69.54	69.20	67.64	69.88
18	70.56	71.05	69.42	70.22	69.79	70.22	70.28	70.86
22	72.28	71.29	69.30	69.30	68.93	69.63	69.97	70.92
26	71.72	72.24	68.44	69.97	69.17	71.02	67.88	70.37
30	71.08	71.35	—	70.80	68.38	71.02	69.39	71.85
34	69.70	71.97	—	70.95	—	70.80	—	72.24
38	—	72.74	—	71.26	—	71.94	—	71.72
42	—	72.18	—	70.37	—	71.02	—	70.74
46	—	71.17	—	69.94	—	71.57	—	70.62
50	—	70.31	—	70.43	—	71.05	—	71.85

sets is shown in Figure 6.2. In this plot the results for ML using k-means and iterative mixture splitting have been given the labels ML-KM and ML-IMS respectively, and the corresponding results for VB have been labeled VB-KM and VB-IMS. From this plot one can see that iterative mixture splitting clearly improved ML's robustness towards numerical problems, and that iterative mixture splitting works better than k-means initialization for these training sets. Iterative mixture splitting also works better in the VB case. In addition, it can be seen that VB with mixture splitting obtains a stable and good performance at around 14 mixture components, and that it outperforms ML.

Statistical significance tests when comparing the iterative mixture splitting version of ML and VB can be found in Table F.4 in Appendix F. This table shows results for each of the four training sets for some of the model sizes.

We also examined the free energy. Figure 6.3 shows the recognition performance and free energy for both k-means initialization, and iterative mixture splitting. In both cases the result has been averaged over the four training sets. We can see that the shape of the free energy plot is similar to the shape of the word accuracy plot in both cases. However, when using k-means the word accuracy is not as stable as when using iterative mixture

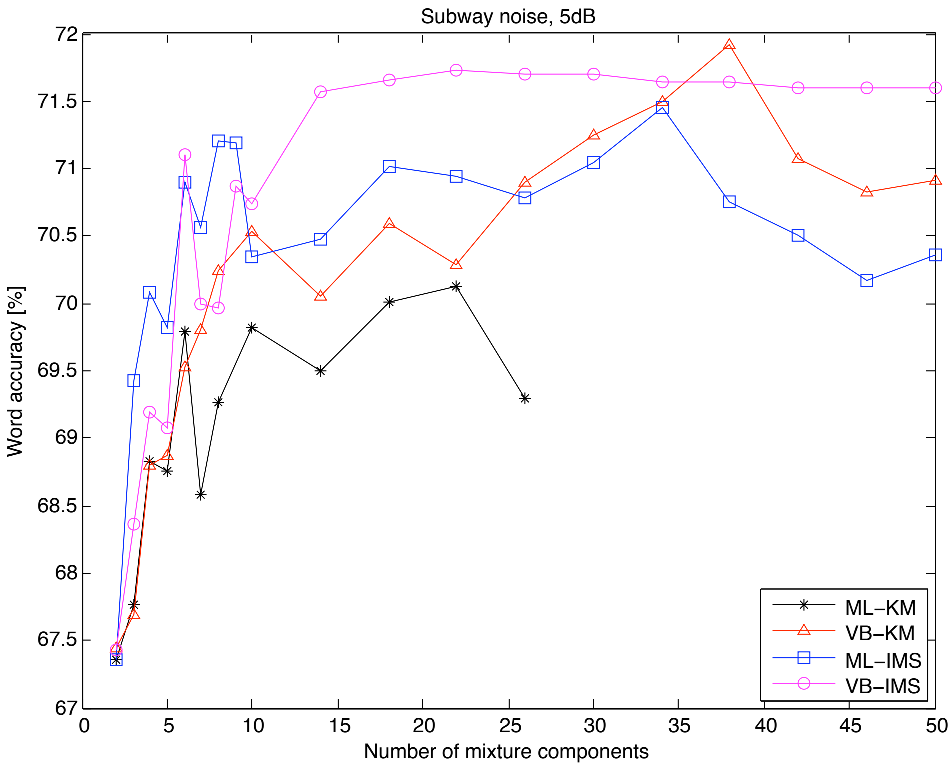

Figure 6.2: Feature enhancement in log-spectral domain with Algonquin. Recognition performance for ML and VB.

splitting, and this is not reflected in the free energy. When using iterative mixture splitting both the word accuracy and the free energy have stable areas above 14 mixture components.

6.6.2 Experiments on Other Noise Types

In Section 6.6.1 we ran experiments on subway noise at 5dB, and found that iterative mixture splitting worked better than k-means initialization. Moreover, the results indicated that VB had an advantage compared to ML. In this section we will give results for the remaining three noise types that are found in test set A of Aurora2: babble, car, and exhibition. Experiments are run with the same speech GMMs (those trained using iterative mixture splitting) and setup that was used in Section 6.6.1. The results are shown in Figure 6.4, Figure 6.5, and Figure 6.6. As can be seen in these plots, VB does not have the same performance advantage compared to ML for these

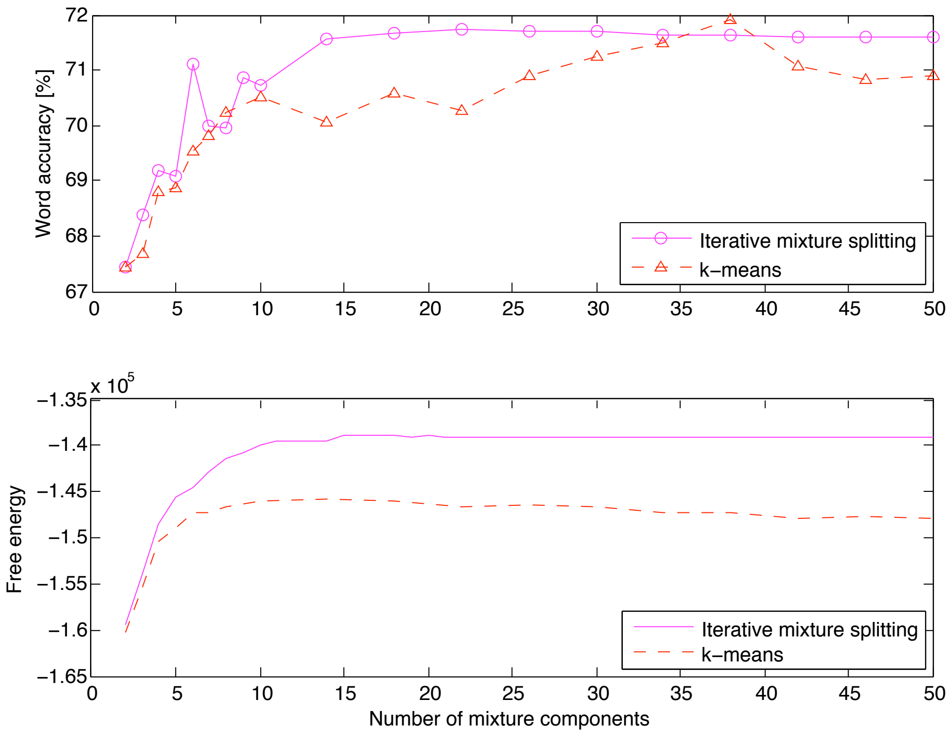

Figure 6.3: Feature enhancement in log-spectral domain with Algonquin. Recognition performance for VB and free energy

noise types. This result is unexpected, since the same speech GMMs are used in all cases. The performance on babble noise is almost equal for ML and VB, while on car noise ML performs slightly better. On exhibition, the performance is approximately the same for ML and VB.

Statistical significance tests comparing ML and VB for these three noise types can be found in Table F.5, Table F.6, and Table F.7 in Appendix F.

The VB performance enters a stable area at around 14 to 18 mixture components for all noise types, and it has a shape similar to the free energy which was shown in Figure 6.3. This means that the free energy could be used for model selection. However, in order to be sure to obtain a model in the flat area of the performance curve, it seems that one should select a model that is slightly larger than the point where the flat area of the free energy curve begins.

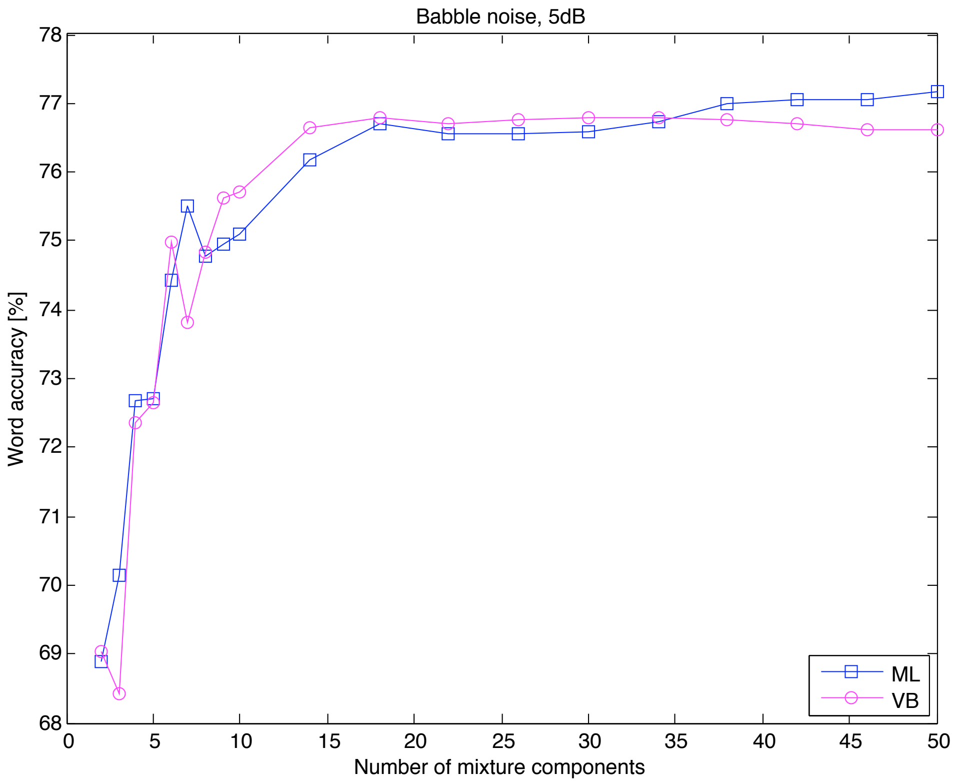

Figure 6.4: Feature enhancement in log-spectral domain with Algonquin. Recognition performance for ML and VB on babble noise at 5dB.

6.7 Experiments with MBFE in the Cepstral Domain

The setup for experiments with MBFE in the cepstral domain was essentially the same as the setup used in Section 6.6.1. Since there is much less correlation between neighboring feature vector elements in the cepstral domain than in the log-spectral domain, the GMMs were now trained with diagonal covariance matrices. The feature vectors were 13-dimensional MFCCs. Moreover, based on the results from the log-spectral domain, iterative mixture splitting was preferred over the k-means approach.

Note that going from Algonquin to MBFE is not just a change of domains. These two methods are different in the way they approach the problem of finding the MMSE estimate. The goal in Algonquin is to find the parameters of the joint clean speech and noise posterior distribution. Hav-

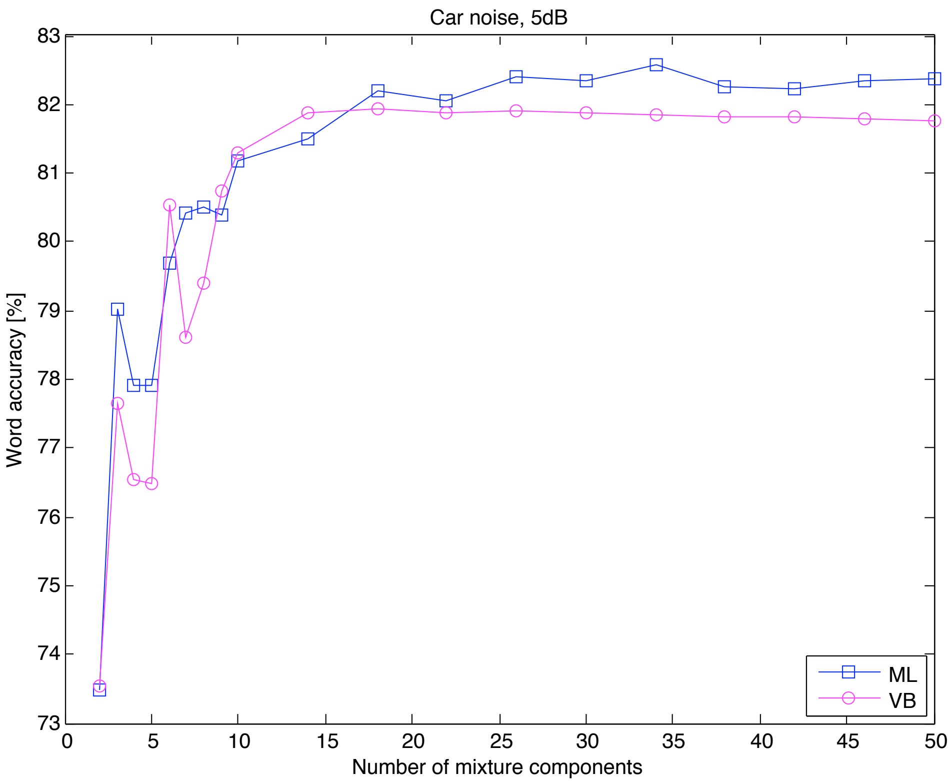

Figure 6.5: Feature enhancement in log-spectral domain with Algonquin. Recognition performance for ML and VB on car noise at 5dB.

ing found these parameters, the resulting distribution is used for finding the clean speech estimate. The approach of MBFE is to use the clean speech model and noise model to find a model for noisy speech. Then, based on the likelihoods of different components in the noisy speech model, a clean speech estimate is found. Thus, since Algonquin and MBFE use different approaches for finding the clean speech MMSE estimate, they can behave differently.

Plots comparing the performance of ML and VB are given in Figures 6.7-6.10. Based on the results from the log-spectral domain, the performance of VB was expected to be at least comparable to the performance of ML. Whereas this was the case for subway noise and car noise, the performance of ML was significantly better than VB for babble noise and exhibition noise. This result was quite unexpected. It is difficult to explain this behavior, but it seems that differences between the Algonquin and MBFE algorithms

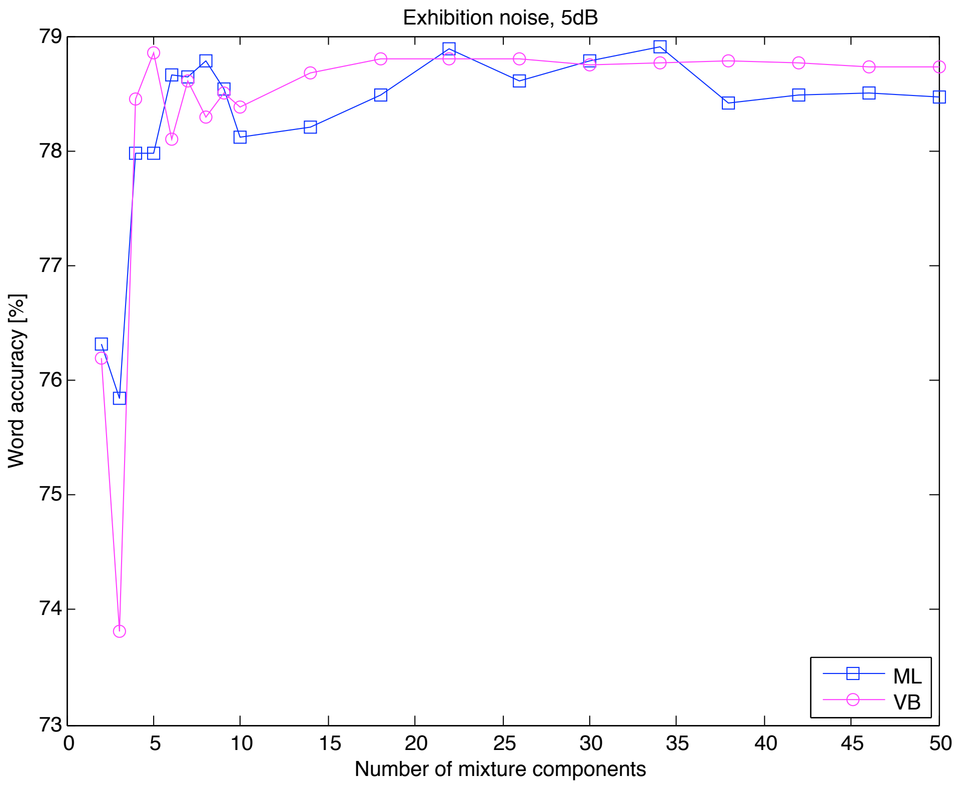

Figure 6.6: Feature enhancement in log-spectral domain with Algonquin. Recognition performance for ML and VB on exhibition noise at 5dB.

lead to different model preferences.

