Subspace Gaussian Mixture Models for Automatic Speech Recognition

Liang Lu

Doctor of Philosophy

Institute for Language, Cognition and Computation
School of Informatics
University of Edinburgh

2013

Abstract

In most of state-of-the-art speech recognition systems, Gaussian mixture models (GMMs) are used to model the density of the emitting states in the hidden Markov models (HMMs). In a conventional system, the model parameters of each GMM are estimated directly and independently given the alignment. This results a large number of model parameters to be estimated, and consequently, a large amount of training data is required to fit the model. In addition, different sources of acoustic variability that impact the accuracy of a recogniser such as pronunciation variation, accent, speaker factor and environmental noise are only weakly modelled and factorized by adaptation techniques such as maximum likelihood linear regression (MLLR), maximum a posteriori adaptation (MAP) and vocal tract length normalisation (VTLN). In this thesis, we will discuss an alternative acoustic modelling approach — the subspace Gaussian mixture model (SGMM), which is expected to deal with these two issues better. In an SGMM, the model parameters are derived from low-dimensional model and speaker subspaces that can capture phonetic and speaker correlations. Given these subspaces, only a small number of state-dependent parameters are required to derive the corresponding GMMs. Hence, the total number of model parameters can be reduced, which allows acoustic modelling with a limited amount of training data. In addition, the SGMM-based acoustic model factorizes the phonetic and speaker factors and within this framework, other source of acoustic variability may also be explored.

In this thesis, we propose a regularised model estimation for SGMMs, which avoids overtraining in case that the training data is sparse. We will also take advantage of the structure of SGMMs to explore cross-lingual acoustic modelling for low-resource speech recognition. Here, the model subspace is estimated from out-domain data and ported to the target language system. In this case, only the state-dependent parameters need to be estimated which relaxes the requirement of the amount of training data. To improve the robustness of SGMMs against environmental noise, we propose to apply the joint uncertainty decoding (JUD) technique that is shown to be efficient and effective. We will report experimental results on the Wall Street Journal (WSJ) database and GlobalPhone corpora to evaluate the regularisation and cross-lingual modelling of SGMMs. Noise compensation using JUD for SGMM acoustic models is evaluated on the Aurora 4 database.

Acknowledgements

First and foremost, I would like to thank my supervisor Prof. Steve Renals for his expert guidance, deep insight in the speech technology, patience and encouragement all the way through my doctoral study. It is a great pleasure to work with this wonderful mentor. I also owe a huge gratitude to Dr. Arnab Ghoshal, a colleague, friend and advisor, without whom, I would definitely have a much tougher experience to reach this thesis. Thanks also to KK Chin — my industry supervisor in Toshiba Cambridge research laboratory (CRL). It is a valuable and fruitful experience to work with KK during my research visit which directly leads to the results reported in Chapter 6, and lays the foundation for the work reported in Chapter 7.

I would like to express my gratitude to Dr. Michael Seltzer and Dr. Junichi Yamagishi to be my thesis examiners and for their insightful comments, and to Prof. Simon King to be the chair of my oral defence committee. Thanks to all the folks in CSTR for creating such a friendly and inspiring place to work. The coffee breaks and lunch hours are always full of joy, and it helps me to improve my English as well. I'm particularly thankful to Peter, Cassia, Oliver and Michael for sharing office with me and for all the technical and non-technical discussions. A big thank you also to the former CSTR members, Dong, Le for answering me many questions and helping me to settle down in my first year. The weekly pub with you is memorable. Thanks also to Ravi for the help with my experiments.

The visit to CRL was wonderful and thanks to Drs. Kate Knill, Javier Latorre and Masami Akamine for making it happen, and to Vincent for the help with the experiments. I'm also grateful to the Marie Curie fellowship for the generous funding which supports my living in Edinburgh as well as the conference travelling and research visit. Thanks to other research fellows within the SCALE project. It's an exciting experience to work toward a Ph.D with you all together and share ideas in the SCALE workshops. Also thanks to Prof. Dietrich Klakow and Cornelia Koeck for managing this project.

Thanks to the Kaldi research group for releasing the toolkit. It provides me a platform to explore my own ideas. Thanks also to Caroline, Avril, Nicola, Julie and Claire for the administrative support.

Finally, I owe too much to my parents for the lack of phone calls and visits. Thanks for your understanding and support and also to my parents-in-law for the delicious food. The biggest thanks goes to my wife Ji for always being there.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Liang Lu*)

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Automatic speech recognition (ASR) has achieved significant progress during the past couple of decades, and applications have been deployed in personal computers, the internet and mobile networks. The task of ASR is to transcribe the audio into the text, and due to the sequential nature of speech signals, the hidden Markov model (HMM) (Baum, 1972) has been proven to be an efficient framework for this task. The HMM model attempts to address the characteristics of a probabilistic sequence of observations that may not be a fixed function but instead changes according to a Markov chain. In the early years of HMM-based speech recognition systems, simple discrete density probabilistic function was used and the systems were mainly used to handle isolated, small vocabulary and speaker-dependent tasks (Barker, 1975; Jelinek, 1976). However, the Sphinx system (Lee, 1988) demonstrated that it is possible to build a HMM-based speech recogniser to perform continuous, large vocabulary and speaker-independent speech recognition. Later on, the application of the Gaussian mixture model (GMM) in the HMM-based speech recognisers was proven to be highly successful for large vocabulary tasks (Woodland et al., 1994) and it is still used in most of today's speech recognition systems.

In an HMM-GMM based speech recognition system, the GMM is used to model the state density which has several advantages. The acoustic variations introduced by pronunciation variation, accent, speaker factor and environmental noise etc, can be more accurately modelled by using mixtures of Gaussians compared to a single form of density function. This is vital for continuous speaker-independent tasks. Due to its relatively simple mathematical structure, an efficient model parameter estimation algorithm is available, which plays an important role for the success of HMM-GMM framework. In addition, advances within this framework in the 1990s including

model adaptation using techniques of the maximum a posteriori (MAP) and maximum likelihood linear regression (MLLR) families (Gauvain and Lee, 1994; Leggetter and Woodland, 1995), adaptive training (Anastasakos et al., 1996), noise-robustness (Moreno et al., 1996; Gales, 1995) etc, further improved the power of HMM-GMM speech recognisers.

Despite of the success of HMM-GMM framework, there are several shortcomings that need to be addressed. In a conventional system, the GMM parameters for each HMM state are estimated independently given the alignment. This results in a very large number of model parameters to be trained, especially for context-dependent acoustic models, and consequently, a large amount of training data is required to fit the model. Parameter tying techniques such as tree-based state tying (Young et al., 1994) can reduce the number of effective parameters, but in order to maintain the discrimination power of the model among states, it can not entirely solve the problem. On the other hand, all kinds of acoustic variations caused by pronunciation variation, speaker factors and environmental noise etc., which significantly affect the recognition accuracy, are not factorized in conventional GMM-based acoustic models. Model adaptation techniques like MLLR only addresses this issue in a crude fashion regardless of the source of variation. Not surprisingly, factorizing the underlying acoustic variations and coping with them accordingly would provide further gains as indicated by recent works (Wang and Gales, 2011; Seltzer and Acero, 2011).

These issues lead us to investigate the subspace Gaussian mixture models (SGMMs) proposed by Povey et al. (2011a). This type of model uses the concept of basis decomposition to reduce the model parameters. This has been extensively explored in the field of speech processing through techniques such as eigenvoices (Kuhn et al., 2000) and cluster adaptive training (CAT) (Gales, 1998a) for speaker adaptation and maximum likelihood linear transformation (MLLT) (Gales, 1998b) as well as its extended version (Olsen and Gopinath, 2002), and SPAM model (Axelrod et al., 2005) for full covariance modelling. It also borrows the idea of factor analysis from speaker recognition (Kenny, 2005) to factorize the variations in the model. In an SGMM, the model parameters are derived from the globally shared model subspace with very low-dimensional state-dependent vectors. The model subspace captures the major variations among the phonetic states. With this informative prior, only a small number of additional parameters are required to derive the state-dependent GMMs. This reduces the total number of model parameters and allows more accurate model estimation with a limited amount of training data. In addition, a speaker subspace can also be intro-

duced which enable SGMMs to factorize the phonetic and speaker factors in the model domain. Recent research has shown that an SGMM acoustic model may result in more accurate speech recognition in both monolingual, multilingual and cross-lingual settings (Povey et al., 2011a; Burget et al., 2010; Lu et al., 2011a). Recently, we have also shown that it is possible for an SGMM to outperform its GMM counterpart in noisy environment (Lu et al., 2012a).

1.1 Contribution of the Thesis

After giving a detailed review of the SGMM-based acoustic modelling method, we first present the regularized model estimation for SGMMs (Lu et al., 2011b). In particular, we introduce a penalty to the original objective function of (sub-)state vectors to improve the estimation accuracy when the amount of training data is very limited. We studied both the ℓ_1-norm and ℓ_2-norm regularization penalties, as well as their combined form, the elastic-net penalty, and compare their performance on the WSJ-5k speech recognition task.

Following (Burget et al., 2010), we applied the SGMM acoustic model to the cross-lingual task using the GlobalPhone corpus (Schultz, 2002). We took the advantage of the structure of SGMM-based acoustic model (that the globally shared parameters do not depend on the HMM topology) and we estimated these parameters from out-of-domain data. We then applied them to the target language system with very limited training data, and only the sate-dependent parameters were estimated. This approach can significantly improve the recognition accuracy of the speech recognition system in limited resource conditions (Lu et al., 2011a, 2013b). In addition, the method of regularized (sub-)state vector estimation can also be applied which allows a larger dimensional subspace been used. Finally, we adapted the model subspace using the MAP criterion to reduce the mismatch between the out-domain and in-domain data (Lu et al., 2012c).

The accuracy of speech recognition systems normally degrades dramatically in a noisy environment. To improve robustness against background noise remains one of the focuses of research on speech recognition. Recent research has indicated that an SGMM acoustic model may result in more accurate speech recognition compared to its GMM counterpart, in both monolingual and multilingual settings (Povey et al., 2011a; Burget et al., 2010; Lu et al., 2011a), however, we will show in this thesis that the standard SGMM acoustic model suffers similar problems to conventional GMMs in noisy

conditions and that the gains disappear in this case. We improved the robustness of SGMMs against noise by applying the joint uncertainty decoding (JUD) approach (Liao and Gales, 2005). We found that JUD can be successfully applied to SGMMs, resulting in state-of-the-art performance on the Aurora 4 corpus (Lu et al., 2012a, 2013a). Unscented transforms were also studied for noise compensation of SGMMs (Lu et al., 2012b). With multi-style training data, we also investigated the noise adaptive training (NAT) technique for model training of SGMMs (Lu et al., 2013c).

Some of the ideas and results in this thesis have been published in reviewed conference and journal papers as follows:

- L. Lu, A. Ghoshal and S. Renals, "Regularized subspace Gaussian mixture models for speech recognition", in *IEEE Signal Processing Letters*, 2011.

- L. Lu, A. Ghoshal and S. Renals, "Regularized subspace Gaussian mixture models for cross-lingual speech recognition", in *Proc. ASRU*, 2011.

- L. Lu, A. Ghoshal and S. Renals, "Maximum a posteriori adaptation of subspace Gaussian mixture models for cross-lingual speech recognition", in *Proc. ICASSP*, 2012.

- L. Lu, KK Chin, A. Ghoshal and S. Renals, "Noise compensation for subspace Gaussian mixture models", in *Proc. Interspeech*, 2012.

- L. Lu, A. Ghoshal and S. Renals, "Joint uncertainty decoding with unscented transforms for noise robust subspace Gaussian mixture models", in *Proc. SAPA-SCALE workshop*, 2012.

- L. Lu, A. Ghoshal and S. Renals, "Noise adaptive training for subspace Gaussian mixture models", in *Proc. Interspeech*, 2013.

- L. Lu, KK Chin, A. Ghoshal and S. Renals, "Joint uncertainty decoding for noise robust subspace Gaussian mixture models", IEEE Transactions on Audio, Speech and Language Processing, 2013

- L. Lu, A. Ghoshal and S. Renals, "Cross-lingual subspace Gaussian mixture models for low-resource speech recognition", IEEE Transactions on Audio, Speech and Language Processing, 2013 (submitted).

1.2 Structure of the Thesis

The reminder of the thesis is organised as follows:

- In Chapter 2, we present an overview of an automatic speech recognition system based on the HMM-GMM framework including front-end feature processing, acoustic modelling and language modelling, as well as decoding and adaptation.

- Chapter 3 reviews SGMM-based acoustic modelling in detail. We start with a discussion of related techniques, and then move on to presenting detailed model training procedure. Model estimation based on the maximum likelihood criterion is discussed for different parameter types in an SGMM. Possible model extension techniques are also described.

- Chapter 4 describes the regularized estimation of SGMMs in which a regularization penalty is introduced to the auxiliary function to avoid model overfitting. We compare three types of regularization penalties and present the optimization algorithm to solve the ℓ_1-norm problem.

- In Chapter 5, we study cross-lingual SGMM acoustic models for low-resource speech recognition. We investigate the estimation of the globally shared parameter set in a multilingual fashion for the target language system and apply ℓ_1-norm regularization to the state-dependent parameters to avoid overtraining. MAP adaptation of model subspace and cross-lingual speaker adaptive training using the speaker subspace are also investigated.

- Chapter 6 presents our implementation of joint uncertainty decoding for noise compensation of SGMMs. We present detailed mathematical derivations of gradient-based noise model estimation. We evaluate the performance of JUD/SGMM system on the Aurora 4 database and compare it's recognition accuracy to GMM-based systems with both VTS and JUD noise compensation. The unscented transforms (UT) based sampling technique is also investigated in the framework of JUD for compensating SGMMs agains noise.

- In Chapter 7, we describe the noise adaptive training (NAT) algorithm for SGMM acoustic models which is based on the generative reformulation of JUD. NAT allows for acoustic model training with multi-style training data and brings further improvement for the noise-robust speech recognition task.

- In Chapter 8, we summarise the thesis and point out directions for future works.

1.3 Notation

We try to maintain a consistency and avoid ambiguity over the mathematical notation throughout the thesis. In some cases, however, a symbol is inevitably reused to represent a different parameter in order to be consistent with the reference. In these cases, it will be made clear by the context. Overall, the following symbols are frequently used throughout the thesis.

$\boldsymbol{\mu}_{jm}$	Mean vector of Gaussian m, state j in an GMM acoustic model
$\boldsymbol{\Sigma}_{jm}$	(Diagonal) covariance matrix of Gaussian m, state j in an GMM acoustic model
w_{jm}	Weight of Gaussian m, state j in an GMM acoustic model
$\boldsymbol{\mu}_{jki}$	Mean vector of Gaussian i, sub-state k, state j in an SGMM acoustic model
$\boldsymbol{\Sigma}_i$	(Full) global covariance matrix i in an SGMM acoustic model
$\mathbf{M}_i$	Model projection matrix i in an SGMM acoustic model
$\mathbf{N}_i$	Speaker projection matrix i in an SGMM acoustic model
$\mathbf{w}_i$	Weight projection vector i in an SGMM acoustic model
$\mathbf{v}_{jk}$	Sub-state vector of sub-state k, state j in an SGMM acoustic model
$\mathbf{v}^{(s)}$	Speaker vector of speaker index s in an SGMM acoustic model
w_{jki}	Weight of Gaussian i, sub-state k, state j in an SGMM acoustic model
c_{jk}	Weight of sub-state k, state j in an SGMM acoustic model
$\gamma_{jm}(t)$	Posterior probability of Gaussian m of state j in an GMM acoustic model at time t
γ_{jm}	Summed posterior probability over time index t: $\gamma_{jm} = \sum_t \gamma_{jm}(t)$
$\gamma_{jki}(t)$	Posterior probability for Gaussian i, sub-state k, state j for an SGMM acoustic model at time t
γ_{jki}	Summed posterior probability over time index t: $\gamma_{jki} = \sum_t \gamma_{jki}(t)$
$\mathcal{N}(\mathbf{x}\|\boldsymbol{\mu},\boldsymbol{\Sigma})$	Multivariate Gaussian distribution for random variable $\mathbf{x}$ with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$
$Q(\theta)$	Expectation-maximisation auxiliary function for a particular parameter θ
$b_j(\cdot)$	Output density function of state j
$\mathbf{n}_t$	Additive noise variable at time t
$\mathbf{h}_t$	Channel noise variable at time t
$\boldsymbol{\mu}_n$	Additive noise mean, only for static feature
$\boldsymbol{\Sigma}_n$	(Diagonal) additive noise covariance, only for statice feature
$\boldsymbol{\mu}_h$	Channel noise mean, only for static feature
$\mathbf{y}_t$	(Noisy) speech observation at time t. $\mathbf{y}_t$ is a concatenation of its static coefficients $\mathbf{y}_{s,t}$, delta coefficients $\Delta\mathbf{y}_t$ and delta-delta coefficients $\Delta^2\mathbf{y}_t$
$\mathbf{x}_t$	Hidden clean speech variable at time t
q_t	State index at time t
$\mathbf{Q}$	State sequence $\mathbf{Q} = (q_1, \ldots, q_T)$

1.4 Abbreviation

CMN	Cepstral mean normalisaiton
CVN	Cepstral variance normalisation
CMLLR	Constraint maximum likelihood linear regression
DPMC	Data-driven parallel model combination
DFT	Discrete Fourier transform
DCT	Discrete cosine transform
EM	Expectation maximisation
GMM	Gaussian mixture model
HMM	Hidden Markov model
JUD	Joint uncertainty decoding
JFA	Joint factor analysis
MFCC	Mel frequency cepstrum coefficient
MAP	Maximum a posteriori
MLLR	Maximum likelihood linear regression
MAPLR	Maximum a posteriori linear regression
NCMLLR	Noisy constraint maximum likelihood linear regression
NAT	Noise adaptive training
PLP	Perceptual linear prediction
PMC	Parallel model combination
SGMM	Subspace Gaussian mixture model
SPLICE	Stereo piece-wise linear compensation for environments
UBM	Universal background model
UT	Unscented transform
VTS	Vector Taylor series
WER	Word error rate
WSJ	Wall Street Journal

Chapter 2

Automatic Speech Recognition

The task of automatic speech recognition is to transcribe the audio speech $\mathbf{S}$ into its word sequence $\mathcal{W}$ using a recogniser $\mathcal{M}$. This can be done by solving the following equation

$$\mathcal{W}_h' = \arg\max_{\mathcal{W}} P(\mathcal{W}|\mathbf{S}, \mathcal{M}) \tag{2.1}$$

where $\mathcal{W}_h'$ denotes the most likely word sequence. However, it's normally difficult to derive the posterior of a word sequence directly. Bayes' rule can be applied which gives

$$\begin{aligned} \mathcal{W}_h' &= \arg\max_{\mathcal{W}} P(\mathcal{W}|\mathbf{S}, \mathcal{M}) \\ &= \arg\max_{\mathcal{W}} \frac{p(\mathbf{S}|\mathcal{W}, \mathcal{M}_a) P(\mathcal{W}|\mathcal{M}_l)}{p(\mathbf{S}, \mathcal{M})} \\ &= \arg\max_{\mathcal{W}} p(\mathbf{S}|\mathcal{W}, \mathcal{M}_a) P(\mathcal{W}|\mathcal{M}_l) \end{aligned} \tag{2.2}$$

where the recogniser $\mathcal{M}$ is decomposed into the acoustic model $\mathcal{M}_a$ and the language model $\mathcal{M}_l$. $p(\mathbf{S}|\mathcal{W}, \mathcal{M}_a)$ is the likelihood of the acoustic model parameters $\mathcal{M}_a$ given the audio signal $\mathbf{S}$ for word sequence $\mathcal{W}$, and $P(\mathcal{W}|\mathcal{M}_l)$ denotes the prior probability of $\mathcal{W}$ given the language model parameters $\mathcal{M}_l$. The normalisation probability $p(\mathbf{S}, \mathcal{M})$ is removed since it does not affect the search of the word sequence.

A diagram of a standard speech recogniser is shown in Figure 2.1. It can be seen from equation (2.2) that the task of building a speech recogniser is composed of two sub-tasks, i.e. acoustic modelling and language modelling. The aim of acoustic modelling is to train a model $\mathcal{M}_a$ which can explain the speech signals $\mathbf{S}$ well given a word sequence $\mathcal{W}$. Due to the sequential nature of speech signals, hidden Markov models (HMMs) are found to be effective for this task (Rabiner, 1989). However, the

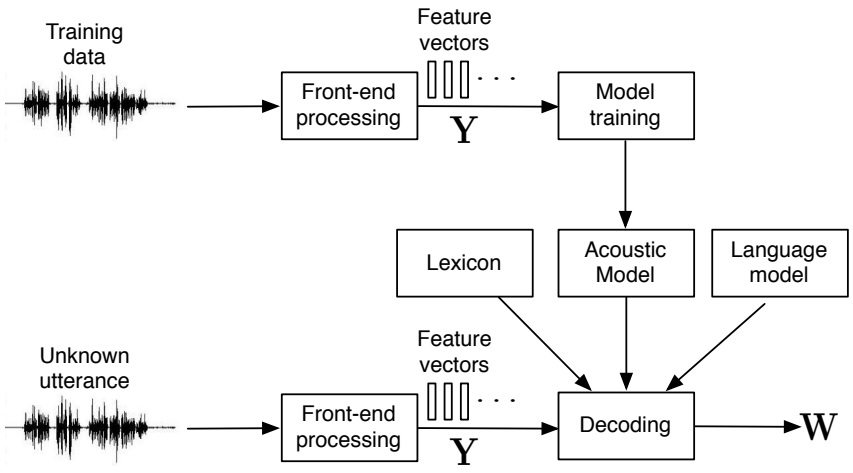


Figure 2.1: Diagram of a standard speech recognition system.

raw speech signal may not be suitable for acoustic modelling, hence there is a need for a front-end processing step which transforms the speech signal **S** into feature vectors **Y**. The ideal feature vectors **Y** should be invariant to extraneous factors to speech recognition such as speaker factors, pronunciation variability and environmental noise. However, in practice, the feature processing step can't normalise all of the variability and the acoustic models are expected to share the task. Language models, on the other hand, try to predict the prior distribution of the word sequence $\mathcal{W}$ before the observation of speech signals. Conventional language models are based on the frequency of n-grams which assume that the distribution of each word depends on the previous $n-1$ words. The rest of this chapter presents an overview of a standard speech recognition system.

2.1 Front-end processing

As stated above, the aim of front-end processing is to extract the feature vectors **Y** suitable for speech recognition from the raw speech signals **S**. To extract the features, a sliding window with overlaps is applied to the speech signals first. The speech signals are assumed to be stationary within each window if the length of the window is sufficiently small, and there are subjected to a series of feature transformation functions to generate the feature vectors, as shown in Figure 2.2. According to the feature transformation functions that been used, different types of feature vectors can be derived. Currently, there are two popular forms of feature representation: mel frequency cepstrum coefficients (MFCC) (Davis and Mermelstein, 1980) and perceptual linear pre-

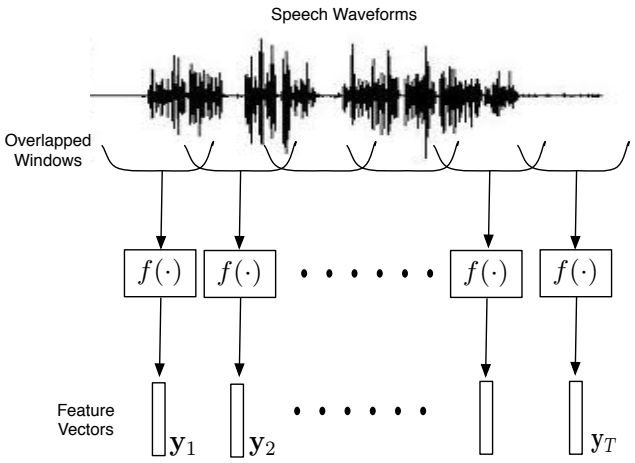


Figure 2.2: Diagram for feature extraction. A window function is first applied to a speech segment, followed by a feature processing function $f(\cdot)$.

diction (PLP) (Hermansky, 1990). MFCC analysis uses a mel-scale filterbank which is designed to model the hair spacings along the basilar membrane. The left side column of Figure 2.3 shows the extraction of MFCC features from speech signals.

PLP is formulated as a method for deriving a more auditory-like spectrum based on linear predictive (LP) analysis of speech. Conventional LP analysis approximates the high energy areas of the spectrum and smooths out the finer harmonic structures. This estimate is applied equally to all frequencies which is inconsistent with human hearing. The auditory-like spectrum in PLP is achieved by making some engineering approximations of the psychophysical attributes of the human hearing process. The right side column of Figure 2.3 shows the extraction of PLP features from speech signals.

2.1.1 MFCC and PLP

The analysis of MFCC and PLP are similar in several stages, which are linked by the broken arrows in Figure 2.3. A more detailed comparison is given as follows (Milner, 1993).

- Discrete Fourier transform (DFT) — Both MFCC and PLP analysis obtain a short-term power spectrum by applying DFT to a frame of windowed speech.
- Critical-band analysis and mel-scale filterbank — Both MFCC and PLP analysis employ an auditory-based warping of the frequency axis derived from the frequency sensitivity of human hearing. At regular points along the two scales,

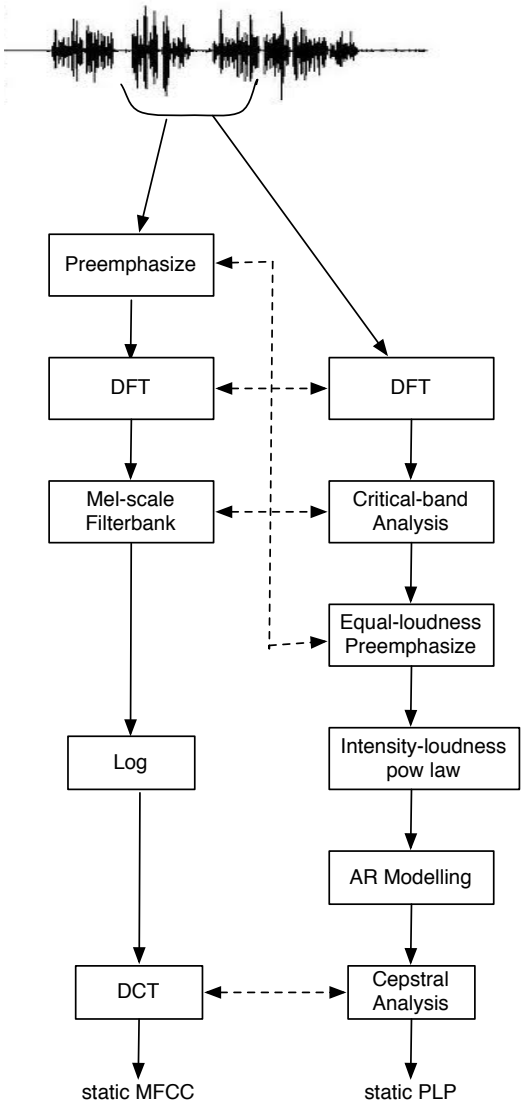


Figure 2.3: Flowcharts for MFCC and PLP feature extraction. The broken lines link the similar processing steps between the two types of feature.

windowing functions are applied which quantise the frequency spectrum. Mel-scale filterbank analysis uses triangular shaped windows whereas in PLP analysis the window shape is designed to simulate critical-band masking curves. Both critical band analysis and mel-filterbank analysis can be viewed as applying a set of basis functions to the power spectrum of the speech signal. Figure 2.4 shows the shape of the windows used in PLP and MFCC analysis.

- Pre-emphasis — To compensate for the unequal sensitivity of human hearing across frequency, PLP analysis scales the critical bands amplitudes according to

an equal-loudness pre-emphasis function, for instance

$$E(\omega) = \frac{\left(\omega^2 + 56.8 \times 10^6\right)\omega^4}{(\omega^2 + 6.3 \times 10^6)^2(\omega^2 + 0.38 \times 10^9)} \tag{2.3}$$

where ω is the angular frequency. In MFCC analysis, pre-emphasis is applied in the time-domain. For instance, a first-order high pass filter is normally used

$$H(z) = 1 - \alpha z^{-1} \tag{2.4}$$

The pre-emphasis coefficient α is normally set to be 0.97.

- Intensity-loudness power law and log algorithm — This processing stage models the non-linear relation between the intensity of sound and its perceived loudness for both MFCC and PLP features. Cubic root compression of critical-band energies is used for PLP analysis whereas for MFCC, logarithmic compression of the mel-filterbank channels is applied. Both operations have a very similar effect.

- Cepstral analysis and discrete cosine transform (DCT) — The spectral features are normally converted into cepstral domain in order to de-correlate the feature coefficients and reduce the dimensionality. MFCC analysis compute the cepstral features form the log mel-filterbank using a DCT function. However in PLP, the critical-band spectrum is converted into a smaller number of LP coefficients by auto-regression (AR) modelling, and cepstral coefficients are computed form the LP coefficients.

2.1.2 Dynamic features

As shown in previous section, the static MFCC and PLP features are extracted based on the assumption that the speech within each window is independent of others. However, this is not true since the windows are overlapped. Dynamic features can be appended to capture this high order correlation between frames close to each other (Furui, 1986). The delta coefficients may be computed by simple differences, e.g. $\Delta\mathbf{y}_{s,t} = \mathbf{y}_{s,t+2} - \mathbf{y}_{s,t-2}$, or a linear regression to approximate the temporal derivative:

$$\Delta\mathbf{y}_t = \frac{\sum_{\delta=1}^{\Delta} \delta(\mathbf{y}_{s,t+\delta} - \mathbf{y}_{s,t-\delta})}{2\sum_{\delta=1}^{\Delta} \delta^2} \tag{2.5}$$

where $\mathbf{y}_{s,t}$ is the static cepstral feature vector, indicated by the subscript $_s$, and $\Delta\mathbf{y}_t$ is the delta feature vector. A higher order delta-delta (acceleration) feature vector $\Delta^2\mathbf{y}_t$ may

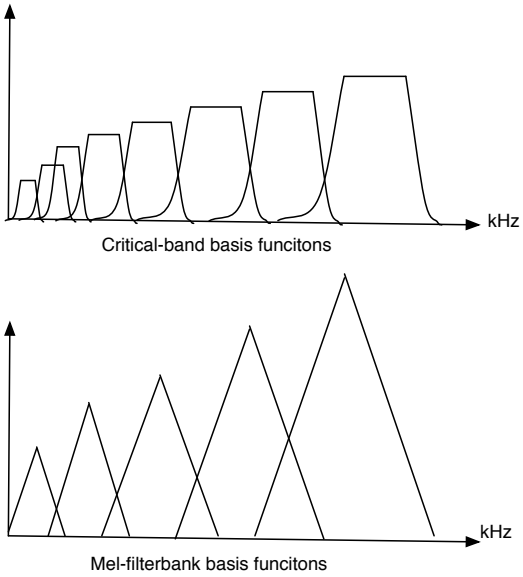


Figure 2.4: Critical-band and mel-filterbank basis functions used for PLP and MFCC feature extraciton.

be computed in the same manner. The final feature vector used for speech recognition is the concatenation of static and dynamic coefficients:

$$\mathbf{y}_t = \begin{bmatrix} \mathbf{y}_{s,t} \\ \Delta \mathbf{y}_t \\ \Delta^2 \mathbf{y}_t \end{bmatrix} \tag{2.6}$$

Higher order dynamic coefficients can also be computed similarly, but they normally do not bring additional notable gains in recognition accuracy.

2.2 Acoustic modelling

Given the feature vectors $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_t, \ldots, \mathbf{y}_T\}$ that are extracted for the speech signal $\mathbf{S}$, the acoustic model $\mathcal{M}_a$ is used to generate the score $p(\mathbf{Y}|\mathcal{W}, \mathcal{M}_a)$ for the word sequence $\mathcal{W}$. The hidden Markov model (HMM) has proven successful in acoustic modelling since it can well estimate the time-varying nature of speech. Good reviews of using HMMs for speech recognition can be seen in (Rabiner, 1989; Gales and Young, 2008). An example of a left-to-right HMM with 3 emitting states and 2 non-emitting states are depicted by Figure 2.5, which is used as building blocks for most state-of-the-art speech recognitions systems. Here, $a_{ij} = p(q_t = j|q_{t-1} = i)$ denotes the state transition probability where q_t is the state at time t. This means that at every time

instance, HMM makes a transition from the current state to one of its connected states according to the transition probability. Given the current state $q_t = j$, the observation will be generated according to the probability density function (pdf) denoted as $b_j(\mathbf{y}_t)$. For the majority of state-of-the-art HMM-based speech recognisers, the emitting density is modelled by a Gaussian mixture model (GMM). For state j, the model can be expressed as

$$\begin{aligned} b_j(\mathbf{y}_t) &= p(\mathbf{y}_t|q_t = j, \mathcal{M}_a) \\ &= \sum_{m=1}^{M} w_{jm}\mathcal{N}(\mathbf{y}_t|\boldsymbol{\mu}_{jm}, \boldsymbol{\Sigma}_{jm}) \end{aligned} \tag{2.7}$$

where m is the Gaussian component index, w_{jm},$\boldsymbol{\mu}_{jm}$ and $\boldsymbol{\Sigma}_{jm}$ are mixture weight, mean and covariance for component m.

The HMM depicted by Figure 2.5 is designed following two important assumptions, i.e. the *first-oder Markov assumption* is used which assume the probability of a state q_t at time t is only dependent on the previous state q_{t-1} as

$$p(q_t|q_{t-1}, q_{t-2}, \ldots, q_1) = p(q_t|q_{t-1}) \tag{2.8}$$

and the *conditional independence assumption* which assumes that given the state sequence, the observations are conditionally independent as

$$p(\mathbf{y}_t|\mathbf{y}_{t-1}, \ldots, \mathbf{y}_1; q_t, \ldots, q_1) = p(\mathbf{y}_t|q_t) \tag{2.9}$$

These two assumptions significantly simplify the application of HMM to speech recognition task. For instance, let's $\mathbf{Q} = \{q_1, q_2, \ldots, q_T\}$ is the possible state sequence for transcription $\mathcal{W}$, by HMMs, the likelihood $p(\mathbf{Y}|\mathcal{W}, \mathcal{M}_a)$ can be computed as

$$p(\mathbf{Y}|\mathcal{W}, \mathcal{M}_a) = \sum_{\mathbf{Q}} p(\mathbf{Y}|\mathbf{Q}, \mathcal{M}_a)p(\mathbf{Q}|\mathcal{W}, \mathcal{M}_a) \tag{2.10}$$

This likelihood funciton is hard to be computed in general since the state sequence $\mathbf{Q}$ is hidden and the space for $\mathbf{Q}$ may be large. However, using the previous two assumptions, the likelihood function (2.10) can be decomposed as

$$\begin{aligned} p(\mathbf{Y}|\mathcal{W}, \mathcal{M}_a) &= \sum_{\mathbf{Q}} \prod_{t=1}^{T} p(\mathbf{y}_t|q_t; \mathcal{M}_a)p(q_t|q_{t-1}; \mathcal{M}_a) \\ &= \sum_{\mathbf{Q}} a_{q_0 q_1} \prod_{t=1}^{T} b_{q_t}(\mathbf{y}_t)a_{q_t q_{t+1}} \end{aligned} \tag{2.11}$$

Here, q_0 and q_{T+1} denote the non-emitting entry and exit states. The acoustic model parameters can now be expressed as $\mathcal{M}_a = \big(\{a_{ij}\}, \{b_j(\cdot)\}\big)$. To build a HMM-based

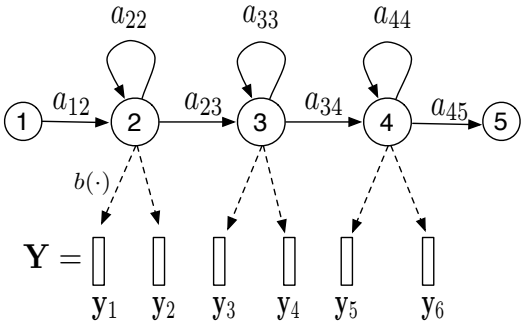


Figure 2.5: Topology of a 5-states left-right HMM with 3 emitting states and 2 non-emitting states. State 1 is the entrance state and state 5 is the exit state and both of them are non-emitting. a_{ij} denotes the state transition probability from state i to state j. $b(\cdot)$ is the density funciton.

speech recognition system, there are three key issues that should be addressed: the likelihood evaluation as equation (2.11); the parameter estimation of the model $\mathcal{M}_a$ and decoding (Rabiner, 1989). The first two issues are briefly address in the following subsections, while the last issue is addressed in section 2.4.

2.2.1 Likelihood evaluation

Since the state sequence $(q_0, q_1, q_2, \ldots, q_T, q_{T+1})$ for the observation $\mathbf{Y}$ is hidden, the likelihood is obtained by summing over all possible state sequences in $\mathbf{Q}$ as in equation (2.11). If there are N states in the HMM, the number of possible state sequences in $\mathbf{Q}$ will be N^T for T frames, and hence the evaluation of this likelihood using exhaustive search has the complexity of $O(TN^T)$ which is computationally infeasible. However, this problem can be addressed by using the *forward-backword* algorithm (Baum et al., 1970) which is an example of the sum-product algorithm (Bishop, 2006). It can reduce the computational complexity to $\mathbf{O}(TN^2)$ which scales linearly, instead of exponentially, with the length of the HMM as we will see later.

First, we denote the forward probability $\alpha_j(t)$ and the backward probability $\beta_j(t)$ as

$$\alpha_j(t) = p(\mathbf{y}_1, \ldots, \mathbf{y}_t, q_t = j | \mathcal{M}_a) \tag{2.12}$$

$$\beta_j(t) = p(\mathbf{y}_{t+1}, \ldots, \mathbf{y}_T | q_t = j; \mathcal{M}_a) \tag{2.13}$$

These two probabilities can be computed recursively as

$$\begin{aligned}\alpha_j(t) &= p(\mathbf{y}_1,\ldots,\mathbf{y}_t,q_t=j|\mathcal{M}_a)\\ &= p(\mathbf{y}_t|q_t=j;\mathcal{M}_a)\sum_{i=1}^{N}p(q_t=j|q_{t-1}=i;\mathcal{M}_a)p(\mathbf{y}_1,\ldots,\mathbf{y}_{t-1},q_{t-1}=i|\mathcal{M}_a)\\ &= b_j(\mathbf{y}_t)\sum_{i=1}^{N}a_{ij}\alpha_i(t-1) \qquad (2.14)\\ \beta_j(t) &= p(\mathbf{y}_{t+1},\ldots,\mathbf{y}_T|q_t=j;\mathcal{M}_a)\\ &= \sum_{i=1}^{N}p(q_{t+1}=i|q_t=j;\mathcal{M}_a)p(\mathbf{y}_{t+1}|q_{t+1}=i;\mathcal{M}_a)p(\mathbf{y}_{t+2},\ldots,\mathbf{y}_T|q_{t+1}=i;\mathcal{M}_a)\\ &= \sum_{i=1}^{N}a_{ji}b_i(\mathbf{y}_{t+1})\beta_i(t+1) \qquad (2.15)\end{aligned}$$

where $1<t\leq T$. The value of α and β are initialized as

$$\alpha_{q_0}(0)=1,\quad \alpha_j(1)=a_{q_0j}b_j(\mathbf{y}_1),\quad a_{q_0j}=1, \qquad (2.16)$$

$$\beta_{q_{T+1}}(T+1)=1,\quad \beta_j(T)=a_{jq_{T+1}}=1,\qquad for\, 1<j<N \qquad (2.17)$$

As stated before, q_0 and q_{T+1} denote the entry and exit states respectively. Now, we can simply obtain the likelihood $p(\mathbf{Y}|\mathcal{M}_a)$ as

$$\begin{aligned}p(\mathbf{Y}|\mathcal{M}_a) &= \sum_{j}^{N}p(\mathbf{y}_1,\ldots,\mathbf{y}_t,\mathbf{y}_{t+1},\ldots,\mathbf{y}_T,q_t=j|\mathcal{M}_a)\\ &= \sum_{j}^{N}\alpha_j(t)\beta_j(t) \qquad (2.18)\end{aligned}$$

This formula can be evaluated for any convenient choice of t. For instance, we may just run the α recursion from the start to the end of the HMM chain to obtain the forward probabilities $\alpha_\bullet(T)$, and the likelihood can then be computed as

$$p(\mathbf{Y}|\mathcal{W};\mathcal{M}_a)=\sum_{j}^{N}\alpha_j(T) \qquad (2.19)$$

where we have made use of the fact that the $\beta_\bullet(T)$ is a unit vector, and hence the β recursion is not required. From the recursion formula of α, we can see that the computation of $\alpha_\bullet(T)$ requires the cost of $O(TN^2)$. The forward-backward algorithm will also be used to estimate the model parameters as discussed in the next subsection.