6.8 Conclusion

This chapter presented a study on the use VB trained front-end models in MMSE feature enhancement. Feature enhancement of noisy speech data was performed based on speech GMMs trained using both ML and VB learning, before running speech recognition on the cleaned data. Both iterative mixture splitting (IMS) and k-means initialization were investigated.

The first set of experiments was run in the log-spectral domain using Algonquin. When using k-means initialization, VB had advantages compared to ML in the sense that numerical problems were avoided, and the performance was better. However, IMS outperformed k-means. When using IMS for both ML and VB, the latter gave improved performance for sub-

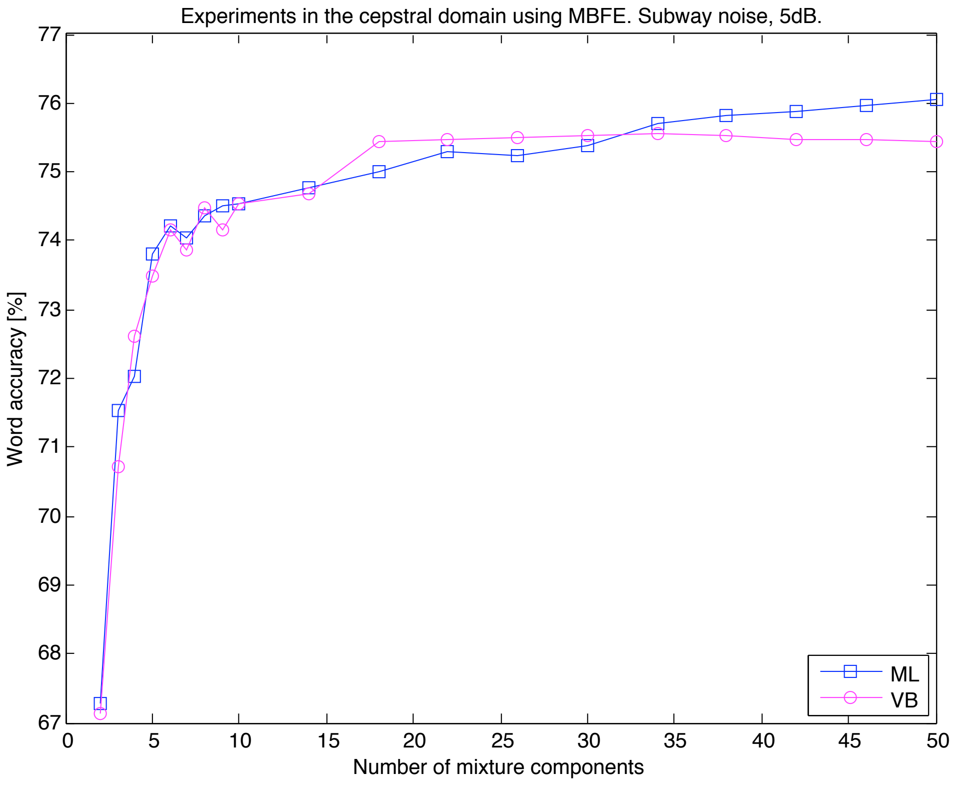

Figure 6.7: Feature enhancement in cepstral domain with MBFE. Recognition performance for ML and VB on subway noise at 5dB.

way noise, but for the other noise types the performance was approximately the same. Thus, the remaining advantage of VB is that the free energy can be used as a criterion for model selection. However, in this particular application, the practical usefulness of model selection is somewhat limited.

The second set of experiments, which was run in the cepstral domain using MBFE, gave very different and unexpected results. While VB and ML showed approximately the same performance on subway and car noise, ML significantly outperformed VB on babble and exhibition noise.

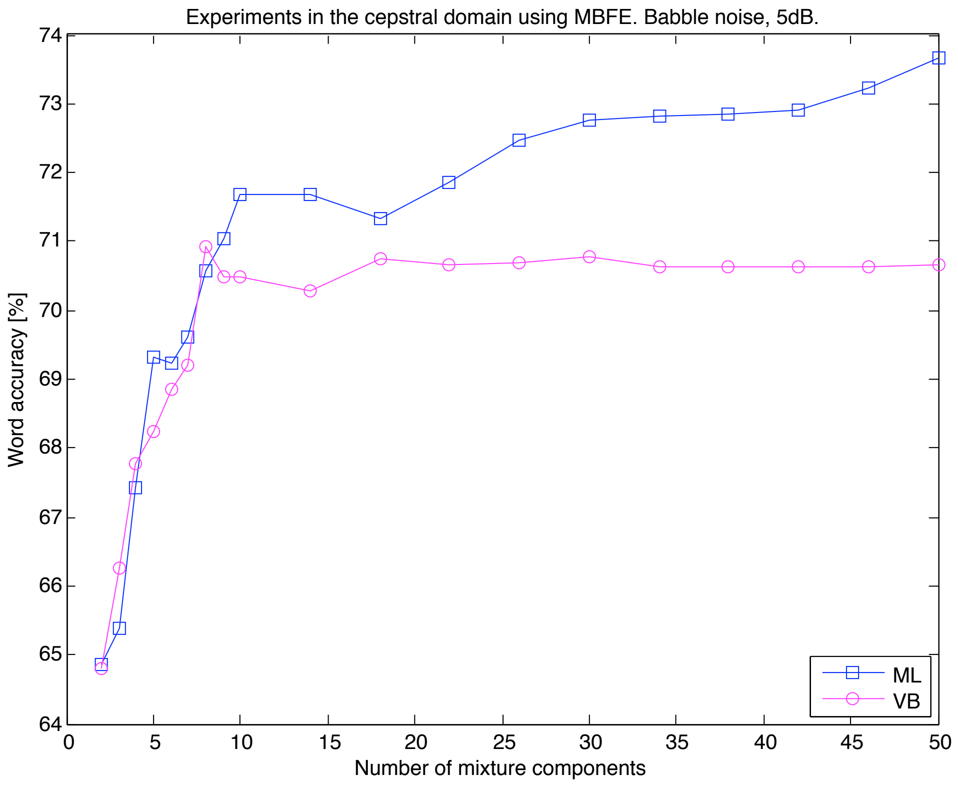

Figure 6.8: Feature enhancement in cepstral domain with MBFE. Recognition performance for ML and VB on babble noise at 5dB.

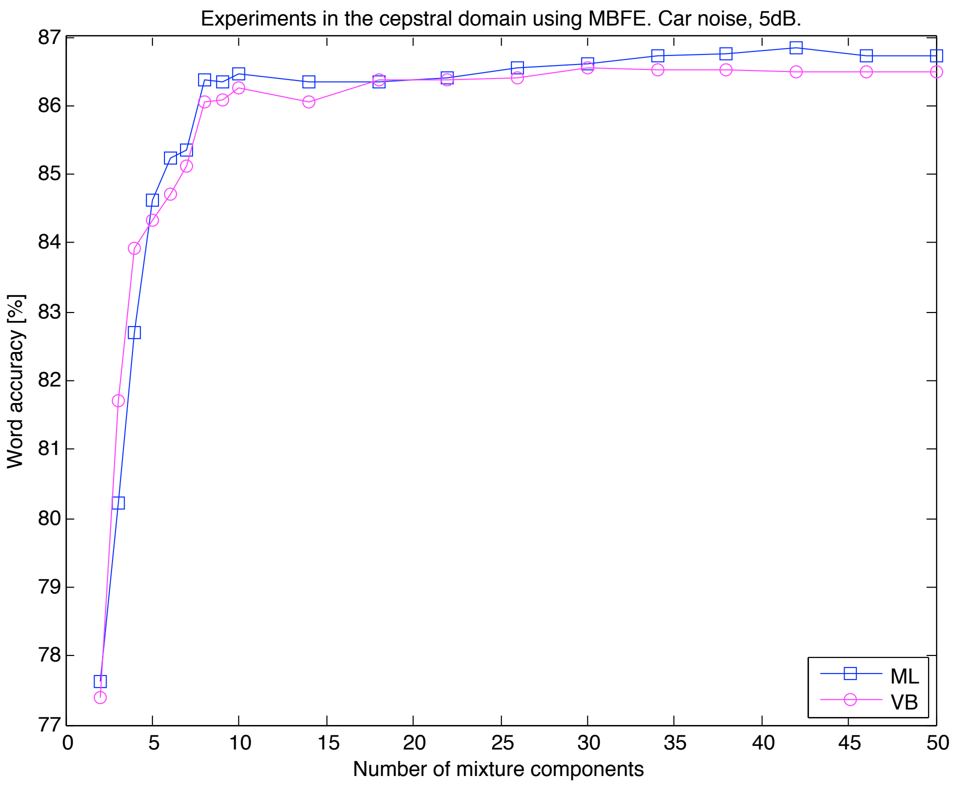

Figure 6.9: Feature enhancement in cepstral domain with MBFE. Recognition performance for ML and VB on car noise at 5dB.

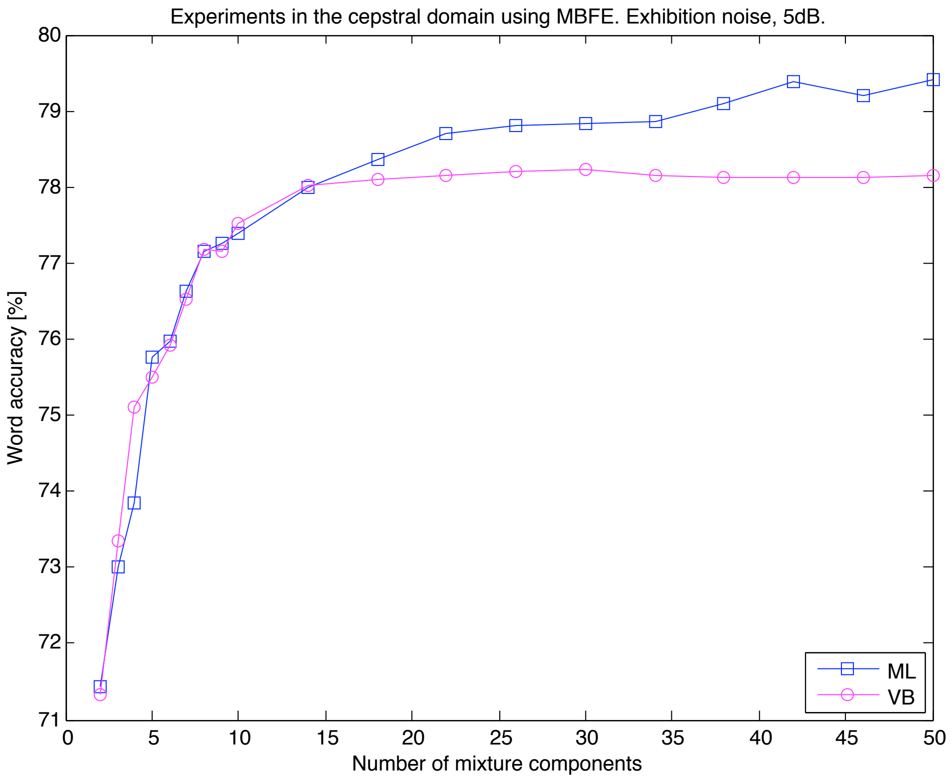

Figure 6.10: Feature enhancement in cepstral domain with MBFE. Recognition performance for ML and VB on exhibition noise at 5dB.

Chapter 7

Improved Noise Modeling for MMSE Feature Enhancement Using Voice Activity Detection

The main difficulty in noise model estimation is that prior information about the noise is usually not available. Consequently, noise data have to be extracted from the current noisy speech utterance. In Section 3.5 we gave a brief overview of different techniques for noise estimation. In an ideal situation, we would like to be able to track how the noise varies from frame to frame. In practice, however, it is very difficult to obtain reliable estimates for every frame. In this chapter we take a simpler approach, by using voice activity detection (VAD) for extracting noise-only frames. The extracted frames can then be used for improving noise models used for MMSE filtering of noisy speech.

Due to the popularity of MFCC features for speech recognition, it is useful to have VAD methods and MMSE filtering algorithms that both work in the MFCC domain. We propose a method for VAD based on the likelihood ratio test (LRT) that works directly on MFCC feature vectors. Detected noise-only frames are collected and used for creating a noise model, which is then used for MMSE filtering. We will also consider a well-known LRT-based VAD algorithm that works in the DFT domain, and compare its performance to the results of our method. Parts of this work were also presented in [55].

7.1 System Description

We consider a system consisting of three stages:

1. Voice activity detection to identify noise-only frames
2. MMSE filtering of noisy speech using the noise-only frames obtained in step 1 for noise modeling
3. Speech recognition based on the MMSE-filtered feature vectors from step 2 using recognizer models trained in clean conditions

A block diagram of the system is shown in Figure 7.1. Performing all steps (after feature extraction) in the MFCC domain has the advantages that no feature vector transformations are necessary between each stage, and we do not need access to features in the frequency domain as is required by many VAD and noise tracking algorithms. The VAD in stage 1 is based on the likelihood ratio test, and takes advantage of prior knowledge about speech in the form of a GMM in the same way as the MMSE filtering in stage 2. Note that the VAD method in stage 1 needs an initial crude estimate of the noise parameters. In practice this can be obtained by using an initial silence detector to extract a few noise-only frames.

The MMSE filtering approach used in this chapter is the GMM-based version of model-based feature enhancement (MBFE) [9, 66] that was described at the end of Section 3.3.3.

Note that our system extracts noise-only frames from the whole utterance before estimating a single set of noise model parameters. We could also have tried an online approach by updating the noise model for each noise-only frame we find. However, the findings of Myrvoll and Nakamura in [52] suggest that there is not a lot to gain from this approach. Consequently, we have chosen the simpler alternative, which also makes our system less computationally demanding since we avoid having to update the noisy speech GMM used by MBFE.

7.2 Cepstral Domain VAD Based on the LRT

Voice activity detection is a binary classification problem, consisting of the classes

$$\begin{aligned}\Omega_0 &: \text{speech absent}\\ \Omega_1 &: \text{speech present.}\end{aligned}$$

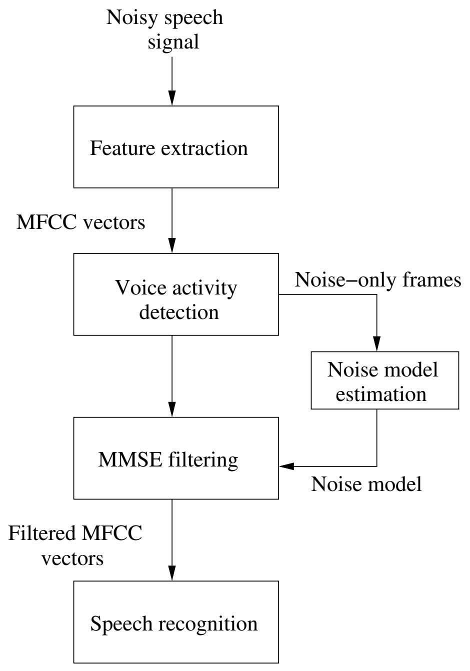

Figure 7.1: Block diagram of the system.