2.2.2 Parameter estimation

Given a set of training data $\mathbf{Y}$, the HMM model parameters can be estimated using the Maximum Likelihood (ML) criterion which is

$$\begin{aligned}\hat{\mathcal{M}}_a &= \arg\max_{\mathcal{M}_a} \log p(\mathbf{Y}|\mathcal{W};\mathcal{M}_a) \\ &= \arg\max_{\mathcal{M}_a} \log \sum_{\mathbf{Q}} p(\mathbf{Y},\mathbf{Q}|\mathcal{W};\mathcal{M}_a)\end{aligned} \tag{2.20}$$

Where $\hat{\mathcal{M}}_a$ denotes the 'new' model parameters. This function can not be solved analytically since the latent variable space for $\mathbf{Q}$ maybe very large, but we can seek a local maximum by using the expectation maximisation (EM) algorithm (Dempster et al., 1977) which iteratively update the model parameters by maximising the lower bound of the log-likelihood function $\log p(\mathbf{Y}|\mathcal{W};\mathcal{M}_a)$, which is also normally referred as auxiliary function in the context of acoustic modelling. The key idea is that the 'old' model parameters are used to estimate the posteriors of the latent variables, and given the posteriors, we can use them as labels to derive the explicit 'pseudo' likelihood function, i.e. the auxiliary function:

$$\begin{aligned}\log p(\mathbf{Y}|\mathcal{W};\mathcal{M}_a) &= \underbrace{\sum_{\mathbf{Q}} P\left(\mathbf{Q}|\mathbf{Y},\mathcal{W};\check{\mathcal{M}}_a\right)}_{=1} \log p\left(\mathbf{Y}|\mathcal{W};\mathcal{M}_a\right) \\ &= \sum_{\mathbf{Q}} P\left(\mathbf{Q}|\mathbf{Y},\mathcal{W};\check{\mathcal{M}}_a\right) \underbrace{\left\{\log p(\mathbf{Y},\mathbf{Q}|\mathcal{W};\mathcal{M}_a) - \log P(\mathbf{Q}|\mathbf{Y},\mathcal{W};\mathcal{M}_a)\right\}}_{=\log p(\mathbf{Y}|\mathcal{W};\mathcal{M}_a)} \\ &= \sum_{\mathbf{Q}} P\left(\mathbf{Q}|\mathbf{Y},\mathcal{W};\check{\mathcal{M}}_a\right) \log p(\mathbf{Y},\mathbf{Q}|\mathcal{W};\mathcal{M}_a) \\ &\quad \underbrace{-\sum_{\mathbf{Q}} P\left(\mathbf{Q}|\mathbf{Y},\mathcal{W};\check{\mathcal{M}}_a\right) \log P(\mathbf{Q}|\mathbf{Y},\mathcal{W};\mathcal{M}_a)}_{\leq 0} \\ &\geq \sum_{\mathbf{Q}} P\left(\mathbf{Q}|\mathbf{Y},\mathcal{W};\check{\mathcal{M}}_a\right) \log p(\mathbf{Y},\mathbf{Q}|\mathcal{W};\mathcal{M}_a) \equiv Q(\mathcal{M}_a;\check{\mathcal{M}}_a)\end{aligned} \tag{2.21}$$

where $\check{\mathcal{M}}_a$ is the 'old' model parameters. Since the auxiliary function $Q(\mathcal{M}_a;\check{\mathcal{M}}_a)$ is the lower bound of the log-likelihood function, we can update the model parameters by improving $Q(\mathcal{M}_a;\check{\mathcal{M}}_a)$, which is to improve the value of $\log p(\mathbf{Y}|\mathcal{W};\mathcal{M}_a)$ implicitly. Hence, we update the model parameters $\mathcal{M}_a$ as

$$\hat{\mathcal{M}}_a = \arg\max_{\mathcal{M}_a} Q(\mathcal{M}_a;\check{\mathcal{M}}_a) \tag{2.22}$$

For the next iteration, we will set $\check{\mathcal{M}}_a \leftarrow \hat{\mathcal{M}}_a$, and the EM steps are repeated until we reach the convergence.

We now turn to the details of model estimation using EM algorithm. We unfold the auxiliary function as

$$\begin{aligned} Q(\mathcal{M}_a;\check{\mathcal{M}}_a) &= \sum_{\mathbf{Q}} p(\mathbf{Q}|\mathbf{Y},\mathcal{W};\check{\mathcal{M}}_a)\log p(\mathbf{Y},\mathbf{Q}|\mathbf{W};\mathcal{M}_a) \\ &= \sum_{t=1}^{T}\left\{\sum_{j=1}^{N} p(q_t=j|\mathbf{Y},\mathcal{W};\check{\mathcal{M}}_a)\log b_j(\mathbf{y}_t)\right. \\ &\quad \left. + \sum_{i=1}^{N}\sum_{j=1}^{N} p(q_{t-1}=i,q_t=j|\mathbf{Y},\mathcal{W};\check{\mathcal{M}}_a)\log a_{ij}\right\} \end{aligned} \tag{2.23}$$

where $p(q_t=j|\mathbf{Y},\mathcal{W};\check{\mathcal{M}}_a)$ and $p(q_{t-1}=i,q_t=j|\mathbf{Y},\mathcal{W};\check{\mathcal{M}}_a)$ are the first-order and second-order state posterior probabilities which can be obtained using the forward-backward algorithm as

$$\begin{aligned} \gamma_j(t) &= p(q_t=j|\mathbf{Y},\mathcal{W};\check{\mathcal{M}}_a) \\ &= \frac{p(q_t=j,\mathbf{Y}|\mathcal{W};\check{\mathcal{M}}_a)}{p(\mathbf{Y}|\mathcal{W};\check{\mathcal{M}}_a)} = \frac{\alpha_j(t)\beta_j(t)}{p(\mathbf{Y}|\mathcal{W};\check{\mathcal{M}}_a)} \end{aligned} \tag{2.24}$$

and

$$\begin{aligned} \zeta_{ij}(t) &= p(q_{t-1}=i,q_t=j|\mathbf{Y},\mathcal{W};\check{\mathcal{M}}_a) \\ &= \frac{p(q_{t-1}=i,q_t=j,\mathbf{Y}|\mathcal{W};\check{\mathcal{M}}_a)}{p(\mathbf{Y}|\mathcal{W};\check{\mathcal{M}}_a)} = \frac{\alpha_i(t-1)\alpha_{ij}b_j(\mathbf{y}_t)\beta_j(t)}{p(\mathbf{Y}|\mathcal{W};\check{\mathcal{M}}_a)} \end{aligned} \tag{2.25}$$

where $\alpha_j(t)$ and $\beta_j(t)$ are defined in equation (4.18) and (C.8). The likelihood $p(\mathbf{Y}|\mathcal{W};\check{\mathcal{M}}_a)$ is obtained by equation (2.19). Using the notation of $\gamma_j(t)$ and $\zeta_{ij}(t)$, the auxiliary function $Q(\mathcal{M}_a;\check{\mathcal{M}}_a)$ can be rewritten as

$$Q(\mathcal{M}_a;\check{\mathcal{M}}_a) = \sum_{t=1}^{T}\left\{\sum_{j=1}^{N}\gamma_j(t)\log b_j(\mathbf{y}_t) + \sum_{i=1}^{N}\sum_{j=1}^{N}\zeta_{ij}(t)\log a_{ij}\right\} \tag{2.26}$$

The acoustic model parameters $\mathcal{M}_a$ can be updated by maximizing the auxiliary function $Q(\mathcal{M}_a;\check{\mathcal{M}}_a)$. This includes the HMM state transition probability a_{ij} and state emitting model parameters $b_j(\mathbf{y}_t)$. If the GMM is used as the emitting state model, then the posterior probability of Gaussian components for each state is also required. For Gaussian component m in state j, the posterior can be computed as

$$\gamma_{jm}(t) = \gamma_j(t)\frac{p(\mathbf{y}_t,m|q_t=j)}{p(\mathbf{y}_t|q_t=j)} \tag{2.27}$$

Hence, the auxiliary function for the GMM-HMM acoustic model can be represented as

$$Q(\mathcal{M}_a;\check{\mathcal{M}}_a)=\sum_{t=1}^{T}\left\{\sum_{j=1}^{N}\sum_{m=1}^{M_j}\gamma_{jm}(t)\left[\log w_{jm}+\log b_{jm}(\mathbf{y}_t)\right]+\sum_{i=1}^{N}\sum_{j=1}^{N}\zeta_{ij}(t)\log a_{ij}\right\} \tag{2.28}$$

By maximizing the auxiliary function with respect to the acoustic model parameters, we can update the model as

$$\hat{a}_{ij}=\frac{\sum_{t=1}^{T}\zeta_{ij}(t)}{\sum_{t=1}^{T}\sum_{k=1}^{N}\zeta_{ik}(t)} \tag{2.29}$$

$$\hat{w}_{jm}=\frac{\sum_{t=1}^{T}\gamma_{jm}(t)}{\sum_{m=1}^{M_j}\sum_{t=1}^{T}\gamma_{jm}(t)} \tag{2.30}$$

$$\hat{\boldsymbol{\mu}}_{jm}=\frac{\sum_{t=1}^{T}\gamma_{jm}(t)\mathbf{y}_t}{\sum_{t=1}^{T}\gamma_{jm}(t)} \tag{2.31}$$

$$\hat{\boldsymbol{\Sigma}}_{jm}^{full}=\frac{\sum_{t=1}^{T}(\mathbf{y}_t-\hat{\mu}_{jm})(\mathbf{y}_t-\hat{\mu}_{jm})^T}{\sum_{t=1}^{T}\gamma_{jm}(t)} \tag{2.32}$$

In conventional GMM-based system, diagonal covariance matrices are normally used since the amount of training data is constraint. This can be obtained by diagonalising the full covariance matrix as

$$\hat{\boldsymbol{\Sigma}}_{jm}=diag\left(\hat{\boldsymbol{\Sigma}}_{jm}^{full}\right) \tag{2.33}$$

Diagonalisation results in lower modelling power to capture the intra-frame correlations. This can be compensated by using mixtures of Gaussians for each HMM state. In SGMM-based acoustic model, the full covariance matrices can be used which does not have this issue.

2.3 Language modelling

In addition to the acoustic model score, the prior distribution of each word sequence $p(\mathcal{W}|\mathcal{M}_l)$ is also required to generate the hypothesis as is shown in equation (2.2). This is normally referred as language modelling in the context of automatic speech recognition, and $\mathcal{M}_l$ denotes the language model parameters. The probability $p(\mathcal{W}|\mathcal{M}_l)$ can be decomposed into the products of conditional probabilities as

$$p(\mathcal{W}|\mathcal{M}_l)=p(w_1|\mathcal{M}_l)\prod_{k=2}^{K}p(w_k|w_{k-1},\ldots,w_1|\mathcal{M}_l) \tag{2.34}$$

where K is the number of words in $\mathcal{W}$. For large vocabulary speech recognition, the conditioning word history in (2.34) is truncated to $n-1$ words which leads to n-gram language model. Since this is a nonparametric modelling approach, we express it by dropping $\mathcal{M}_l$ as

$$p(w_k|w_{k-1},\ldots,w_1) \approx p(w_k|w_{k-1},\ldots,w_{k-n+1}) \tag{2.35}$$

The approximation is more accurate if using larger word history. However, due to reasons of data sparsity, values of n in the range of $1-4$ are typically used.

Estimates of probabilities in n-gram language models are commonly based on maximum likelihood estimates by counting n-gram occurrences on the training text which gives

$$p(w_k|w_{k-1},\ldots,w_{k-n+1}) = \frac{C(w_{k-n+1},\ldots,w_k)}{C(w_{k-n+1,\ldots,w_{k-1}})} \tag{2.36}$$

where $C(\cdot)$ is the count of a given word sequence in the training text. However, due to the data sparsity, the count for some word sequence may be very small or even zero which is not reliable to estimate the n-gram probability. This can be address by some simple smoothing and back-off schemes such as Katz smoothing (Katz, 1987). For instance, for 3-gram language model

$$p(w_k|w_{k-1},w_{k-2}) = \begin{cases} d\frac{C(w_{k-2},w_{k-1},w_k)}{C(w_{k-2},w_{k-1})} & \text{if} \quad 0 \le C \le C' \\ \frac{C(w_{k-2},w_{k-1},w_k)}{C(w_{k-2},w_{k-1})} & \text{if} \quad C \ge C' \\ \alpha(w_{k-1},w_{k-1})p(w_k|w_{k-1}) & \text{otherwise} \end{cases} \tag{2.37}$$

where C' is a count threshold, C denotes $C(w_{k-2},w_{k-1},w_k)$ for brevity. d is a discount coefficient and α is a normalisation constant. There are many variants of this back-off scheme, for instance, the Kneser-Ney smoothing (Kneser and Ney, 1995) is particularly effective when the training data is very sparse.

2.4 Decoding

The decoding process is to generate the most likely word sequence $\hat{\mathcal{W}}$ given a sequence of feature vector $\mathbf{Y}_{1:T}$. This is done by searching all possible state sequences that arise from all possible word sequence that have generated the observation $\mathbf{Y}_{1:T}$.

$$\begin{aligned} \hat{\mathcal{W}} &= \arg\max_{\mathcal{W}} \left\{ \log p(\mathbf{Y}|\mathcal{W},\mathcal{M}_a) + a\log P(\mathcal{W}) + k|\mathcal{W}| \right\} \\ &= \arg\max_{\mathcal{W}} \left\{ \log \left(\sum_{\mathbf{Q}} p(\mathbf{Y},\mathbf{Q}|\mathcal{W},\mathcal{M}_a) \right) + a\log P(\mathcal{W}) + k|\mathcal{W}| \right\} \end{aligned} \tag{2.38}$$

where a and k are language model scale factor and insertion penalty. $|\mathcal{W}|$ indicates the number of words in $\mathcal{W}$. There are introduced to compensate that the acoustic model and language model score may not be in the same range. $\mathbf{Q}$ denotes any possible state sequence for $\mathcal{W}$ that contribute to the marginal likelihood. As mentioned before, exhaustive search in the state sequence is computationally prohibitive. In practice, Viterbi algorithm (Viterbi, 1967) may be applied, which only search the most likely state sequence by an efficient recursive form:

$$\hat{\mathcal{W}} = \arg\max_{\mathcal{W}} \left\{ \log \left(\max_{\mathbf{Q}} p(\mathbf{Y}, \mathbf{Q} | \mathcal{W}, \mathcal{M}_a) \right) + a \log P(\mathcal{W}) + k |\mathcal{W}| \right\} \tag{2.39}$$

The search network may be expended statically prior to decoding, which is normally assumed not feasible for LVCSR tasks since the network would be huge. However, recent advances in weighted finite state transducer (WFST) (Mohri et al., 2002) make it possible to compose the HMM topology, lexicon and language model in a single, large, well optimised network. This approach offers an elegant unified framework for representing all the required knowledge (acoustic model, pronunciation and language model), and therefore is very useful for research and practical applications. In the static network, the search can be performed in either a time-synchronous or a time-asynchronous manner, though in practice, the first choice is often employed.

The other choice is to expend the network dynamically "on-the-fly". The are several decoding algorithms within this category depending on either a time-synchronous or a time-asynchronous search strategy is used. The second choice leads to a stack decoding approach which may involve one or several stacks, and in this case, the search is performed in a sequential, "depth-like" fashion. For the time-synchronous search, there are two main ways of structuring the search space, either on the word histories or on the start time of the words. The first choice leads to the re-entrant tree methods where time is the independent variable and the expansion proceeds in a "breadth-first" manner. The second way makes use of start-synchronous trees. A good review of the decoding algorithms is given by (Aubert, 2002).

2.5 Adaptation

For a particular speech recogniser, the acoustic model $\mathcal{M}_a$ may be trained by tens or hundreds hours of data from many speakers. Despite that, there will always new speakers or environmental conditions that are poorly represented in the training data. This is normally referred as the mismatch between training and testing condition. One of the

solutions to this problem is to use adaptation techniques to adapt the acoustic model $\mathcal{M}_a$ toward the testing condition so that the mismatch can be alleviated. For speech recognition, there are two main categories of adaptation techniques, namely, the maximum a posteriori (MAP) (Gauvain and Lee, 1994) family and maximum likelihood linear regression (MLLR) (Leggetter and Woodland, 1995) family. We present a brief review of these approaches in this section.

2.5.1 Maximum likelihood linear regression

Maximum likelihood linear regression (MLLR) adaptation estimates a set of linear transforms to map the existing model set into a new adapted model set so that the likelihood of the adaptation data is maximised. Since the amount of adaptation data is usually limited compared to that used to train the acoustic model, a regression tree is normally used to cluster the Gaussian components based on acoustic similarity and each cluster shares the same MLLR transform. In this case, the MLLR adaptation may be expressed as

$$\hat{\boldsymbol{\mu}}_m = \mathbf{A}^{(r_m)}\boldsymbol{\mu}_m + \mathbf{b}^{(r_m)}, \quad \hat{\boldsymbol{\Sigma}}_m = \mathbf{H}^{(r_m)}\boldsymbol{\Sigma}_m\mathbf{H}^{(r_m)T} \tag{2.40}$$

where m denotes the Gaussian component and r_m denotes the regression class that m belongs to. $(\mathbf{A}^{(r_m)}, \mathbf{b}^{(r_m)}, \mathbf{H}^{(r_m)})$ are the MLLR transform parameters for regression class r_m where $\mathbf{A}^{(r_m)}$ is a matrix and $\mathbf{b}^{(r_m)}$ is a vector which are used to adapt the means. $\mathbf{H}^{(r_m)}$ which is usually constrained to be a diagonal matrix is used to adapt the covariance matrix. Otherwise, the adapted covariance matrix $\hat{\boldsymbol{\Sigma}}_m$ will be full even though it was originally diagonal. This will significantly increase the computational cost and not feasible for large vocabulary task.

For MLLR there is no constraints between between the transformation applied to the means and covariances. If the two matrix transforms are constrained to be the same, then this turns to be the constrained MLLR (CMLLR) (Gales, 1998b) or FMLLR (Saon et al., 2001) as it can be applied in the feature space. The transform can be represented as

$$\hat{\boldsymbol{\mu}}_m = \tilde{\mathbf{A}}^{(r_m)}\boldsymbol{\mu}_m + \tilde{\mathbf{b}}^{(r_m)}, \quad \hat{\boldsymbol{\Sigma}}_m = \tilde{\mathbf{A}}^{(r_m)}\boldsymbol{\Sigma}_m\tilde{\mathbf{A}}^{(r_m)T} \tag{2.41}$$

In this case, the likelihood can be computed as

$$\mathcal{N}\left(\mathbf{y}_t; \hat{\boldsymbol{\mu}}_m, \hat{\boldsymbol{\Sigma}}_m\right) = |\mathbf{A}^{(r_m)}|\mathcal{N}\left(\mathbf{A}^{(r_m)}\mathbf{y}_t + \mathbf{b}^{(r_m)}; \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m\right) \tag{2.42}$$

where

$$\mathbf{A}^{(r_m)} = \tilde{\mathbf{A}}^{(r_m)-1}, \quad \mathbf{b}^{(r_m)} = -\tilde{\mathbf{A}}^{(r_m)-1}\tilde{\mathbf{b}}^{(r_m)} \tag{2.43}$$

Hence, with this constraint, the model parameters can be left untouched and the transform can be applied in the feature space. This can reduce the computational cost as the transformed frames can be cached.

2.5.2 Maximum a posteriori adaptation

Rather than using a set of affine linear transformations, maximum a posteriori adaptation (MAP) adaptation introduces a prior $P(\boldsymbol{\lambda})$ to the acoustic model parameters $\boldsymbol{\lambda}$, and given some amount of adaptation data $\mathbf{Y}$, the adapted model is obtained by maximising the posterior distribution of the model parameters $p(\mathbf{Y}|\boldsymbol{\lambda})P(\boldsymbol{\lambda})$. For a GMM based acoustic model, using MAP adaptation leads to the following formulae (Gauvain and Lee, 1994) for the Gaussian component mean

$$\hat{\boldsymbol{\mu}}_{jm} = \frac{\tau\boldsymbol{\mu}_{jm} + \sum_t \gamma_{jm}(t)\mathbf{y}_t}{\tau + \sum_t \gamma_{jm}(t)} \tag{2.44}$$

where τ is the smoothing parameter that balance the weight between the model prior and adaptation data. Similar formulas can be derived for MAP adaptation of weights and covariance matrix of Gaussians in (Gauvain and Lee, 1994).

It can be seen that MAP adaptation interpolates the model parameters from the priors and that can be obtained from the adaptation data alone. An attractive property of MAP is that as the amount of adaptation data increases, the model parameters tend asymptotically to the adaptation domain. However, since every Gaussian component should be adapted individually, many of the model parameters will not be adapted if the adaptation data is limited. This makes MAP unsuitable for rapid model adaptation.

2.6 Summary

In this section, we present a brief overview of the conventional HMM based speech recognition framework. We have discussed several key components in a standard system including feature extraction, acoustic modelling, language modelling, model adaptation and decoding. The discussion of acoustic modelling is based on the use of Gaussian mixture models (GMM) as the density function for HMM states. In the next section, we will study a recently proposed acoustic modelling approach based

subspace Gaussian mixture models. In this approach, the HMM state density function is still modelled by GMMs, but the model parameters are derived by an implicit approach. This leads to several advantages including the reduction of the number of model parameters, factorisation of acoustic variability etc. The other components in the standard ASR framework discussed in this section can remain in such acoustic models.

Chapter 3

Subspace Gaussian Mixture Model

3.1 Introduction

As discussed in Chapter 2, in conventional hidden Markov model (HMM) based speech recognisers, the emitting states are modelled by Gaussian mixture models (GMMs), with parameters estimated directly from the training data. However, in a subspace Gaussian mixture model (SGMM), the GMM parameters are inferred via a low dimensional model subspace which capture the correlations among the triphone states and speaker variabilities. In the SGMM acoustic model (Povey et al., 2011a), the HMM state is modelled as:

$$p(\mathbf{y}_t|j,s) = \sum_{k=1}^{K_j} c_{jk} \sum_{i=1}^{I} w_{jki}\, \mathcal{N}(\mathbf{y}_t|\boldsymbol{\mu}_{jki}^{(s)}, \boldsymbol{\Sigma}_i) \tag{3.1}$$

$$\boldsymbol{\mu}_{jki}^{(s)} = \mathbf{M}_i \mathbf{v}_{jk} + \mathbf{N}_i \mathbf{v}^{(s)} \tag{3.2}$$

$$w_{jki} = \frac{\exp \mathbf{w}_i^T \mathbf{v}_{jk}}{\sum_{i'=1}^{I} \exp \mathbf{w}_{i'}^T \mathbf{v}_{jk}} \tag{3.3}$$

where j is the HMM state index, s denote the speaker, k is a sub-state (Povey et al., 2011a), I is the number of Gaussian components in a sub-state, and $\boldsymbol{\Sigma}_i$ is the i-th covariance matrix. $\mathbf{v}_{jk} \in \mathbb{R}^S$ is referred to as the sub-state vector, and S denotes the subspace dimension. $\mathbf{v}^{(s)} \in \mathbb{R}^T$ is referred to as the speaker vector, and T denotes the speaker subspace dimension. The matrices $\mathbf{M}_i$, $\mathbf{N}_i$ and the vectors $\mathbf{w}_i$ span the model subspaces for Gaussian means and weights respectively, and are used to derive the GMM parameters given sub-state vectors. In particular, $\mathbf{M}_i$ models the correlations among the triphone states while $\mathbf{N}_i$ models the correlations among the speakers.

Figure 3.1 shows the structure of a SGMM acoustic model. The total parameters can be split into two sets, the globally shared parameters $\boldsymbol{\Phi}_i = \{\mathbf{M}_i, \mathbf{N}_i, \mathbf{w}_i, \boldsymbol{\Sigma}_i\}$ that do

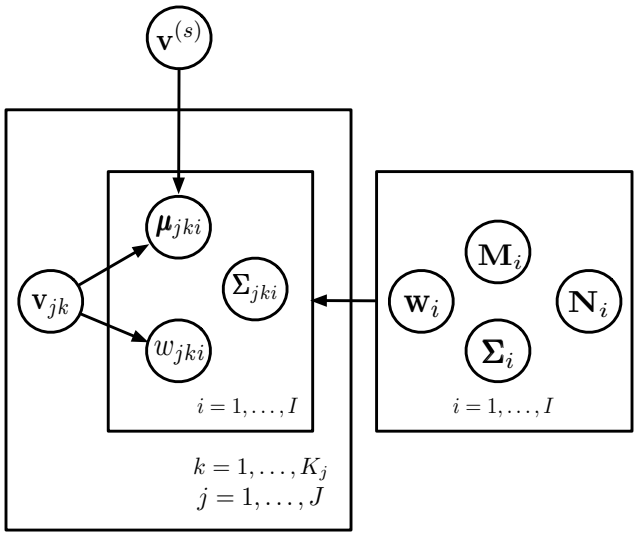


Figure 3.1: Model structure of a SGMM acoustic model, with total J HMM states, and each has K_j sub-states. Each sub-state is modelled by a GMM with I components, whose parameters are derived from $\boldsymbol{\Phi}_i = \{\mathbf{M}_i, \mathbf{N}_i, \mathbf{w}_i, \boldsymbol{\Sigma}_i\}$ and $(\mathbf{v}_{jk}, \mathbf{v}^{(s)})$ using Eq. (3.2) and (3.3), and for covariance $\boldsymbol{\Sigma}_{jki} = \boldsymbol{\Sigma}_i$.

not depend on the state, and the state dependent parameters $\mathbf{v}_{jk}$. The speaker vector $\mathbf{v}^{(s)}$ is used to adapt the model means based on the basis $\mathbf{N}_i$. The sub-state weight c_{jk} is not presented in the figure for clarity. For each Gaussian component, the parameters are derived from both the globally shared and state-dependent parameter sets. This model is quite different from the conventional GMM based acoustic model, as a large portion of the parameters are globally shared, and do not depend on the HMM state. The number of state dependent parameters $(\mathbf{v}_{jk}, c_{jk})$ is relatively small if we use a low dimensional model space. This enables the model to be trained by a relatively small amount of training data since the number of parameters in a SGMM acoustic model can be much smaller that in its GMM based counterpart (Povey et al., 2011a). In addition, since the globally shared parameters do not depend on the model topology, they may be estimated by tying across multiple systems or entirely by out-domain data, which inspires its application in multilingual and cross-lingual speech recognition (Burget et al. (2010) and Chapter 5).

3.1.1 Related techniques

The essence of the SGMM acoustic model is that it factories the phonetic and speaker factors using separate subspace (3.2). In addition, the GMM model parameters are de-

rived from a group of basis vectors (e.g. $\mathbf{M}_i$) together with their corresponding weights $\mathbf{v}_{jk}$.These features are related to techniques used in the field of speaker verification and speech recognition. In speaker verification, joint factor analysis (JFA) (Kenny, 2005) is the most similar approach to SGMMs. In JFA, the Gaussian mean supervector (which is a concatenation of all the Gaussian means) in a speaker and channel dependent GMM model is factorized as

$$M_{s,h} = \mathbf{m} + \mathbf{V}\mathbf{y}_s + \mathbf{U}\mathbf{x}_{s,h} + \mathbf{D}\mathbf{z}_s \tag{3.4}$$

where $\mathbf{m}$ denotes the model origin. s and h denote the speaker and channel index. $\mathbf{y}_s$ is referred as the speaker factor which lies in the subspace spanned by the matrix $\mathbf{V}$ which is a low rank matrix, and is assumed to model the speaker variability. $\mathbf{x}_{s,h}$ is referred as the channel factor which lies in the subspace spanned by the matrix $\mathbf{U}$ which is also a low rank matrix, and is assumed to model the channel variability. $\mathbf{D}$ is a diagonal matrix which spans the whole model space. Together with $\mathbf{z}_s$, it is introduced to capture the residual variability not contained in $\mathbf{U}$ and $\mathbf{V}$.

Both JFA and SGMM factorize the whole model space of GMM means into different subspaces, by which, the distribution of a particular acoustic factor can be modelled a group of basis vectors which are expected to capture most of the corresponding variability. However, there are fundamental differences between the two models as they are used for different tasks. In JFA, the speaker and channel factors are modelled explicitly, in which, the channel factor is viewed as nuisance attribute for speaker verification. SGMMs, however, factorizes the phonetic and speaker factor, and the later is viewed as a nuisance attribute. In addition, the subspace parameters in SGMMs model the correlation among HMM state models, while JFA model the correlations among speaker models.

In the field of speech recognition, closely related techniques to SGMMs include Eigenvoices (Kuhn et al., 2000) and Cluster Adaptive Training (CAT) (Gales, 1998a). They perform speaker adaptation using a basis model and the corresponding weight for the Gaussians means

$$\boldsymbol{\mu}_{ji}^{(s)} = \sum_{k=1}^{K} \lambda_k^{(s)} \boldsymbol{\mu}_{ji}^{(k)} \tag{3.5}$$

where $\boldsymbol{\mu}_{ji}^{(s)}$ denote the Gaussian mean of i_{th} component in j_{th} HMM state for speaker s. The adaptation is performed using the speaker dependent weight $(\lambda_1, \cdots, \lambda_K)$ which is similar to the role of sub-state vector $\mathbf{v}_{jk}$ and a global basis model $(\boldsymbol{\mu}_i, \cdots, \boldsymbol{\mu}_K)$ which

is similar to the model subspace $\mathbf{M}_i$ in the SGMMs. The key difference, however, is that SGMM is used to model the HMM state models while these two approaches are only used to adapt a speaker-independent model to the speaker dependent counterpart.

There are also some similarities between an SGMM and semi-continous GMM acoustic model (Huang and Jack, 1989). They both share some model parameters among the HMM states. However, the main difference is that in a semi-continous GMM acoustic model, all the Gaussian means and covariance matrices are state independent, and only the weights are state dependent, while for an SGMM, the Gaussian means are still state dependent, but they are inferred from globally shared projection matrices. The number of active parameters of an SGMM is much larger than that of a semi-continous GMM, and it can also obtain much higher recognition accuracy as it was found in (Riedhammer et al., 2012).

The idea of using basis representation to reduce the number of model parameters while improve the modelling power is also extensively investigated by many other works in the field of speech recognition. To name a few, maximum likelihood linear transformation (MLLT) (Gales, 1998b) and its extended version, EMLLT (Olsen and Gopinath, 2002) approximate the full covariance matrices for HMM-GMM systems using basis expansion of the precision matrix (i.e. inverse covariance). The SPAM model (Axelrod et al., 2005) extends it further by including the basis representation of the Gaussian means, in which, the product between the precision and mean of each Gaussian is also represented by a linear combination of a set of basis. Hence, it is fundamentally different to an SGMM. The recently proposed Bayesian sensing HMM (Saon and Chien, 2011) can be viewed as another example within this category, which is formulated in a full Bayesian probabilistic framework with different basis construction compared to SGMMs.

3.1.2 Universal background model

In SGMMs, the number of Gaussians in each sub-state is typically very large, i.e. $I = 400$ for most of the system setups in this thesis. This can make direct likelihood evaluation computationally infeasible for both training and decoding. To address this, a universal background model (UBM) which is a mixture of Gaussians with I components, is also introduced to prune the Gaussian indices and initialise the model. The UBM partitions the acoustic space into I regions. The acoustics in region i are assumed to lie in a subspace defined by $\mathbf{M}_i, \mathbf{N}_i$ and $\mathbf{w}_i$. For each acoustic frame $\mathbf{y}_t$, the top P

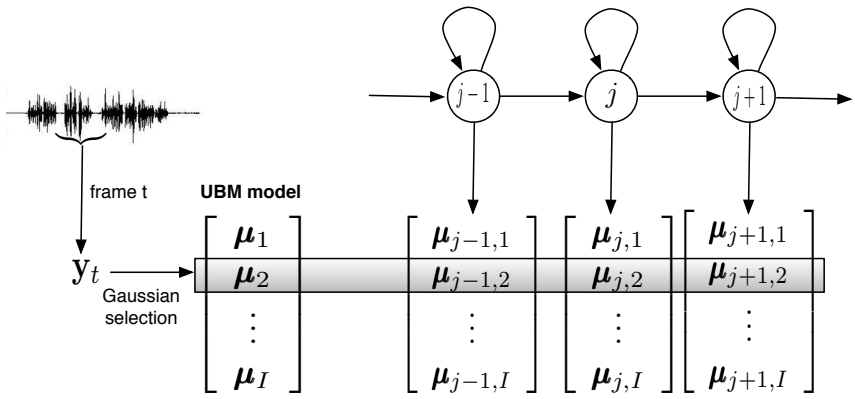


Figure 3.2: An example of Gaussian selection using UBM model. For each acoustic frame $\mathbf{y}_t$, the UBM is used to select the active Gaussian components in the SGMM acoustic models. For instance, in this figure, the second Gaussian represented by $\boldsymbol{\mu}_2$ is among the top P Gaussian components according to its likelihood score for $\mathbf{y}_t$, thus all the second Gaussian in each SGMM sub-state is active for this frame.

(e.g. $P = 15$) Gaussian components with highest likelihood scores in the UBM are selected, and only those Gaussians in SGMMs with the same indices are selected to calculate the likelihood during both acoustic model training and decoding. Thus, the UBM itself provides a clustering of the surface Gaussians in the SGMMs based on the acoustic similarity. Figure 3.2 illustrates the roles of UBM used for Gaussian selection. In Chapter 6, we also show that UBM can serve as the regression class model for joint uncertainty decoding based noise compensation for SGMMs.

3.2 Overview of SGMM acoustic model training

Before going into details, Figure 3.3 presents an overview of SGMM acoustic model training. The UBM model is firstly initialised from a baseline HMM-GMM system by Gaussian clustering and EM re-estimation and then it is used to initialise the SGMM acoustic model as discussed in Section 3.2.1. The baseline HMM-GMM system also provides the alignment of the training data to supervise the model estimation of SGMM acoustic model for the first several iterations. The alignment is updated by the SGMM acoustic model after it is relatively well estimated, and in the following model estimation steps, we optionally increase the number of sub-states (Section 3.4.1) or increase the dimension of subspace (Section 3.4.2). The details are presented in following sections.

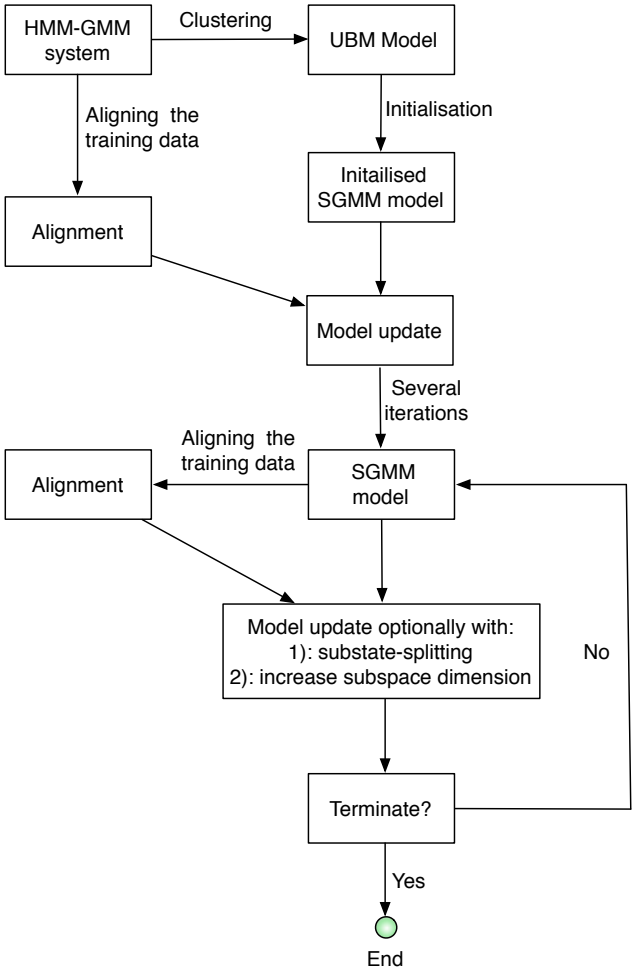


Figure 3.3: Block-diagram of SGMM acoustic model estimation.

3.2.1 Model initialisation

We have discussed the role of UBM model in SGMM acoustic model training and evaluation in Section 3.1.2. Before training and evaluating the SGMM models, it is necessary to initialise the UBM model parameters first. To this end, the following approach is used in (Povey et al., 2011a), namely, a standard HMM-GMM system with diagonal Gaussians is built with the training dataset. The Universal Background Model (UBM), denoted as $(\bar{\boldsymbol{\mu}}_i, \bar{\boldsymbol{\Sigma}}_i, \bar{w}_i, i = 1, \ldots, I)$, is then initialised by the HMM-GMM system. This is done by clustering the diagonal Gaussians in the HMM-GMM system. Since this will normally results into a GMM with much larger number of Gaussian than I, we then merge the pair of Gaussians i and j that would result in the least log-likelihood reduction, computed as follows

$$\Delta\mathcal{L} = \frac{w_i}{2}\log|\boldsymbol{\Sigma}_i| + \frac{w_j}{2}\log|\boldsymbol{\Sigma}_j| - \frac{w_k}{2}\log|\boldsymbol{\Sigma}_k| \tag{3.6}$$

$$w_k = w_i + w_j \tag{3.7}$$

$$\boldsymbol{\mu}_k = (w_i\boldsymbol{\mu}_i + w_j\boldsymbol{\mu}_j)/w_k \tag{3.8}$$

$$\boldsymbol{\Sigma}_k = \text{diag}\left(\frac{w_i}{w_k}(\boldsymbol{\Sigma}_i + \boldsymbol{\mu}_i\boldsymbol{\mu}_i^T) + \frac{w_j}{w_k}(\boldsymbol{\Sigma}_j + \boldsymbol{\mu}_j\boldsymbol{\mu}_j^T) - \boldsymbol{\mu}_k\boldsymbol{\mu}_k^T\right) \tag{3.9}$$

where $|\cdot|$ denotes the determinant of a matrix, k is the index of the merged Gaussian. After clustering, we train the resulting GMM with several EM iterations of full-covariance re-estimation on all the available speech data[1]. In each update we set the weight to all be the same, to encourage even distribution of data among the Gaussians.

After initialising the UBM, we need to initialise the SGMM acoustic model as the starting point to train the model. This involves the initialisation of the globally shared parameters $\boldsymbol{\Phi}_i$ and state-dependent parameters $(\mathbf{v}_{jk}, c_{jk})$[2]. To this end, a feature normalisation transformation matrix $\mathbf{J}$ is computed first, which is similar to the inverse of an LDA transformation but without dimensionality loss. Given the UBM model

[1] Since training the UBM requires unlabelled data only, the out-domain data may be used if the amount of in-domain training is limited. This idea is explored in multilingual and cross-lingual speech recognition tasks using SGMMs (Burget et al. (2010) and Chapter 5)

[2] The number of sub-states K_j is initialised to be 1 for every state j, and the number can be increased during the model training stages as described in Section 3.4.1.

parameters, we compute:

$$\boldsymbol{\Sigma}_W = \sum_{i=1}^{I} \bar{w}_i \bar{\boldsymbol{\Sigma}}_i \tag{3.10}$$

$$\boldsymbol{\mu} = \sum_{i=1}^{I} \bar{w}_i \bar{\boldsymbol{\mu}}_i \tag{3.11}$$

$$\boldsymbol{\Sigma}_B = \left(\sum_{i=1}^{I} \bar{w}_i \bar{\boldsymbol{\mu}}_i \bar{\boldsymbol{\mu}}_i^T \right) - \boldsymbol{\mu}\boldsymbol{\mu}^T \tag{3.12}$$

$$\boldsymbol{\Sigma}_W = \mathbf{L}\mathbf{L}^T \quad \text{(Cholesky decomposition)} \tag{3.13}$$

$$\mathbf{S} = \mathbf{L}^{-1} \boldsymbol{\Sigma}_B \mathbf{L}^{-T} \tag{3.14}$$

$$\mathbf{S} = \mathbf{U}\mathbf{D}\mathbf{V}^T \quad \text{(Single value decomposition)} \tag{3.15}$$

$$\mathbf{J} = \mathbf{L}\mathbf{U} \tag{3.16}$$

where $\boldsymbol{\Sigma}_W$ and $\boldsymbol{\Sigma}_B$ are analogous to the within-class and between-class covariance matrices. After we obtain $\mathbf{J}$, we can initialise the model as

$$M_j = 1 \tag{3.17}$$

$$c_{j1} = 1 \tag{3.18}$$

$$\mathbf{v}_{j1} = \mathbf{e}_1 \in \mathbb{R}^S \tag{3.19}$$

$$\mathbf{M}_i = [\bar{\boldsymbol{\mu}}_i, \mathbf{j}_1, \ldots, \mathbf{j}_{S-1}] \tag{3.20}$$

$$\mathbf{N}_i = [\mathbf{j}_1, \ldots, \mathbf{j}_T] \tag{3.21}$$

$$\mathbf{w}_i = \mathbf{0} \in \mathbb{R}^S \tag{3.22}$$

$$\boldsymbol{\Sigma}_i = \bar{\boldsymbol{\Sigma}}_i \tag{3.23}$$

where we require $S \leq D+1$ and $T \leq D$, $\mathbf{e}_1 = [1\,0 \ldots 0]$ is a unit vector in the first dimension, and $\mathbf{j}_i$ is the i_{th} column of the matrix $\mathbf{J}$. The idea behind the initialisation is that the initial values of $\boldsymbol{\mu}_{j1i}$ are the same as the UBM mean $\bar{\boldsymbol{\mu}}_i$, which is computed as

$$\boldsymbol{\mu}_{j1i} = [\bar{\boldsymbol{\mu}}_i, \mathbf{j}_1, \ldots, \mathbf{j}_{S-1}] \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \bar{\boldsymbol{\mu}} \tag{3.24}$$

Hence, the matrix $\mathbf{J}$ does not affect the initialisation of the Gaussian component mean value, however, it need to be of full rank so that it will not introduce numerical instability when update the state vectors. The elements of the state vector $\mathbf{v}_{j1}$ and the speaker vector $\mathbf{v}^{(s)}$ are offsets on the means in the LDA-normalised space. $\mathbf{v}^s$ can be

initialised to be a zero vector. We are also able to optionally increase S or T during model training stages as in Section 3.4.2.

3.2.2 Likelihood and posterior evaluation

Before we update the model parameters, we first show how to calculate the likelihood and posterior of the model parameter given the data. We have mentioned that a UBM model is used to prune the Gaussians to save computation. Another option to reduce the computational cost is to pre-compute the global terms that does not depend on the acoustic frame and cache them rather than compute them repeatedly for each frame. For instance, the likelihood for state j, sub-state k and Gaussian component i is decomposed as

$$\log p(\mathbf{y}_t, k, i|j) = n_i(t) + n_{jki} + \mathbf{z}_i^T(t)\mathbf{v}_{jk} \tag{3.25}$$

where n_{jki} is a global normalization term which does not depend on the acoustic frames, while $n_i(t)$ and $\mathbf{z}_i^T(t)$ depend on each acoustic frame. n_{jki} can be expressed as

$$n_{jki} = \log c_{jk} + \log w_{jki} - 0.5\left(\log|\boldsymbol{\Sigma}_i| + D\log(2\pi) + \boldsymbol{\mu}_{jki}^T\boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_{jki}\right) \tag{3.26}$$

where $\boldsymbol{\mu}_{jki} = \mathbf{M}_i\mathbf{v}_{jk}$. It requires a large amount of computation to calculate this term. However, it needs to be calculated only once and then it can be cached. The frame dependent terms $n_i(t)$ and $\mathbf{z}_i^T(t)$ are computed by

$$\mathbf{z}_i(t) = \mathbf{M}_i^T\boldsymbol{\Sigma}_i^{-1}\mathbf{y}_{ti} \qquad n_i(t) = -0.5\mathbf{y}_{ti}^T\boldsymbol{\Sigma}_i^{-1}\mathbf{y}_{ti} \tag{3.27}$$

where $\mathbf{y}_{ti} = \mathbf{y}_t - \mathbf{N}_i\mathbf{v}^{(s)}$ are the speaker adapted features. If the speaker subspace $\mathbf{N}_i$ is not used, then $\mathbf{y}_{ti} = \mathbf{y}_t$. From this decomposition, only the frame dependent term $n_i(t)$ and $\mathbf{z}_i^T(t)$ are calculated for each new acoustic frame for both training and decoding. This can significantly save computation. The total likelihood for state j can be obtained by summing over the Gaussians and sub-states

$$\log p(\mathbf{y}_t|j) = \log\sum_{k,i} p(\mathbf{y}_t, k, i|j) \tag{3.28}$$

To update the model parameters, we need to compute the posterior probability of state j, sub-state k and Gaussian component i. The Gaussian component posterior probability can be obtained by

$$\gamma_{jki}(t) \equiv p(j,k,i|\mathbf{y}_t) = \gamma_j(t)\frac{p(\mathbf{y}_t, k, i|j)}{p(\mathbf{y}_t|j)} \tag{3.29}$$

where $p(\mathbf{y}_t, k, i|j)$ and $p(\mathbf{y}_t|j)$ are given by Eq. (3.25) and (3.28). The state posterior probability $\gamma_j(t) \equiv p(j|\mathbf{y}_t)$ can be obtained by the forward-backward or Viterbi algorithm.

3.3 Maximum likelihood estimation

Having obtained the posteriors, in this section we present the model estimation algorithms for SGMM using the maximum likelihood (ML) criterion (Povey, 2009). Similar to conventional GMM based acoustic models, no closed form solution is available to estimate the model parameters since the labels for the HMM states j and Gaussian components i are hidden. However, the standard expectation-maximization (EM) algorithm can be used to address this issue by optimizing the lower bound of the ML objective function, i.e. the auxiliary function. The parameters to be updated for an SGMM acoustic model are $\mathbf{M}_i, \mathbf{N}_i, \mathbf{w}_i, \mathbf{\Sigma}_i, \mathbf{v}_{jk}, \mathbf{v}^{(s)}$ and c_{jk}. In the following sections, we use the EM algorithm to update the model parameters of an SGMM.

3.3.1 Update for sub-state vectors

To update the sub-state vector $\mathbf{v}_{jk}$, the other SGMM model parameters are assumed to be fixed. The same principle applies to the update for other model parameters. The auxiliary function for the sub-state vector $\mathbf{v}_{jk}$ can be obtained as

$$\begin{aligned} Q(\mathbf{v}_{jk}) &= \sum_{t=1}^{T}\sum_{i=1}^{I} \underbrace{p(j,k,i|\mathbf{y}_t)}_{=\gamma_{jki}(t)} \log p(\mathbf{y}_t, j, k, i|\mathbf{v}_{jk}) \\ &= \sum_{t=1}^{T}\sum_{i=1}^{I} \gamma_{jki}(t)\Big(-0.5\mathbf{v}_{jk}^T\mathbf{M}_i^T\mathbf{\Sigma}_i^{-1}\mathbf{M}_i\mathbf{v}_{jk} + \mathbf{y}_t^T\mathbf{\Sigma}_i^{-1}\mathbf{M}_i\mathbf{v}_{jk} + \log w_{jki}\Big) + const \end{aligned} \tag{3.30}$$

where *const* denotes a constant value that does not depend on the parameter to be optimized. $\gamma_{jki}(t)$ is the posterior of frame $\mathbf{y}_t$ that defined in equation (3.29). To simplifiy the derivation, we define

$$\gamma_{jki} = \sum_t \gamma_{jki}(t) \tag{3.31}$$

$$\mathbf{H}_i = \mathbf{M}_i^T \Sigma_i^{-1} \mathbf{M}_i \tag{3.32}$$

$$\mathbf{q}_{jk} = \sum_{i,t} \gamma_{jki}(t) \mathbf{M}_i^T \mathbf{\Sigma}_i^{-1} \mathbf{y}_t \tag{3.33}$$

Then

$$Q(\mathbf{v}_{jk}) = -0.5\mathbf{v}_{jk}^T \left(\sum_i \gamma_{jki} \mathbf{H}_i \right) \mathbf{v}_{jk} + \mathbf{q}_{jk}^T \mathbf{v}_{jk} + \sum_i \gamma_{jki} \log w_{jki} + const. \tag{3.34}$$

Apart from $\log w_{jki}$, the rest of the auxiliary function $Q(\mathbf{v}_{jk})$ is a quadratic function which is simple to optimise. However, since the weight w_{jki} is a softmax function of the sub-state vector $\mathbf{v}_{jk}$ as equation (3.3), this makes the auxiliary function $Q(\mathbf{v}_{jk})$ nonlinear with respect to $\mathbf{v}_{jk}$. In order to simplify the optimisation, we approximate the auxiliary function by a quadratic function as follows.

$$\begin{aligned}
\sum_i \gamma_{jki} \log w_{jki} &= \sum_i \gamma_{jki} \left(\mathbf{w}_i^T \mathbf{v}_{jk} - \log \sum_{i'=1}^{I} \exp \mathbf{w}_{i'}^T \mathbf{v}_{jk} \right) \\
&= \sum_i \gamma_{jki} \left(\mathbf{w}_i^T \mathbf{v}_{jk} - \log \frac{\sum_{i'=1}^{I} \exp \mathbf{w}_{i'}^T \mathbf{v}_{jk}}{\sum_{i'=1}^{I} \exp \mathbf{w}_{i'}^T \check{\mathbf{v}}_{jk}} \right) - \underbrace{\sum_i \gamma_{jki} \log \sum_{i'=1}^{I} \exp \mathbf{w}_{i'}^T \check{\mathbf{v}}_{jk}}_{const} \\
&= \sum_i \gamma_{jki} \left(\mathbf{w}_i^T \mathbf{v}_{jk} + \log \frac{\sum_{i'=1}^{I} \exp \mathbf{w}_{i'}^T \mathbf{v}_{jk}}{\sum_{i'=1}^{I} \exp \mathbf{w}_{i'}^T \check{\mathbf{v}}_{jk}} \right) + const,
\end{aligned} \tag{3.35}$$

where $\check{\mathbf{v}}_{jk}$ denotes the 'old' value of $\mathbf{v}_{jk}$. We then use the inequality $1 - (x/\check{x}) \leq -\log(x/\check{x})$ to get rid of the log function as

$$\begin{aligned}
\sum_i \gamma_{jki} \log w_{jki} &= \sum_i \gamma_{jki} \left(\mathbf{w}_i^T \mathbf{v}_{jk} - \log \frac{\sum_{i'=1}^{I} \exp \mathbf{w}_{i'}^T \mathbf{v}_{jk}}{\sum_{i'=1}^{I} \exp \mathbf{w}_{i'}^T \check{\mathbf{v}}_{jk}} \right) + const \\
&\geq \sum_i \gamma_{jki} \left(\mathbf{w}_i^T \mathbf{v}_{jk} - \frac{\sum_{i'=1}^{I} \exp \mathbf{w}_{i'}^T \mathbf{v}_{jk}}{\sum_{i'=1}^{I} \exp \mathbf{w}_{i'}^T \check{\mathbf{v}}_{jk}} \right) + const.
\end{aligned} \tag{3.36}$$

Since the exponential function is still nonlinear, we further approximate it by a quadratic function using the second-order vector Taylor series approximation as

$$\begin{aligned}
\exp(x) &\approx \exp(x_0) \left(1 + (x - x_0) + 0.5(x - x_0)^2 \right) \\
&= \exp(x_0)(x(1 - x_0) + 0.5x^2) + const.
\end{aligned} \tag{3.37}$$

Then we will have

$$
\begin{aligned}
&\sum_i \gamma_{jki} \log w_{jki} \\
&\geq \sum_i \gamma_{jki} \left(\mathbf{w}_i^T \mathbf{v}_{jk} - \frac{\sum_{i'=1}^I \exp \mathbf{w}_{i'}^T \mathbf{v}_{jk}}{\sum_{i'=1}^I \exp \mathbf{w}_{i'}^T \check{\mathbf{v}}_{jk}} \right) + const \\
&\approx \sum_i \gamma_{jki} \left(\mathbf{w}_i^T \mathbf{v}_{jk} - \sum_{i'=1}^I \underbrace{\frac{\exp \mathbf{w}_{i'}^T \check{\mathbf{v}}_{jk}}{\sum_{i'=1}^I \exp \mathbf{w}_{i'}^T \check{\mathbf{v}}_{jk}}}_{=\check{w}_{jki}} \left(\mathbf{w}_{i'}^T \mathbf{v}_{jk} (1 - \mathbf{w}_{i'}^T \check{\mathbf{v}}_{jk}) + 0.5 (\mathbf{w}_{i'}^T \mathbf{v}_{jk})^2 \right) \right) + const \\
&= \sum_i \gamma_{jki} \left(\mathbf{w}_i^T \mathbf{v}_{jk} - \sum_{i'=1}^I \check{w}_{jki} \left(\mathbf{w}_{i'}^T \mathbf{v}_{jk} (1 - \mathbf{w}_{i'}^T \check{\mathbf{v}}_{jk}) + 0.5 (\mathbf{w}_{i'}^T \mathbf{v}_{jk})^2 \right) \right) + const \\
&= -0.5 \mathbf{v}_{jk}^T \left(\gamma_{jk} \sum_i \check{w}_{jki} \mathbf{w}_i \mathbf{w}_i^T \right) \mathbf{v}_{jk} + \sum_i \gamma_{jki} \mathbf{w}_i^T \mathbf{v}_{jk} - \gamma_{jk} \sum_{i=1}^I \check{w}_{jki} \mathbf{w}_i^T \mathbf{v}_{jk} (1 - \mathbf{w}_i^T \check{\mathbf{v}}_{jk}) + const \\
&= -0.5 \mathbf{v}_{jk}^T \left(\gamma_{jk} \sum_i \check{w}_{jki} \mathbf{w}_i \mathbf{w}_i^T \right) \mathbf{v}_{jk} + \sum_i \left(\gamma_{jki} - \gamma_{jk} \check{w}_{jki} (1 - \mathbf{w}_i^T \check{\mathbf{v}}_{jk}) \right) \mathbf{w}_i^T \mathbf{v}_{jk} + const,
\end{aligned}
\tag{3.38}
$$

where $\check{w}_{jki}$ is the 'old' weight derived by $\check{\mathbf{v}}_{jk}$. Now, we have approximated $\sum_i \gamma_{jki} \log w_{jki}$ in the auxiliary function $Q(\mathbf{v}_{jk})$ by a quadratic function, and therefore, $Q(\mathbf{v}_{jk})$ can be approximated by a quadratic function as

$$
Q(\mathbf{v}_{jk}) = -0.5 \mathbf{v}_{jk}^T \mathbf{H}_{jk} \mathbf{v}_{jk} + \mathbf{v}_{jk}^T \mathbf{g}_{jk} + const \tag{3.39}
$$

where

$$
\mathbf{g}_{jk} = \mathbf{q}_{jk} + \sum_i \left(\gamma_{jki} - \gamma_{jk} \check{w}_{jki} (1 - \mathbf{w}_i^T \check{\mathbf{v}}_{jk}) \right) \mathbf{w}_i \tag{3.40}
$$

$$
\mathbf{H}_{jk} = \sum_i \left(\gamma_{jki} \mathbf{H}_i + \gamma_{jk} \check{w}_{jk} \mathbf{w}_i \mathbf{w}_i^T \right) \tag{3.41}
$$

If the matrix $\mathbf{H}_{jk}$ is well conditioned, the updated value of $\mathbf{v}_{jk}$ can be readily available as

$$
\hat{\mathbf{v}}_{jk} = \mathbf{H}_{jk}^{-1} \mathbf{g}_{jk} \tag{3.42}
$$

When $\mathbf{H}_{jk}$ is poorly conditioned, namely, the condition number of $\mathbf{H}_{jk}$ is large, directly inversing the matrix $\mathbf{H}_{jk}$ may introduces the numerical instability. In this case, a more numerically stable estimation is given in (Povey et al., 2011a), where the eigenvalues of the matrix $\mathbf{H}_{jk}$ are floored before the inversion when they are below a pre-defined threshold.

3.3.2 Update for model projections

The auxiliary function for the phonetic projection matrix $\mathbf{M}_i$ is

$$\begin{aligned} Q(\mathbf{M}_i) &= \sum_{jkt} \gamma_{jki}(t) \log p(\mathbf{y}_t, j, k, i|\mathbf{M}_i) \\ &= -0.5 \sum_{jkt} \gamma_{jki}(t) \mathbf{v}_{jk}^T \mathbf{M}_i^T \boldsymbol{\Sigma}_i^{-1} \mathbf{M}_i \mathbf{v}_{jk} + \sum_{jkt} \gamma_{jki}(t) \mathbf{y}_t^T \boldsymbol{\Sigma}_i^{-1} \mathbf{M}_i \mathbf{v}_{jk} + const \\ &= -0.5 tr\left(\boldsymbol{\Sigma}_i^{-1} \mathbf{M}_i \mathbf{Q}_i \mathbf{M}_i^T\right) + tr\left(\mathbf{M}_i^T \boldsymbol{\Sigma}_i^{-1} \mathbf{Y}_i\right) + const, \end{aligned} \tag{3.43}$$

where $\mathbf{Y}_i$ and $\mathbf{Q}_i$ are accumulated statistics as

$$\mathbf{Y}_i = \sum_{jkt} \gamma_{jki}(t) \mathbf{y}_t \mathbf{v}_{jk}^T \tag{3.44}$$

$$\mathbf{Q}_i = \sum_{jkt} \gamma_{jki}(t) \mathbf{v}_{jk} \mathbf{v}_{jk}^T \tag{3.45}$$

The derivative of $Q(\mathbf{M}_i)$ with respect to $\mathbf{M}_i$ is

$$\begin{aligned} \frac{\partial Q(\mathbf{M}_i)}{\partial \mathbf{M}_i} &= -0.5 \mathbf{Q}_i^T \mathbf{M}_i^T \boldsymbol{\Sigma}_i^{-1} - 0.5 \mathbf{Q}_i \mathbf{M}_i^T \boldsymbol{\Sigma}_i^{-1} + \mathbf{Y}_i^T \boldsymbol{\Sigma}_i^{-1} \\ &= \left(-\mathbf{Q}_i \mathbf{M}_i^T + \mathbf{Y}_i^T\right) \boldsymbol{\Sigma}_i^{-1} \end{aligned} \tag{3.46}$$

The second line of equation is obtained since $\mathbf{Q}_i$ is symmetric. By setting the derivative to be zero, $\mathbf{M}_i$ can be computed as

$$\mathbf{M}_i = \mathbf{Q}_i^{-1} \mathbf{Y}_i \tag{3.47}$$

if the matrix $\mathbf{Q}_i$ is well conditioned. A more numerically stable algorithm is given in (Povey et al., 2011a) in case that the matrix $\mathbf{Q}_i$ is poorly conditioned.

3.3.3 Update for weight projections

The auxiliary function for the weight projection vector $\mathbf{w}_i$ is

$$\begin{aligned} Q(\mathbf{w}) &= \sum_{jkit} \gamma_{jki}(t) \log p(\mathbf{y}_t, j, k, i|\mathbf{w}_i) \\ &= \sum_{jkit} \gamma_{jki}(t) \log w_{jki} + const. \end{aligned} \tag{3.48}$$

The derivation is analogous to that in section 3.3.1 where the nonlinear objective function is approximated by a quadratic function. Here, however, the parameters $\mathbf{w}_i$ are to

be optimized, rather than $\mathbf{v}_{jk}$. We rewrite the derivations with only several key steps. First, the log function is removed using the previous inequality

$$\begin{aligned}
Q(\mathbf{w}) &= \sum_{jkit} \gamma_{jki}(t) \left(\mathbf{w}_i^T \mathbf{v}_{jk} - \log \sum_{i'=1}^{I} \exp \mathbf{w}_{i'}^T \mathbf{v}_{jk} \right) \\
&= \sum_{jkit} \gamma_{jki}(t) \left(\mathbf{w}_i^T \mathbf{v}_{jk} + \log \frac{\sum_{i'=1}^{I} \exp \mathbf{w}_{i'}^T \mathbf{v}_{jk}}{\sum_{i'=1}^{I} \exp \check{\mathbf{w}}_{i'}^T \mathbf{v}_{jk}} \right) + const \\
&\geq \sum_{jki} \gamma_{jki} \mathbf{w}_i^T \mathbf{v}_{jk} - \sum_{jki} \gamma_{jk} \frac{\exp \mathbf{w}_i^T \mathbf{v}_{jk}}{\sum_{i'=1}^{I} \exp \check{\mathbf{w}}_{i'}^T \mathbf{v}_{jk}} + const = Q'(\mathbf{w}),
\end{aligned} \tag{3.49}$$

where $\check{\mathbf{w}}_{i'}$ are the 'old' values of $\mathbf{w}_i$. Since the new objective function $Q'(\mathbf{w})$ is still nonlinear, we use Newton's approach to optimize it as

$$\hat{\mathbf{w}}_i = \mathbf{w}_i - \left(\frac{\partial^2 Q'(\mathbf{w})}{\partial^2 \mathbf{w}_i} |_{\check{\mathbf{w}}_i} \right)^{-1} \left(\frac{\partial Q'(\mathbf{w})}{\partial \mathbf{w}_i} |_{\check{\mathbf{w}}_i} \right)^T, \tag{3.50}$$

where the gradient and Hessian matrix can be computed as

$$\begin{aligned}
\frac{\partial Q'(\mathbf{w})}{\partial \mathbf{w}_i} |_{\check{\mathbf{w}}_i} &= \sum_{jk} \gamma_{jki} \mathbf{v}_{jk}^T - \gamma_{jk} \underbrace{\frac{\exp \check{\mathbf{w}}_i^T \mathbf{v}_{jk}}{\sum_{i'} \exp \check{\mathbf{w}}_i' \mathbf{v}_{jk}}}_{=w_{jki}} \mathbf{v}_{jk}^T \\
&= \sum_{jk} \left(\gamma_{jki} - \gamma_{jk} w_{jki} \right) \mathbf{v}_{jk}^T
\end{aligned} \tag{3.51}$$

$$\frac{\partial^2 Q'(\mathbf{w})}{\partial^2 \mathbf{w}_i} |_{\check{\mathbf{w}}_i} = -\sum_{jk} \gamma_{jk} w_{jki} \mathbf{v}_{jk} \mathbf{v}_{jk}^T \tag{3.52}$$

In practice, however this approach is not stable and it is often necessary to halve the step size many times (Povey, 2009). The reason is that Newton's approach approximates the nonlinear objective function by its second order Taylor series expansion using the 'old' estimate as the expansion point. However, it is a very poor approximation to the exponential function if the step size is large. To make the estimation more stable, the term $\gamma_{jk} w_{jki}$ in equation (3.52) is replaced with $\max(\gamma_{jki}, \gamma_{jk} w_{jki})$. Since without the subspace constraint, the ML estimate of w_{jki} would be γ_{jki}/γ_{jk}, this means that if the value of this term around the unconstrained ML solution would be larger with the intuition that the weights will probably get closer to the unconstrained ML solution. Taking the larger one will make the optimization more stable, as this makes the Hessian more negative and consequently a smaller step size will be used for each iteration. After this manipulation, the Hessian matrix is

$$\frac{\partial^2 Q'(\mathbf{w})}{\partial^2 \mathbf{w}_i} |_{\check{\mathbf{w}}_i} = -\sum_{jk} \max(\gamma_{jki}, \gamma_{jk} w_{jki}) \mathbf{v}_{jk} \mathbf{v}_{jk}^T \tag{3.53}$$

3.3.4 Update for speaker projections

The auxiliary function for the speaker projections $\mathbf{N}_i$ is analogous to that for the phonetic projections $\mathbf{M}_i$, which can be expressed as

$$\begin{aligned} Q(\mathbf{N}_i) &= \sum_{jkst} \gamma_{jki}(t) \log p(\mathbf{y}_t, j, k, i, s|\mathbf{N}_i) \\ &= -0.5 \sum_{jkst} \gamma_{jki}(t) \mathbf{v}^{(s(t))} \mathbf{N}_i^T \boldsymbol{\Sigma}_i^{-1} \mathbf{N}_i \mathbf{v}^{(s(t))} + \sum_{jkst} \mathbf{y}_{jki}(t) \boldsymbol{\Sigma}_i^{-1} \mathbf{N}_i \mathbf{v}^{(s(t))} + const \\ &= -0.5 tr\left(\boldsymbol{\Sigma}_i^{-1} \mathbf{N}_i \mathbf{R}_i \mathbf{N}_i^T\right) + tr\left(\mathbf{N}_i^T \boldsymbol{\Sigma}_i^{-1} \mathbf{Z}_i\right) + const, \end{aligned} \tag{3.54}$$

where $s(t)$ is the speaker active in frame t, and

$$\mathbf{y}_{jki}(t) = \mathbf{y}_t - \mathbf{M}_i \mathbf{v}_{jk} \tag{3.55}$$

If we use $\mathcal{T}(s)$ to denote the set of frames valid for the speaker s, then we can represent $\mathbf{R}_i$ and $\mathbf{Z}_i$ as

$$\mathbf{Z}_i = \sum_s \left(\sum_{t \in \mathcal{T}(s), jk} \gamma_{jki}(t) \right) \mathbf{y}_{jki}(t) \mathbf{v}^{(s)T} \tag{3.56}$$

$$\mathbf{R}_i = \sum_s \left(\sum_{t \in \mathcal{T}(s), jk} \gamma_{jki}(t) \right) \mathbf{v}^{(s)} \mathbf{v}^{(s)T} \tag{3.57}$$

By setting the derivative of $Q(\mathbf{N}_i)$ with respect to $\mathbf{N}_i$ to be zero, we obtain

$$\mathbf{N}_i = \mathbf{Z}_i \mathbf{R}_i^{-1} \tag{3.58}$$

Again, a more numerically stable algorithm is given in (Povey et al., 2011a) in case that $\mathbf{R}_i$ is not well conditioned.

3.3.5 Update for speaker vectors

The update for speaker vectors $\mathbf{v}^{(s)}$ is analogous to that for the sub-sate vectors $\mathbf{v}_{jk}$ except that there is no extra term relating to the weight.

$$\begin{aligned} Q(\mathbf{v}^{(s)}) &= \sum_{t \in \mathcal{T}(s), jki} \gamma_{jki}(t) \log p\left(\mathbf{y}_t, j, k, i, s|\mathbf{v}^{(s)}\right) \\ &= -0.5 \mathbf{v}^{(s)T} \mathbf{H}^{(s)} \mathbf{v}^{(s)} + \mathbf{v}^{(s)T} \mathbf{y}^{(s)} + const, \end{aligned} \tag{3.59}$$

where $\mathbf{H}^{(s)}$ and $\mathbf{y}^{(s)}$ are defined as

$$\mathbf{H}^{(s)} = \sum_i \left(\sum_{t \in \mathcal{T}(s), jk} \gamma_{jki}(t) \right) \mathbf{N}_i^T \boldsymbol{\Sigma}_i^{-1} \mathbf{N}_i \tag{3.60}$$

$$\mathbf{y}^{(s)} = \sum_{t \in \mathcal{T}(s), jki} \gamma_{jki}(t) \mathbf{N}_i^T \boldsymbol{\Sigma}_i^{-1} \mathbf{y}_{jki}(t) \tag{3.61}$$

where $\mathbf{y}_{jki}(t)$ is defined in equation (3.55). If the matrix $\mathbf{H}^{(s)}$ is well conditioned, then the update can be obtained as

$$\mathbf{v}^{(s)} = \mathbf{H}^{(s)-1}\mathbf{y}^{(s)} \tag{3.62}$$

3.3.6 Update for within-class covariances

The auxiliary function for the within-class covariance matrix $\mathbf{\Sigma}_i$ can be expressed as

$$\begin{aligned} Q(\mathbf{\Sigma}_i) &= \sum_{jkt} \gamma_{jki}(t) \log p(\mathbf{y}_t, j, k, i|\mathbf{\Sigma}_i) \\ &= -0.5 \sum_{jkt} \gamma_{jki}(t) \left(\log |\mathbf{\Sigma}_i| + tr\left(\mathbf{\Sigma}_i^{-1} \left(\mathbf{y}_t - \mathbf{M}_i \mathbf{v}_{jk} \right) \left(\mathbf{y}_t - \mathbf{M}_i \mathbf{v}_{jk} \right)^T \right) \right) + const. \end{aligned} \tag{3.63}$$

The update can be readily obtained as

$$\mathbf{\Sigma}_i = \frac{1}{\gamma_i} \left(\mathbf{S}_i + \bar{\mathbf{S}}_i - \mathbf{Y}_i \mathbf{M}_i^T - \mathbf{M}_i \mathbf{Y}_i^T \right) \tag{3.64}$$

where $\mathbf{Y}_i$ is defined in equation (3.44), $\gamma_i = \sum_{jk} \gamma_{jki}(t)$, and

$$\mathbf{S}_i = \sum_{jkt} \gamma_{jki}(t) \mathbf{y}_t \mathbf{y}_t^T \tag{3.65}$$

$$\bar{\mathbf{S}}_i = \sum_{jk} \gamma_{jki}(t) \mathbf{M}_i \mathbf{v}_{jk} \mathbf{v}_{jk}^T \mathbf{M}_i^T \tag{3.66}$$

As with conventional GMM systems, the covariance matrix $\mathbf{\Sigma}_i$ may be poorly conditioned. Variance flooring can be applied to improve the numerical stability.

3.3.7 Update for sub-state weights

The auxiliary function for sub-state weight c_{jk} is

$$Q(c_{jk}, 1 \leq j \leq J, 1 \leq k \leq K_j) = \sum_{jk} \gamma_{jk} \log c_{jk} + const \tag{3.67}$$

The update is subjected to the constraint that the sub-state weights must sum to 1 over all k for a particular state j, so the update formula is

$$c_{jk} = \frac{\gamma_{jk}}{\sum_{k=1}^{K_j} \gamma_{jk}} \tag{3.68}$$

3.4 Model extension

In the previous sections, we have shown the model parameter estimation using ML criterion. This is based on a fixed number of triphone states J, Gaussian components I as well as the subspace dimensionality, etc. For SGMMs, an important issue is to extend the model capacity, or the number of effective parameters according to the amount of available training data. This is found to improve the recognition accuracy of SGMM based acoustic models. In general, there are two ways to extend the model capacity: first by increasing the number of sub-states K_j, and second by increasing the dimensionality of model projections and speaker projections. This section describes these two techniques.

3.4.1 Sub-state splitting

In order to increase the total number of sub-states to be the desired number N, a robust way to assign the number of sub-states for each state and sub-state pair (j,k) is to have them proportional to a small power p, typically 0.2, of the total data count γ_{jk} for that state. The target number of sub-states for state j, $N(j)$ would be $N(j) = \max(1, [0.5 + \alpha\gamma_j^p])$, where $\gamma_j = \sum_{ki}\gamma_{jki}$ and $\alpha = N/\sum_j \gamma_j^p$ so that $\sum_j N(j)$ is close to the target. Suppose we are splitting the sub-state vector $\mathbf{v}_{jki}$ into two vectors $\mathbf{v}_{jk}$ and $\mathbf{v}_{jk'}$, where k' is a newly allocated sub-state index. We would split the weight and the vectors as follows:

$$\hat{c}_{jk} = 0.5 c_{jk} \tag{3.69}$$

$$\hat{c}_{jk'} = 0.5 c_{jk} \tag{3.70}$$

$$\hat{\mathbf{v}}_{jk} = \mathbf{v}_{jk} + 0.1\mathbf{H}^{(sm)-0.5}\mathbf{r} \tag{3.71}$$

$$\hat{\mathbf{v}}_{jk'} = \mathbf{v}_{jk} - 0.1\mathbf{H}^{(sm)-0.5}\mathbf{r} \tag{3.72}$$

$$\mathbf{H}^{(sm)} = \frac{1}{\sum_i \gamma_i}\sum_i \gamma_i \mathbf{H}_i \tag{3.73}$$

where $\mathbf{r}$ is a normally distributed random vector, $\mathbf{H}_i$ is defined in equation (3.32), and $\mathbf{H}^{(sm)-0.5}$ is obtained by Cholesky decomposition of $\mathbf{H}^{sm}$. This formula is analogous to increasing the number of Gaussians in conventional GMM systems (Young et al., 2002).

Table 3.1: SGMM acoustic model size. Q denotes the total number of sub-sates. $Q = 24000, I = 400, S = 40, D = 39$.

Type	globally shared			state dependent	
	$\mathbf{M}_i$	$\mathbf{w}_i$	$\mathbf{\Sigma}_i(full)$	$\mathbf{v}_{jk}$	c_{jk}
#Parameters	$I \cdot D \cdot S$	$I \cdot S$	$I \cdot (D^2 + D)/2$	$Q \cdot D$	Q
Example count	6.24×10^5	1.6×10^4	3.12×10^5	9.36×10^5	1.6×10^4
Total	1.904×10^6				

Table 3.2: GMM acoustic model size. M denotes the total number of Gaussian components. $M \approx 50000, D = 39$.

Type	$\boldsymbol{\mu}_{jm}$	w_{jm}	$\mathbf{\Sigma}_{jm}(diag)$
#Parameters	$D \cdot M$	M	$D \cdot M$
Example count	1.95×10^6	5×10^4	1.95×10^6
Total	3.905×10^6		

3.4.2 Increasing the subspace dimension

The dimension of model projection $\mathbf{M}_i$ and speaker projection $\mathbf{N}_i$ can be increased during the model update. This can be done by

$$\mathbf{M}_i \leftarrow [\mathbf{M}_i \quad \mathbf{j}_1 \ldots \mathbf{j}_{S'-S}] \tag{3.74}$$

$$\mathbf{N}_i \leftarrow [\mathbf{N}_i \quad \mathbf{j}_1 \ldots \mathbf{j}_{T'-T}] \tag{3.75}$$

where S' and T' are the target dimension of model and speaker projection. $\mathbf{j}$ is defined in equation (3.16). By this approach, the amount $S' - S$ and $T' - T$ can not be larger than the feature dimension D for each iteration. The sub-state vectors $\mathbf{v}_{jk}$ and the weight projections $\mathbf{w}_i$ would be extended by appending $S' - S$ zeros. If the speaker vectors $\mathbf{v}^{(s)}$ are being stored between iterations rather than being computed afresh each time we would also append $T' - T$ zeros to the speaker vectors $\mathbf{v}^{(s)}$.

3.5 Model size

The number of active parameters in an SGMM acoustic model can be much smaller than its GMM based counterpart. The reason is that a large proportion of the model parameters are globally shared, and the number of state-dependent parameters, i.e.

sub-state vectors $\mathbf{v}_{jk}$ and sub-state weight c_{jk} is small. We take the baseline systems in Chapter 4 as an example to illustrate this perspective. The systems were built on the WSJ0 dataset which contains around 14.8 hours training data. Both systems were tuned to achieve their best performance. For reference, the word error rate (WER) of the GMM system is 10.3% whereas for SGMM system, it is 8.3% which is significantly better. Table 3.1 and 3.2 show the total number of parameters of these two systems. It demonstrates that the total number of parameters in the SGMM system is around half that of an equivalent GMM system and half of the SGMM parameters are globally shared. Due to these features, SGMM acoustic model has a particular advantage for low-resource speech recognition which will be discussed further in Chapter 5.

3.6 Adaptation

There are approaches to adapt an SGMM acoustic model. The first one is to use the speaker subspace $\mathbf{N}_i$ as shown in equation (3.2). However, since the dimensionality of the speaker vector $\mathbf{v}^{(s)}$ is normally very low, this approach is more suitable for adaptation with very limited adaptation data, e.g. only 1 or a few utterances. Another approach is to use the MLLR or CMLLR transformation which is the same as adaptation of GMM acoustic models. The main difference is that SGMM normally use full-covariance matrices, rather than diagonal-covariances usually used by GMM acoustic models. Hence, full-covariance MLLR or CMLLR transformations have to be used for SGMMs, and examples of this method can be found in (Povey and Saon, 2006; Ghoshal et al., 2010). Recently, Ghalehjegh and Rose (2013) applied a MLLR type of transformation over the state vectors $\mathbf{v}_{jk}$ for speaker adaptation and observed some improvement.

3.7 Decoding

In principle, the same decoding algorithm which is used for conventional GMM acoustic models is applicable to an SGMM acoustic model. However, there is a minor difference from the engineering point of view. Since the number of Gaussian components of an SGMM acoustic model is normally much larger than its GMM counterpart, evaluating all the Gaussian components will make the decoder very slow. To speedup the decoder, the Gaussian selection algorithm using the UBM is applied in the decoding step. This is the same as that used to train the SGMMs which is shown in Figure 3.2.

The rationale behind this idea is that for each frame, most of the Gaussians will have very likelihood score except a few which is close to this frame in the acoustic space. In practice, if we only select a handful Gaussians for each frame, we can achieve a comparable decoding speed without having any noticeable accuracy loss.

3.8 Summary

In this chapter, we have reviewed subspace Gaussian mixture model (SGMM) for acoustic modelling, and shown the model estimation using ML criterion. Compared to conventional GMM based acoustic models, SGMMs reformulate the model parameters into globally shared model subspace and low-dimensional state vectors. The model subspace is introduced to capture the correlations among the phonetic states and speakers. It can be used as informative priors to estimate the state dependent model, especially when the amount of training data is limited. This will be discussed further with experimental studies in the following chapters.

Chapter 4

Regularized Subspace Gaussian Mixture Model

4.1 Introduction

Acoustic modeling for large vocabulary speech recognition often needs to address the problem of robust model estimation from limited acoustic data. There has recently been a renewed interest in regularization approaches to address this problem of data sparsity and model complexity. For instance, Sivaram et al. (2010) introduced an approach to obtain sparse features from an auto-associative network using an ℓ_1-norm regularization function, and Sainath et al. (2010a,b) combined ℓ_1 and ℓ_2 regularization to obtain a sparse exemplar-based representation for phoneme recognition. Regularised MLLR for speaker adaptation is proposed by Omar (2007).

In Chapter 3, we have described the SGMM based acoustic model, which reformulates a conventional HMM-GMM system. Although such a formulation significantly reduces the total number of parameters (Povey et al., 2011a), ML training may still suffer from overfitting with insufficient training data. The state-dependent parameters are especially prone to overfit, as the amount of acoustic data attributed to each state tends to be small. To be specific, the log-likelihood function of a particular (sub-)state vector $\mathbf{v}$ is approximated by a quadratic funciton which comes from the EM auxiliary function of state vectors as equation (3.39):

$$Q(\mathbf{v}) \simeq -\frac{1}{2}\mathbf{v}^T\mathbf{H}\mathbf{v} + \mathbf{g}^T\mathbf{v} + const, \tag{4.1}$$

where we have removed the subscript $_{jk}$ for brevity; $\mathbf{g}$ is a S-dimensional vector and

$\mathbf{H}$ is a $S \times S$ matrix, representing the first- and second-order statistics respectively.[1] Although the state vectors are normally low-dimensional, the amount of data for computing the statistics $\mathbf{H}$ and $\mathbf{g}$ may still be insufficient. Some heuristic approaches may be applied, for instance $\mathbf{H}$ and $\mathbf{b}$ may be smoothed by the global statistics:

$$\hat{\mathbf{H}} = \mathbf{H} + \tau \mathbf{H}^{sm}, \qquad \hat{\mathbf{g}} = \mathbf{g} + \tau \mathbf{g}^{sm} \tag{4.2}$$

where $\mathbf{H}^{sm}$ and $\mathbf{b}^{sm}$ denotes the smoothing term calculated based on all the HMM states (see (Povey, 2009) for details), and $\tau \in \mathbb{R}$ is the tuning parameter. Povey et al. (2011a) also discuss some numeric controls to tackle the poor conditioning of $\mathbf{H}$. In this chapter we address the problem using an explicit regularization function.

To regularize the estimation of the state vectors, we introduce an element-wise penalty term to the original ML objective function in order to smooth the output variables, giving:

$$\hat{\mathbf{v}} = \arg\max_{\mathbf{v}} Q(\mathbf{v}) - J_{\boldsymbol{\lambda}}(\mathbf{v}). \tag{4.3}$$

$J_{\boldsymbol{\lambda}}(\mathbf{v})$ denotes the regularization function for $\mathbf{v}$ parametrised by $\boldsymbol{\lambda}$. We may interpret $-J_{\boldsymbol{\lambda}}(\mathbf{v})$ as a log-prior for the state vector, in which case we can interpret (5.7) as a MAP estimate. However, in this paper, we treat the problem more in terms of the design and analysis of regularization functions, rather than giving an explicit Bayesian treatment as used in JFA-based speaker recognition where Gaussian priors are applied to both speaker and channel factors (Zhao et al., 2009).

In this chapter, we investigate ℓ_1- and ℓ_2-norm regularization (Hastie et al., 2005) in this context, as well as a combination of ℓ_1 and ℓ_2, sometimes referred to as the elastic net (Zou and Hastie, 2005). we introduce the regularization penalties in Section 4.2, and in Section 4.3, we present the optimisation algorithm based on gradient projection to solve the regularised objective function. Finally, in section 4.4, we present experiments on the 5,000 word Wall Street Journal (WSJ-5k) speech transcription task.

4.2 Regularization penalties

We may formulate a family of regularization penalties in terms of a penalty parameter λ, and an exponent $q \in \mathbb{R}$:

$$J_{\boldsymbol{\lambda}}(\mathbf{v}) = \lambda \sum_i |v_i|^q \quad s.t. \quad \lambda \geq 0. \tag{4.4}$$

[1] If the state is split, (4.1) should be the objective function of sub-state vectors—regularization is employed at the sub-state level in this work.

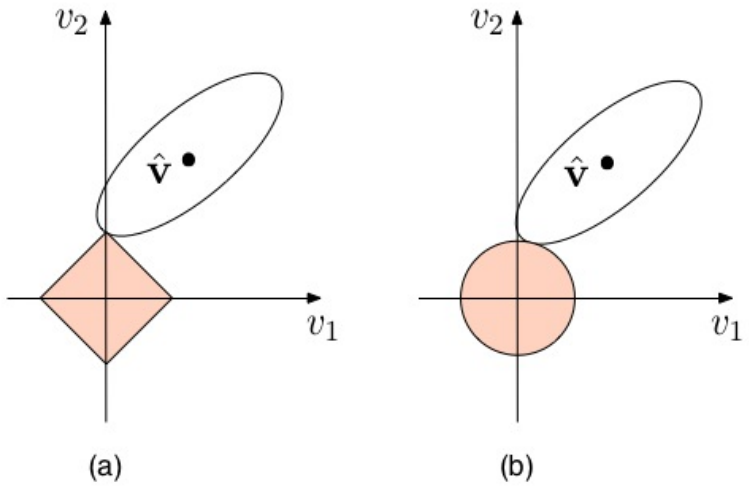


Figure 4.1: An example of ℓ_1/ℓ_2-norm penalty for a quadratic objective function in two-dimensional space. The shaded areas denote the feasible region defined by the regularization in terms of a constraint, and it is like a square for ℓ_1-norm penalty, while a circle for ℓ_2-norm penalty. Without the penalty, the solution of the quadratic function is denoted by the point in the centre of the contour. With the penalty, the solution moves to the tangent point between the contour and the feasible region.

The case $q = 1$ corresponds to ℓ_1-norm regularization, sometimes referred to as the lasso (Tibshirani, 1996), and the case $q = 2$ corresponds to ℓ_2-norm regularization, which is referred to as ridge regression (Hastie et al., 2005) or weight decay.

Both ℓ_1- and ℓ_2-norm penalties perform an element-wise shrinkage of $\mathbf{v}$ towards zero in the absence of an opposing data-driven force (Hastie et al., 2005), which enables more robust estimation. The ℓ_1-norm penalty has the effect of driving some elements to be zero, thus leading to a kind of variable selection, and inspiring its application in basis pursuit denoising (Chen et al., 2001), compressed sensing (CS) in the signal processing literature (Donoho, 2006) and sparse representation of speech features (Sivaram et al., 2010; Sainath et al., 2010a). It is possible to seek a compromise between the ℓ_1 and ℓ_2 penalties by simply setting $1 < q < 2$ which is sometimes referred to as a bridge penalty. However, the nonlinearity of the bridge penalty brings increased computational complexity. Alternatively, the ℓ_1- and ℓ_2-norm penalties can both be applied, as in elastic net regularization (Zou and Hastie, 2005):

$$\begin{aligned} J_{\boldsymbol{\lambda}}(\mathbf{v}) &= \lambda_1 \sum_i |v_i| + \lambda_2 \sum_i |v_i|^2, \\ s.t. \quad & \lambda_1, \lambda_2 \geq 0. \end{aligned} \tag{4.5}$$

This is much less computationally demanding than the bridge penalty. In this chapter,

we investigate the ℓ_1-norm, ℓ_2-norm and elastic net regularization for the estimation of SGMM state vectors.

4.2.1 Reformulation as a quadratic program

Given the regularized objective function for state vector estimation (5.7), a closed form solution is readily available for the ℓ_2-norm penalty:

$$\begin{aligned}\hat{\mathbf{v}} &= \arg\max_{\mathbf{v}} -\frac{1}{2}\mathbf{v}^T\mathbf{H}\mathbf{v} + \mathbf{g}^T\mathbf{v} - \lambda\|\mathbf{v}\|_{\ell_2} \\ &= (\mathbf{H}+\lambda\mathbf{I})^{-1}\mathbf{g}\end{aligned}$$

However, there is no such closed form solutions for the ℓ_1-norm and elastic net penalties as the derivatives of their objective functions are not continuous. In both the optimization and signal processing fields, there have been numerous approaches proposed to solve the ℓ_1-norm penalty problem and here we adopt the gradient projection algorithm of Figueiredo et al. (2007). The same approach may be applied to the elastic net penalty as it can be formulated in terms of the ℓ_1 penalty:

$$\hat{\mathbf{v}} = \arg\max_{\mathbf{v}} -\frac{1}{2}\mathbf{v}^T(\mathbf{H}+\lambda_2\mathbf{I})\mathbf{v} + \mathbf{g}^T\mathbf{v} - \lambda_1\|\mathbf{v}\|_{\ell_1}, \tag{4.6}$$

given the regularization parameters λ_1 and λ_2. A proper scaling factor should applied to the result of (4.6) to get the exact elastic net solution, but we did not do it in this work which corresponds to the naive elastic net (Zou and Hastie, 2005).