Given an observation $\mathbf{y}$ this problem can be formulated as a statistical hypothesis test where the null hypothesis (H_0) is that $\mathbf{y}$ belongs to Ω_0 and the alternative hypothesis (H_1) is that $\mathbf{y}$ belongs to Ω_1. According to the Neyman-Pearson Lemma, an optimal hypothesis test is obtained by using the *likelihood ratio test.* The likelihood ratio is given by

$$\mathrm{LR}(\mathbf{y}) = \frac{L(\mathbf{y}|H_0)}{L(\mathbf{y}|H_1)} = \frac{p(\mathbf{y}|\Omega_0)}{p(\mathbf{y}|\Omega_1)} \tag{7.1}$$

where $L(\mathbf{y}|\cdot)$ denotes the likelihood of the given hypothesis and $p(\mathbf{y}|\cdot)$ denotes the probability density function of an observation from the given class. The likelihood ratio test is performed by comparing the likelihood ratio to a threshold τ, and making a decision, i.e.,

$$\begin{aligned}\text{Accept } H_0 \text{ if } \mathrm{LR}(\mathbf{y}) \geq \tau\\ \text{Reject } H_0 \text{ if } \mathrm{LR}(\mathbf{y}) < \tau.\end{aligned}$$

When using this test for VAD, $\mathbf{y}$ is a frame from a noisy speech signal. Thus, the frame will be classified as only noise if $\mathrm{LR}(\mathbf{y}) \geq \tau$, and as a mixture of noise and speech if $\mathrm{LR}(\mathbf{y}) < \tau$.

In practice, the LRT is usually performed in the log domain by defining a log likelihood ratio, i.e.,

$$\mathrm{LLR}(\mathbf{y}) = \log \frac{p(\mathbf{y}|\Omega_0)}{p(\mathbf{y}|\Omega_1)} \tag{7.2}$$

$$= \log p(\mathbf{y}|\Omega_0) - \log p(\mathbf{y}|\Omega_1). \tag{7.3}$$

It has been shown [60] that for VAD it can be beneficial to use an LRT that takes into account several consecutive observations. More specifically, for a frame l, this is done by summing up the log likelihood ratios for the current frame and for m frames in both directions. Thus, for frame l we evaluate the sum of LLRs on the set $\mathbf{Y}_{l-m}^{l+m} = (\mathbf{y}_{l-m}, \dots, \mathbf{y}_l, \dots, \mathbf{y}_{l+m})$, and get the following:

$$\begin{aligned}\mathrm{LLR}_m(\mathbf{Y}_{l-m}^{l+m}) &= \sum_{k=l-m}^{l+m} \log \frac{p(\mathbf{y}_k|\Omega_0)}{p(\mathbf{y}_k|\Omega_1)} \\ &= \sum_{k=l-m}^{l+m} \log p(\mathbf{y}_k|\Omega_0) - \sum_{k=l-m}^{l+m} \log p(\mathbf{y}_k|\Omega_1).\end{aligned} \tag{7.4}$$

In order to use this test, models for noisy speech and noise are needed. Thus, we will again make use of the first N frames of each utterance, which are assumed to consist only of noise.[1] These frames are used to estimate an initial (single component) Gaussian noise model with mean $\boldsymbol{\mu}_\mathbf{n}$ and covariance matrix $\boldsymbol{\Sigma}_\mathbf{n}$. In addition, we make use of a GMM for clean speech. In our system, this clean speech GMM is the same model that is used for MMSE filtering in the next stage of processing. As was described in Section 3.2.2, we can determine the statistical parameters for noisy speech by applying VTS and using the known statistical parameters of clean speech and noise. Thus, using first-order VTS, the mean and covariance matrix of mixture component j of the noisy speech GMM are given by[2]

$$\boldsymbol{\mu}_{\mathbf{y},j} = \boldsymbol{\mu}_{\mathbf{s},j} + \mathbf{C}\log[\mathbf{1} + \exp(\mathbf{C}^{-1}(\boldsymbol{\mu}_\mathbf{n} - \boldsymbol{\mu}_{\mathbf{s},j}))] \tag{7.5}$$

$$\boldsymbol{\Sigma}_{\mathbf{y},j} = \mathbf{F}_j \boldsymbol{\Sigma}_{\mathbf{s},j} \mathbf{F}_j^T + \mathbf{G}_j \boldsymbol{\Sigma}_\mathbf{n} \mathbf{G}_j^T \tag{7.6}$$

[1]Note that LRT-based VAD methods working in the DFT domain usually assume a known initial noise variance, which in practice means that one needs some data to estimate it from. Hence, these methods also require some initial noise-only frames.

[2]Note that this is the same method as is used in the GMM-based version of MBFE when only a single component noise model is used.

where

$$\mathbf{F}_j = \mathbf{C}\ \operatorname{diag}\left(\frac{\mathbf{1}}{\mathbf{1} + \exp(\mathbf{C}^{-1}(\boldsymbol{\mu}_{\mathbf{n}} - \boldsymbol{\mu}_{\mathbf{s},j}))}\right)\mathbf{C}^{-1} \tag{7.7}$$

$$\mathbf{G}_j = \mathbf{I} - \mathbf{F}_j. \tag{7.8}$$

Then, using these models we can write the resulting LLR as

$$\mathrm{LLR}_m(\mathbf{Y}_{l-m}^{l+m}) = \sum_{k=l-m}^{l+m} \log \frac{\mathcal{N}(\mathbf{y}_k; \boldsymbol{\mu}_{\mathbf{n}}, \boldsymbol{\Sigma}_{\mathbf{n}})}{\sum_{j=1}^{J} w_j \mathcal{N}(\mathbf{y}_k; \boldsymbol{\mu}_{\mathbf{y},j}, \boldsymbol{\Sigma}_{\mathbf{y},j})}. \tag{7.9}$$

7.3 VAD in the DFT Domain Based on the LRT

In the experimental section we will compare our VAD method to a well known DFT-based approach proposed by Sohn *et al.* in [65]. In this section we will briefly review this method.

Let $\tilde{\mathbf{s}}$, $\tilde{\mathbf{n}}$, and $\tilde{\mathbf{y}}$ denote the DFTs of clean speech, noise, and noisy speech respectively. Moreover, let $\tilde{S}[k]$, $\tilde{N}[k]$, and $\tilde{Y}[k]$ denote the kth elements of the respective DFTs. Sohn *et al.*'s method is based on a statistical model where the DFT coefficients are assumed to be asymptotically independent complex Gaussian random variables [19]. Then, the probability densities of the classes Ω_0 and Ω_1 are given by

$$p(\tilde{\mathbf{y}}|\Omega_0) = \prod_{k=1}^{K} \frac{1}{\pi\lambda_N(k)} \exp\left\{\frac{-|\tilde{Y}[k]|^2}{\lambda_N(k)}\right\} \tag{7.10}$$

$$p(\tilde{\mathbf{y}}|\Omega_1) = \prod_{k=1}^{K} \frac{1}{\pi[\lambda_N(k) + \lambda_S(k)]} \exp\left\{\frac{-|\tilde{Y}[k]|^2}{\lambda_N(k) + \lambda_S(k)}\right\}, \tag{7.11}$$

where $\lambda_N(k)$ and $\lambda_S(k)$ denote the variances of $\tilde{N}[k]$ and $\tilde{S}[k]$ respectively, and K is the dimension of the DFT vectors. Note that since this method operates in the DFT domain, it avoids having to deal with the non-linear relationship in (3.16). For the kth DFT coefficient, the likelihood ratio can be written as[3]

$$\Psi_k = \frac{p(\tilde{Y}[k]|\Omega_0)}{p(\tilde{Y}[k]|\Omega_1)} = (1 + \xi_k) \exp\left\{-\frac{\gamma_k \xi_k}{1 + \xi_k}\right\}, \tag{7.12}$$

[3]Note that we have defined this LR as the inverse of the LR used in [65] in order to match our definition in (7.1).

where $\xi_k = \lambda_S(k)/\lambda_N(k)$ and $\gamma_k = |\tilde{Y}[k]|^2/\lambda_N(k)$ are the so-called *a priori* and *a posteriori* signal to noise ratios [19]. The LLR for one frame is then calculated as the geometric mean of LLRs for each DFT coefficient, i.e.,

$$\log \Psi = \frac{1}{K} \sum_{k=1}^{K} \log \Psi_k. \tag{7.13}$$

In order to avoid clipping of weak speech tails, Sohn *et al.*'s method uses a HMM-based hang-over scheme. For more details on this see [65].

7.4 Experiments and Results

To evaluate the proposed approach, we run speech recognition experiments on set A of the Aurora2 database. The recognition performance when using VAD for improved noise modeling will be compared to the recognition performance when only using the first 20 frames for noise modeling. We will also compare the effectiveness of our VAD approach to Sohn *et al.*'s method, which was briefly reviewed in Section 7.3. Note that this method works in the DFT domain and hence does not fit directly into the block diagram in Figure 7.1. In addition, we will compare the results to an approximate upper bound for the VAD method, which has been generated by running forced alignment on the clean speech data using the clean speech recognizer models. This method gives a close to optimal speech activity labeling.

The proposed algorithm was tested using a clean speech GMM with 32 mixture components. The value of m in (7.4) was set to 4. For both the proposed approach and Sohn *et al.*'s method, we have optimized the thresholds to give the best performance when averaging over all noise types, and SNRs from 0 to 20dB.

During VAD and MMSE filtering, 13-dimensional MFCC feature vectors are used, with C_0 as energy. During recognition the feature vectors are 39-dimensional, consisting of static, delta and acceleration coefficients.

Now, we compare the results to only using the first 20 frames and Sohn *et al.*'s method, as well as the close to optimal labeling obtained from forced alignment (FrcAlgn). The results are given in Table 7.1, where the proposed approach has been given the label LRT-CEP.

The proposed LRT-CEP approach outperforms the use of only the first 20 frames for noise modeling. When comparing our VAD method to Sohn *et al.*'s method, we can see that the proposed approach on average performs better on babble and exhibition noise. On the other hand, Sohn *et al.*'s method performs better on subway and car noise. When averaging over all SNRs and noise types, the performance of these two methods is very similar.

Table 7.1: Word accuracies (%) on Aurora2

Noise	SNR	First20	Sohn	LRT-CEP	FrcAlgn
subway	20	96.75	97.05	96.90	97.91
	15	93.89	94.75	94.72	96.22
	10	86.34	89.07	89.41	92.63
	5	75.50	80.44	80.32	85.82
	0	56.16	61.01	59.87	67.92
	-5	28.09	29.66	27.51	34.73
babble	20	96.61	97.19	97.49	98.25
	15	93.74	94.74	95.77	97.58
	10	85.97	88.97	90.81	94.47
	5	72.52	77.75	79.78	84.73
	0	44.53	48.64	46.49	54.20
	-5	14.93	17.26	15.24	21.16
car	20	98.33	98.51	98.57	98.60
	15	97.44	97.88	97.70	98.03
	10	94.78	95.76	95.17	95.76
	5	87.21	88.67	87.92	89.20
	0	66.84	64.06	60.01	65.85
	-5	28.57	24.46	21.12	25.56
exhib.	20	97.69	97.81	97.72	98.21
	15	95.00	95.80	95.96	96.54
	10	90.25	91.02	91.98	92.69
	5	79.91	82.17	83.52	84.60
	0	61.59	65.17	66.52	66.65
	-5	35.51	37.15	36.19	38.11
Avg. (0–20dB)		83.55	85.32	85.33	87.79

Statistical significance tests comparing First20, Sohn *et al.*'s method, and the proposed approach can be found in Table F.8 and Table F.9 in Appendix F.

The improvement of the proposed approach versus only the first 20 frames is 1.78% absolute, which corresponds to a reduction in word error rate of 10.8%. The gap in recognition accuracy between using only the first 20 frames and the forced alignment is 4.24% absolute. This means that we are able to close about 40% of the gap.

Tables showing the relative amounts of frames that were classified as noise by the VAD methods can be found in Appendix E. The tables also show the relative amounts of frames (FrcAlgn) that were classified as silence by the forced alignment upper bound. Note that as the SNR decreases, speech sounds with low energy may be masked by noise. Thus, it may be beneficial to include more frames for noise modelling than those detected by forced alignment on clean speech data. The amount of frames selected by a given method depends on how we set the threshold, and thus the results in Appendix E represent a compromise of selected frames when optimizing thresholds for the maximum word accuracy averaged over all noise types and SNRs (0–20dB).

From the tables in Appendix E, it can be observed that the proposed approach generally classifies a larger amount of frames as noise than Sohn *et al.*'s method. For babble noise and exhibition noise, Sohn *et al.*'s method always selects fewer frames for noise modelling than FrcAlgn. These are also the noise types where the proposed approach works better than Sohn *et al.*'s method for SNRs higher than 0dB. The proposed approach selects more frames than FrcAlgn in almost all cases. At 0dB and -5dB, the number of selected frames is probably too high in most cases, and thus Sohn *et al.*'s method performs better.

As a final comment, we note that the results presented in this chapter do not match the results reported in [66]. This is probably due to the more complex system setup that was used in [66], which included convolutional noise removal and more mixture components in both front-end and back-end.

7.5 Conclusion

This chapter investigated the use of voice activity detection for obtaining improved noise models for use in MMSE feature enhancement. An LRT-based method for VAD working directly on MFCC features was proposed. The VAD method was included in a system where, in addition to VAD, both MMSE filtering and speech recognition was performed in the MFCC domain. Noise modeling based on the proposed VAD algorithm was shown to give improved recognition performance compared to simply using the assumed speech-free regions from the beginning of each utterance. In addition, the VAD approach gave performance comparable to that of Sohn *et al.*'s LRT-based approach working in the discrete Fourier transform domain.

Chapter 8

Conclusions

In the following, we will sum up the most important conclusions that we have drawn in this thesis. Note that since the experiments have been limited to the Aurora2 task, the conclusions do not necessarily generalize to all databases, noise types, and recognizer setups. From the previous four chapters, the most important conclusions we have drawn are:

- For model compensation, more accurate approximations of the non-linear relationship between speech, noise, and noisy speech do not give significant performance gains. The results suggest that the traditional PMC approximation is good enough in practice.

- The joint BPC-PMC approach, which combines model compensation with a robust decision rule, does improve performance at low SNR for several noise types. Since this method involves an increase in variance, prior scaling is needed when the background noise consists mainly of speech.

- Although variational Bayesian learning of the front-end speech model for Algonquin feature enhancement gave some promising results on subway noise, the results on other noise types and with another feature enhancement algorithm showed that the method did not generalize. In most cases traditional maximum likelihood learning was equal to or better than variational Bayesian learning.

- Voice activity detection (VAD) was shown to be an effective way of obtaining additional noise data, and consequently better noise models, for MMSE feature enhancement. The proposed cepstrum domain approach based on the likelihood ratio test gave a reduction in word error rate of 10.8% compared to using the first 20 frames of each utterance for noise modeling. In addition, the average performance was similar to Sohn *et al.*'s well-known approach working in the DFT domain.

8.1 Future Work

In the following we will give some suggestions for directions of future research in this area.

The comparative study of model compensation approximations in Chapter 4 was performed using only static feature vector coefficients. This was done in order to investigate the basic influence of different approximations without any "disturbing" factors. In practice, however, compensation of delta and delta-delta coefficients is very important for the ASR performance. Thus, a study on different approximations including delta compensation schemes would be of interest.

The joint BPC-PMC approach proposed in Chapter 5 was based on the idea of compensating for uncertainty in the parameter estimates of the compensated models. This was done by applying BPC with priors that had mean values equal to the PMC noisy speech mean, and a variance determined by neighborhoods that were originally proposed for the minimax decision rule. A limitation of this approach is that, for a given feature vector element, the prior variance is the same of all HMM states and mixture components. It would be interesting to consider methods for obtaining individual prior variances for each state and mixture component. Further work is needed to determine how to approach this problem.

The variational Bayesian learning for front-end GMMs in MMSE feature enhancement only gave improvements in a few cases in our experiments in Chapter 6. One issue that could be investigated further is the setting of the prior distribution. Whereas we used a non-informative prior, it would be interesting to make it more dependent on the data. An interesting approach in this respect is that of Constantinopoulos and Likas in [8]. In their method the model selection problem is treated locally, in a region of the feature space, in order to set more informative priors.

The results obtained in Chapter 7 using VAD for improved noise modeling in MMSE feature enhancement were promising. Noise frames were extracted from an entire utterance before using the result to estimate the noise model. It is also possible to apply the VAD in an online approach, where the noise models of both the VAD algorithm and MMSE filtering algorithm are updated each time a noise frame is found. It would be interesting to see how the results of this approach would compare to that presented in Chapter 7. It would also be very useful to do a comparison of VAD-based approaches and noise tracking algorithms in the context of MMSE filtering of noisy speech.