Expressing (4.3) with the ℓ_1 penalty results in the following objective function:

$$\hat{\mathbf{v}} = \arg\min_{\mathbf{v}} \frac{1}{2}\mathbf{v}^T\mathbf{H}\mathbf{v} - \mathbf{g}^T\mathbf{v} + \lambda\|\mathbf{v}\|_{\ell_1}, \quad \lambda > 0. \tag{4.7}$$

As the derivative of the objective function is not continuous, which makes the search of global optimum difficult, we introduce two auxiliary vectors $\mathbf{x}$ and $\mathbf{y}$ such that:

$$\mathbf{v} = \mathbf{x} - \mathbf{y}, \quad \mathbf{x} \geq 0, \quad \mathbf{y} \geq 0, \tag{4.8}$$

where, $\mathbf{x} = [\mathbf{v}]_+$ which takes the positive entries of $\mathbf{v}$ while keeping the rest as 0, i.e. $x_i = \max\{0, v_i\}$ for all $i = 1, \ldots, S$. Similarly, $\mathbf{y} = [-\mathbf{v}]_+$. In this case, equation (4.7) can be rewritten as

$$\begin{aligned}(\hat{\mathbf{x}}, \hat{\mathbf{y}}) = \arg\min_{\mathbf{x},\mathbf{y}} &\frac{1}{2}(\mathbf{x}-\mathbf{y})^T\mathbf{H}(\mathbf{x}-\mathbf{y}) \\ &- \mathbf{g}^T(\mathbf{x}-\mathbf{y}) + \lambda\mathbf{1}_S^T\mathbf{x} + \lambda\mathbf{1}_S^T\mathbf{y} \\ s.t. \quad &\mathbf{x} \geq 0, \mathbf{y} \geq 0\end{aligned} \tag{4.9}$$

where $\mathbf{1}_S$ denotes an S-dimensional vector whose elements are all 1. We can reformulate (4.9) further as a more standard bound-constraint quadratic program

$$\hat{\mathbf{z}} = \arg\min_{\mathbf{z}} \frac{1}{2}\mathbf{z}^T\mathbf{B}\mathbf{z} + \mathbf{c}^T\mathbf{z} \quad s.t. \quad \mathbf{z} \geq 0 \tag{4.10}$$

where we have set

$$\mathbf{z} = \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}, \mathbf{c} = \lambda\mathbf{1}_{2S} + \begin{bmatrix} -\mathbf{g} \\ \mathbf{g} \end{bmatrix}, \text{and } \mathbf{B} = \begin{bmatrix} \mathbf{H} & -\mathbf{H} \\ -\mathbf{H} & \mathbf{H} \end{bmatrix}.$$

The objective function (4.10) does not suffer the nonlinearity problem of the original objective function (4.7). If we denote $\mathcal{F}(\mathbf{z})$ as the objective function

$$\mathcal{F}(\mathbf{z}) = \frac{1}{2}\mathbf{z}^T\mathbf{B}\mathbf{z} + \mathbf{c}^T\mathbf{z} \tag{4.11}$$

its gradient is readily available as

$$\nabla\mathcal{F}(\mathbf{z}) = \mathbf{B}\mathbf{z} + \mathbf{c}, \tag{4.12}$$

which forms the basis of the gradient projection algorithm as we discussed in the following section.

4.3 Gradient projection algorithms

To solve the constraint quadratic program of equation (4.10), the underlying principle of the gradient projection algorithms described in this section is the same as the steepest descent algorithm, by which we would update the value of $\mathbf{z}$ from iteration k to $k+1$ as

$$\mathbf{z}_{k+1} = \mathbf{z}_k - \alpha_k\nabla\mathcal{F}(\mathbf{z}) \tag{4.13}$$

where the positive scalar α_k is the step length, and the negative gradient $-\nabla\mathcal{F}(\mathbf{z})$ is the decent direction. However, since $\mathbf{z}$ is positive constraint, we would like to ensure the updated $\mathbf{z}_{k+1}$ satisfies this constraint. To this end, we take the positive entries of the updated estimate as

$$\mathbf{w}_k = [\mathbf{z}_k - \alpha_k\nabla\mathcal{F}(\mathbf{z})]_+ \tag{4.14}$$

and then we interpolate between the 'old' and 'new' values by

$$\mathbf{z}_{k+1} = (1-\lambda_k)\mathbf{z}_k + \lambda_k\mathbf{w}_k \tag{4.15}$$

where $\lambda_k \in [0,1]$ is chosen subjected to that the new value $\mathbf{z}_{k+1}$ will decrease the objective function $\mathcal{F}(\mathbf{z})$. The two approaches described next differ in their choices of α_k and λ_k.

Table 4.1: The basic gradient projection algorithm.

Step 0 (Initialisation): Given $\mathbf{z}_0$, choose parameters $\beta \in (0,1)$
and $\mu \in (0, 0.5)$; set $k = 0$.
Step 1: Compute α_0 from (4.18), and replace α_0 by mid$(\alpha_{min}, \alpha_0, \alpha_{max})$.
Step 2: (backtracking line search): Choose α_k to be the first number
in the sequence $\alpha_0, \beta\alpha_0, \beta^2\alpha_0, \ldots$ such that
$\mathcal{F}(\mathbf{w}_k) - \mathcal{F}(\mathbf{z}_k) \leq \mu \nabla \mathcal{F}(\mathbf{z}_k)^T (\mathbf{z}_k - \mathbf{w}_k)$
which means we can decrease the objective function a sufficient amount
by updating $\mathbf{z}$. We then set $\mathbf{z}_{k+1} = \mathbf{w}_k$, i.e. $\lambda_k = 1$ in equation (4.15).
Step 3: Perform convergence test and terminate with $\mathbf{z}_{k+1}$ if it's satisfied.
Otherwise set $k \leftarrow k+1$ and go to Step 1.

4.3.1 Basic gradient projection

In the basic approach, we search from each iteration $\mathbf{z}_k$ along the negative gradient $-\nabla\mathcal{F}(\mathbf{z})$, projecting onto the nonnegative orthant, and performing a backtracking line search of α_k until a sufficient decrease is obtained in $\mathcal{F}(\cdot)$. We use an initial guess for α_k that would yield the exact minimiser of $\mathcal{F}(\cdot)$ along this direction if no new bounds were to be encountered. Specifically, we define the vector $\mathbf{p}_k$ by

$$(\mathbf{p}_k)_i = \begin{cases} (\nabla\mathcal{F}(\mathbf{z}_k))_i, & \text{if } (\mathbf{z}_k)_i \geq 0 \text{ or } (\nabla\mathcal{F}(\mathbf{z}_k))_i \leq 0 \\ 0, & \text{otherwise} \end{cases} \tag{4.16}$$

where $(\cdot)_i$ denotes the i_{th} element of the vector. Equation (4.16) means that after the projection, $(\mathbf{p}_k)_i$ is non-zero unless it is in the positive orthant, or its value is negative, which means it may switch sign by moving along this gradient direction even though it lies in the negative orthant. We project the gradient since $\mathbf{z}$ is positive constraint.

We then choose the initial guess to be

$$\alpha_0 = \arg\min_{\alpha} \mathcal{F}(\mathbf{z}_k - \alpha\mathbf{p}_k) \tag{4.17}$$

By setting the derivative of $\mathcal{F}(\mathbf{z}_k - \alpha\mathbf{p}_k)$ with respect to α to be zero, we can compute α_0 explicitly as

$$\alpha_0 = \frac{(\mathbf{p}_k)^T \mathbf{p}_k}{(\mathbf{p}_k)^T \mathbf{B}\mathbf{p}_k} \tag{4.18}$$

To avoid that values of α_0 are too small or too large, we confine it to the interval $[\alpha_{min}, \alpha_{max}]$, where $0 < \alpha_{min} < \alpha_{max}$. The complete algorithm is defined in Table 4.1.

Table 4.2: The Barzilai-Borwein gradient projection algorithm.

Step 0 (initialisation): Given $\mathbf{z}_0$, choose parameters $\alpha_{min}, \alpha_{max}, \alpha_0 \in [\alpha_{min}, \alpha_{max}]$, and set $k = 0$.
Step 1: Compute step: $\boldsymbol{\delta}_k = [\mathbf{z}_k - \alpha_k \nabla \mathcal{F}(\mathbf{z}_k)]_+ - \mathbf{z}_k$.
Step 2 (line search): Find the scalar λ_k that minimises $\mathcal{F}(\mathbf{z}_k + \lambda_k \boldsymbol{\delta}_k)$ on the interval $\lambda_k \in [0,1]$, and set $\mathbf{z}_{k+1} = \mathbf{z}_k + \lambda_k \boldsymbol{\delta}_k$.
Step 3 (Update α): Compute $\gamma_k = \boldsymbol{\delta}_k^T \mathbf{B} \delta_k$. If $\gamma_k = 0$, let $\alpha_{k+1} = \alpha_{max}$, otherwise $\alpha_{k+1} = \text{mid}\left(\alpha_{min}, \frac{\|\boldsymbol{\delta}_k\|_2^2}{\gamma_k}, \alpha_{max}\right)$.
Step 4: Perform convergence test and terminate with $\mathbf{z}_{k+1}$ if it's satisfied. Otherwise set $k \leftarrow k+1$ and go to Step 1.

4.3.2 Barzilai-Borwein gradient projection

The basic gradient projection algorithm described above ensures that the objective function $\mathcal{F}(\cdot)$ decrease at every iteration. Here, we describe another approach proposed by Barzilai and Borwein (1988) which does not have this property but is still efficient to search the optimum. This approach was originally developed in the context of unconstrained minimisation of a smooth nonlinear function $\mathcal{F}(\cdot)$. It calculates each step by the formula $\boldsymbol{\delta}_k = -H_k^{-1} \nabla \mathcal{F}(\mathbf{z}_k)$, which is an approximation to the Hessian of $\mathcal{F}(\mathbf{z}_k)$. Barzilai and Borwein (1988) set H_k to be a multiple of the identity $H_k = \eta_k I$, where η_k is chosen so that H_k has similar behaviour to the true Hessian over the most recent step, i.e.

$$\nabla \mathcal{F}(\mathbf{z}_k) - \nabla \mathcal{F}(\mathbf{z}_{k-1}) \approx \eta_k [\mathbf{z}_k - \mathbf{z}_{k-1}] \tag{4.19}$$

with η_k chosen to satisfy this relationship in the least-squares sense. In the unconstrained setting, the update formula is

$$\mathbf{z}_{k+1} = \mathbf{z}_k - (\eta_k)^{-1} \nabla \mathcal{F}(\mathbf{z}_k) \tag{4.20}$$

and this step is taken even if it yields an increase in $\mathcal{F}(\cdot)$.

The Barzilai-Borwein approach has been extended to solve the bound-constraint quadratic optimisation problems (Serafini et al., 2005) and we have used this algorithm for the study here. We choose λ_k in equation (4.15) as the exact minimiser over the interval $[0,1]$ and choose the value of α_k where $\alpha_k = (\eta_k)^{-1}$. The complete algorithm is given in Table 4.2.

4.3.3 Termination criterion

For both of the basic and Barzilai-Borwein gradient projection algorithms, we need to set a criterion to terminate the algorithms. Ideally, we wish for the approximated solution $\mathbf{z}$ to be reasonably close the a solution $\mathbf{z}^*$ and/or that the function value $\mathcal{F}(\mathbf{z})$ be reasonably close to $\mathcal{F}(\mathbf{z}^*)$. Figueiredo et al. (2007) discussed several termination conditions. In this work, we used the following criterion that takes account of the nonzero indices of $\mathbf{z}$ and of how these indices have changed in recent iterations. In this criterion, termination is declared when the set of nonzero indices of an iterate $\mathbf{z}_k$ changes by a relative amount of less than a specified threshold ε. Specifically, we define

$$I_k = \{i | [\mathbf{z}_k]_i \neq 0\}$$
$$\mathcal{C}_k = \{i | i \in I_k \text{ and } i \notin I_{k-1} \text{ or } i \notin I_k \text{ and } i \in I_{k-1}\}$$

Namely, I_k denotes the nonzero indices at iteration k and $\mathcal{C}_k$ denotes the indices that have changed from/to zero from iteration $k-1$ to k. We terminate the algorithm if

$$|\mathcal{C}_k| / |I_k| \leq \varepsilon \tag{4.21}$$

The value of ε should be set beforehand. In addition to this criterion, we also introduce the minimum and maximum number of iterations for the algorithm.

4.4 Experiments

We use the WSJ-5k data for our speech transcription experiments. We follow the setup described in (Woodland et al., 1994). The training set contains 7137 utterances with a total duration of about 14 hours (after removing silence). For testing, we use subset of the WSJ1-5k development set obtained by deleting sentences with out-of-vocabulary words giving a total of 248 sentences from 10 speakers. We use the standard 5k non-verbalised punctuation bigram language model (LM) for decoding. Standard 13-dimension MFCC$+\Delta+\Delta\Delta$ features were used with cepstral mean and variance normalisation. The following results were obtained by tuning the LM scaling factor and word insertion penalty to get the best word error rate (WER).

4.4.1 Baseline system

We first train a conventional HMM-GMM baseline recognizer using the HTK speech recognition toolkit (Young et al., 2002). The baseline system has 3093 tied cross-

word triphone states, each with a 16-component GMM with diagonal covariance. Our baseline result of of 10.3% WER on the test set is comparable to the 10.48% WER reported in (Woodland et al., 1994) using a similar configuration. Starting from the HTK baseline system, we train the SGMM system according to the recipe using the Kaldi software described in (Povey et al., 2011a), using 400 Gaussian components in the universal background model (UBM) and 40-dimensional phonetic subspace (i.e., $S = 40$). State splitting was applied to increase the number of sub-states for large model capacity. The best performance of SGMM baseline is 8.6%, which gives more than 15% relative improvement compared to the conventional system.

4.4.2 SGMM results with smoothing and renormalization

We first compare the performance of ad-hoc smoothing shown in equation (4.2). The results are given in Table 4.3 for different values of the smoothing parameter τ. We also present the results by renormalization (Appendix K in (Povey, 2009)) denoted as $R(\mathbf{v})$ in Table 4.3. While we do not observe much improvements from the ad-hoc smoothing approach, from the results of using a small smoothing term ($\tau = 5$) compared to the non-smoothed case ($\tau = 0$), the smoothing terms can indeed help to address the overfitting issue, albeit rather mildly. Renormalization, however, is beneficial to both system performance and model robustness. While theoretically, renormalization does not change the model, in practice it makes a difference due to issues such as numerical stability of the updates, flooring, and condition limiting of matrices.

4.4.3 SGMM results with regularization

Here the regularization is applied at the sub-state level for systems with sub-state splitting. The regularization parameter is set to be global and constant for different numbers of sub-states, and except for regularized estimation of the sub-state vectors, the SGMM training follows the recipe in (Povey et al., 2011a).

Table 4.4 shows the results of regularization with ℓ_1, ℓ_2 as well as elastic net penalty for systems with and without renormalization. To simplify the tuning of the two reguarization parameters for elastic net system, we set them to be equal. The Barzilai-Borwein gradient projection optimization algorithm described in section 4.3.2 was used in our experiments. For the systems without renormalization, the regularization parameters are set to be 10 for all ℓ_1, ℓ_2 and elastic net systems (i.e. $\lambda_1 = \lambda_2 = 10$ in equation 4.5). Compared to the baseline, the SGMM system with regularization is

Table 4.3: Word error rates of SGMM acoustic model with ad-hoc smoothing or renormalization, $S = 40$

GMM baseline: 10.3					
SGMM with ad-hoc smoothing or renormalization					
#Substates	$R(\mathbf{v})$	$\tau = 0$	$\tau = 5$	$\tau = 10$	$\tau = 20$
3k	9.7	9.8	9.9	10.0	10.1
4.5k	9.7	9.6	9.7	9.7	9.8
6k	9.7	9.4	9.4	9.5	9.6
9k	9.2	9.1	9.2	9.2	9.2
12k	9.0	8.8	8.9	9.1	9.1
16k	8.8	**8.6**	8.8	**8.9**	**8.6**
20k	8.8	8.7	8.7	9.3	8.9
24k	**8.3**	8.8	8.6	9.1	8.8
28k	8.5	8.7	8.7	9.1	8.8
32k	8.7	9.0	**8.5**	9.4	9.7

Table 4.4: Comparison of SGMM acoustic model with regularized (sub-)state vector estimation, $S = 40$

#Sub	without renormalization				with renormalization			
-states	-	ℓ_1	ℓ_2	*eNet*	-	ℓ_1	ℓ_2	*eNet*
3k	9.8	9.7	9.9	9.9	9.7	10.2	9.7	9.9
4.5k	9.6	9.4	9.7	9.6	9.7	9.8	9.7	9.9
6k	9.4	9.4	9.4	9.4	9.7	9.7	9.4	9.6
9k	9.1	9.1	9.1	9.3	9.2	9.2	9.2	9.5
12k	8.8	9.0	8.8	8.9	9.0	8.8	9.1	9.5
16k	**8.6**	8.8	**8.4**	8.7	8.8	8.9	8.9	9.1
20k	8.7	**8.3**	8.8	8.6	8.8	8.7	**8.4**	9.2
24k	8.8	8.4	8.7	**8.5**	**8.3**	8.5	8.6	**9.0**
28k	8.7	8.4	8.5	8.5	8.5	8.4	8.7	9.2
32k	9.0	8.5	8.5	8.8	8.7	**8.3**	9.0	9.2

Table 4.5: Results of SGMM system with ℓ_1-norm regularization, $S = 60$

#Sub-states	3k	4.5k	6k	9k	12k	16k	20k
Baseline	9.6	9.5	**9.1**	9.3	9.2	9.2	9.3
ℓ_1-norm	9.6	9.2	9.0	9.0	9.0	8.9	**8.9**

less likely to suffer from overfitting, as the best results are achieved by models with large capacity, and also obtain moderate improvement, which agrees with the argument of regularization in this chapter. We do not observe a significant difference between the ℓ_1 and ℓ_2-norm penalties in terms of performance, and the elastic net does not give further gains. In our experiments, the ℓ_1 penalty does not give sparse solution when the number of sub-states is small, however, with further sub-state splitting, a considerable amount of sub-state vectors are driven to be sparse, e.g. the proportion of zero entries can be 10-20% for some of them.

With renormalization, the regularization is still efficient in avoiding model overfitting with larger models, as shown by the results in Table 4.4. However, we do not observe performance gains in this case. This shows that, in the previous setting, regularization was providing better performance by improving the numerical stability of the updates. It is worth noting that with renormalization, the regularization parameters need to be much smaller, for example we use $\lambda_1 = \lambda_2 = 2$ for these experiments. Also, the system is more sensitive to the choice of the regularization parameters. This corroborates the assumption that without renormalization, the updates of the globally-shared parameters $\mathbf{M}$ and $\mathbf{w}$ can ameliorate over-penalization of the state-vectors to an extant.

4.4.4 Extensions

In this chapter, we focused on the regularized estimation of the state-dependent parameters. However, this approach can be extended to the estimation of the global shared parameters, i.e. $\mathbf{M}$, $\mathbf{w}$ and $\mathbf{\Sigma}$, which we will explore in future work. As in our experiments, we observe that except for the state vectors, these state independent parameters may also suffer from the data sparsity problem which limits the model capacity, especially for higher dimensional subspaces.

Table 4.5 shows the results of SGMM model with ℓ_1-norm regularization (without renormalization), in which the dimension of state vector is increased to 60. Compared to the 40-dimensional subspace SGMM systems in Table 4.4, we do not achieve any

improvement but notable degradation for both baseline and ℓ_1 regularized systems, which is partly due to the poor estimation of the globally shared parameters. Based on the approach presented here, extending the regularized estimation to the state independent parameters is not difficult, as we can reformulate the objective functions of these parameters into their quadratic forms, by which the code used for state vector regularization can be shared.

4.5 Summary

In this chapter, we have investigated regularized state model estimation for the subspace GMM acoustic model. Given the original ML based objective function, we added regularization penalties based on the ℓ_1-norm and the ℓ_2-norm, as well as their combined form, the elastic net. From our experimental results on WSJ-5k speech transcription task, we have observed reductions in word error rate and improved model robustness by all the three types of regularization. While the performance gains are found to be mostly due to improved numerical stability of the updates, which can also be achieved by renormalizing the phonetic subspaces, regularization is shown to prevent overfitting with larger models. This may prove helpful in training acoustic models with lesser resources. This will be demonstrated in the cross-lingual experiments in Chapter 5.

Chapter 5

Cross-lingual Subspace Gaussian Mixture Model

5.1 Introduction

Large vocabulary continuous speech recognition systems rely on the availability of substantial resources including transcribed speech for acoustic model estimation, in-domain text for language model estimation, and a pronunciation dictionary. Building a speech recognition system from scratch for a new language thus requires considerable investment in gathering these resources. For a new language with limited resources, conventional approaches to acoustic modelling normally result in much lower accuracy. There has been extensive amount of work to improve the accuracy of speech recognizers in low-resource conditions, focusing on estimating models from limited amounts of transcribed audio in the target language (Schultz and Waibel, 1997, 1998; Byrne et al., 2000; Le and Besacier, 2005; Thomas et al., 2010) or lack of a pronunciation dictionary (Slobada and Waibel, 1996; Singh et al., 2000; Goel et al., 2010). In this chapter, we study cross-lingual acoustic modelling with the objective of porting information from one or more source language systems, built using larger amounts of training data, to build a system for a target language for which only limited amounts of transcribed audio are available. However, owing to differences such as different sets of subword units, sharing the knowledge among multiple languages is not a straightforward task. The main approaches to cross-lingual acoustic modelling, discussed below, include the use of *global phone sets*, cross-lingual *phone/acoustic* mapping, cross-lingual *tandem features* and the use of *KL-divergence HMMs*.

5.1.1 Global phone set approach

Schultz and colleagues (Schultz and Waibel, 1997, 1998, 2001; Schultz, 2002) investigated the construction of language-independent speech recognition systems by pooling together all the phoneme units, as well as the acoustic training data, from a set of monolingual systems. The resultant multilingual acoustic model may be used to perform transcription directly, or may serve as a seed model to be adapted to the target language (Schultz and Waibel, 1997, 2001). However, an important problem with this approach is that the number of phone units grows as the number of languages to be covered increases. This may lead to inconsistent parameter estimation and, consequently, degradation in accuracy (Kohler, 1996), especially in case of context-dependent modelling. To overcome this problem, instead of using a universal phone set, a set of universal speech attributes may be used which represent similar sounds across language than phone units (Siniscalchi et al., 2012). The fundamental speech attributes which can be viewed as a clustering of phonetic features, such as voicing, nasality and frication, can be modelled from a particular source language and shared across many different target languages. In practice, a bank of detectors using neural networks (Siniscalchi et al., 2012), for instance, may be employed to extract the universal attributes.

5.1.2 Cross-lingual phone/acoustic mapping

Rather than constructing a global phone set, the mismatch of phone units between source and target languages may be addressed by a direct cross-lingual mapping between phones or between acoustic models. Both knowledge-based (Byrne et al., 2000; Le and Besacier, 2005) and data-driven (Sim and Li, 2008; Sim, 2009) approaches have been investigated. Given a cross-lingual mapping, either the target acoustic model is derived from the source acoustic model, or the transcription of target speech is performed using the mapped source acoustic model (Sim, 2009).

5.1.3 Cross-lingual tandem features

Tandem features, based on phone posterior probability estimates, were originally proposed to improve monolingual speech recognition (Hermansky et al., 2000), but they have also proven effective in the cross-lingual setting. In this approach, multi-layer perceptrons (MLPs), trained using source language acoustic data, are used to generate MLP phone posterior features for the target language (Stolcke et al., 2006; Çetin et al.,

2007; Thomas et al., 2010; Qian et al., 2011; Plahl et al., 2011; Lal, 2011). In addition, the training data of the target language may also be used to adapt the MLPs to fit the target system better (Thomas et al., 2010). Recent advances in using MLPs with multiple hidden layers (deep neural networks, DNNs) (Dahl et al., 2012) have shown great promise for DNN-based cross-lingual acoustic modelling (Swietojanski et al., 2012).

5.1.4 Cross-lingual KL-HMMs

KL-divergence HMM based acoustic modelling (Aradilla et al., 2008) is a recently proposed approach which has shown good performance in low-resource condition (Imseng et al., 2011, 2012). In this framework, a global phone set is first obtained by manually mapping the phones in the different languages to a common phone set (for example, IPA or X-SAMPA). A multilingual MLP phoneme classifier is trained using the data from all the source languages. For the target language system, the phoneme posterior features are extracted given the MLP. Each HMM state is parameterised by a multinomial distribution, and the model is estimated by minimizing the KL-divergence between the posterior features and HMM state multinomial coefficients. The benefits of this approach are that the multilingual information can be explored by the MLP classifier and the number of multinomial parameters is much smaller than conventional GMMs which is particular suitable for low-resource speech recognition.

5.1.5 Overview of this chaper

In this chapter, we will show that the SGMM acoustic model is particularly effective for low-resource speech recognition (Lu et al., 2011a, 2012c) by utilising the structure of the model. The discussion in this chapter will focus on: 1) while the accuracy of conventional speech recognizers degrade significantly in low-resource condition, a cross-lingual SGMM acoustic model can achieve remarkable gain since a large proportion of the model parameters can be estimated using the training data of source languages; 2) building systems with limited training data may encounter numerical and overfitting, as we observed in cross-lingual SGMMs. To overcome it, ℓ_1-norm regularization was developed to improve the robustness of model estimation, and higher recognition accuracy was obtained; 3) a potential mismatch may exist between the training data from the source and target languages owing to phoneme characteristic, corpus recording conditions and speaking style. This may hurt the gain achieved by sharing multilingual information in cross-lingual SGMMs. To address this issue, max-

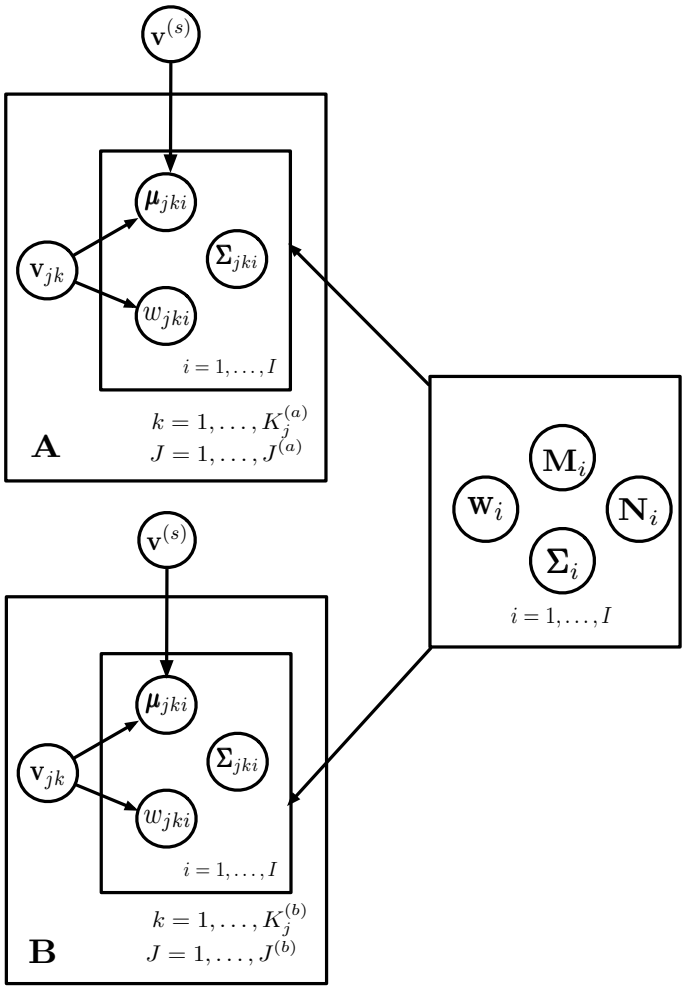


Figure 5.1: An example of multilingual estimation of the globally shared parameters $\boldsymbol{\Phi}_i = (\mathbf{M}_i, \mathbf{N}_i, \mathbf{w}_i, \boldsymbol{\Sigma}_i)$ where we tie them across two source language system $\mathbf{A}$ and $\mathbf{B}$.

imum a posteriori (MAP) adaptation is investigated to adapt the parameters toward the target system using the training data; 4) in low-resource condition, the number of speakers may be small and it is not sufficient to estimate the speaker subspace directly for the SGMMs for speaker adaptive training. However, by utilizing the model structure, the speaker subspace can also be estimated from source languages and applied to the target system.

5.2 Cross-lingual model estimation

As mentioned in Section 5.1, One of the main barriers preventing acoustic knowledge being shared across different languages is the mismatch of phone units between languages. Conventional methods tackle this problem by using global phone units or

through the use of tandem features. However in an SGMM acoustic model the globally shared parameters $\mathbf{\Phi}_i = (\mathbf{M}_i, \mathbf{N}_i, \mathbf{w}_i, \mathbf{\Sigma}_i)$ do not depend on the HMM topology, and hence are independent of the definition of the phone units. Therefore, when using SGMMs for cross-lingual acoustic modelling, the phoneme unit mismatch problem is to some degree bypassed, since we can estimate the globally shared parameters using multilingual training data by tying the globally shared parameters across the available source language systems.

Figure 5.1 demonstrates an example of the multilingual SGMM system in which source language systems **A** and **B** may have different phone units and HMM topologies, provided that the acoustic feature parameterisation and the dimensionality of model subspace are the same. By training a multilingual SGMM system in this way the accuracy for each of the source languages may be improved (Burget et al., 2010), and the multilingual globally shared parameters can be ported to a new target language system with limited training data (Burget et al., 2010; Lu et al., 2011a). In an SGMM the globally shared parameters typically account for a large proportion of the total number of parameters (Table 5.1). The reuse of the globally shared parameters across languages thus significantly reduces the required amount of acoustic training data — only the state dependent parameters $(\mathbf{v}_{jk}, c_{jk})$ need be estimated from target language data.

Using multiple source language systems to estimate the globally shared parameters $\mathbf{\Phi}_i$ involves some modifications in the SGMM training procedure. However, these modifications are minor and relatively simple, since given $\mathbf{\Phi}_i$ each source language system is independent — therefore the statistics for each source language system can be accumulated in the standard fashion using either the Viterbi alignment or the Baum-Welch algorithm. In each iteration, the corresponding statistics are then summed across languages to update the globally shared parameters. The state dependent parameters $(\mathbf{v}_{jk}, c_{jk})$ are updated in the standard fashion, for each language separately. Consider $\mathbf{M}_i$: for the system of Figure 5.1, after obtaining the statistics for each source language system $(\mathbf{Y}_i^{(a)}, \mathbf{Y}_i^{(b)})$ and $(\mathbf{Q}_i^{(a)}, \mathbf{Q}_i^{(b)})$, the final statistics are obtained simply by

$$\mathbf{Y}_i = \mathbf{Y}_i^{(a)} + \mathbf{Y}_i^{(b)}, \quad \mathbf{Q}_i = \mathbf{Q}_i^{(a)} + \mathbf{Q}_i^{(b)}. \tag{5.1}$$

Then $\mathbf{M}_i$ can be updated as usual (3.47). Similar approach can be used to update $\mathbf{N}_i, \mathbf{w}_i$ and $\mathbf{\Sigma}_i$ using the multilingual data. To build a cross-lingual SGMM system, these parameters are ported into target language system directly, and only the state dependent parameters $\mathbf{v}_{jk}$ and c_{jk} are estimated using the (limited) in-domain training

Table 5.1: The number of parameters of an SGMM acoustic model. Q denotes the total number of sub-sates. A large portion of the total parameters, e.g. more than 60% for systems in Chapter 4, are globally shared.

Type	globally shared				state dependent	
	$\mathbf{M}_i$	$\mathbf{N}_i$	$\mathbf{w}_i$	$\mathbf{\Sigma}_i$	$\mathbf{v}_{jk}$	c_{jk}
#Parm	$I \cdot D \cdot S$	$I \cdot D \cdot T$	$I \cdot S$	$I \cdot (D^2+D)/2$	$Q \cdot D$	Q
Total	$I(D^2/2+DS+DT+S+D/2)$				$Q(D+1)$	

data. Our previous experimental results (Lu et al., 2011a) indicate that this approach can significantly reduce the WER in the low-resource condition.

5.3 Cross-lingual model adaptation

In the cross-lingual SGMM system, the globally shared parameters are trained using the out-domain data. This may introduce a mismatch between the target language system and these parameters, owing to differences in phonetic characteristics, corpus recording conditions, and speaking styles. Since the amount of training data may not be sufficient to allow the global parameters to be updated using maximum likelihood (ML), the mismatch may be alleviated by adaptation approach based maximum a posteriori (MAP) criterion. In particular, we have studied the adaptation of $\mathbf{M}_i$ using MAP (Lu et al., 2012c).

In ML estimation of the phonetic subspace (Povey et al., 2011a), the auxiliary function for $\mathbf{M}_i$ is given by equation (3.43). If a prior term is introduced, then the auxiliary function becomes:

$$\tilde{Q}(\mathbf{M}_i) = Q(\mathbf{M}_i) + \tau \log P(\mathbf{M}_i), \tag{5.2}$$

where $P(\mathbf{M}_i)$ denotes the prior distribution of matrix $\mathbf{M}_i$, and τ is the smoothing parameter which balances the relative contributions of the likelihood and prior. Although any valid form of $P(\mathbf{M}_i)$ may be used, in practical MAP applications a conjugate prior distribution is often preferred for reasons of simplicity. As in (Lu et al., 2012c), $P(\mathbf{M}_i)$ is set to be a Gaussian distribution which is conjugate to the auxiliary $Q(\mathbf{M}_i)$.

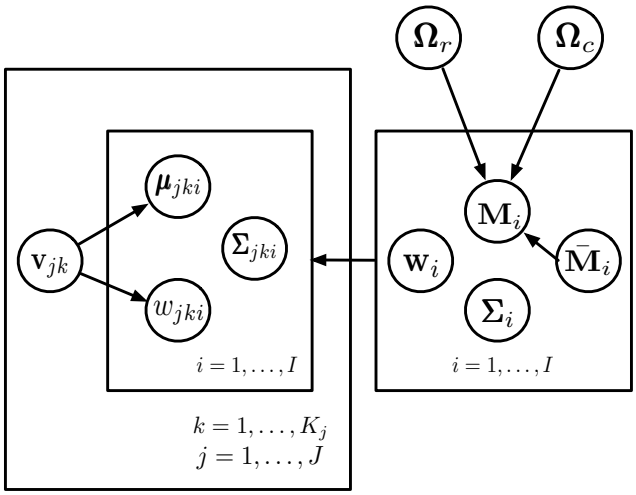


Figure 5.2: MAP adapation of $\mathbf{M}_i$ in SGMM acoustic model. $(\bar{\mathbf{M}}_i, \boldsymbol{\Omega}_r, \boldsymbol{\Omega}_c)$ denote the hyper-parameters of the Gaussian prior $P(\mathbf{M}_i)$, in which the mean $\bar{\mathbf{M}}_i$ is indexed by I while the covariances $\boldsymbol{\Omega}_r$ and $\boldsymbol{\Omega}_c$ are global.

5.3.1 Matrix variate Gaussian prior

The Gaussian distribution of random matrices is well understood (Gupta and Nagar, 1999). A typical example of its application in speech recognition is maximum a posteriori linear regression (MAPLR) (Siohan et al., 2001) for speaker adaptation, in which a matrix variate prior was used for the linear regression transformation matrix. The Gaussian distribution of a $D \times S$ matrix $\mathbf{M}$ is defined as:

$$\log P(\mathbf{M}) = -\frac{1}{2}\Big(DS\log(2\pi) + D\log|\boldsymbol{\Omega}_r| + S\log|\boldsymbol{\Omega}_c| + tr\big(\boldsymbol{\Omega}_r^{-1}(\mathbf{M}-\bar{\mathbf{M}})\boldsymbol{\Omega}_c^{-1}(\mathbf{M}-\bar{\mathbf{M}})^T\big)\Big), \quad (5.3)$$

where $\bar{\mathbf{M}}$ is a matrix containing the expectation of each element of $\mathbf{M}$, and $\boldsymbol{\Omega}_r$ and $\boldsymbol{\Omega}_c$ are $D \times D$ and $S \times S$ positive definite matrices representing the covariance between the rows and columns of $\mathbf{M}$, respectively. $|\cdot|$ and $tr(\cdot)$ denote the determinant and trace of a square matrix. This prior distribution is conjugate to auxiliary function (3.43). This matrix density Gaussian distribution may be written as:

$$Vec(\mathbf{M}) \sim \mathcal{N}(Vec(\bar{\mathbf{M}}), \boldsymbol{\Omega}_r \otimes \boldsymbol{\Omega}_c), \quad (5.4)$$

where $Vec(\cdot)$ is the vectorization operation which maps a $D \times S$ matrix into a $DS \times 1$ vector, and $\otimes$ denotes the Kronecker product of two matrices. In this formulation, only $\boldsymbol{\Omega}_r \otimes \boldsymbol{\Omega}_c$ is uniquely defined, and not the individual covariances $\boldsymbol{\Omega}_r$ and $\boldsymbol{\Omega}_c$, since for any $\alpha > 0$, $(\alpha\boldsymbol{\Omega}_r, \frac{1}{\alpha}\boldsymbol{\Omega}_c)$ would lead to the same distribution. However, this is not of

concern in the current application to MAP adaptation. Figure 5.2 illustrates the concept of using the Gaussian prior to adapt the model subspace $\mathbf{M}_i$. In this case, the auxiliary function for MAP adaptation would be

$$\tilde{\mathcal{Q}}(\mathbf{M}_i) \propto tr\Big(\mathbf{M}_i^T\boldsymbol{\Sigma}_i^{-1}\mathbf{Y}_i + \tau\mathbf{M}_i^T\boldsymbol{\Omega}_r^{-1}\bar{\mathbf{M}}_i\boldsymbol{\Omega}_c^{-1}\Big) - \frac{1}{2}tr\Big(\boldsymbol{\Sigma}_i^{-1}\mathbf{M}_i\mathbf{Q}_i\mathbf{M}_i^T + \tau\boldsymbol{\Omega}_r^{-1}\mathbf{M}_i\boldsymbol{\Omega}_c^{-1}\mathbf{M}_i^T\Big). \quad (5.5)$$

5.3.2 Prior distribution estimation

To apply MAP, the prior distribution $P(\mathbf{M}_i)$ for each $\mathbf{M}_i$, should be estimated first. This requires the estimation of the mean matrices $\bar{\mathbf{M}}_i$, and the row and column covariances $\boldsymbol{\Omega}_r$ and $\boldsymbol{\Omega}_c$. Given a set of samples generated by $P(\mathbf{M}_i)$, the ML estimation of the mean, and the row and column covariances, is described by Dutilleul (Dutilleul, 1999). This is used with some heuristic rules for cross-lingual SGMMs (Lu et al., 2012c), in which, the MAP formulation is based on the assumption that the multilingual estimate of the global subspace parameters serves a good starting point, which has been empirically verified earlier (Lu et al., 2011a). To apply MAP adaptation, we set these multilingual parameters to be the mean of the prior $P(\mathbf{M}_i)$ and update both the state-specific $\mathbf{v}_{jm}$ and the global $\mathbf{M}_i$. With a sufficiently large value of τ in (5.2), we can shrink the system back to the cross-lingual baseline, whereas $\tau = 0$ corresponds to the ML update.

The covariance matrices for each $P(\mathbf{M}_i)$ are set to be global in order to reduce the number of hyper-parameters in the prior distributions. In (Lu et al., 2012c), we compared different forms of the two covariance matrices $(\boldsymbol{\Omega}_r, \boldsymbol{\Omega}_c)$ and the experimental results indicated that using the identity matrix $\mathbf{I}$ for $\boldsymbol{\Omega}_r$ and $\boldsymbol{\Omega}_c$ worked well. Using this configuration, the MAP adaptation of $\mathbf{M}_i$ is equivalent to applying ℓ_2-norm regularizaiton by setting the multilingual estimate as the model origin. In this case, the auxiliary function (A.1) will become

$$\tilde{\mathcal{Q}}(\mathbf{M}_i) \propto tr\Big(\mathbf{M}_i^T\boldsymbol{\Sigma}_i^{-1}\mathbf{Y}_i + \tau\mathbf{M}_i^T\bar{\mathbf{M}}_i\Big) - \frac{1}{2}tr\Big(\boldsymbol{\Sigma}_i^{-1}\mathbf{M}_i\mathbf{Q}_i\mathbf{M}_i^T + \tau\mathbf{M}_i\mathbf{M}_i^T\Big). \quad (5.6)$$

The solution can be obtained in (Povey, 2009; Lu et al., 2012c). In this work, this configuration is adopted in the MAP adaptation experiments. The solution is given in Appendix A. While the formula for the matrix form MAP adaptation has already described by many others, e.g. (Povey, 2009; Siohan et al., 2001), the major novelty here is its application to the cross-lingual setting.

5.4 Cross-lingual model regularization

As discussed before, in a cross-lingual SGMM system, the globally shared parameters can be estimated from the out-domain or multilingual data. In this case, only the state-dependent parameters should be estimated using the in-domain target training data. This can significantly reduce the amount of data required to build the target system, and hence, it's suitable for low-resource speech recognition scenario (Lu et al., 2011a, 2012c). However, when the training data is extremely sparse, e.g. with only 1 hour training data, estimating the state dependent parameter $\mathbf{v}_{jk}$ directly may face overtraining or numerical instabilities since only very few acoustic frames may aligned to each HMM state. In this case, regularizing the estimation of state vectors has been proven beneficial to avoid model overtraining (Lu et al., 2011b). In addition, with regularizaiton it also enable us to use larger size of model subspace in the globally shared parameters as the out-domain data may be rich. In this case, more informative knowledge may be explored from the out-domain data and result in higher accuracy for the cross-lingual system (Lu et al., 2011a).

In Chapter 4, we studied the regularized state vector estimation using ℓ_1-norm, ℓ_2-norm as well as their combined form, the elastic net penalty. Regularization using ℓ_1-norm penalty was found particular effective, and it will be shown that it's also the case for cross-lingual systems in this chapter. With the ℓ_1-norm penalty, the auxiliary function for sub-state vector estimation becomes

$$\hat{\mathbf{v}} = \arg\max_{\mathbf{v}} Q(\mathbf{v}) - \lambda||\mathbf{v}||_{\ell_1}, \quad \lambda > 0. \tag{5.7}$$

where we have dropped the subscript $_{jk}$ for brevity. λ is the global penalty parameter. Intuitively, ℓ_1-norm penalty perform an element-wise shrinkage of $\mathbf{v}$ towards zero in the absence of an opposing data-driven force (Hastie et al., 2005), which enables more robust estimation. The ℓ_1-norm penalty also has the effect of driving some elements to be zero, thus leading to a kind of variable selection, and has been used in sparse representation of speech features (Sivaram et al., 2010; Sainath et al., 2010a), and compressed sensing (Donoho, 2006) for signal processing due to this property. For the case of cross-lingual SGMMs, ℓ_1-norm penalty can select the relevant basis in $\mathbf{M}_i$ according to the amount of available data to estimate $\mathbf{v}_{jk}$ while avoiding overtraining led by ML criterion. However, with ℓ_1-norm penalty, the solution of the auxiliary function is not readily available since the derivative of the auxiliary function is not continuos. In (Lu et al., 2011b), we applied the gradient projection based optimization approach proposed in (Figueiredo et al., 2007) to obtain the solution. Note that, the

Table 5.2: The number of phones and speakers, the amount of training data (hours) for the 4 languages used in this chapter.

Language	#Phones	#Speakers	Trn(h)
German (GE)	44	77	14.8
Spanish (SP)	43	97	17.2
Portuguese (PT)	48	101	22.6
Swedish (SW)	52	98	17.4

idea of regularization can also be applied to other types of parameters in SGMMs. In fact, we has shown in Section 5.3 that, if we set the two covariance matrices $(\mathbf{\Omega}_r, \mathbf{\Omega}_c)$ to be $\mathbf{I}$, applying MAP adaptation to $\mathbf{M}_i$ using the Gaussian prior is equivalent to the ℓ_2-norm regularizaiton by using the prior mean as the model origin.

5.5 Experiments

Cross-lingual experiments using SGMMs were carried out on the GlobalPhone corpus (Schultz, 2002), which contains 19 languages including Arabic, Chinese and a number of European languages. The corpus comprises speech from about 100 speakers per language, reading newspapers in their native language. Recordings were made under relatively quiet conditions using close-talking microphones. Acoustic conditions may vary within a language and between languages, hence acoustic mismatches may affect the performance of cross-lingual systems. In our experiments, German (GE) was used as the target language, and Spanish (SP), Portuguese (PT), and Swedish (SW) as the source languages. Table 5.2 describes the data for each language used in the experiments in terms of the number of phonemes and speakers, and the amount of available audio.

To investigate of limited acoustic training data, we constructed two randomly selected training subsets of the target language German data each containing 1 hour (8 speakers) and 5 hours (40 speakers) of data, both using 7–8 minutes of recorded speech for each of the selected speakers. We used these data subsets, in addition to the full 14.8 hours (referred to as 15 hours) of German training data, as the three target language training sets in the following experiments.

Table 5.3: WERs of baseline GMM and SGMM systems using 1 hour, 5 hour and 14.8 hour training data

System	1 hour		5 hour		14.8 hour	
	dev	eval	dev	eval	dev	eval
GMM	23.2	34.1	18.5	28.0	15.4	24.8
SGMM	20.4	31.4	14.9	24.9	13.0	22.1
#states	831		1800		2537	

Table 5.4: Total trace of covariance and subspace matrices given by the source SGMM systems, $S = 40$.

	SP	PT	SW	Multilingual
# of states	1926	2929	2400	-
# of sub-states	20k	20k	20k	-
$\sum_i tr(\mathbf{\Sigma}_i)/I$	1037	1065	1056	1051
$\sum_i tr(\mathbf{M}_i\mathbf{M}_i^T)/I$	5997	3605	2735	2963

5.5.1 Baseline monolingual systems

We constructed baseline systems using the three training sets (1h / 5h / 15h) in a monolingual fashion, using conventional GMM and SGMM acoustic modelling. The systems were built using the Kaldi speech recognition toolkit (Povey et al., 2011b). We used 39-dimensional MFCC feature vectors for the experiments. Each feature vector consisted of 13-dimensional static features with the zeroth cepstral coefficent and their delta and delta-delta components. Cepstral mean and variance normalization (CMN/CVN) was then applied on a per speaker basis. The GMM and SGMM systems shared the same decision tree to determine the tied state clustering used for context-dependent phone modelling; therefore, the differences in recognition accuracies of the GMM and SGMM systems are purely due to the different parameterisation of the GMMs. In the SGMM systems, we set the number of UBM Gaussians $I = 400$, and phonetic subspace dimension $S = 40$ for 15 hour training data case, whereas we use $S = 20$ when the training data is limited to 1 hour and 5 hours. Since the estimation of UBM model does not require the labels, we estimated it on the whole training dataset and use it for all German SGMM systems. Table 5.3 shows the word error rates (WERs) of baseline systems. As expected, the WERs for both the GMM and SGMM systems increase significantly as the amount of training data is reduced. The mono-

lingual SGMM system has a significantly lower WER than the monolingual GMM system for each of the three training conditions.

There is a large gap between WERs achieved on the development and evaluation dataset in Table 5.3. This is due to the language model that we used. In (Lu et al., 2011a) we used a trigram language model obtained with an earlier release of the GlobalPhone corpus, and achieved accuracies on the development dataset that were comparable to these on the evaluation dataset. Here, we interpolated that previously used language model with one estimated on the training corpus, and we obtained a significant reduction in WER on the development dataset (e.g. 24.0% in (Lu et al., 2011a) to 13.1% for SGMM system with 15 hour training data). But the improvements disappear on the evaluation dataset which indicates that the text in the training set matches the text of the development set better than that of the evaluation dataset. In the cross-lingual acoustic modelling presented in this chaper we observe similar trends on both the development and evaluation sets (as will be shown in Section 5.5.7), so the linguistic variation between training, development, and evaluation sets is not a confounding factor.

5.5.2 Cross-lingual system configuration

Each cross-lingual SGMM used the same context dependent tied state clustering as the corresponding monolingual SGMM trained on the same data set. Sharing global parameters between source languages, together with the constraints imposed by the structure of the SGMM, leads to better parameter estimates with limited amounts of training data. This also allows bigger models to be trained, either using more context-dependent tied states (Burget et al., 2010), or using a model with the same state clustering, but with more substates per state. We do the latter in this chaper. In both cases, the combination of improved parameter estimation and bigger models, is predicted to lead to lower WER.

The UBM was the same as the one that used to train the globally shared parameters $\mathbf{\Phi}_i$ on the source language(s). This is important, since the globally shared parameters correspond to the segmentation of the acoustic space as determined by the UBM (Povey et al., 2011a). First, we train $\mathbf{\Phi}_i$ for the source language systems in either a monolingual or a multilingual fashion. We then ported the shared parameters to the corresponding cross-lingual SGMM system. In the baseline SGMM systems, all the parameters in equations (3.1–3.3) were updated: the sub-state vectors $\mathbf{v}_{jm}$ and

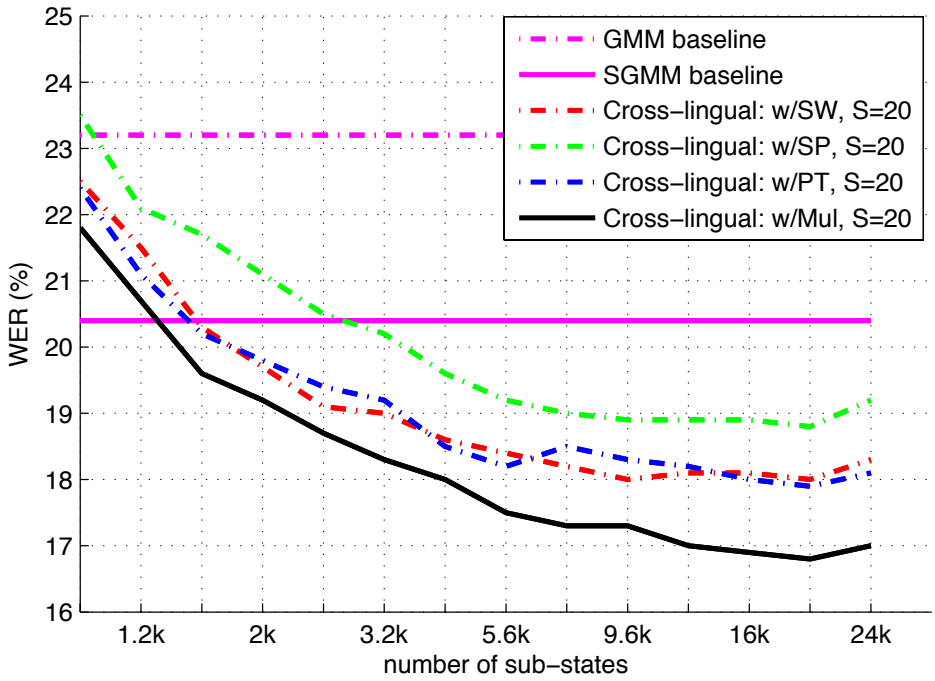


Figure 5.3: WER of baseline cross-lingual systems, 1h training data, tested on the development dataset. The "SGMM baseline" corresponds to the system with optimal number of sub-states using the monolingual setting.

the globally shared parameters $\mathbf{\Phi}_i$. In a cross-lingual system, however, only the sub-state vectors $\mathbf{v}_{jm}$ were re-estimated, with the globally shared parameters fixed unless stated otherwise. Table 5.4 shows the trace of covariance matrix and model subspace obtained by monolingual and multilingual systems.

5.5.3 Cross-lingual experiments: baseline

The baseline results of the cross-lingual systems are shown for 1h, 5h, and 15h training data (Figures 5.3–5.5). We contrast the shared parameters $\mathbf{\Phi}_i$ obtained from each of the source language systems, as well as the tied multilingual system. In these initial experiments, we do not use the speaker subspace $\boldsymbol{N}_i$. The dimension of sub-state vectors is set to be $S = 20$. With 1 hour training data, we achieved a relative WER reduction of up to 17% by reusing the globally shared parameters from source language systems trained in either a monolingual or multilingual fashion, demonstrating that out-domain knowledge can be used to improve significantly the accuracy of a target language system. In addition, we also observe that the system with multilingually trained subspace parameters "w/Mul" in Figure 5.3 results in considerably lower WERs compared with

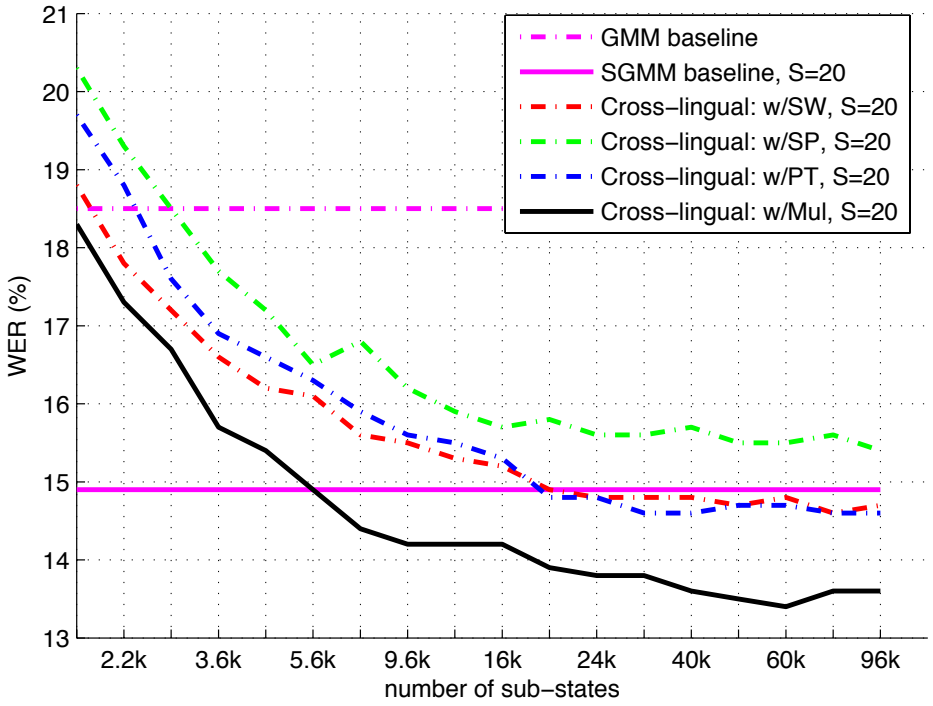


Figure 5.4: WER of baseline cross-lingual systems, 5h training data, tested on the development dataset. The "SGMM baseline" corresponds to the system with optimal number of sub-states using the monolingual setting.

the other cross-lingual systems derived from a single source language. This may because that there is much larger amount of training data in the multilingual system, and furthermore, the linguistic differences and corpus mismatch may be averaged out by the multilingual estimation which alleviate the mismatch between the multilingual parameters and target language system.

We observed the similar trend in the 5 hour training data case (Figure 5.4), although in this case the WER reduction is smaller (up to 10% relative) which is expected as the amount of training data increases. In order to evaluate if the cross-lingual frameworks can achieve improvement when the target training data is more abundant, we carried out the experiments using the entire 15 hour training data. Since we can drawn the conclusion from the previous experiments that the multilingual $\mathbf{\Phi}_i$ perform better than their monolingual counterparts, we only use the multilingual parameters for the cross-lingual setups. Results are shown in Figure 5.5 where the dimensions of sub-state vectors were set to be $S = 40$. In this case, the cross-lingual SGMM system still reduces the WER by 8% relative (1% absolute).

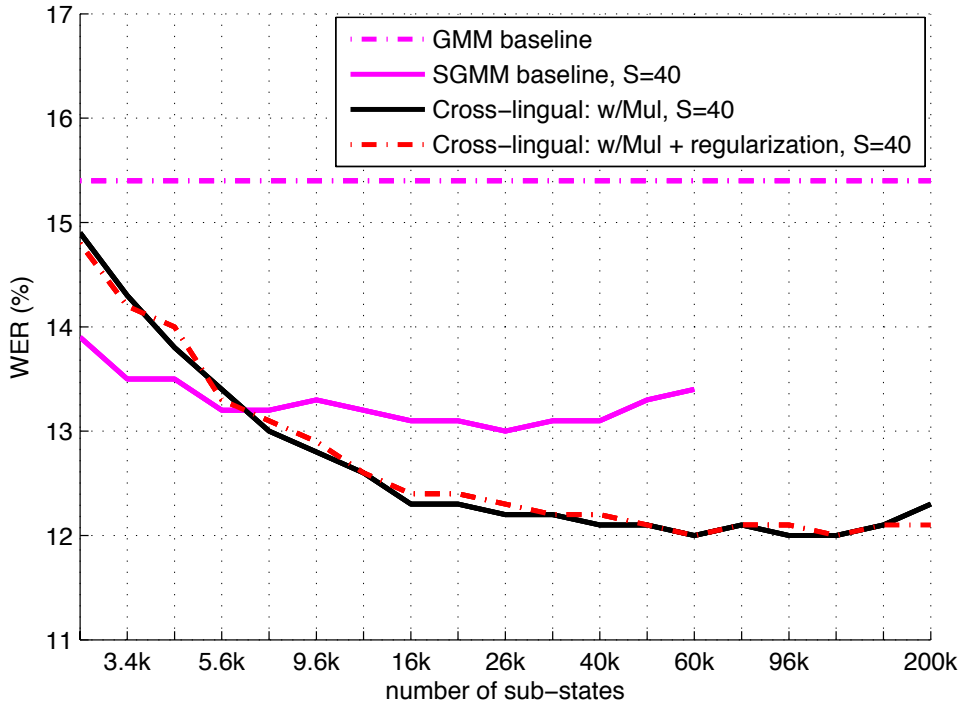


Figure 5.5: WER of baseline cross-lingual systems, 15h training data, tested on the development dataset.

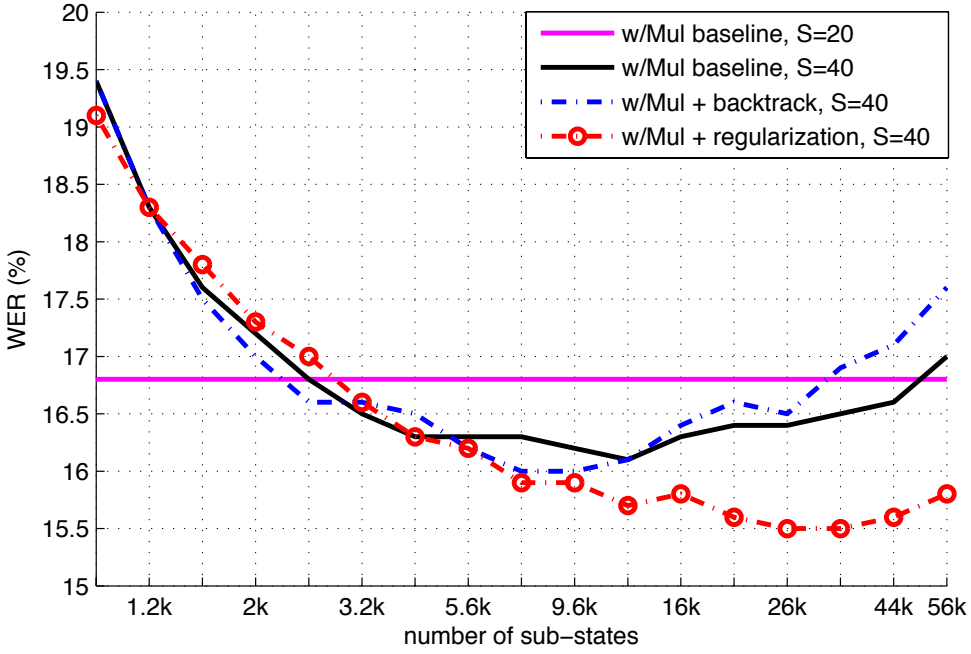


Figure 5.6: WER of regularized cross-lingual systems, 1h training data, tested on the development dataset.

5.5.4 Cross-lingual experiments: with regularization

With limited amounts of training data, it is often necessary to limit the dimensionality of the state vectors $\mathbf{v}_{jk}$, since increasing the phonetic subspace dimension S increases the number of both global and state-specific parameters. When the global parameters $\mathbf{\Phi}_i$ are trained on separate data, state vectors of larger dimensionality may be used. Comparing figures 5.3 and 5.6, we see that for the cross-lingual system trained on 1 hour of speech using a phonetic subspace dimension of $S = 40$ lowers the WER compared to a subspace of dimension $S = 20$[1].

Figure 5.6 also compares the standard ML update with a more conservative one that "backtracks" to the previous parameter values if the auxiliary function decreases due to the update. Models trained using both these criteria are found to have larger WER when the number of substates is increased, showing that the models tend to overtrain when using very small amounts of training data. However, when the state vectors are estimated with the ℓ_1-norm regularization, the updates are more stable and allow models with larger number of substates to be trained leading to lower WER overall. In fact, the WER of 15.5% achieved by the cross-lingual SGMM trained on 1 hour of speech using ℓ_1-norm regularization is comparable to the GMM baseline with the entire 15 hour training data.

Figure 5.7 shows the results with 5 hour training data. Not surprisingly, the difference between the regularized model and the one without regularization is smaller than that seen when training on 1 hour of data. However, when the number of sub-states is very large, regularization still helps to avoid model overfitting and results in a small gain in terms of accuracy. Again, the more conservative update with backtracking did not work better than the regularized update. After increasing the amount of training data to be 15 hours, we did not obtain improvement by applying the ℓ_1-norm regularization as shown in Figure 5.5. This is in agreement with our previous experience of using ℓ_1-norm regulariation for SGMMs (Lu et al., 2011b) on a different task.

5.5.5 Cross-lingual experiments: with MAP adaptation

As discussed above, if $\mathbf{\Phi}_i$ is estimated from out-domain data, then there may be a mismatch between the target language system and these parameters. One approach to address this mismatch is via MAP adaptation of $\mathbf{\Phi}_i$. We applied MAP adaptation

[1] In (Lu et al., 2011a), we used a preliminary version of Kaldi toolkit that was used in (Povey et al., 2011a) and faced numerical instability when building the baseline system without regularization. We did not have that experience using a more recent version of Kaldi (revision 710).

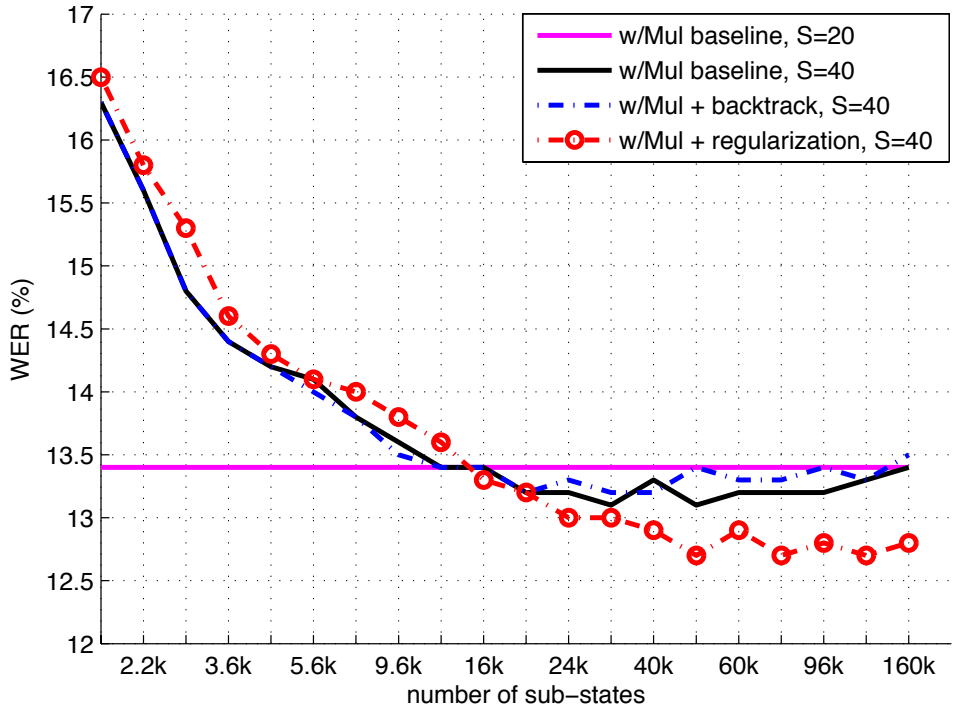


Figure 5.7: WER of regularized cross-lingual systems, 5h training data, tested on the development dataset.

of $\mathbf{M}_i$ to the systems "w/Mul baseline, S=40" and "w/Mul+regularization, S=40" to the 1h and 5h training data conditions (Figures 5.8 and 5.9). As stated in section 5.3, the two covariance matrices $\mathbf{\Omega}_r$ and $\mathbf{\Omega}_c$ are set to be the identity matrix $\mathbf{I}$. For the 1h training data case, we set smoothing parameter used in equation (5.2) $\tau = 500$. By using MAP adaptation, we obtained a small reduction in WER (2% relative) compared to the regularized system. The improvement is not comparable to our previous results (Lu et al., 2012c) since the baseline is much stronger here. When we applied MAP adaptation to the baseline without regularization, we did not observe a reduction in WER when the number of sub-states was large. This may be because the sub-state vectors $\mathbf{v}_{jk}$ are not well estimated due to overfitting and hence we do not have sufficient and accurate statistics for equation (3.47) to adapt $\mathbf{M}_i$.

In the 5h training data case (Figure 5.9), we did not observe any reduction in WER for either the baseline of regularized systems, even though the amount of adaptation data was increased. When applying MAP adaptation, the likelihood on the training data increased, but the higher WER suggests that it overfits to the training data. We also increased the smoothing term τ to much larger value but it only pushed the adapted

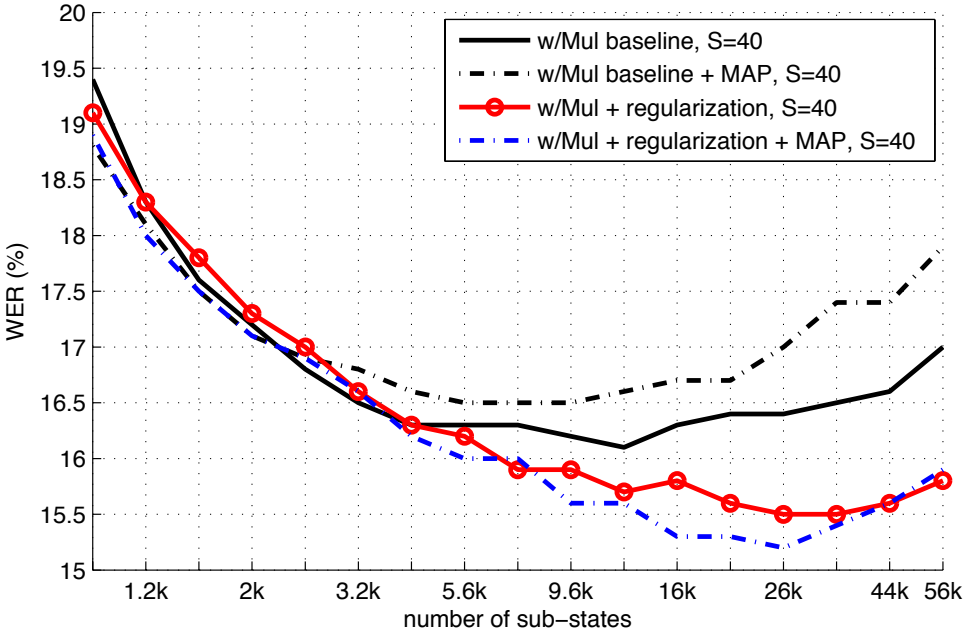


Figure 5.8: WER of MAP-adapted cross-lingual systems, 1h training data, tested on the development dataset.

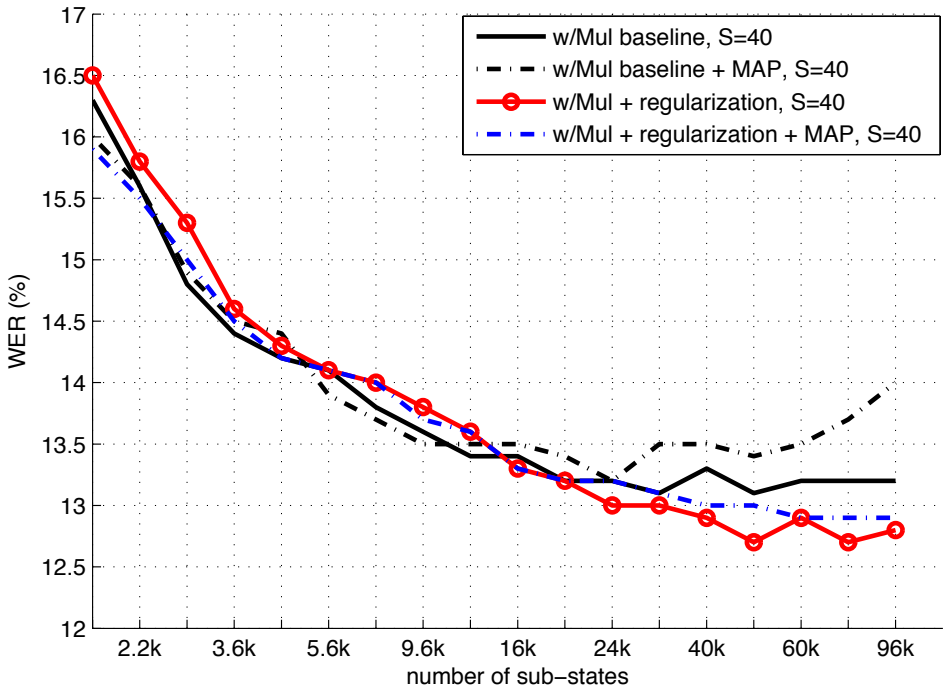


Figure 5.9: WER of MAP-adapted cross-lingual systems, 5h training data, tested on the development dataset.

system closer to the baseline while no gain was observed. This may further demonstrate that the multilingual parameters are more robust and match the target training data well. Again we did not achieve gains by using MAP adaptation of $\mathbf{M}_i$ in the 15h training data case.

For the 15h training data case, we investigated the update of globally shared parameters $\mathbf{\Phi}_i$. We updated $\mathbf{w}_i$ and $\mathbf{M}_i$ to maximize the likelihood for the cross-lingual system. While this resulted in lower WER for models with less number of sub-states, the results were worse for larger models (Figure 5.10). These results are not entirely unexpected. The multilingual estimation of $\mathbf{w}_i$ and $\mathbf{M}_i$ is expected to be more accurate and robust than monolingual estimate. It is worth noting that while updating $\mathbf{M}_i$ and $\mathbf{w}_i$ makes the results worse than keeping them fixed at the multilingually estimated values, the results are comparable to the monolingual system in Figure 5.5, and in some cases slightly better than those. This shows that starting the iterative ML updates of the subspace parameters from a better starting point, that is the multilingually trained parameters, makes no substantial difference eventually. We also carried out the experiments where $\mathbf{\Sigma}_i$ were updated but those showed similar trends as the ML updates of $\mathbf{M}_i$ and $\mathbf{w}_i$.

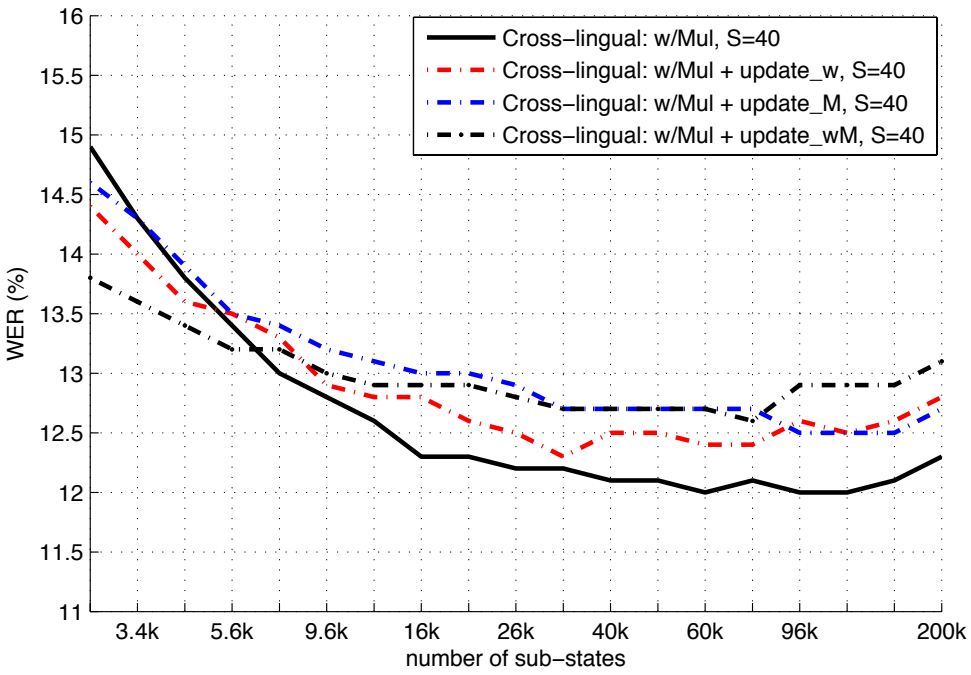


Figure 5.10: WER of cross-lingual systems with global parameter update, 15h training data, tested on the development dataset.

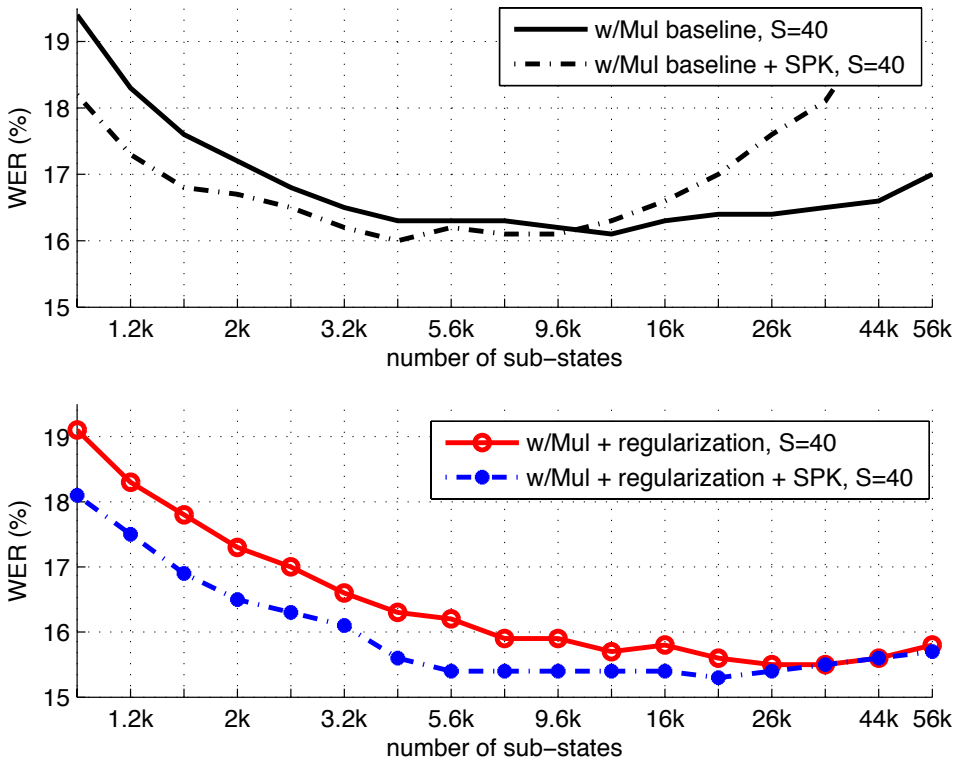


Figure 5.11: WER of baseline (above) and regularized (below) cross-lingual systems using speaker subspace, 1h training data, tested on the development dataset.

5.5.6 Cross-lingual experiments: with speaker subspace

Our final set of experiments concerned speaker adaptive training using the speaker subspace for cross-lingual SGMM systems for the 1h, 5h, and 15h training data cases (Figures 5.11–5.13). In the 1h training data case, there are only 8 speakers in the training set, which is not sufficient to train the speaker subspace $\mathbf{N}_i$ on per speaker basis for our baseline SGMM system. We trained $\mathbf{N}_i$ on per utterance basis for the baseline but did not observe an improvement. However, we can estimate $\mathbf{N}_i$ in multilingual fashion by tying it across the source language system as other types of globally shared parameters. We then rebuilt the target system "w/Mul + regularization, S=40" by using the resultant speaker subspace. Results are given in Figure 5.11. Here the dimension of speaker vector was set to be $T = 39$. We can see that for the regularized system, using the multilingual $\mathbf{N}_i$ results in significant gains when the number of sub-states is relative small. The gains, however, vanish as we further increased the number of sub-states. For the system without regularization, it is more prone to overtraining with speaker subspace adaptive training.

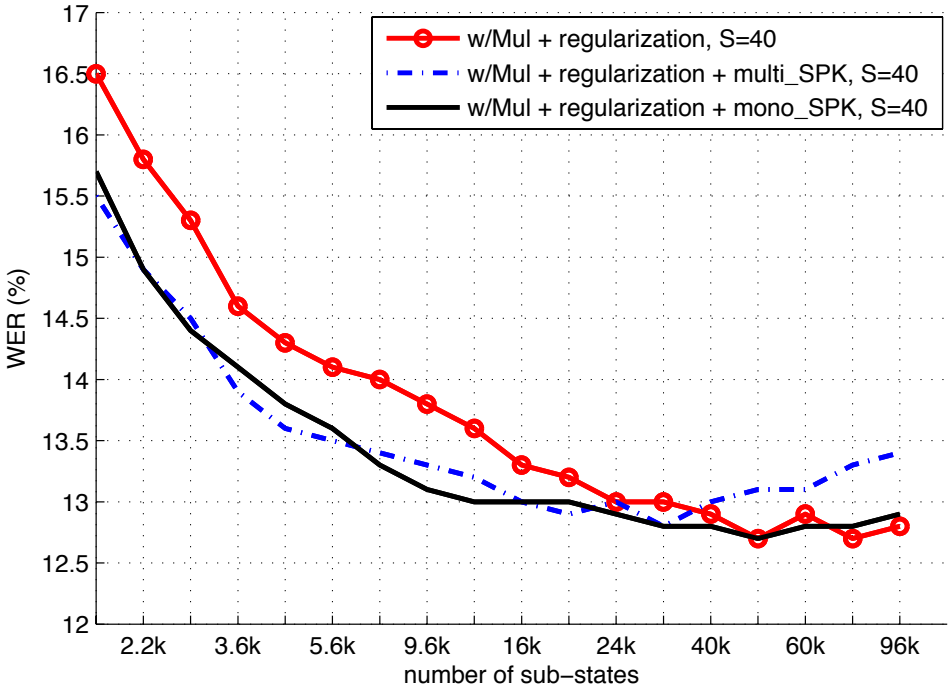


Figure 5.12: WER of regularized cross-lingual systems using speaker subspace, 5h training data, tested on the development dataset.

In the 5h training data case, there are 40 speakers in the training set, enough to estimate $\mathbf{N}_i$ from the in-domain data. This system is referred as "w/Mul + regularization + mono_SPK, S=40" in Figure 5.12. For the system using the multilingual speaker subspace $\mathbf{N}_i$, we refer it as "w/Mul + regularization + multi_SPK, S=40". In both systems, $T = 39$. We can see that both systems achieve large reductions in WER when the number of sub-states is small — again, the gains vanish when using large number of sub-states. In addition, the multilingual speaker subspace $\mathbf{N}_i$ achieves a similar WER to the monolingual one. This indicates that the speaker information from the out-domain data can fit the target system well.

We did not observe notable WER differences when using either monolingual or multilingual speaker subspace in the 15h training data case (Figure 5.13), as for the 5h training data case. Just as with 1 hour and 5 hours of training data, using the speaker subspace lowers the WER for smaller model sizes, but the difference between the adaptively trained and unadapted models vanish when using a very large number of substates. Although the speaker adaptive training does not provide a overall reduction in WER, it provides a practical advantage: it is computationally cheaper to use a smaller model with speaker subspace than a larger model without it. In the future, we

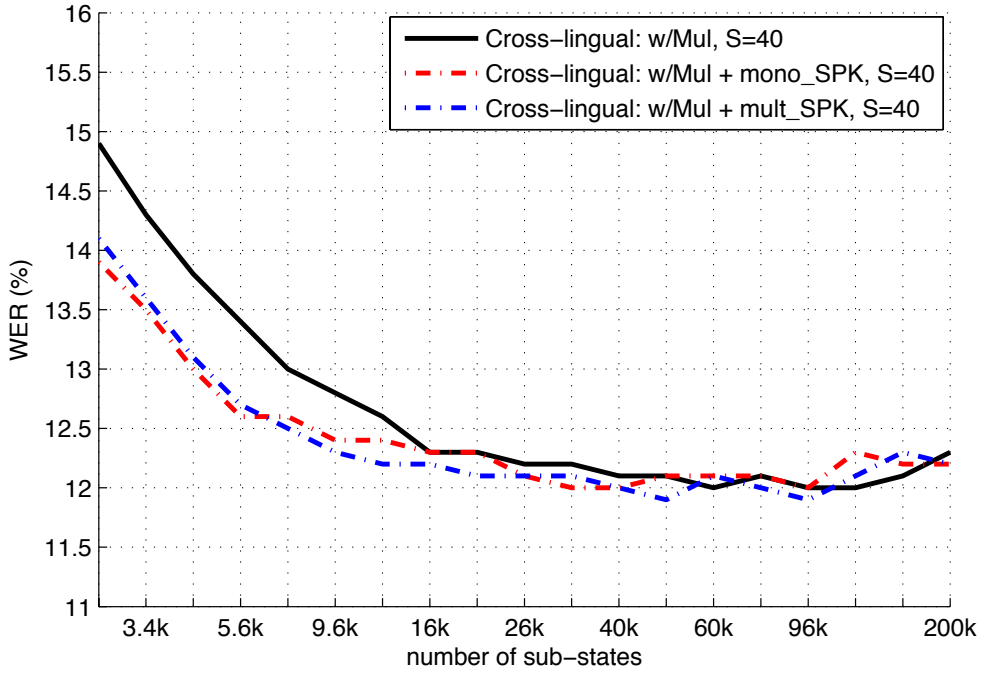


Figure 5.13: WER of cross-lingual systems using speaker subspace, 15h training data, tested on the development dataset.

plan to investigate using feature space (constrained) MLLR for cross-lingual speaker adaptive training as a comparison to the results using the speaker subspace.

5.5.7 Cross-lingual experiments: summary

Table 5.5 summarizes the results on the development and evaluation datasets with 1h training data. We observed a similar trend of results on both datasets. The lowest WER on the evaluation set (26.7%) was achieved by using multilingual parameter estimation with regularization, followed by speaker subspace adaptive training. This is significantly better than the GMM and SGMM baseline using the same training data (34.1% and 31.4%) and it is only 2% worse than the GMM baseline using the entire 15h training dataset (24.8%). Hence, by leveraging on the out-domain data, the cross-lingual SGMM system can mitigate the performance loss due to the limitation of the training data.

Table 5.6 summarizes the WERs of systems with 5h training data on both the development and evaluation datasets. Using multilingual parameter estimation and ℓ_1-norm regularization, the cross-lingual system obtains 12.7% on the development dataset and 22.1% on the evaluation dataset, a reduction of about 2% absolute compared to the

speaker adaptively trained SGMM baseline using monolingual subspace.

A summary of the results using the entire 15h training data is given in Table 5.7. In this condition, the cross-lingual system outperformed the baseline with speaker subspace adaptive training by 0.4% absolute on the development dataset and they achieved around the same accuracy on the evaluation dataset.

Table 5.5: Results of Cross-lingual SGMM systems with 1 hour training data on the development (Dev) and evaluation dataset (Eval).

System	Dev	Eval
GMM baseline	23.2	34.1
SGMM baseline	20.4	31.4
X-SGMM w/SP, $S = 20$	18.8	32.4
X-SGMM w/PO, $S = 20$	17.9	30.9
X-SGMM w/SW, $S = 20$	18.0	31.0
X-SGMM w/Mul, $S = 20$	16.8	29.3
X-SGMM w/Mul + ℓ_1, $S = 40$	15.5	26.9
+speaker subspace	15.3	26.7

Table 5.6: Results of Cross-lingual SGMM systems with 5 hour training data on the development (Dev) and evaluation dataset (Eval).

System	Dev	Eval
GMM baseline	18.5	28.0
SGMM baseline	14.9	24.9
+speaker subspace	14.6	24.7
X-SGMM w/SP, $S = 20$	15.4	26.5
X-SGMM w/PO, $S = 20$	14.6	25.2
X-SGMM w/SW, $S = 20$	14.6	25.4
X-SGMM w/Mul, $S = 20$	13.4	24.5
X-SGMM w/Mul + ℓ_1, $S = 40$	12.7	22.1

Table 5.7: Results of Cross-lingual SGMM systems with 15 hour training data for development (Dev) and evaluation dataset (Eval).

System	Dev	Eval
GMM baseline	15.4	24.8
SGMM baseline	13.0	22.1
+speaker subspace	12.4	21.5
X-SGMM w/Mul + ℓ_1, $S = 40$	12.0	21.6

5.6 Conclusions

In this chapter, we have studied cross-lingual speech recognition using SGMM acoustic models in low-resource conditions. We first present a systematic review of the techniques used to build the cross-lingual SGMM system. We then carried out a set of experiments using the GlobalPhone corpus with three source languages (Portuguese, Spanish, and Swedish), using German as the target language. Our results indicate that the globally shared parameters in the SGMM acoustic model can be borrowed from the source language system. This leads to large reductions in WER when the amount of target language training data is limited (e.g. 1 hour). In addition, estimating the globally shared parameters using multilingual training data is particularly beneficial. We observed that the cross-lingual system using the multilingual parameters outperforms other cross-lingual systems using the monolingual parameters.