Appendix A

The Aurora2 Database

The experiments described in this thesis have been performed on the Aurora2 database [32]. In this chapter we give a short description of this database.

The Aurora2 database has been designed to evaluate the performance of algorithms for robust speech recognition in noisy conditions. It was released by the Aurora working group, which is a part of the European Telecommunication Standards Institute (ETSI) under the technical committee for Speech processing, Transmission planning, and Quality service aspects (STQ).

In the Aurora2 database, noisy speech was created artificially by adding noise to the clean speech recordings of the TIDigits database [45]. The TIDigits database contains connected digit strings spoken in American English. The recordings that were used to create Aurora2 consist of sequences of up to seven digits. The original 20kHz recordings were downsampled to 8kHz before distortions were added. Moreover, in order to consider realistic frequency characteristics of terminals and equipment in the telecommunications area, an additional filtering was applied. Two "standard" frequency characteristics which have been defined by the International Telecommunication Union (ITU) were used. These were created by filtering with the so-called G.712 and MIRS filters (see [32] for plots and details).

Recordings of noise were added to the clean speech recordings at SNRs of 20dB, 15dB, 10dB, 5dB, 0dB, and -5dB. To be able to add noise at a given level, one first has to define the term SNR. Since this is dependent on the selected frequency range, the SNR was defined as the ratio of signal to noise energy after filtering both signals with the G.712 characteristic.

The speech energy was determined by applying the ITU recommendation P.56 and using the corresponding ITU software. The noise energy was calculated as root mean square (RMS) of the noise segment. For each utterance a randomly chosen noise segment of the same length as the clean speech recording is cut out from the noise recording. Noise recordings have been collected from different places.

- Suburban train
- Crowd of people (babble)
- Car
- Exhibition hall
- Restaurant
- Street
- Airport
- Train station

Some of these noise recordings are fairly stationary (e.g. car and exhibition hall), while others contain non-stationary segments (e.g. street and airport).

Two training modes have been defined for the Aurora2 database. One is training on clean speech data, while the other is training on clean and noisy data. The latter is referred to as multi-condition training. In this thesis we focus on the use of models trained in clean conditions. For clean training, 8440 utterances are selected from the training set of TIDigits, containing recordings of 55 male and 55 female adults. These recordings are filtered with the G.712 characteristic.

Three test sets are defined, using a total of 4004 utterances from 52 male and 52 female speakers in the TIDigits test set. These utterances are split into 4 subsets with 1001 utterances in each. All speakers are present in each subset. Each noise signal is added to one of these subsets at different SNRs. The clean files are included as a seventh condition. Both speech and noise is filtered with the G.712 characteristic before adding.

The first test set, called *test set A*, contains the following noise types: suburban train (subway), babble, car, and exhibition. In total, this set consists of $4 \cdot 7 \cdot 1001 = 28028$ utterances. The second test set, called *test set B*, is created in the same way as test set A, but contains different noise types. These are restaurant, street, airport, and train station. There is also a third set, which contains only 2 of the 4 subsets with 1001 utterances in each. This set is called *test set C*. On this set speech and noise are filtered with a MIRS characteristic before adding noise at different SNRs. The noise types used for this set are subway and street. The purpose of this set is to show the influence on recognition performance from mismatch in frequency characteristics.

The reference recognizer is based on the Hidden Markov Model Toolkit (HTK) [77]. This recognizer uses one whole-word HMM for each of 11 English digits, including “oh”. Each HMM has 16 states with a simple left-to-right model structure without skips, and each state uses a three-component GMM with diagonal covariance matrices. There is also a silence model, as well as a model for short pause. The silence model has three states with six mixture components in each state, while the model for short pause only has a single state which is tied to the middle state of the silence model. Note that some research papers use a more complex version of this back-end recognizer, which has more mixture component per state. In this thesis, however, we use the simple version of the back-end recognizer.

The feature vectors used by the reference recognizer are MFCCs with a dimension of 39. This includes 12 cepstral coefficients (without C_0) and logarithmic frame energy plus the corresponding delta and acceleration coefficients. Note that in this thesis we replace the log energy by C_0, since most the algorithms we use require knowledge about C_0. When calculating MFCC vectors the frame length is set to 25ms and the frame shift is set to 10ms. The number of bands in the mel-scale filter bank is 23.

Appendix B

Functions and Probability Distributions

B.1 The Digamma Function

The digamma function is defined as the logarithmic derivative of the gamma function $\Gamma(x)$, i.e.,

$$\psi(x) = \frac{d}{dx}\log\Gamma(x) = \frac{1}{\Gamma(x)}\frac{d}{dx}\Gamma(x). \tag{B.1}$$

By substituting an integral representation for the derivative of the gamma function, it can be written as

$$\psi(x) = \frac{1}{\Gamma(x)}\int\limits_0^\infty t^{x-1}e^{-t}\log t\,dt. \tag{B.2}$$

B.2 The Dirichlet Distribution

Suppose we have M random variables $\{Y_1,\ldots,Y_M\}$ distributed according to a Dirichlet distribution of order M. Given the parameters $\lambda = \{\lambda_1,\ldots,\lambda_M\}$, where $\lambda_i > 0$ for all i, the Dirichlet distribution has a pdf given by

$$p(y_1,\ldots,y_M) = \mathcal{D}(\lambda) = \frac{\Gamma\left(\sum_{j=1}^M \lambda_j\right)}{\prod_{j=1}^M \Gamma(\lambda_j)}\prod_{i=1}^M y_i^{\lambda_i-1} \tag{B.3}$$

for all $y_1,\ldots,y_M \geq 0$ such that $\sum_{i=1}^M y_i = 1$.

Defining $\tilde{\lambda} = \sum_{j=1}^{M} \lambda_j$, the mean value of Y_i is given by

$$\mathrm{E}[Y_i] = \frac{\lambda_i}{\tilde{\lambda}}, \tag{B.4}$$

and the variance is given by

$$\mathrm{Var}[Y_i] = \frac{\lambda_i(\tilde{\lambda} - \lambda_i)}{\tilde{\lambda}^2(\tilde{\lambda} + 1)}. \tag{B.5}$$

B.3 The Wishart Distribution

Suppose we have a matrix $\mathbf{A}$ of dimension $D \times D$ that has a Wishart distribution. Then, the pdf is given by

$$p(\mathbf{A}) = \mathcal{W}(\nu, \mathbf{\Phi}) = \frac{|\mathbf{A}|^{(\nu-D-1)/2}}{2^{\nu D/2}|\mathbf{\Phi}|^{-\nu/2}\Gamma_D(\nu/2)} \exp\left[-\frac{1}{2}\,\mathrm{tr}(\mathbf{A}\mathbf{\Phi})\right] \tag{B.6}$$

where $\mathbf{\Phi}$ is a positive definite matrix of size $D \times D$, and $\nu \geq D$ denotes degrees of freedom. The function $\Gamma_D(\nu/2)$ is the multivariate gamma function given by

$$\Gamma_D(\nu/2) = \pi^{D(D-1)/4} \prod_{j=1}^{D} \Gamma\left(\frac{\nu + 1 - j}{2}\right). \tag{B.7}$$

The mean of $\mathbf{A}$ is given by

$$\mathrm{E}[\mathbf{A}] = \nu \mathbf{\Phi}^{-1}. \tag{B.8}$$

B.4 The Gamma Distribution

Suppose we have a random variable Y that has a gamma distribution. Then, the pdf is given by

$$p(y) = \mathcal{G}(\nu, \phi) = \frac{(\frac{\phi}{2})^{\nu/2}}{\Gamma(\nu/2)} y^{\nu/2-1} e^{-\frac{\phi}{2}y} \tag{B.9}$$

for $y > 0$.

The mean value of Y is given by

$$\mathrm{E}[Y] = \frac{\nu}{\phi}, \tag{B.10}$$

and the variance is given by

$$\mathrm{Var}[Y] = \frac{2\nu}{\phi^2}. \tag{B.11}$$

B.5 The Multivariate t-Distribution

Suppose we have a random vector $\mathbf{y}$ of dimension D that has a multivariate t-distribution. Then, the pdf is given by

$$p(\mathbf{y}) = t_\kappa(\boldsymbol{\rho}, \boldsymbol{\Omega}) = \frac{\Gamma\left(\frac{\kappa+D}{2}\right)}{\Gamma\left(\frac{\kappa}{2}\right)(\pi\kappa)^{D/2}|\boldsymbol{\Omega}|^{1/2}\left[1+\frac{1}{\kappa}(\mathbf{y}-\boldsymbol{\rho})^T\boldsymbol{\Omega}^{-1}(\mathbf{y}-\boldsymbol{\rho})\right]^{\frac{\kappa+D}{2}}}, \tag{B.12}$$

where $\boldsymbol{\rho}$ is the mean, $\boldsymbol{\Omega}$ is the covariance matrix, and κ denotes degrees of freedom.

Appendix C

Derivation of VB Learning for the GMM

In this chapter we will show how to derive the VB learning algorithm which was described in Section 6.3. As a starting point for the derivation, we will use a general formulation of the VB learning algorithm for conjugate-exponential models which was given by Beal in [5].

C.1 General Formulation of VB for Conjugate-Exponential Models

As in Section 6.2, we let $\mathbf{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$ denote an observed dataset consisting of N independent and identically distributed items, $\mathbf{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_N\}$ denote hidden variables and $\boldsymbol{\theta}$ denote the parameters. For a model to be conjugate-exponential, it has to satisfy to following two conditions [5].

1. The complete-data likelihood must be in the exponential family, i.e.,

$$p(\mathbf{y}_n, \mathbf{z}_n|\boldsymbol{\theta}) = g(\boldsymbol{\theta}) f(\mathbf{y}_n, \mathbf{z}_n) \exp\left\{\boldsymbol{\phi}(\boldsymbol{\theta})^T \mathbf{u}(\mathbf{y}_n, \mathbf{z}_n)\right\} \tag{C.1}$$

   where $\boldsymbol{\phi}(\boldsymbol{\theta})$ is the vector of natural parameters, and $\mathbf{u}(\mathbf{y}_n, \mathbf{z}_n)$ is the vector of sufficient statistics.

2. The parameter prior must be conjugate to the complete-data likelihood, i.e.,

$$p(\boldsymbol{\theta}|\eta, \boldsymbol{\xi}) = h(\eta, \boldsymbol{\xi}) g(\boldsymbol{\theta})^\eta \exp\left\{\boldsymbol{\phi}(\boldsymbol{\theta})^T \boldsymbol{\xi}\right\} \tag{C.2}$$

   where η and $\boldsymbol{\xi}$ are hyperparameters of the prior.

For a model that satisfies these two conditions, the steps of the VB EM algorithm can be formulated as follows [5].

1. The VB E-step is given by

$$q(\mathbf{z}_n) \propto f(\mathbf{y}_n, \mathbf{z}_n) \exp\left\{\bar{\boldsymbol{\phi}}^T \mathbf{u}(\mathbf{y}_n, \mathbf{z}_n)\right\} \tag{C.3}$$

$$q(\mathbf{Z}) = \prod_{n=1}^{N} q(\mathbf{z}_n) \tag{C.4}$$

   where the natural parameter vector $\bar{\boldsymbol{\phi}}$ is given by

$$\bar{\boldsymbol{\phi}} = \mathrm{E}_{\boldsymbol{\theta}}[\boldsymbol{\phi}(\boldsymbol{\theta})]. \tag{C.5}$$

   Note that the factorization in (C.4) turns out to be optimal as a consequence of the assumption that the elements in $\mathbf{Y}$ are independent and identically distributed (see [5] for more details).

2. The VB M-step results in the conjugate posterior $q(\boldsymbol{\theta})$ with hyperparameters $\tilde{\eta}$ and $\tilde{\boldsymbol{\xi}}$, i.e.,

$$q(\boldsymbol{\theta}) = h(\tilde{\eta}, \tilde{\boldsymbol{\xi}}) g(\boldsymbol{\theta})^{\tilde{\eta}} \exp\left\{\boldsymbol{\phi}(\boldsymbol{\theta})^T \tilde{\boldsymbol{\xi}}\right\}. \tag{C.6}$$

   The new hyperparameters are obtained as

$$\tilde{\eta} = \eta + N \tag{C.7}$$

$$\tilde{\boldsymbol{\xi}} = \boldsymbol{\xi} + \sum_{n=1}^{N} \bar{\mathbf{u}}(\mathbf{y}_n) \tag{C.8}$$

   where

$$\bar{\mathbf{u}}(\mathbf{y}_n) = \mathrm{E}_{\mathbf{Z}}[\mathbf{u}(\mathbf{y}_n, \mathbf{z}_n)]. \tag{C.9}$$

C.2 The GMM as a Conjugate-Exponential Model

The GMM satisfies the two conditions given in Section C.1, and is therefore a conjugate-exponential model. Note that in the case of a GMM, the latent variables $\mathbf{z}_n$ are scalars, and thus denoted as z_n. In order to use the formulation from Section C.1 we need to identify the natural parameter $\boldsymbol{\phi}(\boldsymbol{\theta})$ of the GMM. The complete-data likelihood is given by

$$\begin{aligned} p(\mathbf{y}_n, z_n = z|\boldsymbol{\theta}) &= p(z_n = z|\boldsymbol{\theta}) p(\mathbf{y}_n | z_n = z, \boldsymbol{\theta}) \\ &= w_z \frac{|\boldsymbol{\Gamma}_z|^{1/2}}{(2\pi)^{D/2}} \exp\left\{-\frac{1}{2}(\mathbf{y}_n - \boldsymbol{\mu}_z)^T \boldsymbol{\Gamma}_z (\mathbf{y}_n - \boldsymbol{\mu}_z)\right\}. \end{aligned} \tag{C.10}$$

We will now rewrite this as a product over all mixture components by defining

$$\delta_j(z) = \begin{cases} 1 & \text{if } z = j \\ 0 & \text{otherwise.} \end{cases} \tag{C.11}$$

Then, we can write (C.10) as

$$p(\mathbf{y}_n, z_n|\boldsymbol{\theta}) = \frac{1}{(2\pi)^{\frac{D}{2}}} \prod_{j=1}^{M} \left(w_j |\boldsymbol{\Gamma}_j|^{\frac{1}{2}} \exp\left\{ -\frac{1}{2}(\mathbf{y}_n - \boldsymbol{\mu}_j)^T \boldsymbol{\Gamma}_j (\mathbf{y}_n - \boldsymbol{\mu}_j) \right\} \right)^{\delta_j(z_n)}. \tag{C.12}$$

Taking both exp and log of (C.12) yields

$$\begin{aligned} p(\mathbf{y}_n, z_n|\boldsymbol{\theta}) = \frac{1}{(2\pi)^{\frac{D}{2}}} \exp\Bigg\{ & \sum_{j=1}^{M} \delta_j(z_n) \log w_j + \frac{1}{2} \sum_{j=1}^{M} \delta_j(z_n) \log |\boldsymbol{\Gamma}_j| \\ & + \sum_{j=1}^{M} \delta_j(z_n) \left[-\frac{1}{2}(\mathbf{y}_n - \boldsymbol{\mu}_j)^T \boldsymbol{\Gamma}_j (\mathbf{y}_n - \boldsymbol{\mu}_j) \right] \Bigg\}. \end{aligned} \tag{C.13}$$

The expression inside the square brackets of the last term can be rewritten as follows:

$$-\frac{1}{2}(\mathbf{y}_n - \boldsymbol{\mu}_j)^T \boldsymbol{\Gamma}_j (\mathbf{y}_n - \boldsymbol{\mu}_j) = -\frac{1}{2} \operatorname{tr}(\boldsymbol{\Gamma}_j \mathbf{y}_n \mathbf{y}_n^T) + \boldsymbol{\mu}_j^T \boldsymbol{\Gamma}_j \mathbf{y}_n - \frac{1}{2} \boldsymbol{\mu}_j^T \boldsymbol{\Gamma}_j \boldsymbol{\mu}_j. \tag{C.14}$$