Our results also demonstrate the effectiveness of regularization using an ℓ_1-norm penalty for the state vectors. With a limited amount of training data, regularization is able to improve the numerical stability of the system, enabling the use of a model subspace of higher dimension and with more sub-state vectors. The benefits were demonstrated by experimental results using 1 hour and 5 hour training data in our study, in which substantial reductions in WER were obtained by using a higher dimensional model subspace together with regularization.

We also investigated the MAP adaptation of the model subspace, and cross-lingual speaker adaptive training using a speaker subspace. In both cases, our findings indicated that the resulting reductions in WER can also be achieved by multilingual parameter estimation and regularization. In addition, we compared the speaker adaptive training using monolingual and multilingual speaker subspace and obtained comparable recognition accuracy in 5 hour and 15 hour training data conditions. This indicates that the speaker subspace may also be portable across languages. Although our work

has focused on speech recognition, we view this approach to cross-lingual modelling and factorization as potentially useful across speech technology: such speaker and language factorization has recently been studied — and proven beneficial — for the task of speech synthesis (Zen et al., 2012).

Chapter 6

Noise compensation for Subspace Gaussian Mixture Model

6.1 Introduction

Speech recognition accuracy is significantly degraded in the noisy environments that are characteristic of many real world applications. There is an extensive literature on methods to compensate for the mismatch between the speech recognition model and noise-corrupted data (Droppo and Acero, 2008). There are two broad categories of techniques for noise robust speech recognition, compensation in the feature domain and compensation in the model domain. In the feature domain, approaches referred to as feature enhancement or de-noising aim to estimate the unobserved clean speech features given the observed noisy features. Many feature domain approaches have been proposed including spectral subtraction, cepstral mean and variance normalization (CMN/CVN), cepstral maximum mean square error (MMSE) estimation (Yu et al., 2008), SPLICE (Deng et al., 2000), Algonquin (Frey et al., 2001) and feature space vector Taylor series (VTS) compensation (Moreno et al., 1996). Conventional feature domain methods use a point estimate of the hidden clean speech features, which is used as an observation vector for a speech recognition system. A number of approaches have moved beyond point estimation of clean speech features and have considered the observation uncertainties (Arrowood and Clements, 2002; Droppo et al., 2002; Deng et al., 2005; Liao and Gales, 2005). Such approaches have been shown to be more effective at improving recognition accuracy given the noisy observations.

In contrast, model domain techniques adapt the model parameters in order to better explain the noisy observations. Purely data-driven model domain techniques include

approaches in the maximum likelihood linear regression (MLLR) family (Woodland et al., 1996), such as noisy constrained MLLR (NCMLLR) (Kim and Gales, 2011). These approaches are not affected by the parameterisation of the acoustic features, since they use a generic compensation scheme—typically an affine transform—instead of an explicit model of the distortion caused by the noise. Hence, they may be combined with other feature-space compensation techniques. However, their performance is normally limited by the sparsity of adaptation data. Knowledge-based model domain approaches can overcome this limitation by estimating a mismatch function between the clean and noise-corrupted speech features in order to estimate the compensation parameters (Acero, 1990). Examples of such techniques include model space VTS and joint uncertainty decoding (JUD) (Liao, 2007), parallel model combination (PMC) (Gales, 1995) and a linear spline interpolation model (Kalgaonkar et al., 2009). These approaches can achieve good results without requiring a large amount of adaptation data, but are limited to only spectral or cepstral features, and combination with other feature space techniques is challenging.

In this chapter, we present a model-based noise compensation scheme for subspace Gaussian mixture models (SGMMs) (Povey et al., 2011a). As we have discussed in the previous chapters, in an SGMM the parameters of each Gaussian component are derived from a low dimensional model subspace. This allows a much larger number of surface Gaussians to be used by each HMM state while the total parameters to be estimated is typically smaller compared to conventional HMM/GMM acoustic models. Recent research has shown that an SGMM acoustic model is more accurate than its GMM counterpart in both monolingual and multilingual settings (Povey et al., 2011a,c; Lu et al., 2011b; Burget et al., 2010; Lu et al., 2011a). However, uncompensated SGMMs suffer similar problems to GMMs in noise mismatched conditions.

There are many more component Gaussians in a typical SGMM compared with a conventional GMM. Model-based compensation schemes developed for conventional GMMs which explicitly compensate the parameters of each component Gaussian, such as standard VTS compensation, will be computationally expensive if applied directly to SGMMs. Direct compensation of the surface Gaussians in an SGMM is also inelegant, since it does not take account of the structure of the model. JUD can address this problem, since the entire set of Gaussian components in the model is clustered into a small number of classes, typically using a regression tree (Gales, 1996). The mapping between a clean speech model and a noise-corrupted speech model is assumed to be common to all the Gaussians belonging to the same regression class. Moreover,

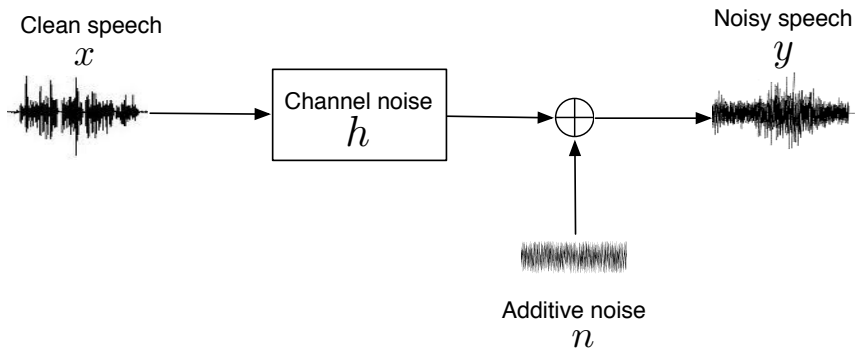


Figure 6.1: The relationship between clean speech $\mathbf{x}$, additive and channel noise $(\mathbf{n}, \mathbf{h})$ and noise corrupted speech $\mathbf{y}$.

JUD compensates the model using a feature space transformation (together with a bias term for the covariances), which is compatible with the compact model structure of an SGMM.

In this chapter, we develop model domain noise compensation for SGMMs, and report on a number of experiments using the Aurora 4 corpus. These experiments indicate that, by using a smaller regression model, the computational cost is relatively low while the accuracy is significantly improved in noise mismatched conditions. In addition, the SGMM system is more accurate than similar GMM systems using both VTS and JUD noise compensation.

6.2 Mismatch function

In discrete time domain, the relationship between noise-corrupted speech $y(t)$, clean speech $x(t)$, additive noise $n(t)$ and the channel's impulse response $h(t)$ can be formulated as

$$y(t) = x(t) * h(t) + n(t). \tag{6.1}$$

where t is the time frame index. This is shown in Figure 6.1. Applying the discrete Fourier transform (DFT) to both sides, the equivalent relationship in the frequency domain, for the k-th frequency bin of the Mel-scale warped filterbank, is

$$y_{k,t} = x_{k,t} h_{k,t} + n_{k,t} \tag{6.2}$$

$$\approx x_{k,t} h_k + n_{k,t}. \tag{6.3}$$

The channel distortion is assumed to be time invariant[1], so the subscript t may be dropped from $h_{k,t}$. The power spectrum of the noisy speech can then be obtained as

$$\begin{aligned}|y_{k,t}|^2 &\approx |x_{k,t}h_k + n_{k,t}|^2 \\ &= |x_{k,t}|^2|h_k|^2 + |n_{k,t}|^2 + 2|x_{k,t}||h_k||n_{k,t}|\cos\theta_{kt}\end{aligned} \tag{6.4}$$

where θ_{kt} denotes the (random) angle between the two complex variables $(x_{k,t}h_k)$ and $n_{k,t}$. By taking logarithm and multiplying by the truncated discrete cosine transform (DCT) matrix $\mathbf{C}$ on both sides, the distortion function in cepstral domain can be expressed as

$$\begin{aligned}\mathbf{y}_{s,t} &= f(\mathbf{x}_{s,t}, \mathbf{h}, \mathbf{n}_t, \boldsymbol{\alpha}_t) \\ &= \mathbf{x}_{s,t} + \mathbf{h} + \mathbf{C}\log\Big[\mathbf{1} + \exp\left(\mathbf{C}^{-1}\left(\mathbf{n}_t - \mathbf{x}_{s,t} - \mathbf{h}\right)\right) \\ &\quad + 2\boldsymbol{\alpha}_t \bullet \exp\left(\mathbf{C}^{-1}(\mathbf{n}_t - \mathbf{x}_{s,t} - \mathbf{h})/2\right)\Big],\end{aligned} \tag{6.5}$$

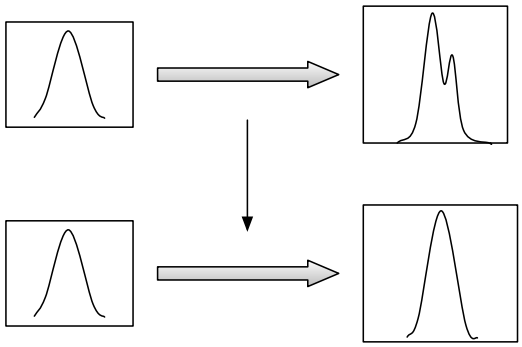
where $\mathbf{x}_{s,t}, \mathbf{y}_{s,t}, \mathbf{n}_t$, and $\mathbf{h}$ are the vector-valued clean and noise-corrupted speech, additive noise, and channel noise, respectively, at time frame t; $\mathbf{C}^{-1}$ is the pseudoinverse of the truncated DCT matrix $\mathbf{C}$ and $\mathbf{1}$ is a vector with each element set to 1; $\log(\cdot)$, $\exp(\cdot)$, and $\bullet$ denote the element-wise logarithm, exponentiation, and multiplication, respectively. $\boldsymbol{\alpha}_t$ is a random variable that may be interpreted as a factor making the mismatch function sensitive to the phase between the clean speech and noise (Deng et al., 2004; Li et al., 2009). The interpretation of $\boldsymbol{\alpha}_t$ as a phase factor suggests that the possible range of values for each dimension of $\boldsymbol{\alpha}_t$ is $[-1.0, 1.0]$ (Deng et al., 2004). In many noise compensation applications, $\boldsymbol{\alpha}_t$ is often assumed to be zero. We introduce the subscript s to the speech variable to indicate it only relates to the static feature. This is not done to the noise variable $\mathbf{n}_t, \mathbf{h}$ as they are always relates to the static feature.

Following (Li et al., 2009), we make a simplifying assumption that the value of $\boldsymbol{\alpha}_t$ does not depend on t, and rewrite the mismatch function as

$$\begin{aligned}\mathbf{y}_{s,t} &= f(\mathbf{x}_{s,t}, \mathbf{h}, \mathbf{n}_t, \boldsymbol{\alpha}) \\ &= \mathbf{x}_{s,t} + \mathbf{h} + \mathbf{C}\log\Big[1 + \exp\left(\mathbf{C}^{-1}\left(\mathbf{n}_t - \mathbf{x}_{s,t} - \mathbf{h}\right)\right) \\ &\quad + 2\boldsymbol{\alpha} \bullet \exp\left(\mathbf{C}^{-1}(\mathbf{n}_t - \mathbf{x}_{s,t} - \mathbf{h})/2\right)\Big].\end{aligned} \tag{6.6}$$

Note that $\boldsymbol{\alpha} = \mathbf{0}$ corresponds to compensation in power domain, while $\boldsymbol{\alpha} = \mathbf{1}$ corresponds to magnitude domain compensation (Gales and Flego, 2010). In subsequent

[1]This is a safe assumption since the noise compensation is applied on a per-utterance basis.



Gaussian approximation

Figure 6.2: After the corruption of noise, the distribution of noisy speech may not be Gaussian even though the original clean speech is Gaussian distributed, but we still use Gaussian approximation for GMM- or SGMM-based recognisers.

sections, we drop the subscript t from the vectors $\mathbf{x}_{s,t}, \mathbf{y}_{s,t}$ and $\mathbf{n}_t$, in order to simplify the notation, wherever the dependence on the time index is obvious. Note that while the mismatch function is valid for Mel cepstra (static features), it is customary to append the first and second order difference vectors (delta and acceleration features) to obtain the complete observation vector. These dynamic coefficients are derived using a continuous-time approximation (Gopinath et al., 1995). For example, the delta coefficients are given by:

$$\begin{aligned}\boldsymbol{\Delta}\mathbf{y}_t \approx \frac{\partial \mathbf{y}_s}{\partial t}|_t &= \frac{\partial \mathbf{y}_s}{\partial \mathbf{x}_s}\frac{\partial \mathbf{x}_s}{\partial t}|_t + \frac{\partial \mathbf{y}_s}{\partial \mathbf{n}}\frac{\partial \mathbf{n}}{\partial t}|_t \\ &\approx \frac{\partial \mathbf{y}_s}{\partial \mathbf{x}_s}\boldsymbol{\Delta}\mathbf{x}_t + \frac{\partial \mathbf{y}_s}{\partial \mathbf{n}}\boldsymbol{\Delta}\mathbf{n}_t. \end{aligned} \tag{6.7}$$

and the acceleration coefficients, $\boldsymbol{\Delta}^2\mathbf{y}_t$, are derived similarly. In model-based compensation, the compensated dynamic mean and covariance parameters are obtained by taking the expectation (cf. Section 6.3.1).

In most noise compensation schemes, the additive noise is assumed to be Gaussian distributed, while the constant channel noise is represented by its "mean" for notational symmetry:

$$\mathbf{n}_t \sim \mathcal{N}(\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n), \qquad \mathbf{h} = \boldsymbol{\mu}_h. \tag{6.8}$$

The clean speech $\mathbf{x}_t$ is normally assumed to be Gaussian distributed, and the noise-corrupted speech $\mathbf{y}_t$ is still approximated by a Gaussian distribution in order fit the recognizers, although its "true" distribution may be very complex. This is illustrated

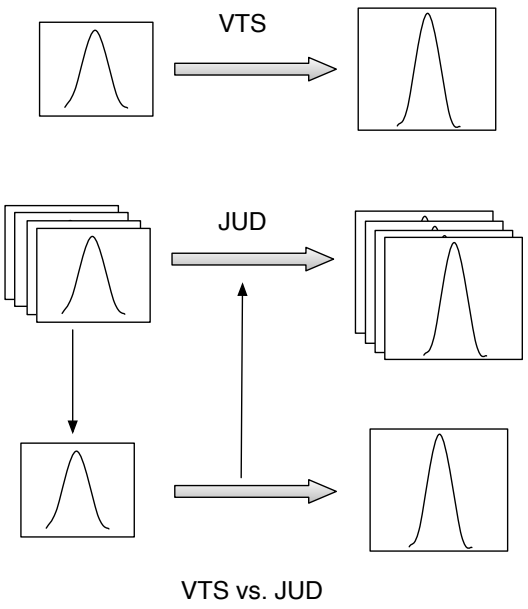


Figure 6.3: A comparison of VTS and JUD noise compensation. VTS is performed on per component basis, while for JUD, a cluster of components share the same compensation parameters.

by Figure 6.2. Under this Gaussianity assumption, the aim of noise compensation is to estimate the mean $\boldsymbol{\mu}_y$ and covariance $\boldsymbol{\Sigma}_y$ of the noise-corrupted speech. However, as the mismatch function (6.6) is highly nonlinear, no closed-form solution is available. A solution may obtained either by using sampling techniques, such as data-driven parallel model combination (DPMC) (Gales, 1995), unscented transform (UT) (e.g. (Julier and Uhlmann, 2004; Hu and Huo, 2006; Xu and Chin, 2009a; Li et al., 2010)), or by using a polynomial approximation such as vector Taylor series (VTS) (Moreno et al., 1996). Sampling techniques draw samples from a noise model and a clean speech model to synthesise the corresponding noisy speech samples using the mismatch function (6.6). They can achieve very good results with a sufficiently large number of samples, but this comes at a higher computational cost, thus limiting their applicability.

VTS approximates the nonlinear mismatch function by a truncated vector Taylor series expansion, by which a closed-form solution can be obtained for the noisy speech model. First order VTS is typically used, although recent results show that improvements can be obtained by a second or higher order VTS expansion (Xu and Chin, 2009b; Du and Huo, 2011). Compared to sampling, VTS compensation is relatively effective and efficient. However, since the parameters for each Gaussian component in the acoustic model are individually compensated in this approach, it is still

computationally demanding, especially when the number of Gaussians is large. Joint uncertainty decoding (JUD) (Liao and Gales, 2005) provides a more efficient way of performing noise compensation, by clustering the Gaussian components into a relatively small number of classes, and sharing the compensation parameters among the Gaussians in each class. This significantly reduces the computational cost without a large sacrifice in accuracy. Figure 6.3 presents the general idea while the details are given in the next section.

6.3 Joint uncertainty decoding

In the framework of joint uncertainty decoding (JUD) (Liao and Gales, 2005), the relationship between the observed noisy speech $\mathbf{y}$, the underlying clean speech vector $\mathbf{x}$, and Gaussian component m can be expressed as

$$p(\mathbf{y}|m) = \int p(\mathbf{x},\mathbf{y}|m)d\mathbf{x} = \int p(\mathbf{y}|\mathbf{x},m)p(\mathbf{x}|m)d\mathbf{x} \tag{6.9}$$

where the conditional distribution $p(\mathbf{y}|\mathbf{x},m)$ models the effect of noise on clean speech for Gaussian component m. If the dependency on m is removed from the conditional distribution, that is:

$$p(\mathbf{y}|\mathbf{x},m) \approx p(\mathbf{y}|\mathbf{x}), \tag{6.10}$$

then it results in a simplified uncertainty decoding rule, used for many feature domain approaches, such as SPLICE with uncertainty(Droppo et al., 2002).

Although each of the Gaussians in the model could be compensated using (6.9), such an approach is not computationally feasible in practice. Instead, the Gaussians are grouped into a relatively small number of classes based on their acoustic similarities. One way of clustering the Gaussians is to use a regression tree (Gales, 1996), first proposed in the context of speaker adaptation. Equation (6.9) is approximated by replacing Gaussian component m with its regression class r_m:

$$p(\mathbf{y}|m) \approx \int p(\mathbf{y}|\mathbf{x},r_m)p(\mathbf{x}|m)d\mathbf{x} \tag{6.11}$$

The conditional distribution $p(\mathbf{y}|\mathbf{x},r_m)$ is derived from the joint distribution of clean and noise-corrupted speech which is assumed to be Gaussian. For r^{th} regression class

$$p\left(\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix} \mid r\right) := \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\mu}_x^{(r)}\\ \boldsymbol{\mu}_y^{(r)}\end{bmatrix}, \begin{bmatrix}\boldsymbol{\Sigma}_x^{(r)} & \boldsymbol{\Sigma}_{xy}^{(r)}\\ \boldsymbol{\Sigma}_{yx}^{(r)} & \boldsymbol{\Sigma}_y^{(r)}\end{bmatrix}\right), \tag{6.12}$$

which gives conditional distribution $p(\mathbf{y}|\mathbf{x},r)$, with parameters:

$$\boldsymbol{\mu}_{y|x}^{(r)} = \boldsymbol{\mu}_y^{(r)} + \boldsymbol{\Sigma}_{yx}^{(r)} \boldsymbol{\Sigma}_x^{(r)^{-1}} \left(\mathbf{x} - \boldsymbol{\mu}_x^{(r)}\right) \tag{6.13}$$

$$\boldsymbol{\Sigma}_{y|x}^{(r)} = \boldsymbol{\Sigma}_y^{(r)} - \boldsymbol{\Sigma}_{yx}^{(r)} \boldsymbol{\Sigma}_x^{(r)^{-1}} \boldsymbol{\Sigma}_{xy}^{(r)} \tag{6.14}$$

By marginalizing out the clean speech distribution $p(\mathbf{x}|m)$ using (6.11), the likelihood of corrupted speech for m^{th} component may be expressed as a Gaussian with transformed features and an additive covariance bias:

$$p(\mathbf{y}|m) \approx |\mathbf{A}^{(r_m)}| \mathcal{N}\left(\mathbf{A}^{(r_m)}\mathbf{y} + \mathbf{b}^{(r_m)}; \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m + \boldsymbol{\Sigma}_b^{(r_m)}\right), \tag{6.15}$$

where the JUD transform parameters are obtained as:

$$\mathbf{A}^{(r)} = \boldsymbol{\Sigma}_x^{(r)} \boldsymbol{\Sigma}_{yx}^{(r)^{-1}}, \tag{6.16}$$

$$\mathbf{b}^{(r)} = \boldsymbol{\mu}_x^{(r)} - \mathbf{A}^{(r)} \boldsymbol{\mu}_y^{(r)}, \tag{6.17}$$

$$\boldsymbol{\Sigma}_b^{(r)} = \mathbf{A}^{(r)} \boldsymbol{\Sigma}_y^{(r)} \mathbf{A}^{(r)^T} - \boldsymbol{\Sigma}_c^{(r)}. \tag{6.18}$$

The clean speech parameters, $\boldsymbol{\mu}_x^{(r)}$ and $\boldsymbol{\Sigma}_x^{(r)}$, may be derived from the clean speech model using a regression tree. The corresponding parameters for noise-corrupted speech, $\boldsymbol{\mu}_y^{(r)}$, $\boldsymbol{\Sigma}_y^{(r)}$, and the cross covariance $\boldsymbol{\Sigma}_{yx}^{(r)}$, are obtained from the mismatch function (6.6). In practise, only the parameters for the static cepstral coefficients, $\boldsymbol{\mu}_{y_s}^{(r)}$, $\boldsymbol{\Sigma}_{y_s}^{(r)}$, and $\boldsymbol{\Sigma}_{y_s x_s}^{(r)}$, are computed using (6.6), given an estimate of the noise parameters $\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n$, and $\boldsymbol{\mu}_h$. Certainly, the true noise parameters are unknown and they need to be estimated jointly with the transform parameters. Details of noise model estimation are provided in Section 6.3.3.

The means and covariances of the dynamic coefficients are computed using the continuous time approximation (6.7), as described in the following section. Cross-correlations between the static and dynamic components are assumed to be zero, which leads to a block-diagonal structure for the matrices appearing in equations (6.16)–(6.18). As an alternative to the continuous time approximation for dynamic features, the static coefficients $\mathbf{y}_{s,t}$ may be extended by appending the static coefficients of the preceding and succeeding frames, and calculating the dynamic coefficients as a linear transform of this extended vector. Although there is evidence that this approach improves upon the continuous time approximation (van Dalen and Gales, 2009), it has a much higher computational cost and hence not considered for this thesis.

6.3.1 Transformation estimation

In this thesis, we use a first-order VTS approximation (Moreno et al., 1996) to linearise the mismatch function, around the expansion point $\{\boldsymbol{\mu}_x^{(r)}, \boldsymbol{\mu}_h, \boldsymbol{\mu}_n\}$ which results in:

$$\mathbf{y}_s|r \approx f(\boldsymbol{\mu}_{x_s}^r, \boldsymbol{\mu}_h, \boldsymbol{\mu}_n, \boldsymbol{\alpha}) + \mathbf{G}_x^{(r)} \left(\mathbf{x}_s - \boldsymbol{\mu}_{x_s}^{(r)}\right) + \mathbf{G}_n^{(r)} \left(\mathbf{n} - \boldsymbol{\mu}_n\right) \tag{6.19}$$

where $\mathbf{G}_x^{(r)}$ and $\mathbf{G}_n^{(r)}$ denote the Jacobian matrices

$$\mathbf{G}_x^{(r)} = \frac{\partial f(\cdot)}{\partial \mathbf{x}_s}\Big|_{\boldsymbol{\mu}_{x_s}^{(r)}, \boldsymbol{\mu}_h, \boldsymbol{\mu}_n}, \tag{6.20}$$

$$\mathbf{G}_n^{(r)} = \frac{\partial f(\cdot)}{\partial \mathbf{n}}\Big|_{\boldsymbol{\mu}_{x_s}^{(r)}, \boldsymbol{\mu}_h, \boldsymbol{\mu}_n} = \mathbf{I} - \mathbf{G}_x^{(r)}. \tag{6.21}$$

Here, note that the two Jacobian matrices $\mathbf{G}_x^r$ and $\mathbf{G}_n^r$ are functions of the phase factor $\boldsymbol{\alpha}$. Hence, the tuning the value of $\boldsymbol{\alpha}$ can lead to different Jacobian matrices, and consequently control the value of JUD transformation parameters. Bearing in mind this will help to understand the experiments on the phase factor $\boldsymbol{\alpha}$ in section 6.4.3. The mean and covariance of $\mathbf{y}$ can then be obtained by taking expectations:

$$\begin{aligned} \boldsymbol{\mu}_{y_s}^{(r)} &= \mathbb{E}\left[\mathbf{y_s}|r\right] \\ &= f\left(\boldsymbol{\mu}_{x_s}^{(r)}, \boldsymbol{\mu}_h, \boldsymbol{\mu}_n, \boldsymbol{\alpha}\right), \end{aligned} \tag{6.22}$$

$$\begin{aligned} \boldsymbol{\Sigma}_{y_s}^{(r)} &= \mathbb{E}\left[\mathbf{y}_s\mathbf{y}_s^T|r\right] - \boldsymbol{\mu}_{y_s}^{(r)}\boldsymbol{\mu}_{y_s}^{(r)T} \\ &= \mathbf{G}_x^{(r)}\boldsymbol{\Sigma}_{x_s}^{(r)}\mathbf{G}_x^{(r)T} + \mathbf{G}_n^{(r)}\boldsymbol{\Sigma}_n\mathbf{G}_n^{(r)T}. \end{aligned} \tag{6.23}$$

To obtain the delta parameters, we similarly take expectations on both sides of equation (6.7):

$$\boldsymbol{\mu}_{\Delta y}^{(r)} \approx \mathbf{G}_x^{(r)}\boldsymbol{\mu}_{\Delta x}^{(r)}, \tag{6.24}$$

$$\boldsymbol{\Sigma}_{\Delta y}^{(r)} \approx \mathbf{G}_x^{(r)}\boldsymbol{\Sigma}_{\Delta x}^{(r)}\mathbf{G}_x^{(r)T} + \mathbf{G}_n^{(r)}\boldsymbol{\Sigma}_{\Delta n}\mathbf{G}_n^{(r)T}. \tag{6.25}$$

Here we have assumed $\mathbb{E}\left[\boldsymbol{\Delta}\mathbf{n}\right] = \mathbf{0}$. This assumption was relaxed in (Li et al., 2009), but the results showed no improvements when compensating the static or dynamic parts of the variances. Similar expressions can be obtained for the acceleration coefficients, where we assume $\mathbb{E}\left[\boldsymbol{\Delta}^2\mathbf{n}\right] = \mathbf{0}$ as well.

The cross covariance $\boldsymbol{\Sigma}_{y_s x_s}$ is calculated as:

$$\boldsymbol{\Sigma}_{y_s,x_s}^{(r)} = \mathbb{E}\left[\mathbf{y}_s\mathbf{x}_s^T|r\right] - \boldsymbol{\mu}_{y_s}^{(r)}\boldsymbol{\mu}_{x_s}^{(r)T}. \tag{6.26}$$

By substituting the VTS approximation of $\mathbf{y}_s$ from equation (6.19) and $\boldsymbol{\mu}_{y_s}^{(r)}$ from equation (6.22) into (6.26), we obtain:

$$\boldsymbol{\Sigma}_{y_s x_s}^{(r)} \approx \mathbf{G}_x^{(r)}\boldsymbol{\Sigma}_{x_s}^{(r)}. \tag{6.27}$$

Again, a continuous time approximation can be used to derive the dynamic coefficients, which gives

$$\boldsymbol{\Sigma}^{(r)}_{\Delta y \Delta x} \approx \mathbf{G}^{(r)}_{x} \boldsymbol{\Sigma}^{(r)}_{\Delta x}, \qquad \boldsymbol{\Sigma}^{(r)}_{\Delta^2 y \Delta^2 x} \approx \mathbf{G}^{(r)}_{x} \boldsymbol{\Sigma}^{(r)}_{\Delta^2 x}. \tag{6.28}$$

Note that even if $\boldsymbol{\Sigma}^{(r)}_{x}$ and $\boldsymbol{\Sigma}_{n}$ are diagonal, $\boldsymbol{\Sigma}^{(r)}_{y}$ and $\boldsymbol{\Sigma}^{(r)}_{yx}$ are not, since the Jacobian matrices $\mathbf{G}^{(r)}_{x}$ and $\mathbf{G}^{(r)}_{n}$ are full. This makes the covariance bias term $\boldsymbol{\Sigma}^{(r)}_{b}$ block-diagonal, which is incompatible with standard HMM/GMM based speech recognizers that use diagonal covariance matrices. To obtain a final diagonal compensated covariance matrices, elements of the joint distribution are diagonalized for GMM based systems (Xu et al., 2011) as

$$\begin{bmatrix} \operatorname{diag}\left(\boldsymbol{\Sigma}^{(r)}_{x}\right) & \operatorname{diag}\left(\boldsymbol{\Sigma}^{(r)}_{xy}\right) \\ \operatorname{diag}\left(\boldsymbol{\Sigma}^{(r)}_{yx}\right) & \operatorname{diag}\left(\boldsymbol{\Sigma}^{(r)}_{y}\right) \end{bmatrix}. \tag{6.29}$$

Diagonalising is expected to limit the compensation power of JUD. For SGMMs, however, diagonalising is not applied since the model uses full or block-diagonal covariance matrices.

6.3.2 Compensating subspace Gaussian mixture models

Since JUD noise compensation takes the form of a feature transform with an additive covariance bias, it is well suited to the SGMM framework. By contrast, VTS compensates each Gaussian individually, which is computationally infeasible (and inelegant) for an SGMM system which has a large number of surface Gaussians—for instance, in the experiments presented in this paper the models have 6.4 million surface Gaussians. However, to apply JUD compensation, a regression model is needed which clusters all the surface Gaussians of SGMMs. It is certainly possible to use a conventional clustering algorithm, as in GMM based models, to derive the regression model, however, it is computationally expensive, and furthermore, the covariance matrices will not be globally shared after compensation since the covariance bias term depends on the regression class. This will also increase the computational cost. This can be circumvented by using UBM as the regression model.

Using JUD with the UBM as the regression model, the likelihood of noise-corrupted speech becomes:

$$P(\mathbf{y}_t|j,\mathcal{M}_n) = \sum_{k=1}^{K_j} c_{jk} \sum_{i=1}^{I} w_{jki} |\mathbf{A}^{(i)}| \mathcal{N}\left(\mathbf{A}^{(i)}\mathbf{y}_t + \mathbf{b}^{(i)}; \boldsymbol{\mu}_{jki}, \boldsymbol{\Sigma}_i + \boldsymbol{\Sigma}^{(i)}_{b}\right) \tag{6.30}$$

Table 6.1: Procedure for JUD noise-compensation using gradient-based noise model estimation. In this paper, we used the Viterbi alignment for the SGMM system. Step 3 is required for the first loop, but can be skipped after that which means only the alignment will be updated using the new noise model.

1. Given a test utterance U, initialize the noise model $\mathcal{M}_n$.
2. Estimate the JUD transforms $\{\mathbf{A}^{(i)}, \mathbf{b}^{(i)}, \boldsymbol{\Sigma}^{(i)}\}$ using current $\mathcal{M}_n$.
3. If required, decode U and generate the hypothesis $\mathcal{H}_u$ given the clean acoustic model $\mathcal{M}_s$, and the JUD transforms $\{\mathbf{A}^{(i)}, \mathbf{b}^{(i)}, \boldsymbol{\Sigma}^{(i)}\}$.
4. Given U, $\mathcal{M}_s$ and $\mathcal{H}_u$, accumulate the statistics λ_u by Viterbi alignment.
5. Update the noise model:
   for $i = 1$; $i \leq$ #iter1; i++
   1) Given λ_u, $\mathcal{M}_s$ and the 'old' noise model $\mathcal{M}_n$, update the noise model means $\boldsymbol{\mu}_n, \boldsymbol{\mu}_h$ (6.35).
   2) Compute the auxiliary function (6.31), and if its value decrease, back-off the noise model means. (6.39, 6.40).
   for $j = 1$; $j \leq$ #iter2; j++
   3) Given λ_u, $\mathcal{M}_s$ and the 'old' noise model $\mathcal{M}_n$, update the noise model variance $\boldsymbol{\Sigma}_n$ (6.36).
   4) Compute the auxiliary function (6.31), and if its value decrease, back-off the noise model variance.
   end
   end
6. Go to step 2. if not converged.
7. Decode the utterance to obtain the final results.

where $\mathbf{A}^{(i)}, \mathbf{b}^{(i)}$ and $\boldsymbol{\Sigma}_b^{(i)}$ correspond to the i^{th} Gaussian in the UBM; and $\mathcal{M}_n = \{\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n, \boldsymbol{\mu}_h\}$ denotes the noise model. Since the covariances are compensated using an additive bias, the data-independent normalisation terms in the SGMM likelihood computation (cf. (Povey et al., 2011a), section 3) need to be recomputed on a per-utterance basis. This extra computation may be saved by using a predictive CMLLR method (Gales and Van Dalen, 2007) that computes a set of feature transforms to minimise the Kullback-Leibler divergence between the CMLLR-adapted and JUD-compensated distributions (Xu et al., 2011). The effect of the covariance bias terms is subsumed in the second-order statistics used for the estimation of the CMLLR transforms, thereby keeping the original covariances unchanged.

6.3.3 Noise model estimation

Noise compensation using the mismatch function (6.6) requires knowledge of the noise parameters $\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n$, and $\boldsymbol{\mu}_h$. Given the clean speech model, the noise parameters and the

JUD transforms can be estimated alternately following the procedure outlined in Table 6.1. The noise model may be estimated either using expectation-maximization (EM), which treats the noise parameters as latent variables (Kim et al., 1998); or using a gradient based optimization approach (Liao, 2007; Li et al., 2009). A comparison between the two approaches (Zhao and Juang, 2010) showed the gradient-based approach to converge faster than EM, and to provide comparable or better recognition accuracy.

In this thesis we use the gradient-based approach. The auxiliary function for noise model update is

$$Q(\hat{\mathcal{M}}_n;\check{\mathcal{M}}_n)=\sum_{jkit}\gamma_{jki}(t)\left[\log|\mathbf{A}^{(i)}|+\log\mathcal{N}\left(\mathbf{A}^{(i)}\mathbf{y}_t+\mathbf{b}^{(i)};\boldsymbol{\mu}_{jki},\boldsymbol{\Sigma}_i+\boldsymbol{\Sigma}_b^{(i)}\right)\right], \quad (6.31)$$

where $\hat{\mathcal{M}}_n$ and $\check{\mathcal{M}}_n$ are the 'new' and 'old' estimates of the noise model, respectively. $\gamma_{jki}(t)$ is the Gaussian component posterior, which is defined as:

$$\gamma_{jki}(t)=p\left(j,k,i|\mathbf{y}_t\right). \quad (6.32)$$

In (Liao, 2007), the derivatives of the objective function are computed numerically for JUD/GMM based noise model estimation, while in this thesis, we derive the gradients and Hessian matrices using explicit mathematical derivation for the JUD/SGMM based noise model estimation. However, these derivations are not new as they are very similar to the VTS based noise model estimation in (Liao, 2007; Li et al., 2009), except for a difference in estimating the additive noise variance $\boldsymbol{\Sigma}_n$ since we use the block-diagonal covariance matrices for SGMM acoustic model rather than diagonal covariance matrices in GMMs (Liao, 2007; Li et al., 2009). We present the overview of the estimation here while leaving the details to the appendixes.

6.3.3.1 Update the additive and channel noise mean

To update the additive and channel noise means, we first fix the Jacobian matrices $\mathbf{G}_x^{(r)}$, $\mathbf{G}_n^{(r)}$ and the covariance bias terms $\boldsymbol{\Sigma}_b^{(r)}$. Taking the derivatives of $Q(\cdot)$ with respect to $\hat{\boldsymbol{\mu}}_n$ and $\hat{\boldsymbol{\mu}}_h$, we obtain

$$\frac{\partial Q(\cdot)}{\partial\hat{\boldsymbol{\mu}}_n}=\mathbf{d}-\mathbf{E}\hat{\boldsymbol{\mu}}_n-\mathbf{F}\hat{\boldsymbol{\mu}}_h, \quad (6.33)$$

$$\frac{\partial Q(\cdot)}{\partial\hat{\boldsymbol{\mu}}_h}=\mathbf{u}-\mathbf{V}\hat{\boldsymbol{\mu}}_n-\mathbf{W}\hat{\boldsymbol{\mu}}_h, \quad (6.34)$$

where $\mathbf{d},\mathbf{E},\mathbf{F}$ and $\mathbf{u},\mathbf{V},\mathbf{W}$ are defined in equations (B.7 - B.9) and (B.11 - B.13) in Appendix B. By setting the two derivatives to zero, we obtain the additive and channel

noise means as a solution to the following linear system:

$$\begin{bmatrix} \mathbf{E} & \mathbf{F} \\ \mathbf{V} & \mathbf{W} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{\mu}}_n \\ \hat{\boldsymbol{\mu}}_h \end{bmatrix} = \begin{bmatrix} \mathbf{d} \\ \mathbf{u} \end{bmatrix}. \tag{6.35}$$

Here, we jointly estimate $\boldsymbol{\mu}_n$ and $\boldsymbol{\mu}_h$, which is similar to the VTS noise model estimation in (Liao, 2007) (Chapter 4). This approach is slightly different from that used in Li et al. (2009), in which $\boldsymbol{\mu}_h$ is updated first, and $\boldsymbol{\mu}_n$ is estimated using the updated $\boldsymbol{\mu}_h$. The detailed derivation can be found in Appendix B.

6.3.3.2 Update the additive noise variance

Unlike the additive and channel noise means, there is no closed-form solution for additive noise variance $\boldsymbol{\Sigma}_n$. In this paper, we use Newton's algorithm to update it. Denote $\sigma^2_{n,d}$ as the d_{th} coefficient of $\boldsymbol{\Sigma}_n$,

$$\hat{\sigma}^2_{n,d} = \sigma^2_{n,d} - \zeta \left(\frac{\partial^2 Q(\cdot)}{\partial (\sigma^2_{n,d})^2} \right)^{-1} \left(\frac{\partial Q(\cdot)}{\partial \sigma^2_{n,d}} \right), \tag{6.36}$$

where ζ is the learning rate. The gradient and Hessian are defined as:

$$\frac{\partial Q(\cdot)}{\partial \sigma^2_{n,d}} = -\frac{1}{2} \sum_{i=1}^{I} (\gamma_i \kappa_{id} - \beta_{id}), \tag{6.37}$$

$$\frac{\partial^2 Q(\cdot)}{\partial (\sigma^2_{n,d})^2} = -\frac{1}{2} \sum_{i=1}^{I} (2\kappa_{id}\beta_{id} - \gamma_i \kappa^2_{id}), \tag{6.38}$$

where κ_{id}, β_{id} and $\boldsymbol{\Omega}_i$ are defined in equations (C.5), (C.8) and (C.7) in Appendix C, and $\gamma_i = \sum_{jkt} \gamma_{jki}(t)$. Note that in practice, the variance may be negative if eq (C.18) is applied directly. To enforce the positivity, the logarithm of variance is estimated as in (Li et al., 2009; Kalinli et al., 2010). Details of derivation are given in Appendix C.

6.3.4 Implementation Details

Since the noise model only accounts for the static features and the Jacobian matrices are fixed during estimation, updating $\boldsymbol{\mu}_n$ and $\boldsymbol{\mu}_h$ according to equation (6.35) does not guarantee an increase in the auxiliary function (6.31). We used a simple back-off scheme (Liao, 2007) that interpolates between the 'old' and 'new' model parameters:

$$\hat{\boldsymbol{\mu}}_h = \eta \boldsymbol{\mu}_h^{old} + (1-\eta) \boldsymbol{\mu}_h^{new}, \tag{6.39}$$

$$\hat{\boldsymbol{\mu}}_n = \eta \boldsymbol{\mu}_n^{old} + (1-\eta) \boldsymbol{\mu}_n^{new}, \tag{6.40}$$

where $\eta \in [0,1]$ is chosen by line search such that the auxiliary function does not decrease. A similar back-off scheme is also applied to additive noise variance $\mathbf{\Sigma}_n$. In our experiments, we found that the back-off scheme is important for noise model estimation, similar to (Liao, 2007). Finally, the auxiliary function (6.31) needs to be efficiently computed, since it is evaluated multiple times during the iterative update of the noise model. We do this by computing the sufficient statistics for each Gaussian compoent in the UBM over the entire utterance and caching them.

6.4 Experiments

We performed experiments using the Aurora 4 corpus which is derived from the Wall Street Journal (WSJ0) 5,000-word (5k) closed vocabulary transcription task. The clean training set contains about 15 hours of audio, and Aurora 4 provides a noisy version, which enables multi-condition training (MTR). The test set has 300 utterances from 8 speakers. The first test set, set A (`test01`), was recorded using a close talking microphone, similar to the clean training data. The data comprising set B (`test02` to `test07`) was obtained by adding six different types of noise, with randomly selected signal-to-noise ratios ranging from 5dB to 15dB, to set A. Set C (`test08`) was recording using a desk-mounted secondary microphone and the same type of noise used for set B was added to this test set forming set D (`test09` to `test14`). In the following experiments for both GMM and SGMM based systems, we used 39 dimensional feature vectors: 12th order mel frequency cepstral coefficients, plus energy, with delta and acceleration features. We used the standard WSJ0 5k bigram language model.

6.4.1 Results of GMM based systems

The GMM systems were built using the HTK software (Young et al., 2006). Table 6.2 shows the results of VTS and JUD noise compensation on a conventional GMM system, without the phase term (ie $\boldsymbol{\alpha} = \mathbf{0}$). Here, the clean and MTR models each have about 3,100 triphone states, with each speech state modelled using 16 Gaussian components and 32 Gaussian components for the silence state model. As expected, the clean model results in a high word error rate (WER) on the noisy test data, whereas the MTR model can alleviate the mismatch, resulting in significant reductions in WER, on average. For the JUD system, we used a regression model with 112 Gaussian components, in which 48 components were used for silence and the remaining 64 for speech.

Table 6.2: WER of VTS and JUD based on GMM systems with $\boldsymbol{\alpha} = \mathbf{0}$.

Methods	A	B	C	D	Avg
Clean model	7.7	56.6	46.7	72.8	**59.3**
MTR model	12.7	18.6	31.7	36.8	**26.9**
VTS-init	8.7	22.4	43.0	48.0	33.9
+ 1st iter	7.1	15.8	17.3	28.6	20.8
+ 2nd iter	7.3	14.8	12.1	24.8	**18.3**
JUD-init	8.4	23.8	42.6	47.1	34.0
+1st iter	7.2	17.3	24.1	31.8	23.3
+2nd iter	7.0	16.6	16.3	28.7	**21.1**

Table 6.3: WERs of noise compensation by JUD on SGMM systems with $\boldsymbol{\alpha} = \mathbf{0}$.