Thus, comparing (C.13) to (C.1), we get

$$g(\boldsymbol{\theta}) = \frac{1}{(2\pi)^{\frac{D}{2}}}, \tag{C.15}$$

and the natural parameter vector becomes

$$\boldsymbol{\phi}(\boldsymbol{\theta}) = \begin{bmatrix} \log w_1 \\ \vdots \\ \log w_M \\ \log|\boldsymbol{\Gamma}_1| \\ \vdots \\ \log|\boldsymbol{\Gamma}_M| \\ \operatorname{vec}(\boldsymbol{\Gamma}_1) \\ \boldsymbol{\Gamma}_1\boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_1^T\boldsymbol{\Gamma}_{\mathbf{1}}\boldsymbol{\mu}_1 \\ \vdots \\ \operatorname{vec}(\boldsymbol{\Gamma}_M) \\ \boldsymbol{\Gamma}_M\boldsymbol{\mu}_M \\ \boldsymbol{\mu}_M^T\boldsymbol{\Gamma}_M\boldsymbol{\mu}_M \end{bmatrix}. \tag{C.16}$$

The vector $\mathbf{u}(\mathbf{y}_n, z_n)$ is

$$\mathbf{u}(\mathbf{y}_n, z_n) = \begin{bmatrix} \delta_1(z_n) \\ \vdots \\ \delta_M(z_n) \\ \frac{1}{2}\delta_1(z_n) \\ \vdots \\ \frac{1}{2}\delta_M(z_n) \\ -\frac{1}{2}\delta_1(z_n)\operatorname{vec}(\mathbf{y}_n\mathbf{y}_n^T) \\ \delta_1(z_n)\mathbf{y}_n \\ -\frac{1}{2}\delta_1(z_n) \\ \vdots \\ -\frac{1}{2}\delta_M(z_n)\operatorname{vec}(\mathbf{y}_n\mathbf{y}_n^T) \\ \delta_M(z_n)\mathbf{y}_n \\ -\frac{1}{2}\delta_M(z_n) \end{bmatrix}. \tag{C.17}$$

Now we will consider the prior, which is given by

$$\begin{aligned} p(\boldsymbol{\theta}|\boldsymbol{\xi}) &= p(\{w_z\}) \prod_{j=1}^{M} p(\boldsymbol{\mu}_j|\boldsymbol{\Gamma}_j)p(\boldsymbol{\Gamma}_j) \\ &= \mathcal{D}(\{\lambda_{0z}\}) \prod_{j=1}^{M} \mathcal{N}(\boldsymbol{\rho}_0, \beta_0\boldsymbol{\Gamma}_j)\mathcal{W}(\nu_0, \boldsymbol{\Phi}_0). \end{aligned} \tag{C.18}$$

Note that in this case we do not have the η hyperparameter from (C.2). Also note that in this chapter, we will use subscript zeros instead of superscript zeros to indicate prior parameters. When ignoring constants in the prior, we have

$$\begin{aligned} p(\boldsymbol{\theta}|\boldsymbol{\xi}) \propto & \prod_{i=1}^{M} w_i^{\lambda_{0i}-1} \\ & \cdot \prod_{j=1}^{M} \Bigg(|\beta_0 \boldsymbol{\Gamma}_j|^{\frac{1}{2}} \exp\left\{ -\frac{1}{2} (\boldsymbol{\mu}_j - \boldsymbol{\rho}_0)^T \beta_0 \boldsymbol{\Gamma}_j (\boldsymbol{\mu}_j - \boldsymbol{\rho}_0) \right\} \\ & \cdot |\boldsymbol{\Gamma}_j|^{(\nu_0 - D - 1)/2} \exp\left[-\frac{1}{2} \operatorname{tr}(\boldsymbol{\Gamma}_j \boldsymbol{\Phi}_0) \right] \Bigg). \end{aligned} \tag{C.19}$$

Taking the exp and log, we obtain

$$\begin{aligned} p(\boldsymbol{\theta}|\boldsymbol{\xi}) \propto \exp\Bigg\{ & \sum_{i=1}^{M} (\lambda_{0i} - 1) \log w_i + \frac{1}{2} \sum_{j=1}^{M} (\log \beta_0^D + \log |\boldsymbol{\Gamma}_j|) \\ & - \frac{\beta_0}{2} \sum_{j=1}^{M} (\boldsymbol{\mu}_j^T \boldsymbol{\Gamma}_j \boldsymbol{\mu}_j - 2\boldsymbol{\mu}_j^T \boldsymbol{\Gamma}_j \boldsymbol{\rho}_0 + \boldsymbol{\rho}_0^T \boldsymbol{\Gamma}_j \boldsymbol{\rho}_0) \\ & + \sum_{j=1}^{M} \frac{\nu_0 - D - 1}{2} \log |\boldsymbol{\Gamma}_j| - \frac{1}{2} \sum_{j=1}^{M} \operatorname{tr}(\boldsymbol{\Gamma}_j \boldsymbol{\Phi}_0) \Bigg\}. \end{aligned} \tag{C.20}$$

This gives the following hyperparameter $\boldsymbol{\xi}$:

$$\boldsymbol{\xi} = \begin{bmatrix} \lambda_{01} - 1 \\ \vdots \\ \lambda_{0M} - 1 \\ \nu_0/2 \\ \vdots \\ \nu_0/2 \\ -\frac{1}{2} \operatorname{vec}(\beta_0 \boldsymbol{\rho}_0 \boldsymbol{\rho}_0^T + \boldsymbol{\Phi}_0) \\ \beta_0 \boldsymbol{\rho}_0 \\ -\frac{1}{2}\beta_0 \\ \vdots \\ -\frac{1}{2} \operatorname{vec}(\beta_0 \boldsymbol{\rho}_0 \boldsymbol{\rho}_0^T + \boldsymbol{\Phi}_0) \\ \beta_0 \boldsymbol{\rho}_0 \\ -\frac{1}{2}\beta_0 \end{bmatrix}. \tag{C.21}$$

C.3 The E-Step

For the E-step we need to calculate $\bar{\boldsymbol{\phi}}$. This is given by

$$\bar{\boldsymbol{\phi}} = \begin{bmatrix} \log \tilde{w}_1 \\ \vdots \\ \log \tilde{w}_M \\ \log \tilde{\Gamma}_1 \\ \vdots \\ \log \tilde{\Gamma}_M \\ \mathrm{vec}(\bar{\boldsymbol{\Gamma}}_1) \\ \bar{\boldsymbol{\Gamma}}_1 \boldsymbol{\rho}_1 \\ \boldsymbol{\rho}_1^T \bar{\boldsymbol{\Gamma}}_1 \boldsymbol{\rho}_1 + \frac{D}{\beta_1} \\ \vdots \\ \mathrm{vec}(\bar{\boldsymbol{\Gamma}}_M) \\ \bar{\boldsymbol{\Gamma}}_M \boldsymbol{\rho}_M \\ \boldsymbol{\rho}_M^T \bar{\boldsymbol{\Gamma}}_M \boldsymbol{\rho}_M + \frac{D}{\beta_M} \end{bmatrix} \tag{C.22}$$

where $\log \tilde{w}_j$, $\log \tilde{\Gamma}_j$, and $\bar{\boldsymbol{\Gamma}}_j$ are as in (6.16), (6.17), and (6.18) respectively. Note that

$$\begin{aligned} \mathrm{E}_{\boldsymbol{\theta}}[\boldsymbol{\mu}_j^T \boldsymbol{\Gamma}_j \boldsymbol{\mu}_j] &= \mathrm{E}_{\boldsymbol{\theta}}[\mathrm{tr}(\boldsymbol{\Gamma}_j \boldsymbol{\mu}_j \boldsymbol{\mu}_j^T)] \\ &= \mathrm{tr}(\mathrm{E}_{\boldsymbol{\theta}}[\boldsymbol{\Gamma}_j]\, \mathrm{E}_{\boldsymbol{\theta}}[\boldsymbol{\mu}_j \boldsymbol{\mu}_j^T]) \\ &= \mathrm{tr}\left(\bar{\boldsymbol{\Gamma}}_j \left\{ \mathrm{E}_{\boldsymbol{\theta}}\left[\frac{1}{\beta_j}\boldsymbol{\Gamma}_j^{-1}\right] + \boldsymbol{\rho}_j \boldsymbol{\rho}_j^T \right\}\right) \\ &= \mathrm{tr}\left(\bar{\boldsymbol{\Gamma}}_j \left\{ \frac{1}{\beta_j}\bar{\boldsymbol{\Gamma}}_j^{-1} + \boldsymbol{\rho}_j \boldsymbol{\rho}_j^T \right\}\right) \\ &= \mathrm{tr}\left(\frac{1}{\beta_j}\mathbf{I} + \bar{\boldsymbol{\Gamma}}_j \boldsymbol{\rho}_j \boldsymbol{\rho}_j^T\right) \\ &= \boldsymbol{\rho}_j^T \bar{\boldsymbol{\Gamma}}_j \boldsymbol{\rho}_j + \frac{D}{\beta_j}. \end{aligned} \tag{C.23}$$

Plugging (C.22) into (C.3) yields

$$\begin{aligned} q(z_n) \propto \exp\Bigg\{ & \sum_{j=1}^{M} \delta_j(z_n) \log \tilde{w}_j + \frac{1}{2} \sum_{j=1}^{M} \log \tilde{\Gamma}_j \\ & + \sum_{j=1}^{M} \Bigg[-\frac{1}{2} \delta_j(z_n) \operatorname{tr}(\bar{\mathbf{\Gamma}}_j \mathbf{y}_n \mathbf{y}_n^T) + \delta_j(z_n) \boldsymbol{\rho}_j^T \bar{\mathbf{\Gamma}}_j \mathbf{y}_n \\ & - \frac{1}{2} \delta_j(z_n) \left(\boldsymbol{\rho}_j^T \bar{\mathbf{\Gamma}}_j \boldsymbol{\rho}_j + \frac{D}{\beta_j} \right) \Bigg] \Bigg\}. \end{aligned} \tag{C.24}$$

This can be simplified to

$$q(z_n) \propto \prod_{j=1}^{M} \left(\tilde{w}_j \tilde{\Gamma}_j^{\frac{1}{2}} \exp\left\{ -\frac{1}{2} (\mathbf{y}_n - \boldsymbol{\rho}_j)^T \bar{\mathbf{\Gamma}}_j (\mathbf{y}_n - \boldsymbol{\rho}_j) - \frac{D}{2\beta_j} \right\} \right)^{\delta_j(z_n)} \tag{C.25}$$

which is equivalent to

$$\gamma_j^n = q(z_n = j) \propto \tilde{w}_j \tilde{\Gamma}_j^{\frac{1}{2}} \exp\left\{ -\frac{1}{2} (\mathbf{y}_n - \boldsymbol{\rho}_j)^T \bar{\mathbf{\Gamma}}_j (\mathbf{y}_n - \boldsymbol{\rho}_j) - \frac{D}{2\beta_j} \right\}. \tag{C.26}$$

The result in (C.26) is the same as (6.14).

C.4 The M-Step

For deriving the M-step we need to calculate the vector $\bar{\mathbf{u}}(\mathbf{y}_n)$ as in (C.9). Since $\mathrm{E}_{\mathbf{Z}}[\delta_j(z_n)] = \gamma_j^n$, we get

$$\bar{\mathbf{u}}(\mathbf{y}_n) = \begin{bmatrix} \gamma_1^n \\ \vdots \\ \gamma_M^n \\ \frac{1}{2}\gamma_1^n \\ \vdots \\ \frac{1}{2}\gamma_M^n \\ -\frac{1}{2}\gamma_1^n \operatorname{vec}(\mathbf{y}_n \mathbf{y}_n^T) \\ \gamma_1^n \mathbf{y}_n \\ -\frac{1}{2}\gamma_1^n \\ \vdots \\ -\frac{1}{2}\gamma_M^n \operatorname{vec}(\mathbf{y}_n \mathbf{y}_n^T) \\ \gamma_M^n \mathbf{y}_n \\ -\frac{1}{2}\gamma_M^n \end{bmatrix}. \tag{C.27}$$

Now we can find the posterior hyperparameters by plugging (C.21) and (C.27) into (C.8). First, note that the vector of posterior hyperparameters $\tilde{\boldsymbol{\xi}}$ is given by

$$\tilde{\boldsymbol{\xi}} = \begin{bmatrix} \lambda_1 - 1 \\ \vdots \\ \lambda_M - 1 \\ \nu_1/2 \\ \vdots \\ \nu_M/2 \\ -\frac{1}{2}\operatorname{vec}(\beta_1 \boldsymbol{\rho}_1 \boldsymbol{\rho}_1^T + \boldsymbol{\Phi}_1) \\ \beta_1 \boldsymbol{\rho}_1 \\ -\frac{1}{2}\beta_1 \\ \vdots \\ -\frac{1}{2}\operatorname{vec}(\beta_M \boldsymbol{\rho}_M \boldsymbol{\rho}_M^T + \boldsymbol{\Phi}_M) \\ \beta_M \boldsymbol{\rho}_M \\ -\frac{1}{2}\beta_M \end{bmatrix}. \tag{C.28}$$

Thus, for λ_j we get

$$\lambda_j - 1 = \lambda_{0j} - 1 + \sum_{n=1}^{N} \gamma_j^n \tag{C.29}$$

$$\lambda_j = \lambda_{0j} + \bar{N}_j, \tag{C.30}$$

where $\bar{N}_j$ is as defined in (6.23). This result is the same as (6.24) if every λ_{0j} is set to a common constant λ_0. Similarly, for ν_j we obtain

$$\frac{\nu_j}{2} = \frac{\nu_0}{2} + \frac{1}{2}\sum_{n=1}^{N} \gamma_j^n \tag{C.31}$$

$$\nu_j = \nu_0 + \bar{N}_j. \tag{C.32}$$

This result equals (6.25). Before finding update expressions for $\boldsymbol{\rho}_j$ and $\boldsymbol{\Phi}_j$ we will need to find the expression for β_j:

$$-\frac{1}{2}\beta_j = -\frac{1}{2}\beta_0 - \frac{1}{2}\sum_{n=1}^{N} \gamma_j^n \tag{C.33}$$

$$\beta_j = \beta_0 + \bar{N}_j. \tag{C.34}$$

This result equals (6.26). Then, we can find the expression for $\boldsymbol{\rho}_j$.

$$\beta_j \boldsymbol{\rho}_j = \beta_0 \boldsymbol{\rho}_0 + \sum_{n=1}^{N} \gamma_j^n \mathbf{y}_n \tag{C.35}$$

$$\boldsymbol{\rho}_j = \frac{\beta_0 \boldsymbol{\rho}_0 + \bar{N}_j \bar{\boldsymbol{\mu}}_j}{\beta_j} \tag{C.36}$$

Here, $\bar{\boldsymbol{\mu}}_j$ is as defined in (6.20). Plugging in (C.34) gives

$$\boldsymbol{\rho}_j = \frac{\beta_0 \boldsymbol{\rho}_0 + \bar{N}_j \bar{\boldsymbol{\mu}}_j}{\beta_0 + \bar{N}_j}. \tag{C.37}$$

This result equals (6.27). Finally, we find the expression for $\boldsymbol{\Phi}_j$.

$$-\frac{1}{2}(\beta_j \boldsymbol{\rho}_j \boldsymbol{\rho}_j^T + \boldsymbol{\Phi}_j) = -\frac{1}{2}(\beta_0 \boldsymbol{\rho}_0 \boldsymbol{\rho}_0^T + \boldsymbol{\Phi}_0) - \frac{1}{2} \sum_{n=1}^{N} \gamma_j^n \mathbf{y}_n \mathbf{y}_n^T \tag{C.38}$$

This can be rearranged to

$$\boldsymbol{\Phi}_j = \boldsymbol{\Phi}_0 + \beta_0 \boldsymbol{\rho}_0 \boldsymbol{\rho}_0^T - \beta_j \boldsymbol{\rho}_j \boldsymbol{\rho}_j^T + \sum_{n=1}^{N} \gamma_j^n \mathbf{y}_n \mathbf{y}_n^T. \tag{C.39}$$

Then, by plugging in (C.34) and (C.37) for β_j and $\boldsymbol{\rho}_j$ respectively, we get

$$\begin{aligned}
\boldsymbol{\Phi}_j &= \boldsymbol{\Phi}_0 + \beta_0 \boldsymbol{\rho}_0 \boldsymbol{\rho}_0^T \\
&\quad - (\beta_0 + \bar{N}_j) \left(\frac{\beta_0 \boldsymbol{\rho}_0 + \bar{N}_j \bar{\boldsymbol{\mu}}_j}{\beta_0 + \bar{N}_j} \right) \left(\frac{\beta_0 \boldsymbol{\rho}_0^T + \bar{N}_j \bar{\boldsymbol{\mu}}_j^T}{\beta_0 + \bar{N}_j} \right) \\
&\quad + \sum_{n=1}^{N} \gamma_j^n \mathbf{y}_n \mathbf{y}_n^T \\
&= \boldsymbol{\Phi}_0 + \beta_0 \boldsymbol{\rho}_0 \boldsymbol{\rho}_0^T \\
&\quad - \frac{1}{\beta_0 + \bar{N}_j} \left(\beta_0^2 \boldsymbol{\rho}_0 \boldsymbol{\rho}_0^T + \beta_0 \bar{N}_j \boldsymbol{\rho}_0 \bar{\boldsymbol{\mu}}_j^T + \beta_0 \bar{N}_j \bar{\boldsymbol{\mu}}_j \boldsymbol{\rho}_0^T + \bar{N}_j^2 \bar{\boldsymbol{\mu}}_j \bar{\boldsymbol{\mu}}_j^T \right) \\
&\quad + \sum_{n=1}^{N} \gamma_j^n \mathbf{y}_n \mathbf{y}_n^T \\
&= \boldsymbol{\Phi}_0 + \frac{\beta_0(\beta_0 + \bar{N}_j) - \beta_0^2}{\bar{N}_j + \beta_0} \boldsymbol{\rho}_0 \boldsymbol{\rho}_0^T - \frac{\bar{N}_j \beta_0 \boldsymbol{\rho}_0 \bar{\boldsymbol{\mu}}_j^T}{\beta_0 + \bar{N}_j} - \frac{\bar{N}_j \beta_0 \bar{\boldsymbol{\mu}}_j \boldsymbol{\rho}_0^T}{\beta_0 + \bar{N}_j} \\
&\quad - \frac{\bar{N}_j^2 \bar{\boldsymbol{\mu}}_j \bar{\boldsymbol{\mu}}_j^T}{\beta_0 + \bar{N}_j} + \sum_{n=1}^{N} \gamma_j^n \mathbf{y}_n \mathbf{y}_n^T.
\end{aligned} \tag{C.40}$$

Next, we add and subtract the term $\frac{\bar{N}_j\beta_0}{\beta_0+\bar{N}_j}\bar{\boldsymbol{\mu}}_j\bar{\boldsymbol{\mu}}_j^T$. This gives

$$\begin{aligned}\boldsymbol{\Phi}_j &= \boldsymbol{\Phi}_0 + \frac{\bar{N}_j\beta_0}{\beta_0+\bar{N}_j}\left(\boldsymbol{\rho}_0\boldsymbol{\rho}_0^T - \boldsymbol{\rho}_0\bar{\boldsymbol{\mu}}_j^T - \bar{\boldsymbol{\mu}}_j\boldsymbol{\rho}_0^T + \bar{\boldsymbol{\mu}}_j\bar{\boldsymbol{\mu}}_j^T\right)\\ &\quad - \frac{\bar{N}_j\beta_0}{\beta_0+\bar{N}_j}\bar{\boldsymbol{\mu}}_j\bar{\boldsymbol{\mu}}_j^T - \frac{\bar{N}_j^2\bar{\boldsymbol{\mu}}_j\bar{\boldsymbol{\mu}}_j^T}{\beta_0+\bar{N}_j} + \sum_{n=1}^N \gamma_j^n \mathbf{y}_n\mathbf{y}_n^T\\ &= \boldsymbol{\Phi}_0 + \frac{\bar{N}_j\beta_0}{\beta_0+\bar{N}_j}(\bar{\boldsymbol{\mu}}_j-\boldsymbol{\rho}_0)(\bar{\boldsymbol{\mu}}_j-\boldsymbol{\rho}_0)^T\\ &\quad - \frac{\bar{N}_j(\beta_0+\bar{N}_j)}{\beta_0+\bar{N}_j}\bar{\boldsymbol{\mu}}_j\bar{\boldsymbol{\mu}}_j^T + \sum_{n=1}^N \gamma_j^n \mathbf{y}_n\mathbf{y}_n^T\end{aligned} \tag{C.41}$$

Note that the last two terms can be rewritten as

$$\sum_{n=1}^N \gamma_j^n \mathbf{y}_n\mathbf{y}_n^T - \bar{N}_j\bar{\boldsymbol{\mu}}_j\bar{\boldsymbol{\mu}}_j^T = \sum_{n=1}^N \gamma_j^n(\mathbf{y}_n-\bar{\boldsymbol{\mu}}_j)(\mathbf{y}_n-\bar{\boldsymbol{\mu}}_j)^T = \bar{N}_j\bar{\boldsymbol{\Sigma}}_j, \tag{C.42}$$

where $\bar{\boldsymbol{\Sigma}}_j$ is as defined in (6.21). This gives us the final result, i.e.,

$$\boldsymbol{\Phi}_j = \boldsymbol{\Phi}_0 + \frac{\bar{N}_j\beta_0}{\beta_0+\bar{N}_j}(\bar{\boldsymbol{\mu}}_j-\boldsymbol{\rho}_0)(\bar{\boldsymbol{\mu}}_j-\boldsymbol{\rho}_0)^T + \bar{N}_j\bar{\boldsymbol{\Sigma}}_j, \tag{C.43}$$

which is the same as (6.28).

Appendix D

VB Free Energy for GMM

Recall from (6.4) that the free energy can be written as

$$\mathcal{F}_M(q(\mathbf{Z}), q(\boldsymbol{\theta})) = \mathrm{E}_{\mathbf{Z},\boldsymbol{\theta}}\left[\log \frac{p(\mathbf{Y}, \mathbf{Z}|\boldsymbol{\theta}, M)}{q(\mathbf{Z})}\right] - \mathrm{KL}[q(\boldsymbol{\theta})\|p(\boldsymbol{\theta}|M)]. \tag{D.1}$$

In order to calculate the free energy for a GMM, we will start by looking at the first term. This term can be viewed as averaged log likelihood. For notational convenience, we will denote this term by L. Since the GMM assumes that observations are independent, we have

$$\begin{aligned} L = \mathrm{E}_{\mathbf{Z},\boldsymbol{\theta}}\left[\log \frac{p(\mathbf{Y}, \mathbf{Z}|\boldsymbol{\theta}, M)}{q(\mathbf{Z})}\right] &= \mathrm{E}_{\mathbf{Z},\boldsymbol{\theta}}\left[\log \left(\prod_{n=1}^{N} \frac{p(\mathbf{y}_n, z_n|\boldsymbol{\theta}, M)}{q(z_n)}\right)\right] \\ &= \mathrm{E}_{\mathbf{Z},\boldsymbol{\theta}}\left[\sum_{n=1}^{N} \log \frac{p(\mathbf{y}_n, z_n|\boldsymbol{\theta}, M)}{q(z_n)}\right] \\ &= \sum_{n=1}^{N} \mathrm{E}_{z_n,\boldsymbol{\theta}}\left[\log \frac{p(\mathbf{y}_n, z_n|\boldsymbol{\theta}, M)}{q(z_n)}\right] \end{aligned} \tag{D.2}$$

We can now write the expectation as

$$L = \sum_{n=1}^{N} \sum_{z=1}^{M} q(z_n = z) \int q(\boldsymbol{\theta}) \log \frac{p(\mathbf{y}_n, z_n = z|\boldsymbol{\theta})}{q(z_n = z)} d\boldsymbol{\theta}. \tag{D.3}$$

Note that VB assumes that the parameter posterior $q(\boldsymbol{\theta})$ can be factorized as follows

$$q(\boldsymbol{\theta}) = q(\{w_z\}) \prod_{z=1}^{M} q(\boldsymbol{\mu}_z, \boldsymbol{\Gamma}_z). \tag{D.4}$$

Using this factorization, we can split (D.3) into two parts:

$$L = \sum_{n=1}^{N}\sum_{z=1}^{M} q(z_n = z) \int q(\{w_z\}) \log \frac{p(z_n = z|\{w_z\})}{q(z_n = z)} d\{w_z\}$$
$$+ \sum_{n=1}^{N}\sum_{z=1}^{M} q(z_n = z) \int q(\boldsymbol{\mu}_z, \boldsymbol{\Gamma}_z) \log p(\mathbf{y}_n|z_n = z, \boldsymbol{\mu}_z, \boldsymbol{\Gamma}_z) d\boldsymbol{\mu}_z d\boldsymbol{\Gamma}_z. \tag{D.5}$$

We will now consider each of these two terms separately. The terms will be denoted as L_1 and L_2 respectively, i.e., $L = L_1 + L_2$. Considering L_1, this term can again be split into two terms as follows:

$$L_1 = \sum_{n=1}^{N}\sum_{z=1}^{M} q(z_n = z) \int q(w_z) \log w_z dw_z - \sum_{n=1}^{N}\sum_{z=1}^{M} q(z_n = z) \log q(z_n = z). \tag{D.6}$$

As in Section 6.3 we will denote $q(z_n = z)$ by γ_z^n. Now, we can write (D.6) as

$$L_1 = \sum_{n=1}^{N}\sum_{z=1}^{M} \gamma_z^n \log \tilde{w}_z - \sum_{n=1}^{N}\sum_{z=1}^{M} \gamma_z^n \log \gamma_z^n \tag{D.7}$$

where $\log \tilde{w}_z$ is as defined in (6.16).

Now, we can move on to the term L_2. First, we note that $\log p(\mathbf{y}_n|z_n = z, \boldsymbol{\mu}_z, \boldsymbol{\Gamma}_z)$ can be written as

$$\begin{aligned}\log p(\mathbf{y}_n|z_n = z, \boldsymbol{\mu}_z, \boldsymbol{\Gamma}_z) = &-\frac{D}{2}\log 2\pi + \frac{1}{2}\log|\boldsymbol{\Gamma}_z| \\ &- \frac{1}{2}(\mathbf{y}_n - \boldsymbol{\mu}_z)^T \boldsymbol{\Gamma}_z (\mathbf{y}_n - \boldsymbol{\mu}_z).\end{aligned} \tag{D.8}$$

The last term in (D.8) can be written as

$$-\frac{1}{2}(\mathbf{y}_n - \boldsymbol{\mu}_z)^T \boldsymbol{\Gamma}_z (\mathbf{y}_n - \boldsymbol{\mu}_z) = -\frac{1}{2}\left(\mathbf{y}_n^T \boldsymbol{\Gamma}_z \mathbf{y}_n - 2\mathbf{y}_n^T \boldsymbol{\Gamma}_z \boldsymbol{\mu}_z + \mathrm{tr}(\boldsymbol{\Gamma}_z \boldsymbol{\mu}_z \boldsymbol{\mu}_z^T)\right). \tag{D.9}$$

Now, taking the expectation of (D.8) with respect to $q(\boldsymbol{\theta})$ yields

$$\begin{aligned}\mathrm{E}_{\boldsymbol{\theta}}[\log p(\mathbf{y}_n|z_n = z, \boldsymbol{\mu}_z, \boldsymbol{\Gamma}_z)] = &-\frac{D}{2}\log 2\pi + \frac{1}{2}\log \tilde{\Gamma}_z \\ &- \frac{1}{2}\left(\mathbf{y}_n^T \bar{\boldsymbol{\Gamma}}_z \mathbf{y}_n - 2\mathbf{y}_n^T \bar{\boldsymbol{\Gamma}}_z \boldsymbol{\rho}_z + \boldsymbol{\rho}_z^T \bar{\boldsymbol{\Gamma}}_z \boldsymbol{\rho}_z + \frac{D}{\beta_z}\right),\end{aligned} \tag{D.10}$$

where we have used (C.23) from Appendix C to find the expectation of the last term in (D.9). In (D.10), $\log \tilde{\Gamma}_z$ and $\bar{\mathbf{\Gamma}}_z$ are as defined in (6.17) and (6.18) respectively. This results in the following expression for L_2:

$$L_2 = \sum_{n=1}^{N} \sum_{z=1}^{M} \frac{\gamma_z^n}{2} \left(-D \log 2\pi + \log \tilde{\Gamma}_z - (\mathbf{y}_n - \boldsymbol{\rho}_z)^T \bar{\mathbf{\Gamma}}_z (\mathbf{y}_n - \boldsymbol{\rho}_z) - \frac{D}{\beta_z} \right). \tag{D.11}$$

Plugging (D.7) and (D.11) into (D.5) yields the following expression for L:

$$\begin{aligned} L = \sum_{n=1}^{N} \sum_{z=1}^{M} \frac{\gamma_z^n}{2} \Bigg(& 2 \log \tilde{w}_z - 2 \log \gamma_z^n - D \log 2\pi + \log \tilde{\Gamma}_z \\ & - (\mathbf{y}_n - \boldsymbol{\rho}_z)^T \bar{\mathbf{\Gamma}}_z (\mathbf{y}_n - \boldsymbol{\rho}_z) - \frac{D}{\beta_z} \Bigg). \end{aligned} \tag{D.12}$$

Then, we can move on to the last term in (D.1). We have

$$\begin{aligned} \mathrm{KL}[q(\boldsymbol{\theta}) \| p(\boldsymbol{\theta}|M)] &= \int q(\boldsymbol{\theta}) \log \frac{q(\boldsymbol{\theta})}{p(\boldsymbol{\theta}|M)} d\boldsymbol{\theta} \\ &= \int q(\{w_z\}) \prod_{z=1}^{M} q(\boldsymbol{\mu}_z, \mathbf{\Gamma}_z) \log \frac{q(\{w_i\}) \prod_{i=1}^{M} q(\boldsymbol{\mu}_i, \mathbf{\Gamma}_i)}{p(\{w_j\}) \prod_{j=1}^{M} p(\boldsymbol{\mu}_j, \mathbf{\Gamma}_j)} d\boldsymbol{\theta} \\ &= \int q(\{w_z\}) \log \frac{q(\{w_z\})}{p(\{w_z\})} d\{w_z\} \\ &\quad + \int \prod_{z=1}^{M} q(\boldsymbol{\mu}_z, \mathbf{\Gamma}_z) \sum_{i=1}^{M} \log \frac{q(\boldsymbol{\mu}_i, \mathbf{\Gamma}_i)}{p(\boldsymbol{\mu}_i, \mathbf{\Gamma}_i)} d\boldsymbol{\mu}_z d\mathbf{\Gamma}_z. \end{aligned} \tag{D.13}$$

The first term is the KL-distance between the posterior and prior of $\{w_z\}$. These are Dirichlet distributions with parameters $\{\lambda_z\}$ and $\{\lambda_z^0\}$ respectively. We will denote this KL-distance by $\mathrm{KL}_{\mathcal{D}}[\{\lambda_z\} \| \{\lambda_z^0\}]$. The last term in (D.13) can be simplified due to the fact that the parameters of one component are independent of the parameters of other components. Thus, for each component we can integrate out the parameters of all other components, and get

$$\begin{aligned}\mathrm{KL}[q(\boldsymbol{\theta})\|p(\boldsymbol{\theta}|M)] &= \mathrm{KL}_{\mathcal{D}}[\{\lambda_z\}\|\{\lambda_z^0\}] \\ &\quad + \sum_{z=1}^{M} \int q(\boldsymbol{\mu}_z, \boldsymbol{\Gamma}_z) \log \frac{q(\boldsymbol{\mu}_z, \boldsymbol{\Gamma}_z)}{p(\boldsymbol{\mu}_z, \boldsymbol{\Gamma}_z)} d\boldsymbol{\mu}_z d\boldsymbol{\Gamma}_z \\ &= \mathrm{KL}_{\mathcal{D}}[\{\lambda_z\}\|\{\lambda_z^0\}] \\ &\quad + \sum_{z=1}^{M} \int q(\boldsymbol{\mu}_z|\boldsymbol{\Gamma}_z) q(\boldsymbol{\Gamma}_z) \log \frac{q(\boldsymbol{\mu}_z|\boldsymbol{\Gamma}_z)}{p(\boldsymbol{\mu}_z|\boldsymbol{\Gamma}_z)} d\boldsymbol{\mu}_z d\boldsymbol{\Gamma}_z \\ &\quad + \sum_{z=1}^{M} \int q(\boldsymbol{\Gamma}_z) \log \frac{q(\boldsymbol{\Gamma}_z)}{p(\boldsymbol{\Gamma}_z)} d\boldsymbol{\Gamma}_z.\end{aligned} \tag{D.14}$$