Methods	A	B	C	D	Avg
Clean model	5.2	58.2	50.7	72.1	**59.9**
MTR model	6.8	15.2	18.6	32.3	**22.2**
JUD-init	5.5	20.6	36.8	45.6	31.4
+1st iter	5.3	15.3	25.3	32.0	22.5
+2nd iter	5.3	14.7	20.7	28.4	**20.3**

Two separate regression trees were used. For comparison, we carried out VTS-based noise compensation, which may be viewed as JUD when every Gaussian component corresponds to a regression class.

The noise model was initialized by the first and last 20 frames of each test utterance, corresponding to "VTS-init" and "JUD-init" in Table 6.2. The hypotheses generated by the initial decoding were then used to update the noise model, and another decoding pass was conducted, giving results shown as "1st iter". The procedure was repeated to give the results "2nd iter". Table 6.2 indicates that updating the noise model leads to considerable gains in accuracy for both VTS and JUD. In addition, VTS-based systems consistently outperform their JUD counterparts as expected. However, the computation cost for JUD is much lower than that for VTS. The lowest WER given by VTS is 18.3% which is comparable to 17.8% reported in (Wang and Gales, 2011) with a similar system configuration, and that for JUD is 21.1% which is a little better than 22.2% in (Flego and Gales, 2011).

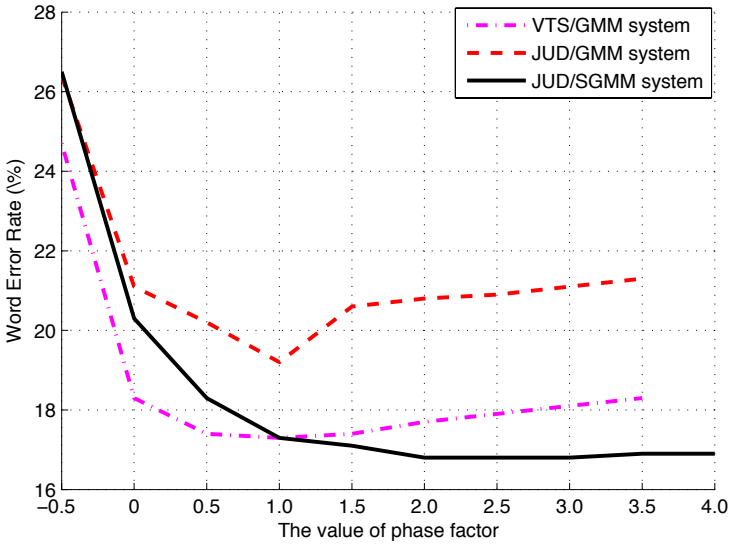


Figure 6.4: Effect of phase term α for both GMM and SGMM system with VTS or JUD style noise compensation. The best result for VTS/GMM is 17.3% ($\boldsymbol{\alpha} = \mathbf{1.0}$), JUD/GMM is 19.2% ($\boldsymbol{\alpha} = \mathbf{1.0}$) and JUD/SGMM is 16.8% ($\boldsymbol{\alpha} = \mathbf{2.5}$).

6.4.2 Results of SGMM based systems

The SGMM systems were built using the Kaldi software toolkit (Povey et al., 2011b). We used $I = 400$ components in the UBM and a subspace dimension $S = 40$ in the SGMM-based systems. There were about 3,900 tied triphone states, and about 16,000 substates were used in total, resulting in a total of 6.4 million Gaussian components. Similar to the GMM-based systems, we separated speech and silence in the regression model, using 100 Gaussian components for silence and 300 for speech in the UBM. Table 6.3 gives the baseline results using clean and MTR models. The SGMM system has a lower WER than the GMM system on clean test data (A; 5.2% vs. 7.7%); however, the improvement disappears in noisy conditions. For the MTR model, where the mismatch is less serious, we observed that the SGMM system has a lower average WER compared with its GMM counterpart (22.2% vs. 26.9%).

We then applied JUD noise compensation to a clean SGMM acoustic model. Table 6.3 shows the results without the phase term, i.e. $\boldsymbol{\alpha} = \mathbf{0}$. Again, the noise model is initialised by the first and last 20 frames of each utterance, and then updated by the algorithm described in section 6.3.3. The results show that JUD compensation lead to lower WERs for SGMM systems in the presence of additive noise compared with the MTR model. Overall, using a three-pass decoding, we achieve 20.3% WER, which is about 2% absolute lower than that obtained using the MTR/SGMM, but is 2% absolute

higher than that obtained by the VTS/GMM system.

We then investigated using a non-zero phase term. We did not optimize the value of $\boldsymbol{\alpha}$ (as in (Deng et al., 2004)) but set all the coefficients of $\boldsymbol{\alpha}$ to a fixed value (Li et al., 2009). As a comparison, we also investigated different values of the phase factor for the GMM-based VTS and JUD systems. Figure 6.4 graphs the average WERs. We find that the phase factor significantly affects both VTS and JUD compensation for GMM and for SGMM systems, consistent with previously reported results (Deng et al., 2004; Li et al., 2009). The phase factor has a large effect on the JUD/SGMM system: tuning $\boldsymbol{\alpha}$ achieves 16.8% WER, significantly lower than the baseline (20.3%), also lower than the best performance of VTS/GMM by 0.5% absolute. Possible reasons for this improvement may be the correlations between noise and speech captured by the phase factor, and the systematic bias introduced by the VTS linearisation error (equation (6.19)) (Deng et al., 2004; Li et al., 2009). In addition, $\boldsymbol{\alpha} = \mathbf{1}$ corresponds to magnitude domain compensation, in contrast to power domain compensation ($\boldsymbol{\alpha} = \mathbf{0}$) (Gales and Flego, 2010).

Table 6.4: WERs of each test set with regards to the value of phase factor for JUD/SGMM system. “restau.” denotes restaurant noise condition.

	A	B						C	D						Avg
α	clean	car	babble	restau.	street	airport	station	clean	car	babble	restau.	street	airport	station	
-0.5	5.2	9.0	20.9	26.3	21.9	16.6	23.1	23.3	29.1	36.4	41.6	41.1	36.2	40.1	26.5
0.0	5.3	7.6	14.6	20.1	15.8	13.4	16.6	20.7	18.5	28.0	32.6	32.4	28.4	30.7	20.3
0.5	5.3	**7.1**	13.0	18.6	**14.2**	12.4	15.5	17.8	14.1	25.6	30.7	29.2	24.9	27.9	18.3
1.0	5.3	7.2	**12.3**	17.5	14.2	11.6	**15.0**	16.2	12.7	24.0	30.0	27.1	23.6	26.2	17.3
1.5	5.3	7.1	12.4	17.5	14.5	**11.1**	15.2	14.2	12.5	23.9	28.8	26.3	23.5	26.3	17.1
2.0	5.2	7.1	12.5	**17.3**	14.4	11.2	15.3	13.1	12.1	**23.3**	28.6	**26.0**	23.2	**26.0**	16.8
2.5	5.1	7.3	12.5	17.5	14.4	11.5	15.5	12.0	**12.0**	23.4	28.2	26.1	23.1	26.2	16.8
3.0	**5.0**	7.4	12.5	17.4	14.8	12.0	15.7	10.8	12.1	24.0	**28.1**	26.0	**22.8**	26.5	16.8
3.5	5.3	7.6	12.8	17.5	14.6	11.8	15.8	10.7	12.1	24.5	28.3	26.3	23.0	26.5	16.9
4.0	5.1	7.6	13.2	17.8	14.9	11.7	16.0	**10.4**	12.2	23.8	28.3	26.4	23.2	26.7	16.9

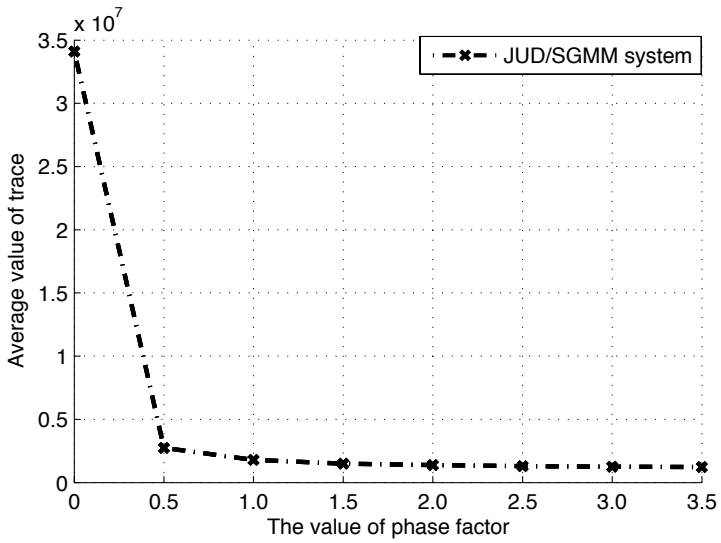


Figure 6.5: Average trace of covariance matrix $\mathbf{\Sigma}_i + \mathbf{\Sigma}_b^{(i)}$ respect to the phase term $\boldsymbol{\alpha}$ for JUD/SGMM systems. $\mathbf{\Sigma}_b^{(i)}$ is large when $\boldsymbol{\alpha}$ is small (e.g. $\boldsymbol{\alpha} = \mathbf{0}$). The value for $\boldsymbol{\alpha} = -0.5$ is much larger, and it is not shown here for clarity.

6.4.3 Analysis of the effect of phase factors

To gain further insight to the effect of phase term, we calculated the total variance of $\mathbf{\Sigma}_i + \mathbf{\Sigma}_b^{(i)}$ and averaged it by I and the number of test utterances. The plot is shown in Figure 6.5 for JUD/SGMM system. The average value of covariance shows a similar trend to that of the WER when using different values of phase factors. This is not unexpected if one interprets the value of covariance as indicating the degree of uncertainty of the model. A small covariance may indicate that the model is more confident in its explanation of the data; if this confidence is gained from more accurate model compensation, it is expected to result in lower WER. However, the absolute value in the figure is not intuitive as the features were first transformed into another feature space by the JUD transformation $(\mathbf{A}^{(i)}, \mathbf{b}^{(i)})$. As shown in the table, we obtain large value of $\mathbf{\Sigma}_b^{(i)}$ when $\boldsymbol{\alpha}$ is small. This is because that the Jacobian matrix $\mathbf{G}_x^{(i)}$ (cf. equation (6.20)) for component i is a function of the phase factor $\boldsymbol{\alpha}$, and we observe that when $\boldsymbol{\alpha}$ is small, $\mathbf{G}_x^{(i)}$ has very small eigenvalues, which lead to large transformation matrix $\mathbf{A}^{(i)}$ since (Liao, 2007):

$$\mathbf{A}^{(i)} \approx \begin{bmatrix} \mathbf{G}_x^{(i)-1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{G}_x^{(i)-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{G}_x^{(i)-1} \end{bmatrix} \tag{6.41}$$

Table 6.5: Confusion matrix of speech and silence separation by UBM model.

	sil	speech
sil	64.8%	35.2%
speech	9.3%	90.7%

Table 6.6: Comparison of UBM model with ('yes/S') and without ('no/S') speech and silence separation for JUD/SGMM system.

$\boldsymbol{\alpha}$	0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5
no/S	20.9	18.7	17.7	17.1	**16.8**	16.8	16.8	17.0
yes/S	20.3	18.3	17.3	17.1	**16.8**	16.8	16.8	16.9

and consequently, $\boldsymbol{\Sigma}_b^{(i)}$ is also large (cf. equation (6.18)). A large value of phase factor is able to smooth the Jacobian matricx $\mathbf{G}_x^{(i)}$, resulting in smaller transformation parameters $(\mathbf{A}^{(i)}, \mathbf{b}^{(i)}, \boldsymbol{\Sigma}_b^{(i)})$.

To investigate the effect of phase factors in different noise conditions, we show the results of JUD/SGMM system on the 14 individual test sets in Table 6.4. For the clean test set A, introducing a non-zero phase term does not improve accuracy notably just as expected. However, for the test set C which is also clean but recorded using a desk-mounted microphone, a large value of phase term increases the accuracy significantly (about 50% relative compared to the result of system without phase term). Since the training data is recorded using close-talking microphone, the mismatch between training and test data is mainly the channel noise including reverberation, which is correlated with speech. This is consistent with the assumption that phase factors model the correlations between noise and speech, and may explain the gains here. Comparing the results of sets B and D in which 6 different types of noise were added to the clean sets A and C respectively, the optimal values of the phase term, as well as the reductions in WER, are larger for D, which is probably because that there is more channel noise in set D which requires larger phase terms to capture the correlations.

6.4.4 Analysis of speech and silence separation in UBM

In the regression model of the JUD/GMM system, the Gaussian components for speech and silence were estimated using different regression trees. The reason for this is that speech and silence show different characteristics in the spectral domain, and the dis-

Table 6.7: WERs (%) of supervised ("SGMM-aux") and unsupervised ("UBM-aux") and hybrid ("Hybrid") noise model estimation for SGMM/JUD system. "#pass" denotes the number of decoding passes.

α	1.0	1.5	2.0	2.5	3.0	3.5	4.0	#pass
UBM-aux	18.2	17.7	17.5	17.4	17.3	**17.2**	17.3	1
SGMM-aux	17.7	17.1	**16.8**	16.8	16.8	17.0	17.1	3
Hybrid	17.5	17.1	16.8	16.8	**16.7**	16.8	16.8	2

tortions resulted from additive and channel noise are also different. This separation is expected to reduce the mismatch between the regression class model and its component Gaussians. We also separate the speech and silence in the UBM in the JUD/SGMM system,. This was done by first identifying the speech and silence frames in the training data using a baseline system, and then building the two UBM models for speech and silence using 100 and 300 Gaussian components, respectively. They were combined to derive the final UBM model. We then used the final UBM model to classify the acoustic frame in the training data, and the results are shown in Table 6.5. We observe that by this approach, we achieve high accuracy to identify the speech frames, but not for silence (90.7% vs. 64.8%). The accuracy for noisy test data may decrease further even after noise compensation. This may undermine the gains achieved by separating speech and silence in the UBM. We compared the results of systems with and without speech and silence separation, which is shown in Table 6.6. Without the phase term, we achieved 0.6% gains relative by speech and silence separation in UBM, but the two system achieve the same accuracy after tuning the phase factor.

6.4.5 Unsupervised noise model estimation

Model-based noise compensation is normally computationally expensive, and not suitable for real time applications. For instance, in our experiments, we performed three decoding passes to obtain the final results, in which the first two were used to generate the hypothesis for the noise model estimation. For applications with limited computational power, feature space noise compensation is normally preferred, but has lower accuracy compared to its model-based counterpart (Li et al., 2012). We have investigated reducing the computational cost of JUD/SGMM by unsupervised noise model estimation. Instead of Equation (6.31), we used the UBM to update the noise model

Table 6.8: Approximation of computational cost for VTS/GMM, JUD/GMM and JUD/SGMM system. M' and R denote the total number Gaussians and regression classes in GMM systems.

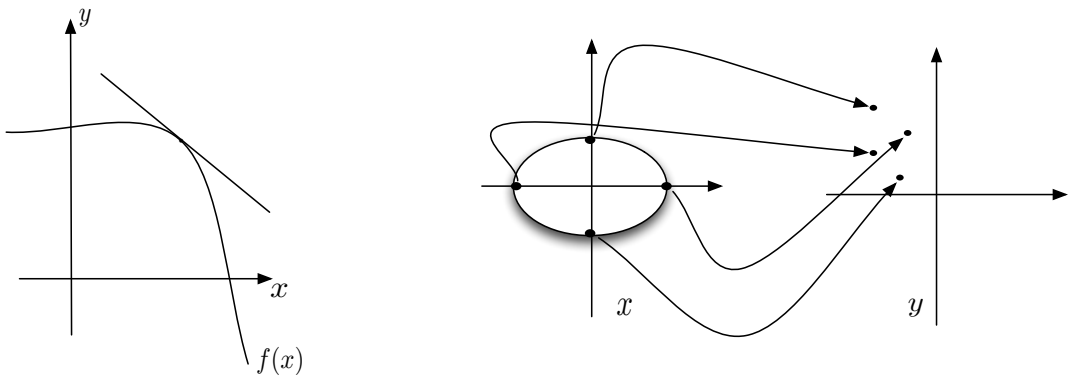
System	Model	Transform Estimation	Compensation
VTS/GMM	diag	$O(M'D^3)$	$O(M'D^2)$
JUD/GMM	diag	$O(RD^3)$	$O(RTD+M'D)$
JUD/SGMM	blck	$O(ID^3)$	$O(ITD^2+ID^2)$

using the following auxiliary function:

$$Q(\hat{\mathcal{M}}_n;\check{\mathcal{M}}_n)=\sum_{it}\gamma_i(t)\left[\log|\mathbf{A}^{(i)}|+\log\mathcal{N}\left(\mathbf{A}^{(i)}\mathbf{y}_t+\mathbf{b}^{(i)};\boldsymbol{\mu}_x^{(i)},\boldsymbol{\Sigma}_x^{(i)}+\boldsymbol{\Sigma}_b^{(i)}\right)\right], \quad (6.42)$$

where $\boldsymbol{\mu}_x^{(i)}$ and $\boldsymbol{\Sigma}_x^{(i)}$ are the mean and covariance the ith UBM component, and $\gamma_i(t)$ is the posterior of the ith component. In this case, the noise model is estimated without needing to generate the hypothesis by decoding the test utterance first, leading to a significant reduction in computational cost. The motivation behind this is similar to feature space VTS, in which a GMM is used to model the acoustic space, and to learn the mapping between a clean model and its noise-corrupted model. However, we do not use the mapping to de-noise the features, but to compensate the noise in the model domain. It also differentiates from the front-end JUD (FE-JUD) (Liao, 2007), in which a GMM is used to model the conditional distribution $p(\mathbf{y}|\mathbf{x})$ in equation (6.10) which is independent of the acoustic model. The transformation for each acoustic frame is globally shared by the whole Gaussian components of the acoustic model in FE-JUD while it depends on the regression class here.

Table 6.7 shows that we can achieve just slightly worse accuracy by using unsupervised noise model estimation (17.3% vs. 16.8% in terms of WER), while significantly reduces the computational cost (with only one-pass decoding). We can also use the unsupervised fashion to initialized the noise model, and then switch to supervised estimation to refine the noise model parameters. We denote this system as "Hybrid" in Table 6.7, in which we only update the noise model once by the supervised fashion. We achieved about the same accuracy compared to "SGMM-aux" but significantly reduced the computational cost.



(a) (b)

Figure 6.6: A comparison between VTS and UT approximation: (a) VTS approximates the nonlinear function $y = f(x)$ by vector Taylor series expansion, and results in a linear function by using first order VTS. (b) UT draws sigma points from the distribution of x and synthesise the corresponding samples of y by the nonlinear function $f(x)$.

6.4.6 JUD with unscented transform

Recently, the unscented transform (UT) (Julier and Uhlmann, 2004) has been applied to noise compensation in both feature and model domains (Hu and Huo, 2006; Faubel et al., 2010; Xu and Chin, 2009a; Li et al., 2010), and has achieved good results. Unlike DPMC, UT draws samples deterministically from the *sigma points*—a set of points chosen to have the same mean and covariance as the original distribution. In UT it assumed that the mean and covariance of the nonlinear system can be derived from sigma points (Julier and Uhlmann, 2004), although a recent review (Gustafsson and Hendeby, 2012) pointed out that this is not guaranteed depending on the nonlinear system and parameterisation of UT. Based on GMM system settings, UT can result in a more accurate estimate compared to first-order VTS, while its computational cost is much lower than DPMC (Xu and Chin, 2009a; Li et al., 2010). Our final set of experiments is to apply UT to compensate an SGMM against noise in the framework of JUD.

Unlike VTS which approximates the nonlinear function by a linear function to estimate the distribution of $\mathbf{y}$, sampling approaches draw samples from the distributions of $\mathbf{x}$ and $\mathbf{n}$ to synthesise noisy samples from which to estimate its distribution[2]. UT is a deterministic sampling approach. Let $\mathbf{z} = \begin{bmatrix} \mathbf{x}_s \\ \mathbf{n}_s \end{bmatrix}$ be the combined vector, then UT

[2]We don't draw samples for $\mathbf{h}$ because we assume its distribution is a delta function.

draws samples as

$$\mathbf{z}_0^{(r)} = \boldsymbol{\mu}_z^{(r)}, \tag{6.43}$$

$$\mathbf{z}_i^{(r)} = \boldsymbol{\mu}_z^{(r)} + \left[\sqrt{(2d+\kappa)\boldsymbol{\Sigma}_z^{(r)}}\right]_i, \tag{6.44}$$

$$\mathbf{z}_{i+d}^{(r)} = \boldsymbol{\mu}_z^{(r)} - \left[\sqrt{(2d+\kappa)\boldsymbol{\Sigma}_z^{(r)}}\right]_i, \tag{6.45}$$

where $i = 1, \ldots, d$, and $\sqrt{\mathbf{A}}$ and $[\mathbf{A}]_i$ denote the Cholesky decomposition and i_{th} column of the matrix $\mathbf{A}$ respectively. κ is a tuning parameter, d is the dimensionality of $\mathbf{z}$, and

$$\boldsymbol{\mu}_z^{(r)} = \begin{bmatrix} \boldsymbol{\mu}_{x_s}^{(r)} \\ \boldsymbol{\mu}_{n_s} \end{bmatrix}, \quad \boldsymbol{\Sigma}_z^{(r)} = \begin{bmatrix} \boldsymbol{\Sigma}_{x_s}^{(r)} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma}_{n_s} \end{bmatrix}. \tag{6.46}$$

After obtaining the noise and clean speech samples $\{\mathbf{n}_0, \ldots, \mathbf{n}_{2d}\}$ and $\{\mathbf{x}_0, \ldots, \mathbf{x}_{2d}\}$, the noise corrupted speech samples $\{\mathbf{y}_0, \ldots, \mathbf{y}_{2d}\}$ can be derived by the mismatch function (6.6) and the static parameters can be obtained by

$$\boldsymbol{\mu}_{y_s}^{(r)} = \sum_{i=0}^{2d} w_i \mathbf{y}_i \tag{6.47}$$

$$\boldsymbol{\Sigma}_{y_s}^{(r)} = \sum_{i=0}^{2d} w_i \mathbf{y}_i \mathbf{y}_i^T - \boldsymbol{\mu}_{y_s} \boldsymbol{\mu}_{y_s}^T, \tag{6.48}$$

$$\boldsymbol{\Sigma}_{y_s x_s}^{(r)} = \sum_{i=0}^{2d} w_i \mathbf{y}_i \mathbf{x}_i^T - \boldsymbol{\mu}_{y_s} \boldsymbol{\mu}_{x_s}^T, \tag{6.49}$$

where the weights are defined in UT as

$$w_0 = \frac{\kappa}{d+\kappa}, \quad w_i = \frac{1}{2(d+\kappa)}. \tag{6.50}$$

In this work, we set $\kappa = 1/2$ to give the equal weight to all the samples (Julier and Uhlmann, 2004) . For the dynamic coefficients, we still use the continuous time approximation which requires linearisation as VTS. Unlike equation (6.20) and (6.21), the Jacobian is obtained by all the samples rather than just the mean as

$$\tilde{\mathbf{G}}_x^{(r)} = \sum_{i=0}^{2d} w_i \frac{\partial f(\cdot)}{\partial \mathbf{x}_{is}} \Big|_{\mathbf{z}_{is}, \boldsymbol{\mu}_{hs}}, \quad \tilde{\mathbf{G}}_n^{(r)} = \mathbf{I} - \tilde{\mathbf{G}}_x^{(r)} \tag{6.51}$$

In this work, however, we found that using the Jacobian (6.51) to linearise the static covariance $\boldsymbol{\Sigma}_{y_s}$ and $\boldsymbol{\Sigma}_{y_s x_s}$ can achieve better results, as the static and dynamic coefficients are derived in a consistent fashion. Figure 6.6 illustrates the principle ideas of VTS and UT approximation.

Table 6.9: WERs of noise compensation by JUD on SGMM systems with $\boldsymbol{\alpha} = \mathbf{0}$.

Methods	A	B	C	D	Avg
Clean model	5.2	58.2	50.7	72.1	**59.9**
MTR model	6.8	15.2	18.6	32.3	**22.2**
JUD-VTS init	5.3	22.5	36.8	47.4	32.9
+1st iter	5.1	15.8	24.6	33.8	23.4
+2nd iter	5.1	15.0	19.8	29.7	**20.9**
+UT re-est	5.0	14.0	20.7	28.4	**20.0**
JUD-UT init	5.2	19.8	36.9	44.7	30.6
+1st iter	4.9	15.0	23.4	30.6	21.6
+2nd iter	4.9	14.3	18.4	26.9	**19.3**

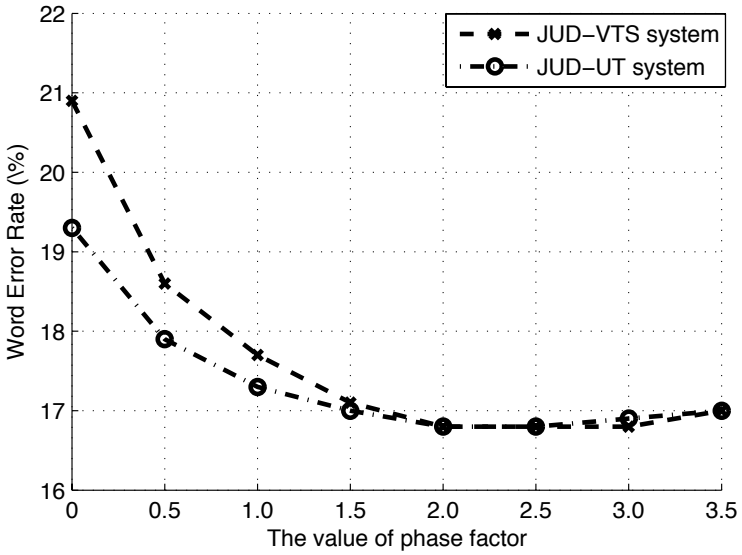


Figure 6.7: Average WER with respect to the phase term $\boldsymbol{\alpha}$ for JUD with VTS and UT compensation for SGMM systems. They achieve almost the same accuracy after increasing the value of phase term.

Table 6.9 gives the baseline results using clean and MTR models. In these experiments, we initialised the noise model by the first and last 20 frames of each utterance, and the results are shown by "JUD-VTS init" and "JUD-UT init". Note that, we do not perform speech and silence separation in the UBM, so the baseline results are different with those in Table 6.3. We then updated the noise model by either UT or VTS using the hypothesis from the previous decoding results. Here, we did not use the phase term, i.e. $\boldsymbol{\alpha} = 0$. The results are shown in Table 6.9 in which, after two decoding passes, the

JUD-VTS system achieves the average WER of 20.9%, indicated by JUD-VTS "+2nd iter". Given these noise model, we re-estimate the JUD compensation parameters using UT and can reduce the WER to be 20.0%. This shows that UT can lead to more accurate estimate in this condition given the same noise model compared to VTS. If we update the noise model from scratch, we achieve 19.3% WER after two decoding passes, which is considerably better than that of 20.9% for JUD-VTS system, and also 22.2% of MTR baseline.

We then tune the value of phase factor $\boldsymbol{\alpha}$. Figure 6.7 graphs the average WERs. Similar to the JUD-VTS system and consistent with the observations in (Li et al., 2010), the phase factor also affects JUD with UT system, and after increasing the value of $\boldsymbol{\alpha}$, the gap between JUD-VTS and JUD-UT system shrinks, and both system achieve the same lowest WER, 16.8%, when $\boldsymbol{\alpha} = \mathbf{2.0}$. Similar results were also obtained by comparing VTS and UT on GMM based systems on another task (Li et al., 2010).

6.5 Discussion and Conclusion

This chapter addresses robust speech recognition based on subspace Gaussian mixture models (SGMMs) using joint uncertainty decoding (JUD) noise compensation. Compared to VTS, JUD reduce the computational cost significantly by sharing the compensation parameters for Gaussian components within the same class. The major computational cost lies in that multiple decoding passes are required to estimate the noise model parameters and compensation parameters. In JUD, for each decoding pass we did not adapt each surface Gaussian of the SGMMs, but used I regression classes to generate I transformations $\{(\mathbf{A}^{(i)}, \mathbf{b}^{(i)}, \boldsymbol{\Sigma}_b^{(i)}), i = 1, \ldots, I\}$. For each test utterance with T frames, each frame is transformed by the I transformations with the computational cost as $O(ITD^2)$. In this chapter, I was set to be 400, and T was between 100 to about one thousand. To update the variance, it requires $O(ID^2)$ and as we also need to update the normalization term as in (Povey et al., 2011a), it requires some additional computation. However, the total computational cost is still significantly lower than VTS compensation which would require $O(MD^3)$ for blog-diagonal covariance matrices used in this paper, in which M is the total number of surface Gaussians of the SGMM acoustic model (6.4 million in this paper). As mentioned before, further computation may be saved for JUD/SGMM by predictive CMLLR (Gales and Van Dalen, 2007) to reduce the covariance bias term $\boldsymbol{\Sigma}_b^{(i)}$, so that the normalization terms of SGMMs can be left untouched.

In Table 6.8, we compare the computational cost in terms of transformation estimation and compensation for VTS/GMM, JUD/GMM and JUD/SGMM systems. For transform estimation, the main computational cost is to estimate the Jacobian matrices (e.g. equation (6.20)), which is linear to the number of Gaussian for VTS/GMM system, while it is linear to the number of regression classes for JUD/GMM and JUD/SGMM systems. In this case, the cost of estimating transformations for JUD/SGMM system is lower than that of VTS/GMM system. For compensation, the computational cost lies in compensating the covariance (e.g. equation (6.23)) for VTS/GMM system, and as we used diagonal covariances, the cost was reduced to $O(M'D^2)$. The number of Gaussians in VTS/GMM system is about 3000, hence $M' < IT$. Thus, the overall computational cost of JUD/SGMM system may be still higher than that of VTS/GMM system. However, the gap will shrink when using larger number of Gaussians in VTS/GMM system for larger vocabulary tasks. To further reduce the computational cost of JUD/SGMM, we also investigated the unsupervised noise model estimation using UBM which removes the need of multiple decoding passes by slightly sacrificing the accuracy, or it can be used to initialize the noise model to reduce the number of decoding passes for supervised fashion.

To summarise our experiments, by empirically tuning the phase factor, we achieved 16.8% WER for JUD/SGMM system on the Aurora 4 dataset, which slightly outperforms 17.3% WER by VTS/GMM system in our experiments, and it is comparable to state-of-the-art results by noise compensation on this task (Wang and Gales, 2011; Ragni and Gales, 2011). Further improvement has been observed by VTS-based noise adaptive training (NAT) (Kalinli et al., 2010). For instance, from the baseline results in (Ragni and Gales, 2011) on the same task, NAT-VTS improved over VTS alone from 17.9% to 16.0% WER, and further gains were obtained by discriminative adaptive training using minimum phone error (MPE) criterion which achieved 15.3% WER (Ragni and Gales, 2011). This points out the direction for our future work on adaptive training for JUD/SGMM system.

Chapter 7

Noise Adaptive Training for Subspace Gaussian Mixture Model

7.1 Introduction

In Chapter 6, we have investigated the noise compensation technique for an SGMM acoustic model which is trained on clean data. However, modern state-of-the-art automatic speech recognition (ASR) systems are normally trained on a large amount of heterogeneous acoustic data recorded from different speakers and in various environmental conditions. This induces nuisance variability in the acoustic data which is irrelevant to the task of speech recognition, and hence reduces the recognition accuracy of an ASR system. Adaptive training is an effective technique to normalise such variability. A typical example is speaker adaptive training (SAT) (Anastasakos et al., 1996), in which speaker-dependent transformations are trained jointly with the acoustic model parameters in order to account for speaker-related variability. The canonical acoustic model trained in this fashion is a better model for the phonetic variabilities in the acoustic data. Similar adaptive training schemes have also been proposed to normalise the variability induced by environmental noise, which is referred to as noise adaptive training (NAT) (Deng et al., 2000; Kalinli et al., 2010), including some variants such as irrelevant variability normalisation (IVN) (Hu and Huo, 2007) and joint adaptive training (JAT) (Liao and Gales, 2007).

The application of NAT depends on the particular choice of the noise compensation algorithms, which may be either feature-domain or model-domain. Several approaches of this nature have been proposed, each with specific strengths and weaknesses. For instance, the vector Taylor series (VTS) (Moreno et al., 1996) and model-based joint

uncertainty decoding (JUD) (Liao, 2007) approaches rely on a mismatch function that models the relationship between clean and noise corrupted speech. Using such a mismatch function has the advantage that the required amount of adaptation data is small, which is suitable for rapid adaptation. But its applicability is limited to spectral or cepstral features. SPLICE (Deng et al., 2000; Droppo et al., 2002) and front-end JUD (Liao and Gales, 2005) remove this constraint by learning a mapping between clean and noisy speech from stereo (both noisy and clean) training data. However, stereo data is normally hard to obtain, and it may not generalise well to unseen noise conditions. Noisy constrained maximum likelihood linear regression (NCMLLR) (Kim and Gales, 2011), which is a purely data-driven method, is more flexible from this perspective. It relies neither on a mismatch function (as with VTS or JUD), nor on having stereo training data (as with SPLICE), but estimates the noise compensation transformations using the maximum likelihood (ML) criterion for each homogeneous block of acoustic data. However, it requires a larger amount of training data to achieve good performance, and hence it is not suitable for rapid adaptation.

In Chapter 6, we extended JUD-based noise compensation to subspace Gaussian mixture models (SGMMs). In this Chapter, we study the application of NAT to SGMMs using JUD transformations. The adaptive training algorithm is derived from the generative nature of the JUD transformation (Kim and Gales, 2011), which leads to an efficient EM-based algorithm to update the acoustic model parameters. Again, the experiments of using the NAT algorithm were performed on the Aurora 4 dataset and some of the results have been presented in (Lu et al., 2013c).

7.2 Generative form of JUD

An introduction of JUD has been given in Chapter 6.3, where it is generally derived from the model adaptation perspective. As shown in (Kim and Gales, 2011), JUD may also be represented as a generative model for each regression class r:

$$\mathbf{y}_t = \mathbf{H}^{(r)}\mathbf{x}_t + \mathbf{g}^{(r)} + \mathbf{e}_t^{(r)}, \quad \mathbf{e}_t^{(r)} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{\Phi}^{(r)}\right) \tag{7.1}$$

where $\mathbf{H}^{(r)}$ is a linear transform, $\mathbf{g}^{(r)}$ denote the bias term and $\mathbf{e}_t^{(r)}$ is a Gaussian additive noise. From equation (7.1), the conditional distribution of $\mathbf{y}_t$ given $\mathbf{x}_t$ for each regression class can be obtained as

$$p(\mathbf{y}_t|\mathbf{x}_t, r) = \mathcal{N}\left(\mathbf{y}_t; \mathbf{H}^{(r)}\mathbf{x}_t + \mathbf{g}^{(r)}, \mathbf{\Phi}^{(r)}\right). \tag{7.2}$$

Given this distribution, the original JUD likelihood function (6.15) can be obtained by substituting equation (7.2) into (6.11) by setting the JUD transformation parameters to be $\mathbf{A}^{(r)} = \mathbf{H}^{(r)-1}, \mathbf{b}^{(r)} = -\mathbf{H}^{(r)-1}\mathbf{g}^{(r)}$ and $\mathbf{\Sigma}_b^{(r)} = \mathbf{A}^{(r)}\mathbf{\Phi}^{(r)}\mathbf{A}^{(r)T}$.

The generative view of JUD is particularly useful, since it makes it possible to estimate the JUD transforms in a data-driven fashion. It is more flexible as it gets rid of the mismatch function (6.6). For instance, a successful example can be found in (Kim and Gales, 2011) which is also known as noisy-CMLLR. Meanwhile, an EM algorithm can also be derived to update the acoustic model parameter for adaptive training as in (Kim and Gales, 2011; Flego and Gales, 2009). Our NAT algorithm for SGMMs is also based on such kind of reformulation of JUD as detailed in the following sections.

7.3 Noise adaptive training

Noise adaptive training (NAT) of the acoustic model involves joint optimisation of the acoustic model parameters $\mathcal{M}$ and the transformation parameters $\mathcal{T}$. For an SGMM acoustic model, the auxiliary function for NAT is

$$\mathcal{Q}\left(\mathcal{M},\mathcal{T};\tilde{\mathcal{M}},\tilde{\mathcal{T}}\right) = \sum_{jkit} \gamma_{jki}(t) \log\left[|\mathbf{A}^{(r)}|\mathcal{N}\left(\mathbf{A}^{(r)}\mathbf{y}_t + \mathbf{b}^{(r)}; \boldsymbol{\mu}_{jki}, \mathbf{\Sigma}_i + \mathbf{\Sigma}_b^{(r)}\right)\right] \quad (7.3)$$

where $\tilde{\mathcal{M}}$ and $\tilde{\mathcal{T}}$ denote the current estimate of the model and transformation parameters, and $\gamma_{jki}(t)$ is the posterior probability, computed based on $\tilde{\mathcal{M}}$ and $\tilde{\mathcal{T}}$. This auxiliary function is for a particular training utterance that the transformation parameters $\mathcal{T}$ depend on. The overall auxiliary function for the entire training set is obtained by summing (7.3) over all utterances.

Directly optimising either $\mathcal{M}$ or $\mathcal{T}$ is computationally demanding, especially for an SGMM, since the auxiliary function is complex. Analogous to SAT (Anastasakos et al., 1996), a common practice is to interleave the update of $\mathcal{M}$ and $\mathcal{T}$ one after another (Kalinli et al., 2010; Liao and Gales, 2007). In this paper, we adopt the same principle for adaptive training of SGMMs. We have previously detailed the estimation of $\mathcal{T}$ given $\mathcal{M}$ in Chapter 6; in this chapter, we focus on the estimation of the acoustic model parameters $\mathcal{M}$ given the estimate of the transformation parameters $\mathcal{T}$.

7.3.1 Optimisation

Two optimisation approaches for the update of the acoustic model parameters $\mathcal{M}$ in NAT have been investigated: second-order gradient-based (Liao and Gales, 2007;

Kalinli et al., 2010) and EM-based (Kim and Gales, 2011).

In the second-order gradient-based approach a particular set of parameters θ in $\mathcal{M}$ is updated by

$$\theta = \tilde{\theta} - \zeta \left[\left(\frac{\partial^2 Q(\cdot)}{\partial^2 \theta} \right)^{-1} \left(\frac{\partial Q(\cdot)}{\partial \theta} \right) \right]_{\theta=\tilde{\theta}} \tag{7.4}$$

where $\tilde{\theta}$ is the current value of θ, ζ is the learning rate and $Q(\cdot)$ denotes the auxiliary function (7.3). Such gradient-based optimisation was used for JUD-GMM systems (Liao and Gales, 2007) and for VTS-GMM systems (Kalinli et al., 2010). Depending on the form of Hessian, it may yield faster convergence. However, the drawbacks of this approach are that the computation of the gradient and Hessian terms in (7.4) can be complex, especially for the SGMM-based acoustic models due to the compact model representation. Furthermore, it is not simple to do gradient-based optimisation when using a discriminative criteria (Flego and Gales, 2009).

The second type of optimisation is based on the EM algorithm, which is derived from viewing the JUD transformation as a generative model (7.1). This method requires computing sufficient statistics of the expected "pseudo-clean" speech feature $\mathbf{x}_t$, which is obtained by computing its conditional distribution given component m:

$$p(\mathbf{x}_t|\mathbf{y}_t, r, m) = \frac{p(\mathbf{y}_t|\mathbf{x}_t, r)p(\mathbf{x}_t|m)}{\int p(\mathbf{y}_t|\mathbf{x}_t, r)p(\mathbf{x}_t|m)d\mathbf{x}_t}. \tag{7.5}$$

As shown in (Kim and Gales, 2011), an analytical solution can be obtained from (7.2), which gives the conditional expectations as

$$\mathbb{E}[\mathbf{x}_t|\mathbf{y}_t, r, m] = \tilde{\mathbf{x}}_t^{(rm)} \tag{7.6}$$

$$\mathbb{E}[\mathbf{x}_t\mathbf{x}_t^T|\mathbf{y}, r, m] = \tilde{\boldsymbol{\Sigma}}_x^{(rm)} + \tilde{\mathbf{x}}_t^{(rm)}\tilde{\mathbf{x}}_t^{(rm)T} \tag{7.7}$$

where

$$\begin{aligned}
\tilde{\mathbf{x}}_t^{(rm)} &= \tilde{\mathbf{A}}^{(rm)}\mathbf{y}_t + \tilde{\mathbf{b}}^{(rm)} \\
\tilde{\boldsymbol{\Sigma}}_x^{(rm)} &= \left(\boldsymbol{\Sigma}_x^{(m)-1} + \boldsymbol{\Sigma}_b^{(r)-1} \right)^{-1} \\
\tilde{\mathbf{A}}^{(rm)} &= \tilde{\boldsymbol{\Sigma}}_x^{(rm)} \boldsymbol{\Sigma}_b^{(r)-1} \mathbf{A}^{(r)} \\
\tilde{\mathbf{b}}^{(rm)} &= \tilde{\boldsymbol{\Sigma}}_x^{(rm)} \left(\boldsymbol{\Sigma}_x^{(m)-1} \boldsymbol{\mu}_x^{(m)} + \boldsymbol{\Sigma}_b^{(r)-1} \mathbf{b}^{(r)} \right)
\end{aligned}$$

where $\boldsymbol{\mu}_x^{(m)}$ and $\boldsymbol{\Sigma}_x^{(m)}$ are the mean and covariance of Gaussian component m. Given the expectations, the statistics can be accumulated in the standard fashion to re-estimate the acoustic model parameters. This method makes the implementation much simpler and hence has been used in this work.