The last term here is the sum of KL-distances between the posterior and prior of each $\boldsymbol{\Gamma}_z$. These have Wishart distributions. Given one component z, the posterior has parameters $\{\nu_z, \boldsymbol{\Phi}_z\}$, and the prior has parameters $\{\nu^0, \boldsymbol{\Phi}^0\}$. We will denote this KL-distance by $\mathrm{KL}_{\mathcal{W}}[\{\nu_z, \boldsymbol{\Phi}_z\}\|\{\nu^0, \boldsymbol{\Phi}^0\}]$. In the second term in (D.14) we have to integrate out $\boldsymbol{\Gamma}_z$ for all components. This results in the sum of KL-distances between posteriors $q(\boldsymbol{\mu}_z)$ and priors $p(\boldsymbol{\mu}_z)$ that have their precision matrices replaced by the expected values according to $q(\boldsymbol{\Gamma}_z)$: $\beta_z\bar{\boldsymbol{\Gamma}}_z$ and $\beta^0\bar{\boldsymbol{\Gamma}}_z$ respectively. For a given component, we will denote this KL-distance by $\mathrm{KL}_{\mathcal{N}}[\{\boldsymbol{\rho}_z, \beta_z\bar{\boldsymbol{\Gamma}}_z\}\|\{\boldsymbol{\rho}^0, \beta^0\bar{\boldsymbol{\Gamma}}_z\}]$. Thus, we can write (D.14) as

$$\begin{aligned}\mathrm{KL}[q(\boldsymbol{\theta})\|p(\boldsymbol{\theta}|M)] &= \mathrm{KL}_{\mathcal{D}}[\{\lambda_z\}\|\{\lambda_z^0\}] \\ &\quad + \sum_{z=1}^{M} \mathrm{KL}_{\mathcal{N}}[\{\boldsymbol{\rho}_z, \beta_z\bar{\boldsymbol{\Gamma}}_z\}\|\{\boldsymbol{\rho}^0, \beta^0\bar{\boldsymbol{\Gamma}}_z\}] \\ &\quad + \sum_{z=1}^{M} \mathrm{KL}_{\mathcal{W}}[\{\nu_z, \boldsymbol{\Phi}_z\}\|\{\nu^0, \boldsymbol{\Phi}^0\}].\end{aligned} \tag{D.15}$$

Closed-form expressions for calculating these KL-distances can be found in [53].

Then, we can write the final expression for the free energy as

$$\begin{aligned}
\mathcal{F}_M(q(\mathbf{Z}), q(\boldsymbol{\theta})) = \sum_{n=1}^{N}\sum_{z=1}^{M} \frac{\gamma_z^n}{2}\Bigg(& 2\log \tilde{w}_z - 2\log \gamma_z^n - D\log 2\pi \\
& + \log \tilde{\Gamma}_z - (\mathbf{y}_n - \boldsymbol{\rho}_z)^T \bar{\mathbf{\Gamma}}_z (\mathbf{y}_n - \boldsymbol{\rho}_z) - \frac{D}{\beta_z}\Bigg) \\
& - \mathrm{KL}_{\mathcal{D}}[\{\lambda_z\} \| \{\lambda_z^0\}] - \sum_{z=1}^{M} \mathrm{KL}_{\mathcal{W}}[\{\nu_z, \mathbf{\Phi}_z\} \| \{\nu^0, \mathbf{\Phi}^0\}] \\
& - \sum_{z=1}^{M} \mathrm{KL}_{\mathcal{N}}[\{\boldsymbol{\rho}_z, \beta_z \bar{\mathbf{\Gamma}}_z\} \| \{\boldsymbol{\rho}^0, \beta^0 \bar{\mathbf{\Gamma}}_z\}].
\end{aligned} \tag{D.16}$$

Appendix E

Amounts of Detected Noise Frames

This appendix lists relative amounts of noise frames selected for noise modelling by the voice activity detection methods described in Chapter 7. Table E.1 shows the amounts as percentages of the total number of frames when excluding the first 20 frames of each utterance, which are used for estimating the initial noise model. Table E.2 shows the amounts as percentages of the total number of frames when including the first 20 frames.

Table E.1: Percentage of frames classified as noise, excluding the first 20.

Noise	SNR	Sohn	LRT-CEP	FrcAlgn
	20	11.05	16.96	18.52
	15	13.38	20.29	”
subway	10	15.69	21.68	”
	5	19.46	25.68	”
	0	23.11	32.63	”
	-5	31.77	47.58	”
	20	6.50	17.27	17.83
	15	7.39	19.15	”
babble	10	8.24	22.36	”
	5	9.22	28.85	”
	0	11.05	39.76	”
	-5	14.51	57.41	”
	20	13.78	21.76	17.40
	15	16.78	25.82	”
car	10	20.10	30.24	”
	5	24.27	36.85	”
	0	30.25	49.14	”
	-5	40.56	70.77	”
	20	6.16	19.65	18.36
	15	7.55	22.08	”
exhib.	10	8.72	25.13	”
	5	9.56	28.14	”
	0	11.74	33.11	”
	-5	13.39	43.03	”

Table E.2: Percentage of frames classified as noise, including the first 20.

Noise	SNR	Sohn	LRT-CEP	FrcAlgn
subway	20	21.30	26.53	27.90
	15	23.36	29.47	”
	10	25.40	30.70	”
	5	28.74	34.24	”
	0	31.96	40.39	”
	-5	39.63	53.62	”
babble	20	17.06	26.61	27.11
	15	17.85	28.29	”
	10	18.61	31.13	”
	5	19.48	36.89	”
	0	21.10	46.57	”
	-5	24.16	62.22	”
car	20	23.54	30.61	26.75
	15	26.20	34.21	”
	10	29.14	38.13	”
	5	32.84	43.99	”
	0	38.14	54.90	”
	-5	47.28	74.08	”
exhib.	20	16.95	28.88	27.74
	15	18.17	31.04	”
	10	19.21	33.74	”
	5	19.96	36.40	”
	0	21.88	40.80	”
	-5	23.34	49.57	”

Appendix F

Significance Testing

When comparing test results of different algorithms for speech recognition, small performance differences can often occur due to chance effects. It is therefore important to check whether differences in performance are statistically significant. In this thesis we will use the matched-pairs test presented by Gillick and Cox in [30] when testing for statistical significance.

F.1 Matched-Pairs Significance Test

This test assumes that it is possible to divide the output from a speech recognition algorithm into segments such that the errors in one segment are statistically independent of the errors in any other segment. Typical candidates for such segments are phrases or sentences. Now, suppose we are to compare the results from two speech recognition algorithms, denoted A_1 and A_2 respectively. Then, we would like to test whether algorithm A_1 on average makes the same amount of errors on a segment as algorithm A_2.

Let the random variables N_1^i and N_2^i denote the number of errors made on segment i by A_1 and A_2 respectively. Then, define the random variable

$$Z_i = N_1^i - N_2^i \tag{F.1}$$

for $i = 1, \ldots, n$, where n is the number of segments. This means that we have a random sample of size n for the difference of the number of errors made by A_1 and A_2 on a given segment.

Based on this random sample, we will check whether there is sufficient evidence to support the hypothesis that $\mu = 0$. Thus, the following hypotheses will be tested:

$$\begin{aligned} H_0 :&\quad \mu = 0 \\ H_1 :&\quad \mu \neq 0. \end{aligned} \tag{F.2}$$

A natural estimator for the mean is the sample mean, i.e.,

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} Z_i, \tag{F.3}$$

The variance can be estimated as

$$\hat{\sigma}^2 = \frac{1}{n-1} \sum_{i=1}^{n} (Z_i - \hat{\mu})^2. \tag{F.4}$$

Now, we define a new random variable W as

$$W = \frac{\hat{\mu}}{\hat{\sigma}/\sqrt{n}}. \tag{F.5}$$

If n is large, W will approximately have a normal distribution $\mathcal{N}(0, 1)$. To test the null hypothesis, we compute

$$\begin{aligned} \mathcal{P} &= 2P(Z \geq |w|) \\ &= 2 \int_{|w|}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-t^2/2} dt \\ &= 1 - \operatorname{erf}\left(\frac{|w|}{\sqrt{2}}\right), \end{aligned} \tag{F.6}$$

where w is the observed value of W. If $\mathcal{P} < \alpha$ for a chosen significance level α, then H_0 is rejected. This means that the differences in the recognition results are found to be statistically significant. If $\mathcal{P} \geq \alpha$, then H_0 cannot be rejected, and the differences in the results might be due to chance effects.

When applying this significance test to our experiments on Aurora2, we will use utterances as segments. Moreover, to determine the number of errors on a given segment, we will use an alignment algorithm based on dynamic programming which finds the number of substitution errors, deletion errors, and insertion errors. The number of errors for a given segment is then found by summing up the number of substitutions, deletions and insertions. The significance level α is chosen to be 5%.

F.2 Comparing PMC, BPC-PMC, and BPC-vad

In this section we compare the results of the methods PMC, BPC-PMC, and BPC-vad from Chapter 5 in order to determine which differences are statistically significant. Table F.1 compares PMC and BPC-PMC, Table F.2 compares BPC-PMC and BPC-vad, and Table F.3 compared PMC and BPC-vad.

Table F.1: Comparison of PMC and BPC-PMC. Boldfaced font indicates a statistically significant difference using a significance level of 5%.

Noise	SNR	PMC	BPC-PMC
subway	20	**97.88**	97.39
	15	96.53	96.01
	10	92.66	92.20
	5	80.35	**82.13**
	0	46.67	**56.00**
	-5	21.31	**25.88**
babble	20	**96.83**	87.48
	15	**92.65**	79.20
	10	**85.04**	63.27
	5	**67.05**	47.22
	0	**39.57**	24.12
	-5	**17.11**	8.16
car	20	**98.30**	97.64
	15	96.96	96.87
	10	91.50	**92.48**
	5	70.27	**76.74**
	0	31.05	**45.75**
	-5	15.72	**18.25**
exhibition	20	**97.41**	96.36
	15	**95.83**	94.35
	10	**91.05**	89.02
	5	75.75	75.13
	0	42.27	**46.74**
	-5	15.46	**19.72**

Table F.2: Comparison of BPC-PMC and BPC-vad. Boldfaced font indicates a statistically significant difference using a significance level of 5%.

Noise	SNR	BPC-PMC	BPC-vad
subway	20	97.39	97.39
	15	96.01	95.95
	10	92.20	91.86
	5	82.13	81.58
	0	**56.00**	52.72
	-5	**25.88**	23.24
babble	20	87.48	**94.50**
	15	79.20	**89.60**
	10	63.27	**79.50**
	5	47.22	**64.33**
	0	24.12	**38.15**
	-5	8.16	**16.60**
car	20	97.64	97.55
	15	96.87	96.75
	10	**92.48**	91.92
	5	**76.74**	74.80
	0	**45.75**	38.56
	-5	**18.25**	16.19
exhibition	20	96.36	**96.82**
	15	94.35	**95.09**
	10	89.02	**89.85**
	5	75.13	**77.45**
	0	46.74	47.39
	-5	19.72	19.13

Table F.3: Comparison of PMC and BPC-vad. Boldfaced font indicates a statistically significant difference using a significance level of 5%.

Noise	SNR	PMC	BPC-vad
subway	20	**97.88**	97.39
	15	**96.53**	95.95
	10	**92.66**	91.86
	5	80.35	**81.58**
	0	46.67	**52.72**
	-5	21.31	**23.24**
babble	20	**96.83**	94.50
	15	**92.65**	89.60
	10	**85.04**	79.50
	5	**67.05**	64.33
	0	**39.57**	38.15
	-5	17.11	16.60
car	20	**98.30**	97.55
	15	96.96	96.75
	10	91.50	91.92
	5	70.27	**74.80**
	0	31.05	**38.56**
	-5	15.72	16.19
exhibition	20	**97.41**	96.82
	15	**95.83**	95.09
	10	**91.05**	89.85
	5	75.75	**77.45**
	0	42.27	**47.39**
	-5	15.46	**19.13**

F.3 Comparing ML and VB Learning for Modeling Speech in Algonquin

In this section we take a closer look at some of the results from Chapter 6 in order to check which differences are statistically significant. For model sizes of 18, 26, 34, 42, and 50 we present detailed results for each of the four training sets that were used to generate Figure 6.2, Figure 6.4, Figure 6.5, and Figure 6.6. These results are shown in Table F.4, Table F.5, Table F.6, and Table F.7 respectively.

Table F.4: Comparing ML and VB on subway noise 5dB for different training sets. Boldfaced font indicates a statistically significant difference using a significance level of 5%.

	Set 1		Set 2		Set 3		Set 4	
M	ML	VB	ML	VB	ML	VB	ML	VB
18	71.88	**73.50**	69.97	70.83	70.99	71.08	71.26	71.23
26	72.09	**73.44**	70.46	70.74	70.19	**71.38**	70.37	71.23
34	72.40	**73.50**	70.77	70.71	70.92	71.17	71.75	71.23
42	70.53	**73.38**	70.00	70.68	69.70	**71.08**	71.78	71.26
50	70.46	**73.38**	70.34	70.74	69.88	**71.05**	70.77	71.23

Table F.5: Comparing ML and VB on babble noise 5dB for different training sets. Boldfaced font indicates a statistically significant difference using a significance level of 5%.

	Set 1		Set 2		Set 3		Set 4	
M	ML	VB	ML	VB	ML	VB	ML	VB
18	77.24	76.72	76.00	**77.78**	76.48	76.78	**77.03**	75.85
26	76.21	76.63	76.84	**77.75**	76.93	76.90	76.24	75.73
34	76.36	76.66	76.54	**77.78**	**77.90**	76.93	76.15	75.76
42	77.12	76.57	76.75	**77.69**	77.30	76.75	**77.00**	75.73
50	77.27	76.45	77.09	77.69	77.30	76.72	**77.03**	75.60

Table F.6: Comparing ML and VB on car noise 5dB for different training sets. Boldfaced font indicates a statistically significant difference using a significance level of 5%.

	Set 1		Set 2		Set 3		Set 4	
M	ML	VB	ML	VB	ML	VB	ML	VB
18	82.52	82.14	82.05	81.99	82.28	82.43	81.93	81.18
26	81.99	82.05	82.17	81.84	83.21	82.52	**82.25**	81.18
34	82.43	81.90	82.20	81.75	83.06	82.55	**82.58**	81.24
42	81.60	81.93	**82.46**	81.63	82.67	82.55	**82.20**	81.21
50	81.99	81.90	82.25	81.60	82.52	82.37	**82.73**	81.15

Table F.7: Comparing ML and VB on exhibition noise 5dB for different training sets. Boldfaced font indicates a statistically significant difference using a significance level of 5%.

	Set 1		Set 2		Set 3		Set 4	
M	ML	VB	ML	VB	ML	VB	ML	VB
18	79.91	79.08	76.95	**78.28**	78.46	79.17	78.62	78.68
26	**80.01**	78.96	77.94	78.22	78.43	79.27	78.03	78.74
34	79.91	79.02	78.25	78.19	78.77	79.11	78.68	78.71
42	**80.19**	79.02	77.82	78.15	77.14	**79.20**	78.80	78.71
50	**80.16**	78.96	77.66	78.06	77.32	**79.20**	78.71	78.71

F.4 Comparing VAD-Methods and the First20 Approach for Noise Model Estimation

In this section we compare the results of different methods from Chapter 7. We compare the First20 method, Sohn *et al.*'s method and the proposed LRT-CEP approach in order to determine which differences are statistically significant. Table F.8 compares First20 to the LRT-CEP approach and Table F.9 compares Sohn *et al.*'s approach to the LRT-CEP approach.

Table F.8: Comparison of First20 and proposed LRT-CEP approach. Boldfaced font indicates a statistically significant difference using a significance level of 5%.

Noise	SNR	First20	LRT-CEP
subway	20	96.75	96.90
	15	93.89	**94.72**
	10	86.34	**89.41**
	5	75.50	**80.32**
	0	56.16	**59.87**
	-5	28.09	27.51
babble	20	96.61	**97.49**
	15	93.74	**95.77**
	10	85.97	**90.81**
	5	72.52	**79.78**
	0	44.53	**46.49**
	-5	14.93	15.24
car	20	98.33	98.57
	15	97.44	97.70
	10	94.78	95.17
	5	87.21	87.92
	0	**66.84**	60.01
	-5	**28.57**	21.12
exhib.	20	97.69	97.72
	15	95.00	**95.96**
	10	90.25	**91.98**
	5	79.91	**83.52**
	0	61.59	**66.52**
	-5	35.51	36.19

Table F.9: Comparison of Sohn *et al.*'s approach and proposed LRT-CEP approach. Boldfaced font indicates a statistically significant difference using a significance level of 5%.

Noise	SNR	Sohn	LRT-CEP
subway	20	97.05	96.90
	15	94.75	94.72
	10	89.07	89.41
	5	80.44	80.32
	0	61.01	59.87
	-5	**29.66**	27.51
babble	20	97.19	**97.49**
	15	94.74	**95.77**
	10	88.97	**90.81**
	5	77.75	**79.78**
	0	**48.64**	46.49
	-5	**17.26**	15.24
car	20	98.51	98.57
	15	97.88	97.70
	10	**95.76**	95.17
	5	**88.67**	87.92
	0	**64.06**	60.01
	-5	**24.46**	21.12
exhib.	20	97.81	97.72
	15	95.80	95.96
	10	91.02	**91.98**
	5	82.17	**83.52**
	0	65.17	**66.52**
	-5	37.15	36.19

Bibliography

[1] A. Acero, L. Deng, T. Kristjansson, and J. Zhang. HMM adaptation using vector Taylor series for noisy speech recognition. In *Proc. Int. Conf. on Spoken Language Processing (ICSLP)*, pages 869–872, 2000.

[2] M. Afify, O. Siohan, and C.-H. Lee. Minimax classification with parametric neighborhoods for noisy speech recognition. In *Proc. European Conf. on Speech Comm. and Technology (Eurospeech)*, pages 2355–2358, 2001.

[3] H. Attias. A variational Bayesian framework for graphical models. In *Proc. Neural Information Processing Systems*, volume 12, pages 209–215, 2000.

[4] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *The Annals of Mathematical Statistics*, 41(1):164–171, 1970.

[5] M. J. Beal. *Variational Algorithms for Approximate Bayesian Inference.* PhD thesis, University College London, 2003.

[6] S. F. Boll. Suppression of acoustic noise in speech using spectral subtraction. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 27:113–120, 1979.

[7] J. T. Chien. A Bayesian prediction approach to robust speech recognition and online environmental learning. *Speech Communication*, 37:321–334, 2002.

[8] C. Constantinopoulos and A. Likas. Unsupervised learning of Gaussian mixtures based on variational component splitting. *IEEE Trans. on Neural Networks*, 18(3):745–755, 2007.

[9] C. Couvreur and H. Van hamme. Model-based feature enhancement for noisy speech recognition. In *Proc. IEEE Int. Conf. on Acoust., Speech, Signal Processing (ICASSP)*, pages 1719–1722, 2000.

[10] S. B. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 28:357–366, 1980.

[11] 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*, 39(1):1–38, 1977.

[12] L. Deng, J. Droppo, and A. Acero. Incremental Bayes learning with prior evolution for tracking nonstationary noise statistics from noisy speech data. In *Proc. IEEE Int. Conf. on Acoust., Speech, Signal Processing (ICASSP)*, pages 672–675, 2003.

[13] 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*, 11(6):568–580, 2003.

[14] L. Deng, J. Droppo, and A. Acero. Enhancement of log mel power spectra of speech using a phase-sensitive model of the acoustic environment and sequential estimation of the corrupting noise. *IEEE Trans. on Speech and Audio Processing*, 12(2):133–143, 2004.

[15] G.-H. Ding. Maximum a posteriori noise log-spectral estimation based on first-order vector Taylor series expansion. *IEEE Signal Processing Letters*, 15:158–161, 2008.

[16] G.-H. Ding, X. Wang, Y. Cao, F. Ding, and Y. Tang. Sequential noise estimation for noise-robust speech recognition based on 1st-order VTS approximation. In *Proc. IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 363–368, 2005.

[17] J. Droppo, A. Acero, and L. Deng. A nonlinear observation model for removing noise from corrupted speech log mel-spectral energies. In *Proc. Int. Conf. on Spoken Language Processing (ICSLP)*, pages 1569–1572, 2002.

[18] J. Droppo, L. Deng, and A. Acero. A comparison of three non-linear observation models for noisy speech features. In *Proc. European Conf. on Speech Comm. and Technology (Eurospeech)*, pages 681–684, 2003.

[19] Y. Ephraim and D. Malah. Speech enhancement using a minimum mean-square error short-time spectral amplitude estimator. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 32:1109–1121, 1984.

[20] F. Faubel, J. McDonough, and D. Klakow. A phase-averaged model for the relationship between noisy speech, clean speech and noise in the log-mel domain. In *Proc. Interspeech*, pages 553–556, 2008.

[21] B. J. Frey, L. Deng, A. Acero, and T. Kristjansson. Algonquin: Iterating laplace's method to remove multiple types of acoustic distortion for robust speech recognition. In *Proc. European Conf. on Speech Comm. and Technology (Eurospeech)*, pages 901–904, 2001.

[22] B. J. Frey, T. T. Kristjansson, L. Deng, and A. Acero. Algonquin - learning dynamic noise models from noisy speech for robust speech recognition. In *Proc. Neural Information Processing Systems*, 2002.

[23] S. Furui and D. Itoh. Neural-network-based HMM adaptation for noisy speech. In *Proc. IEEE Int. Conf. on Acoust., Speech, Signal Processing (ICASSP)*, pages 365–368, 2001.

[24] B. Gajić and K. K. Paliwal. Robust speech recognition in noisy environments based on subband spectral centroid histograms. *IEEE Trans. on Audio, Speech, and Language Processing*, 14(2):600–608, 2006.

[25] M. J. F. Gales. *Model-based techniques for noise robust speech recognition*. PhD thesis, University of Cambridge, 1995.

[26] M. J. F. Gales and S. J. Young. Cepstral parameter compensation for HMM recognition in noise. *Speech Communication*, 12:231–239, 1993.

[27] M. J. F. Gales and S. J. Young. Parallel model combination for speech recognition in noise. Technical Report CUED/F-INFENG/TR 135, University of Cambridge, 1993.

[28] M. J. F. Gales and S. J. Young. A fast and flexible implementation of parallel model combination. In *Proc. IEEE Int. Conf. on Acoust., Speech, Signal Processing (ICASSP)*, pages 133–136, 1995.

[29] M. J. F. Gales and S. J. Young. Robust continuous speech recognition using parallel model combination. *IEEE Trans. on Speech and Audio Processing*, 4(5):352–359, 1996.

[30] L. Gillick and S. Cox. Some statistical issues in the comparison of speech recognition algorithms. In *Proc. IEEE Int. Conf. on Acoust., Speech, Signal Processing (ICASSP)*, pages 532–535, 1989.

[31] H. Hermansky. Perceptual linear predictive (PLP) analysis of speech. *Journal of the Acoustical Society of America*, 87(4):1738–1752, 1990.

[32] H. G. Hirsch and D. Pearce. The AURORA experimental framework for the performance evaluation of speech recognition systems under noisy conditions. In *Proc. ISCA Tutorial and Research Workshop (ITRW) ASR2000*, 2000.

[33] X. Huang, A. Acero, and H.-W. Hon. *Spoken Language Processing.* Prentice Hall, 2001.

[34] Q. Huo. An introduction to decision rules for automatic speech recognition. Technical Report TR-99-07, University of Hong Kong, 1999.

[35] Q. Huo, H. Jiang, and C.-H. Lee. A Bayesian predictive classification approach to robust speech recognition. In *Proc. IEEE Int. Conf. on Acoust., Speech, Signal Processing (ICASSP)*, pages 1547–1550, 1997.

[36] Q. Huo and C.-H. Lee. On-line adaptive learning of the continuous density hidden Markov model based on approximate recursive bayes estimate. *IEEE Trans. on Speech and Audio Processing*, 5(2):161–172, 1997.

[37] Q. Huo and C.-H. Lee. A study of prior sensitivity for Bayesian predictive classification based robust speech recognition. In *Proc. IEEE Int. Conf. on Acoust., Speech, Signal Processing (ICASSP)*, pages 741–744, 1998.

[38] Q. Huo and C.-H. Lee. A Bayesian predictive classification approach to robust speech recognition. *IEEE Trans. on Speech and Audio Processing*, 8(2):200–204, 2000.

[39] H. Jiang, K. Hirose, and Q. Huo. Robust speech recognition based on a Bayesian prediction approach. *IEEE Trans. on Speech and Audio Processing*, 7(4):426–440, 1999.

[40] H. Jiang, K. Hirose, and Q. Huo. A minimax search algorithm for robust continuous speech recognition. *IEEE Trans. on Speech and Audio Processing*, 8(6):688–694, 2000.

[41] B. H. Juang and L. R. Rabiner. Hidden Markov models for speech recognition. *Technometrics*, 33(3):251–272, 1991.

[42] D. K. Kim, K. W. Jang, and J.-H. Chang. A new statistical voice activity detection based on UMP test. *IEEE Signal Processing Letters*, 14(11):891–894, 2007.

[43] D. Y. Kim, C. K. Un, and N. S. Kim. Speech recognition in noisy environments using first-order vector Taylor series. *Speech Communication*, 24(1):39–49, 1998.

[44] T. T. Kristjansson. *Speech Recognition in Adverse Environments: a Probabilistic Approach.* PhD thesis, University of Waterloo, 2002.

[45] R. Leonard. A database for speaker-independent digit recognition. In *Proc. IEEE Int. Conf. on Acoust., Speech, Signal Processing (ICASSP)*, pages 328–331, 1984.

[46] J. Li, L. Deng, D. Yu, Y. Gong, and A. Acero. High-performance HMM adaptation with joint compensation of additive and convolutive distortions via vector Taylor series. In *Proc. IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 65–70, 2007.

[47] J. Makhoul. Spectral linear prediction: Properties and applications. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 23(3):283–296, 1975.

[48] R. Martin. Noise power spectral density estimation based on optimal smoothing and minimum statistics. *IEEE Trans. on Speech and Audio Processing*, 9(5):504–512, 2001.

[49] N. Merhav and C.-H. Lee. A minimax classification approach with application to robust speech recognition. *IEEE Trans. on Speech and Audio Processing*, 1(1):90–100, 1993.

[50] P. J. Moreno. *Speech Recognition in Noisy Environments.* PhD thesis, Carnegie Mellon University, 1996.

[51] P. J. Moreno, B. Raj, and R. M. Stern. A vector Taylor series approach for environment-independent speech recognition. In *Proc. IEEE Int.*

Conf. on Acoust., Speech, Signal Processing (ICASSP), pages 733–736, 1996.

[52] T. A. Myrvoll and S. Nakamura. Online cepstral filtering using a sequential EM approach with Polyak averaging and feedback. In *Proc. IEEE Int. Conf. on Acoust., Speech, Signal Processing (ICASSP)*, pages 261–264, 2005.

[53] W. D. Penny. KL-divergences of normal, gamma, Dirichlet and Wishart densities. Technical report, University College London, 2001.

[54] S. G. Pettersen. Joint Bayesian predictive classification and parallel model combination with prior scaling for robust ASR. In *Proc. Interspeech*, pages 1273–1276, 2008.

[55] S. G. Pettersen and M. H. Johnsen. Cepstral domain voice activity detection for improved noise modeling in MMSE feature enhancement for ASR. In *Proc. Interspeech*, pages 1012–1015, 2008.

[56] S. G. Pettersen, M. H. Johnsen, and T. A. Myrvoll. A comparative study of model compensation methods for robust speech recognition in noisy conditions. In *Proc. ASIDE, ISCA Tutorial and Research Workshop (ITRW) and COST278 Final Workshop*, 2005.

[57] S. G. Pettersen, M. H. Johnsen, and T. A. Myrvoll. Joint Bayesian predictive classification and parallel model combination for robust speech recognition. In *Proc. European Conf. on Speech Comm. and Technology (Interspeech)*, pages 373–376, 2005.

[58] S. G. Pettersen, M. H. Johnsen, and C. J. Wellekens. Variational Bayesian learning of speech GMMs for feature enhancement based on Algonquin. In *Proc. IEEE Int. Conf. on Acoust., Speech, Signal Processing (ICASSP)*, pages 905–908, 2007.

[59] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE*, 77(2):257–286, 1989.

[60] J. Ramírez, J. Segura, C. Benítez, L. García, and A. Rubio. Statistical voice activity detection using a multiple observation likelihood ratio test. *IEEE Signal Processing Letters*, 12(10):689–692, 2005.

[61] C. K. Raut, T. Nishimoto, and S. Sagayama. Model composition by Lagrange polynomial approximation for robust speech recognition in noisy environment. In *Proc. Int. Conf. on Spoken Language Processing (ICSLP)*, pages 2809–2812, 2004.

[62] J. Rissanen. Universal coding, information, prediction, and estimation. *IEEE Trans. on Information Theory*, 30(4):629–636, 1984.

[63] S. Sagayama, Y. Yamaguchi, S. Takahashi, and J. Takahashi. Jacobian approach to fast acoustic model adaptation. In *Proc. IEEE Int. Conf. on Acoust., Speech, Signal Processing (ICASSP)*, pages 835–838, 1997.

[64] S. C. Schwartz and Y. S. Yeh. On the distribution function and moments of power sums with log-normal components. *The Bell System Technical Journal*, 61(7):1441–1462, 1982.

[65] J. S. Sohn, N. S. Kim, and W. Sung. A statistical model-based voice activity detection. *IEEE Signal Processing Letters*, 6(1):1–3, 1999.

[66] V. Stouten. *Robust Automatic Speech Recognition in Time-Varying Environments.* PhD thesis, Katholieke Universiteit Leuven, 2006.

[67] V. Stouten, H. Van hamme, and P. Wambacq. Joint removal of additive and convolutional noise with model-based feature enhancement. In *Proc. IEEE Int. Conf. on Acoust., Speech, Signal Processing (ICASSP)*, pages 949–952, 2004.

[68] V. Stouten, H. Van hamme, and P. Wambacq. Effect of phase-sensitive environment model and higher order VTS on noisy speech feature enhancement. In *Proc. IEEE Int. Conf. on Acoust., Speech, Signal Processing (ICASSP)*, pages 433–436, 2005.

[69] V. Stouten, H. Van hamme, and P. Wambacq. Application of minimum statistics and minima controlled recursive averaging methods to estimate a cepstral noise model for robust ASR. In *Proc. IEEE Int. Conf. on Acoust., Speech, Signal Processing (ICASSP)*, pages 765–768, 2006.

[70] S. Theodoridis and K. Koutroumbas. *Pattern Recognition.* Elsevier Academic Press, second edition, 2003.

[71] C. W. Therrien. *Discrete Random Signals and Statistical Signal Processing.* Prentice Hall, 1992.

[72] F. Valente. *Variational Bayesian Methods for Audio Indexing.* PhD thesis, University of Nice Sophia-Antipolis, 2005.

[73] F. Valente and C. J. Wellekens. Variational Bayesian GMM for speech recognition. In *Proc. European Conf. on Speech Comm. and Technology (Eurospeech)*, pages 441–444, 2003.

[74] S. Watanabe. *Speech Recognition Based on a Bayesian Approach.* PhD thesis, Waseda University, 2006.

[75] S. Watanabe, Y. Minami, A. Nakamura, and N. Ueda. Variational Bayesian estimation and clustering for speech recognition. *IEEE Trans. on Audio, Speech, and Language Processing*, 12:365–381, 2004.

[76] S. Watanabe, A. Sako, and A. Nakamura. Automatic determination of acoustic model topology using variational Bayesian estimation and clustering for large vocabulary continuous speech recognition. *IEEE Trans. on Audio, Speech, and Language Processing*, 14:855–871, 2006.

[77] S. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, X. Liu, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland. *The HTK Book.* Cambridge University Engineering Department, 2006.