7.3.2 Model update

Using the EM-based NAT, described above, it only involves minor changes in the original model estimation formula of the SGMMs presented in (Povey et al., 2011a). Taking the estimation of the Gaussian mean projection $\mathbf{M}_i$ for instance, the auxiliary function is

$$Q(\mathbf{M}_i) = tr\left(\mathbf{M}_i^T \boldsymbol{\Sigma}_i^{-1} \mathbf{Y}_i\right) - \frac{1}{2} tr\left(\mathbf{M}_i^T \boldsymbol{\Sigma}_i^{-1} \mathbf{M}_i \mathbf{Q}_i\right) \tag{7.8}$$

where the sufficient statistics $\mathbf{Y}_i$ and $\mathbf{Q}_i$ are now obtained as

$$\mathbf{Y}_i = \sum_{jkt} \gamma_{jki}(t) \mathbb{E}[\mathbf{x}_t | \mathbf{y}_t, r, m] \mathbf{v}_{jk}^T \tag{7.9}$$

$$\mathbf{Q}_i = \sum_{jkt} \gamma_{jki}(t) \mathbf{v}_{jk} \mathbf{v}_{jk}^T. \tag{7.10}$$

Note that in an SGMM, the Gaussian component index m is replaced by jki as in (6.30), and the regression class index r is replaced by i. It also worth emphasising that the posterior probability $\gamma_{jki}(t)$ needs to be computed using the noisy feature vector $\mathbf{y}_t$ using the likelihood function (6.30) during the adaptive training phase.

Likewise, other types of SGMM acoustic model parameters such as $\mathbf{v}_{jk}$ and $\boldsymbol{\Sigma}_i$ can be estimated in the same fashion using the expectations of the "pseudo-clean" feature vectors. The EM-based algorithm for NAT is similar to some feature enhancement methods which also estimate $\mathbf{x}_t$ given $\mathbf{y}_t$, e.g. (Moreno et al., 1996). However, a fundamental difference is that the conditional expectations directly relate to the acoustic model structure as in (7.6) and (7.7), while for feature enhancement they are normally derived using a front-end GMM. Due to the closer match to the acoustic model, NAT was found to outperform its feature enhancement counterpart in (Li et al., 2012).

Finally, it is worthwhile to point out that the UBM associated with the SGMM acoustic model also needs to be updated during adaptive training. After NAT, the SGMM models the "pseudo-clean" features $\mathbf{x}_t$, while the UBM is originally trained on the noise-corrupted features $\mathbf{y}_t$. Since the UBM provides the regression class for the Gaussian components when applying JUD (Lu et al., 2013a), it needs to be in the same space as the SGMM. In this work, the UBM is updated using the weighted average of

the corresponding Gaussian component in the SGMM as

$$\boldsymbol{\Sigma}_i^{ubm} = \boldsymbol{\Sigma}_i \tag{7.11}$$

$$w_i^{ubm} = \frac{\sum_{jkt} \gamma_{jki}(t)}{\sum_{jkit} \gamma_{jki}(t)} \tag{7.12}$$

$$\boldsymbol{\mu}_i^{ubm} = \sum_{jkt} \frac{\gamma_{jki}(t)}{\sum_{jkt} \gamma_{jki}(t)} \mathbf{M}_i \mathbf{v}_{jk} \tag{7.13}$$

where w_i^{ubm}, $\boldsymbol{\mu}_i^{ubm}$ and $\boldsymbol{\Sigma}_i^{ubm}$ are the weight, mean and covariance matrix for component i in the UBM respectively. Updating the UBM was found to improve the recognition accuracy of the NAT system.

7.3.3 Training recipe

To sum up, the NAT recipe for an SGMM acoustic model used in this paper is as follows.

1. Initialise the acoustic model $\mathcal{M}$ by the standard maximum likelihood training.
2. For each training utterance, initialise the noise model parameters for $\mathbf{n}_t$ and $\mathbf{h}_t$ in (6.6).
3. Re-estimate the noise model parameters given $\mathcal{M}$.
4. Obtain the JUD transformation parameters $\mathcal{T}$.
5. Given $\mathcal{M}$ and $\mathcal{T}$, compute the posterior probability $\gamma_{jki}(t)$ using (6.30).
6. Accumulate the statistics using the conditional expectations (7.6) (7.7) and update $\mathcal{M}$.
7. Go to step 5 until convergence.
8. Update the UBM using equations (7.11) - (7.13).
9. Go to step 2 until the number of iterations is reached.

While this paper focuses on the NAT algorithm for the SGMMs, more details about noise model and JUD transform estimation used in step 2 to step 4 can be found in Chapter 6.

7.4 Experiments

As in Chapter 6, the experiments were performed on the Aurora 4 corpus. We used 39 dimensional feature vectors derived from 12th order mel frequency cepstral coefficients, plus the zeroth order coefficient (C0), with delta and acceleration features. Again, we used the standard WSJ0 5k bigram language model (Paul and Baker, 1992) and the CMU pronunciation dictionary. Same as in Chapter 6, the SGMM systems have about 3900 tied triphone states, 16,000 sub-states, and $I = 400$ Gaussians in the UBM, which results in 6.4 million surface Gaussians. As mentioned before, the phase-sensitive mismatch function (6.6) is used to estimate the JUD transforms. Based on the previous findings in Chapter 6, all the entries in $\boldsymbol{\alpha}$ are empirically set to 2.5 in both training and decoding stages unless otherwise specified.

7.4.1 Results

The experimental results are given in Table 7.1 using the clean, MST and NAT acoustic models. The NAT system is trained following the recipe in section 7.3.3, where we perform 4 iterations in step 7 which yields convergence, and only 1 iteration in step 9. As expected, the MST system is significantly more accurate than the clean trained system without JUD compensation since the mismatch between the training and testing data is reduced. However, with JUD compensation we observe that the clean model is more accurate than MST (16.8% vs. 17.6%). This may be due to the larger variability in the MST model making it less suitable for rapid adaptation towards a particular noise condition using limited adaptation data. The NAT system, on the other hand, normalises the irrelevant variability in the training data using noise dependent JUD transforms. Without JUD in the decoding stage, this model results in higher WER than MST, since it does not match the testing data well. With JUD adaptation, however, it is more accurate than the MST and clean systems with a WER of 15.7%, which is slightly better than the adaptively-trained GMM system using VTS on the same dataset (16.0%) (Flego and Gales, 2012).

Previous work on empirically tuning the phase factor $\boldsymbol{\alpha}$ in (6.6) has shown that it is able to bring significant gains in both VTS- and JUD-based noise robust speech recognition systems (Deng et al., 2004; Li et al., 2009; Lu et al., 2013a). Interpreted as a phase factor, the values of the elements of $\boldsymbol{\alpha}$ should be in the range $[-1, 1]$ (Deng et al., 2004). However, experimental studies have demonstrated that treating $\boldsymbol{\alpha}$ as additional model parameters tuned to mitigate the mismatch between the training and

Table 7.1: Word error rates (WERs) of SGMM systems with and without noise adaptive training.

Methods	A	B	C	D	Avg
Clean model	5.2	58.2	50.7	72.1	59.9
+JUD	5.1	13.1	12.0	23.2	16.8
MST model	6.8	15.2	18.6	32.3	22.2
+JUD	7.4	13.3	14.7	24.1	17.6
NAT model	6.5	20.3	19.8	39.7	27.6
+JUD	6.1	11.3	11.9	22.4	15.7

testing data (Gales and Flego, 2010) results in improved accuracy (Li et al., 2009; Lu et al., 2013a). While previous studies on this issue were mainly based on systems trained on clean data (Li et al., 2009; Lu et al., 2013a), we see similar trends with our MST and NAT systems. Figure 7.1 shows the WER of the systems using the three models by empirically tuning the values of $\boldsymbol{\alpha}$ in the decoding stage as in (Li et al., 2009; Lu et al., 2013a). It shows that tuning the value of $\boldsymbol{\alpha}$ results in gains for all the three systems, e.g. 15.5% ($\boldsymbol{\alpha} = \mathbf{2.0}$) vs. 17.0% ($\boldsymbol{\alpha} = \mathbf{0}$) for the NAT system. However, compared to the MST and NAT systems that are trained on multi-condition data, the improvement is much larger for the highly mismatched system that is trained on clean data, e.g. 16.8% ($\boldsymbol{\alpha} = \mathbf{2.0}$) vs. 20.3% ($\boldsymbol{\alpha} = \mathbf{0}$). These results support the previous argument that tuning $\boldsymbol{\alpha}$ may help to reduce the mismatch between the training and testing conditions. Note that, the results were obtained by tuning $\boldsymbol{\alpha}$ in the decoding phase only; future work will investigate the effect of $\boldsymbol{\alpha}$ on the training stage for NAT system.

7.5 Conclusions

We have investigated the noise adaptive training (NAT) algorithm for an SGMM acoustic model using multi-condition training data. Our method is based on the joint uncertainty decoding (JUD) noise compensation technique. For adaptive training, an EM-based optimisation algorithm is employed which is derived from reformulating JUD adaptation into a generative model. This algorithm has proven to be simple for implementation, and effective in terms of recognition accuracy. Evaluation was carried out using the Aurora 4 dataset; using NAT, the SGMM system achieved the lowest

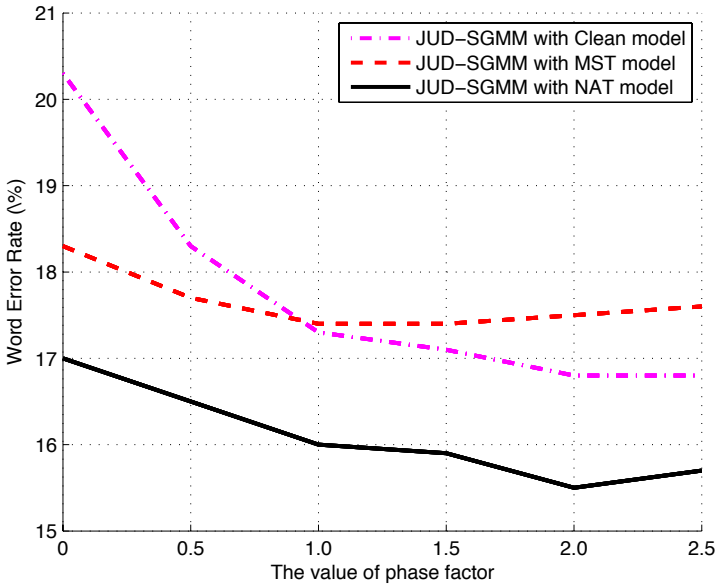


Figure 7.1: Results of tuning the value of phase factor $\boldsymbol{\alpha}$ in the decoding stage.

WER (15.5%) which is considerably better than systems without adaptive training. These experiments are also helpful to understand the effect of phase factor parameter in the mismatch function. Future work will be on applying a discriminative criterion to the adaptively trained system that has been found effective with GMM based systems (Flego and Gales, 2009; Gales and Flego, 2010).

Chapter 8

Conclusion

This thesis investigates the subspace Gaussian mixture model (SGMM) for automatic speech recognition. This type of model differs from the conventional GMM/HMM system for speech recognition in that the state dependent GMM parameters are derived from globally shared model subspace and low-dimensional state-dependent vectors. One of the benefits is that the total number of model parameters may be reduced which makes the model more efficient for low-resource speech recognition task. In addition, acoustic factorisation can be performed by using separated model subspaces which leads to more elegant acoustic modelling and higher recognition accuracy. For instance, phonetic and speaker factors can be modelled in the current framework. Chapter 3 presents an overview of this model.

Chapter 4 describes the regularized model estimation for SGMMs that we proposed in (Lu et al., 2011b), where a regularization penalty was introduced to the maximum likelihood objective function in order to avoid the model overfitting. In this work, regularized estimation was employed on the state vectors, and there different regularization penalties — ℓ_1-norm, ℓ_2-norm penalty, as well as their combined form, the elastic net penalty — were investigated. Experimental results indicate that the ℓ_1-norm penalty leads to better performance in terms of recognition accuracy and model robustness. Regularization is particularly effective when the amount of training data is very limited, as was demonstrated in the experiments in Chapter 5, where significant gains was achieved by using ℓ_1-norm regularization in the cross-lingual systems.

A typical feature of SGMM acoustic model is that a large proportion of model parameters are globally shared, which do not depend on the HMM topology. This property is particularly useful for cross-lingual settings, in which, the globally shared parameters are estimated from source language systems which are data rich. These pa-

rameters can be reused in the target language system and reduce the amount of training data that is required to train the model. In Chapter 5, a comprehensive investigation of using SGMMs for cross-lingual speech recognition is presented including multilingual parameter tying, regularization and cross-lingual model adaptation and speaker adaptive training. This method can achieve excellent results for low-resource task. For instance, our experiments on the GlobalPhone shown that, the cross-lingual SGMM system obtained 26.7% WER with only 1 hour training data, significantly outperformed the GMM and SGMM baseline with WER as 34.1% and 31.4% respectively.

Chapter 6 addresses the issue of noise robustness for SGMM acoustic model using joint uncertainty decoding (JUD) technique. In JUD, the noise compensation parameters are shared among Gaussian components within the same regression class instead of per component noise compensation. This significantly saves the computational cost with only slightly sacrifice on the recognition accuracy. In Chapter 6, JUD with vector Taylor series (VTS) approximation was implemented and studied for SGMM acoustic models. Experimental results on the Aurora 4 corpus indicate that JUD/SGMM system can achieve the state-of-the-art performance with 16.8% WER which is comparable to the conventional VTS/GMM system. We also investigated the unscented transform (UT) approximation with JUD instead of VTS for noise-robust SGMMs. It was found that UT results in higher recognition accuracy than VTS when the phase term between noise and speech was not considered. After introducing the phase term, UT and VTS achieved the same recognition accuracy on the Aurora 4 task.

Chapter 7 investigates the noise adaptive training (NAT) algorithm based JUD for SGMMs. By normalising the noise variability in the multi-style training data, NAT achieves more than 1% absolute WER reduction on the Aurora 4 dataset. As a highly structured model, SGMM provides much room for innovations for future work. Following the work in this thesis. the promising future research directions within the SGMM framework include

- Multilingual speaker adaptive training

  The aim of multilingual speech recognition is hard to achieve mainly because of the mismatch in phone units between languages. However, the SGMM acoustic model can avoid this issue since the globally shared parameters do not depend on the HMM topology and therefore, these parameters can be reused among languages or estimated by multilingual data. Previous work on multilingual SGMM acoustic models does not consider the speaker variance inter- and intra-language (Burget et al., 2010). We expect further gains maybe achieved by performing

multilingual speaker adaptive training to normalise the variance and make the system robust against the mismatch among the corpus of different languages. This can be done by tying the speaker subspace in SGMMs or the speaker linear transforms across multilingual systems and estimate these parameters by multilingual data.

- Log-spectral domain noise compensation
  Log-spectral domain noise compensation aims at getting rid of the DCT and inverse DCT transforms in the mismatch function (6.6). This can be simply done by using log-spectral features rather than cesptral features as MFCCs. This leads to several benefits. For instance, the mismatch function is simplified and may lead to more accurate approximation by either VTS or UT. Furthermore, the Jacobian matrix is diagonal which will results in significant reduction in terms of the computational cost. However, this idea is hard to employ in conventional GMM based acoustic models using diagonal covariance matrices which require the features to be de-correlated by the DCT. An SGMM does not have this constraint since full covariance matrices can be used. This makes the idea feasible to be explored.

- Speaker and noise factorisation

  Chapter 6 and 7 explores noise compensation and adaptive training for an SGMM acoustic model. Though good results have been obtained, the speaker-dependent variability is not considered, which is another one of the major impacts that affect ASR systems. Speaker and noise factorisation has been investigated with standard GMM-based system (Wang and Gales, 2011; Seltzer and Acero, 2011), and have shown promising results. It is interesting to explore this topic with SGMM acoustic models which is more structured. In addition, some unsupervised methods may be worth looking at, which is inspired from our results on unsupervised noise compensation in Chapter 6.

Appendix A

MAP update of phone projection matrix

Here, we present the detailed analytical solution of the MAP estimate of subspace parameters with Gaussian prior by solving the following auxiliary function:

$$\tilde{Q}(\mathbf{M}_i) \propto tr\Big(\mathbf{M}_i^T \boldsymbol{\Sigma}_i^{-1} \mathbf{Y}_i + \tau \mathbf{M}_i^T \boldsymbol{\Omega}_r^{-1} \bar{\mathbf{M}}_i \boldsymbol{\Omega}_c^{-1}\Big) - \frac{1}{2} tr\Big(\boldsymbol{\Sigma}_i^{-1} \mathbf{M}_i \mathbf{Q}_i \mathbf{M}_i^T + \tau \boldsymbol{\Omega}_r^{-1} \mathbf{M}_i \boldsymbol{\Omega}_c^{-1} \mathbf{M}_i^T\Big). \quad \text{(A.1)}$$

The original formulation is given by Povey (Povey, 2009) (App. J), and we summarize the main ideas.

The solution is not readily available by taking the derivative of $\tilde{Q}(\mathbf{M}_i)$ with respect to $\mathbf{M}_i$ and setting it to be zero. Instead, we introduce an intermediate transform $\mathbf{T} = \mathbf{U}^T \mathbf{L}^{-1}$ that simultaneously diagonalises $\boldsymbol{\Sigma}_i^{-1}$ and $\boldsymbol{\Omega}_r^{-1}$, where

$$\boldsymbol{\Sigma}_i^{-1} = \mathbf{L}\mathbf{L}^T \qquad \text{(Cholesky decomposition)}, \quad \text{(A.2)}$$

$$\mathbf{S} = \mathbf{L}^{-1} \boldsymbol{\Omega}_r^{-1} \mathbf{L}^{-T}, \quad \text{(A.3)}$$

$$\mathbf{S} = \mathbf{U} \Lambda \mathbf{U}^T \qquad \text{(Eigenvalue decomposition)}. \quad \text{(A.4)}$$

It is the case that $\mathbf{T}\boldsymbol{\Sigma}_i^{-1}\mathbf{T} = \mathbf{I}$ and $\mathbf{T}\boldsymbol{\Omega}_r^{-1}\mathbf{T} = \boldsymbol{\Lambda}$, where $\mathbf{I}$ is the identity matrix, and $\boldsymbol{\Lambda}$ is a diagonal matrix holding the eigenvalues of matrix $\mathbf{S}$. If we further define $\mathbf{M}_i = \mathbf{T}^T \mathbf{M}_i'$, then equation (A.1) can be rewritten as

$$\tilde{Q}(\mathbf{M}_i') \propto tr\Big(\mathbf{M}_i'^T \mathbf{T}\big(\boldsymbol{\Sigma}_i^{-1} \mathbf{Y}_i + \tau \boldsymbol{\Omega}_r^{-1} \bar{\mathbf{M}}_i \boldsymbol{\Omega}_c^{-1}\big)\Big) - \frac{1}{2} tr\Big(\mathbf{M}_i' \mathbf{Q}_i \mathbf{M}_i'^T + \tau \boldsymbol{\Lambda} \mathbf{M}_i' \boldsymbol{\Omega}_c^{-1} \mathbf{M}_i'^T\Big). \quad \text{(A.5)}$$

Now we can take the derivative of $\tilde{Q}(\mathbf{M}_i^{'})$ with respect to $\mathbf{M}_i^{'}$:

$$\frac{\partial \tilde{Q}(\mathbf{M}_i^{'})}{\partial \mathbf{M}_i^{'}} = \mathbf{T}\left(\mathbf{\Sigma}_i^{-1}\mathbf{Y}_i + \tau\mathbf{\Omega}_r^{-1}\bar{\mathbf{M}}_i\mathbf{\Omega}_c^{-1}\right) - \mathbf{M}_i^{'}\mathbf{Q}_i - \tau\mathbf{\Lambda}\mathbf{M}_i^{'}\mathbf{\Omega}_c^{-1}. \tag{A.6}$$

Setting this derivative to be zero, we obtain the row by row solution of $\mathbf{M}_i^{'}$ as

$$\mathbf{m}_n^{'} = \mathbf{g}_n\left(\mathbf{Q}_i + \tau\lambda_n\mathbf{\Omega}_c^{-1}\right)^{-1}, \tag{A.7}$$

where $\mathbf{m}_n^{'}$ is the n_{th} row of $\mathbf{M}_i^{'}$, λ_n is the n_{th} diagonal element of $\mathbf{\Lambda}$, and $\mathbf{g}_n$ is the n_{th} row of matrix $\mathbf{T}\left(\mathbf{\Sigma}_i^{-1}\mathbf{Y}_i + \tau\mathbf{\Omega}_r^{-1}\bar{\mathbf{M}}_i\mathbf{\Omega}_c^{-1}\right)$. The final solution of $\mathbf{M}_i$ can then be obtained by $\mathbf{M}_i = \mathbf{T}^T\mathbf{M}_i^{'}$. The results of using $\mathbf{\Omega}_r = \mathbf{\Omega}_c = \mathbf{I}$ are also readily available.

Appendix B

Update the additive and channel noise mean

The following derivations are for the static features; the delta and acceleration coefficients may be obtained using a continuous time approximation, as discussed in Chapter 6 (Section 6.2), in which the static and dynamic coefficients are assumed to be independent. Likewise, the JUD transforms $(\mathbf{A}^{(i)}, \mathbf{b}^{(i)}, \boldsymbol{\Sigma}_b^{(i)})$ in these derivations correspond to the static coefficients only.

We denote the clean and noisy UBM models as $\{\boldsymbol{\mu}_x^{(i)}, \boldsymbol{\Sigma}_x^{(i)}; i = 1, \ldots, I\}$, and $\{\boldsymbol{\mu}_y^{(i)}, \boldsymbol{\Sigma}_y^{(i)}; i = 1, \ldots, I\}$, respectively. As stated before, the derivations here are similar to the VTS noise model estimation (Liao (2007), Chapter 4), but with a different accumulation of statistics for the SGMM. We use a similar notations to (Liao, 2007) in order to make clear the relations and difference between the two. We first rewrite the auxiliary function (6.31) for the static coefficients as

$$\begin{aligned} \mathcal{Q}(\cdot) &= \sum_{jkit} \gamma_{jki}(t) \left[\log \mathcal{N}\left(\mathbf{y}_t; \mathbf{A}^{(i)-1} \left(\boldsymbol{\mu}_{jki} - \boldsymbol{\mu}_x^{(i)} \right) + \boldsymbol{\mu}_y^{(i)}, \tilde{\boldsymbol{\Sigma}}_y^{(i)} \right) \right] \\ &= -\frac{1}{2} \sum_{jkit} \gamma_{jki}(t) \left(\log |\tilde{\boldsymbol{\Sigma}}_y^{(i)}| + \tilde{\mathbf{y}}_{jkit}^T \tilde{\boldsymbol{\Sigma}}_y^{(i)-1} \tilde{\mathbf{y}}_{jkit} \right) \end{aligned} \tag{B.1}$$

where

$$\tilde{\mathbf{y}}_{jkit} = \left(\mathbf{y}_t - \mathbf{A}^{(i)-1} \left(\boldsymbol{\mu}_{jki} - \boldsymbol{\mu}_x^{(i)} \right) - \boldsymbol{\mu}_y^{(i)} \right), \tag{B.2}$$

$$\begin{aligned} \tilde{\boldsymbol{\Sigma}}_y^{(i)} &= \mathbf{A}^{(i)-1} \left(\boldsymbol{\Sigma}_i + \boldsymbol{\Sigma}_b^{(i)} \right) \mathbf{A}^{(i)-T} \\ &= \boldsymbol{\Sigma}_y^{(i)} + \mathbf{A}^{(i)-1} \left(\boldsymbol{\Sigma}_i - \boldsymbol{\Sigma}_x^{(i)} \right) \mathbf{A}^{(i)-T}. \end{aligned} \tag{B.3}$$

To update the noise model, we first fix the VTS expansion point, so that the Jacobian matrices are also fixed, and $\boldsymbol{\mu}_y^{(i)}$ is a function of the additive and channel noise

means, $\boldsymbol{\mu}_n$ and $\boldsymbol{\mu}_h$, only. Using the first order VTS expansion around the old noise model parameters $(\check{\boldsymbol{\mu}}_n, \check{\boldsymbol{\mu}}_h)$, $\boldsymbol{\mu}_y^{(i)}$ can be expressed as

$$\begin{aligned}\boldsymbol{\mu}_y^{(i)} &\approx \mathbb{E}\Big\{ f\left(\boldsymbol{\mu}_x^{(i)}, \check{\boldsymbol{\mu}}_n, \check{\boldsymbol{\mu}}_h, \boldsymbol{\alpha}\right) + \mathbf{G}_x^{(i)}\left(\mathbf{x} - \boldsymbol{\mu}_x^{(i)}\right) + \mathbf{G}_n^{(i)}\left(\mathbf{n} - \check{\boldsymbol{\mu}}_n\right) + \mathbf{G}_h^{(i)}\left(\mathbf{h} - \check{\boldsymbol{\mu}}_h\right) \Big\} \\ &= \check{\boldsymbol{\mu}}_y^{(i)} + \mathbf{G}_n^{(i)}\left(\hat{\boldsymbol{\mu}}_n - \check{\boldsymbol{\mu}}_n\right) + \mathbf{G}_h^{(i)}\left(\hat{\boldsymbol{\mu}}_h - \check{\boldsymbol{\mu}}_h\right), \end{aligned} \tag{B.4}$$

where $\mathbf{G}_x^{(i)}$ and $\mathbf{G}_n^{(i)}$ are the Jacobian matrices (6.20) and (6.21). $\mathbf{G}_h^{(i)}$ is defined as

$$\mathbf{G}_h^{(i)} = \frac{\partial f(\cdot)}{\partial \check{\boldsymbol{\mu}}_h}\Big|_{\boldsymbol{\mu}_x^{(i)}, \check{\boldsymbol{\mu}}_h, \check{\boldsymbol{\mu}}_n}. \tag{B.5}$$

Taking the derivative of $Q(\cdot)$ w.r.t. $\hat{\boldsymbol{\mu}}_n$, we obtain

$$\begin{aligned}\frac{\partial Q(\cdot)}{\partial \hat{\boldsymbol{\mu}}_n} &= \sum_{jkit} \gamma_{jki}(t) \mathbf{G}_n^{(i)T} \tilde{\boldsymbol{\Sigma}}_y^{(i)-1} \tilde{\mathbf{y}}_{jkit} \\ &= \sum_{jkit} \gamma_{jki}(t) \mathbf{G}_n^{(i)T} \tilde{\boldsymbol{\Sigma}}_y^{(i)-1} \Big(\mathbf{y}_t - \mathbf{A}^{(i)-1}\left(\boldsymbol{\mu}_{jki} - \boldsymbol{\mu}_x^{(i)}\right) \\ &\quad - \check{\boldsymbol{\mu}}_y^{(i)} - \mathbf{G}_n^{(i)}\left(\hat{\boldsymbol{\mu}}_n - \check{\boldsymbol{\mu}}_n\right) - \mathbf{G}_h^{(i)}\left(\hat{\boldsymbol{\mu}}_h - \check{\boldsymbol{\mu}}_h\right) \Big) \\ &= \mathbf{d} - \mathbf{E}\hat{\boldsymbol{\mu}}_n - \mathbf{F}\hat{\boldsymbol{\mu}}_h, \end{aligned} \tag{B.6}$$

where

$$\mathbf{d} = \sum_{jkit} \gamma_{jki}(t) \mathbf{G}_n^{(i)T} \tilde{\boldsymbol{\Sigma}}_y^{(i)-1} \Big(\mathbf{y}_t - \mathbf{A}^{(i)-1}\left(\boldsymbol{\mu}_{jki} - \boldsymbol{\mu}_x^{(i)}\right) - \check{\boldsymbol{\mu}}_y^{(i)} + \mathbf{G}_n^{(i)} \check{\boldsymbol{\mu}}_n + \mathbf{G}_h^{(i)} \check{\boldsymbol{\mu}}_h \Big), \tag{B.7}$$

$$\mathbf{E} = \sum_i \gamma_i \mathbf{G}_n^{(i)T} \tilde{\boldsymbol{\Sigma}}_y^{(i)-1} \mathbf{G}_n^{(i)}, \tag{B.8}$$

$$\mathbf{F} = \sum_i \gamma_i \mathbf{G}_n^{(i)T} \tilde{\boldsymbol{\Sigma}}_y^{(i)-1} \mathbf{G}_h^{(i)}, \tag{B.9}$$

and $\gamma_i = \sum_{jkt} \gamma_{jki}(t)$. Similarly, taking the derivative of $Q(\cdot)$ w.r.t. $\hat{\boldsymbol{\mu}}_h$ gives

$$\begin{aligned}\frac{\partial Q(\cdot)}{\partial \hat{\boldsymbol{\mu}}_h} &= \sum_{jkit} \gamma_{jki}(t) \mathbf{G}_h^{(i)T} \tilde{\boldsymbol{\Sigma}}_y^{(i)-1} \tilde{\mathbf{y}}_{jkit} \\ &= \mathbf{u} - \mathbf{V}\hat{\boldsymbol{\mu}}_n - \mathbf{W}\hat{\boldsymbol{\mu}}_h, \end{aligned} \tag{B.10}$$

where

$$\mathbf{u} = \sum_{jkit} \gamma_{jki}(t) \mathbf{G}_h^{(i)T} \tilde{\boldsymbol{\Sigma}}_y^{(i)-1} \Big(\mathbf{y}_t - \mathbf{A}^{(i)-1}\left(\boldsymbol{\mu}_{jki} - \boldsymbol{\mu}_x^{(i)}\right) - \check{\boldsymbol{\mu}}_y^{(i)} + \mathbf{G}_n^{(i)} \check{\boldsymbol{\mu}}_n - \mathbf{G}_h^{(i)} \check{\boldsymbol{\mu}}_h \Big), \tag{B.11}$$

$$\mathbf{V} = \sum_i \gamma_i \mathbf{G}_h^{(i)T} \tilde{\boldsymbol{\Sigma}}_y^{(i)-1} \mathbf{G}_n^{(i)}, \tag{B.12}$$

$$\mathbf{W} = \sum_i \gamma_i \mathbf{G}_h^{(i)T} \tilde{\boldsymbol{\Sigma}}_y^{(i)-1} \mathbf{G}_h^{(i)}. \tag{B.13}$$

Setting the two derivatives to be zero we obtain

$$\begin{bmatrix} \mathbf{E} & \mathbf{F} \\ \mathbf{V} & \mathbf{W} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{\mu}}_n \\ \hat{\boldsymbol{\mu}}_h \end{bmatrix} = \begin{bmatrix} \mathbf{d} \\ \mathbf{u} \end{bmatrix}, \tag{B.14}$$

which gives

$$\begin{bmatrix} \hat{\boldsymbol{\mu}}_n \\ \hat{\boldsymbol{\mu}}_h \end{bmatrix} = \begin{bmatrix} \mathbf{E} & \mathbf{F} \\ \mathbf{V} & \mathbf{W} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{d} \\ \mathbf{u} \end{bmatrix}. \tag{B.15}$$

In our implementation, we cached the statistics that are independent of the noise parameter so that they were not computed repeatedly. For instance, $\mathbf{d}$ can be decomposed as follows:

$$\mathbf{d} = \sum_i \gamma_i \mathbf{G}_n^{(i)T} \tilde{\boldsymbol{\Sigma}}_y^{(i)-1} \left(\mathbf{G}_n^{(i)} \breve{\boldsymbol{\mu}}_n + \mathbf{G}_h^{(i)} \breve{\boldsymbol{\mu}}_h - \breve{\boldsymbol{\mu}}_y^{(i)} \right) + \sum_i \mathbf{G}_n^{(i)T} \tilde{\boldsymbol{\Sigma}}_y^{(i)-1} \underbrace{\sum_{jkt} \boldsymbol{\gamma}_{jki}(t) \mathbf{y}_t}_{\text{cached}}$$

$$- \sum_i \mathbf{G}_n^{(i)T} \tilde{\boldsymbol{\Sigma}}_y^{(i)-1} \mathbf{A}^{(i)-1} \underbrace{\sum_{jkt} \boldsymbol{\gamma}_{jki}(t) \left(\boldsymbol{\mu}_{jki} - \boldsymbol{\mu}_x^{(i)} \right)}_{\text{cached}}$$

Caching was also used for the computation of $\mathbf{u}$ in (B.11).

Appendix C

Update the additive noise variance

The derivation here is similar to the estimation of the additive noise variance $\mathbf{\Sigma}_n$ for VTS (Liao (2007), App. C). To update $\mathbf{\Sigma}_n$, we first fix the value of $\boldsymbol{\mu}_n$ and $\boldsymbol{\mu}_h$. Again, the derivations are for static features only. For the dynamic coefficients of $\mathbf{\Sigma}_n$, the derivations are similar. We rewrite the auxiliary function (B.1):

$$Q(\cdot) = -\frac{1}{2}\sum_{jkit}\gamma_{jki}(t)\left(\log|\tilde{\mathbf{\Sigma}}_y^{(i)}| + \tilde{\mathbf{y}}_{jkit}^T\tilde{\mathbf{\Sigma}}_y^{(i)-1}\tilde{\mathbf{y}}_{jkit}\right) \tag{C.1}$$

where $\tilde{\mathbf{y}}_{jkit}$ and $\tilde{\mathbf{\Sigma}}_y^{(i)}$ are defined in (B.2) and (B.3). Note that that $\tilde{\mathbf{\Sigma}}_y^{(i)}$ is full rather than diagonal (unlike Liao (2007)). Therefore, the derivations are slightly different. Since $\tilde{\mathbf{y}}_{jkit}$ does not depend on $\mathbf{\Sigma}_n$, by taking derivative $Q(\cdot)$ w.r.t. to the d^{th} diagonal element of $\mathbf{\Sigma}_n$, we obtain:

$$\frac{\partial Q(\cdot)}{\partial\sigma_{n,d}^2} = -\frac{1}{2}\sum_{jkit}\gamma_{jki}(t)\left[\underbrace{\frac{\partial\log|\tilde{\mathbf{\Sigma}}_y^{(i)}|}{\partial\sigma_{n,d}^2}}_{\text{first part}} + \underbrace{\tilde{\mathbf{y}}_{jkit}^T\frac{\partial\tilde{\mathbf{\Sigma}}_y^{(i)-1}}{\partial\sigma_{n,d}^2}\tilde{\mathbf{y}}_{jkit}}_{\text{second part}}\right] \tag{C.2}$$

The first part of the derivative is

$$\frac{\partial\log|\tilde{\mathbf{\Sigma}}_y^{(i)}|}{\partial\sigma_{n,d}^2} = \frac{1}{|\tilde{\mathbf{\Sigma}}_y^{(i)}|}\frac{\partial|\tilde{\mathbf{\Sigma}}_y^{(i)}|}{\partial\sigma_{n,d}^2} = Tr\left(\tilde{\mathbf{\Sigma}}_y^{(i)-1}\frac{\partial\tilde{\mathbf{\Sigma}}_y^{(i)}}{\partial\sigma_{n,d}^2}\right). \tag{C.3}$$

It can be seen from (B.3) that only $\mathbf{\Sigma}_y^{(i)}$ depends on $\sigma_{n,d}^2$, and from (6.23)

$$\mathbf{\Sigma}_y^{(i)} = \mathbf{G}_x^{(i)}\mathbf{\Sigma}_x^{(i)}\mathbf{G}_x^{(i)T} + \mathbf{G}_n^{(i)}\mathbf{\Sigma}_n\mathbf{G}_n^{(i)T}.$$

Hence

$$\frac{\partial\tilde{\mathbf{\Sigma}}_y^{(i)}}{\partial\sigma_{n,d}^2} = \frac{\partial\mathbf{\Sigma}_y^{(i)}}{\partial\sigma_{n,d}^2} = [\mathbf{G}_n^{(i)}]_d[\mathbf{G}_n^{(i)}]_d^T, \tag{C.4}$$

where $[\mathbf{G}_n^{(i)}]_d$ denotes the d^{th} column of $\mathbf{G}_n^{(i)}$. Substituting (C.4) into (C.3), we obtain the first part of the derivative as

$$\begin{aligned}\kappa_{id} \equiv \frac{\partial \log |\tilde{\mathbf{\Sigma}}_y^{(i)}|}{\partial \sigma_{n,d}^2} &= Tr\left(\tilde{\mathbf{\Sigma}}_y^{(i)-1}[\mathbf{G}_n^{(i)}]_d[\mathbf{G}_n^{(i)}]_d^T\right) \\ &= [\mathbf{G}_n^{(i)}]_d^T\tilde{\mathbf{\Sigma}}_y^{(i)-1}[\mathbf{G}_n^{(i)}]_d \end{aligned} \tag{C.5}$$

Similarly, for the second part of the derivative

$$\begin{aligned}\frac{\partial \tilde{\mathbf{\Sigma}}_y^{(i)-1}}{\partial \sigma_{n,d}^2} &= -\tilde{\mathbf{\Sigma}}_y^{(i)-1}\frac{\partial \tilde{\mathbf{\Sigma}}_y^{(i)}}{\partial \sigma_{n,d}^2}\tilde{\mathbf{\Sigma}}_y^{(i)-1} \\ &= -\tilde{\mathbf{\Sigma}}_y^{(i)-1}[\mathbf{G}_n^{(i)}]_d[\mathbf{G}_n^{(i)}]_d^T\tilde{\mathbf{\Sigma}}_y^{(i)-1} \end{aligned} \tag{C.6}$$

Hence, we can compute it as

$$\begin{aligned}\sum_{jkit}\gamma_{jki}(t)\tilde{\mathbf{y}}_{jkit}^T\frac{\partial \tilde{\mathbf{\Sigma}}_y^{(i)-1}}{\partial \sigma_{n,d}^2}\tilde{\mathbf{y}}_{jkit} &= -\sum_{jkit}\gamma_{jki}(t)\tilde{\mathbf{y}}_{jkit}^T\tilde{\mathbf{\Sigma}}_y^{(i)-1}[\mathbf{G}_n^{(i)}]_d[\mathbf{G}_n^{(i)}]_d^T\tilde{\mathbf{\Sigma}}_y^{(i)-1}\tilde{\mathbf{y}}_{jkit} \\ &= -\sum_{jkit}\gamma_{jki}(t)[\mathbf{G}_n^{(i)}]_d^T\tilde{\mathbf{\Sigma}}_y^{(i)-1}\tilde{\mathbf{y}}_{jkit}\tilde{\mathbf{y}}_{jkit}^T\tilde{\mathbf{\Sigma}}_y^{(i)-1}[\mathbf{G}_n^{(i)}]_d \\ &= -\sum_i[\mathbf{G}_n^{(i)}]_d^T\tilde{\mathbf{\Sigma}}_y^{(i)-1}\mathbf{\Omega}_i\tilde{\mathbf{\Sigma}}_y^{(i)-1}[\mathbf{G}_n^i]_d, \end{aligned}$$

where we have accumulated all the statistics indexed by j,k,t as

$$\mathbf{\Omega}_i = \sum_{jkt}\gamma_{jki}(t)\tilde{\mathbf{y}}_{jkit}\tilde{\mathbf{y}}_{jkit}^T. \tag{C.7}$$

Again, we decompose $\mathbf{\Omega}_i$ and cache the statistics that do not depend on the noise parameters in order to save the computation, but we omit the details here for brevity. If we denote

$$\beta_{id} \equiv [\mathbf{G}_n^{(i)}]_d^T\tilde{\mathbf{\Sigma}}_y^{(i)-1}\mathbf{\Omega}_i\tilde{\mathbf{\Sigma}}_y^{(i)-1}[\mathbf{G}_n^i]_d, \tag{C.8}$$

then the gradient can be expressed as

$$\frac{\partial Q(\cdot)}{\partial \sigma_{n,d}^2} = -\frac{1}{2}\sum_{i=1}^{I}(\gamma_i\kappa_{id} - \beta_{id}). \tag{C.9}$$

As stated before, to enforce positivity, the logarithm of the variance is estimated by Li et al. (2009):

$$\tilde{\sigma}_{n,d}^2 = \log(\sigma_{n,d}^2), \tag{C.10}$$

$$\sigma_{n,d}^2 = \exp(\tilde{\sigma}_{n,d}^2). \tag{C.11}$$

Then we can obtain

$$\frac{\partial Q(\cdot)}{\partial \tilde{\sigma}^2_{n,d}} = \frac{\partial Q(\cdot)}{\partial \sigma^2_{n,d}} \frac{\partial \sigma^2_{n,d}}{\partial \tilde{\sigma}^2_{n,d}} = \frac{\partial Q(\cdot)}{\partial \sigma^2_{n,d}} \sigma^2_{n,d}$$
$$= -\frac{1}{2} \sum_{i=1}^{I} (\gamma_i \kappa_{id} - \beta_{id}) \sigma^2_{n,d}. \tag{C.12}$$

Next, we calculate the Hessian matrix of $Q(\cdot)$ w.r.t. the noise variance $\partial^2 Q(\cdot)/\partial(\sigma^2_{n,d})^2$. From the gradient (C.9) we can obtain:

$$\frac{\partial^2 Q(\cdot)}{\partial(\sigma^2_{n,d})^2} = \frac{\partial}{\partial \sigma^2_{n,d}} \left(\frac{\partial Q(\cdot)}{\partial \sigma^2_{n,d}} \right)$$
$$= \frac{\partial}{\partial \sigma^2_{n,d}} \left(-\frac{1}{2} \sum_{i=1}^{I} (\gamma_i \kappa_{id} - \beta_{id}) \right)$$
$$= -\frac{1}{2} \sum_{i=1}^{I} (\gamma_i \frac{\partial \kappa_{id}}{\partial \sigma^2_{n,d}} - \frac{\partial \beta_{id}}{\partial \sigma^2_{n,d}}). \tag{C.13}$$

From (C.5) and (C.6), we can obtain the first part of the derivative as:

$$\frac{\partial \kappa_{id}}{\partial \sigma^2_{n,d}} = [\mathbf{G}_n^{(i)}]_d^T \frac{\partial \tilde{\mathbf{\Sigma}}_y^{(i)-1}}{\partial \sigma^2_{n,d}} [\mathbf{G}_n^{(i)}]_d$$
$$= -\underbrace{[\mathbf{G}_n^{(i)}]_d^T \tilde{\mathbf{\Sigma}}_y^{(i)-1} [\mathbf{G}_n^{(i)}]_d}_{\kappa_{id}} \underbrace{[\mathbf{G}_n^{(i)}]_d^T \tilde{\mathbf{\Sigma}}_y^{(i)-1} [\mathbf{G}_n^{(i)}]_d}_{\kappa_{id}}$$
$$= -\kappa^2_{id}. \tag{C.14}$$

Using (C.8) and (C.6), we can compute the second part of the derivative $\partial \beta_{id}/\partial \sigma^2_{n,d}$:

$$\frac{\partial \beta_{id}}{\partial \sigma^2_{n,d}} = \frac{\partial \left([\mathbf{G}_n^{(i)}]_d^T \tilde{\mathbf{\Sigma}}_y^{(i)-1} \mathbf{\Omega}_i \tilde{\mathbf{\Sigma}}_y^{(i)-1} [\mathbf{G}_n^{(i)}]_d \right)}{\partial \sigma^2_{n,d}}$$
$$= [\mathbf{G}_n^{(i)}]_d^T \frac{\partial \tilde{\mathbf{\Sigma}}_y^{(i)-1}}{\partial \sigma^2_{n,d}} \mathbf{\Omega}_i \tilde{\mathbf{\Sigma}}_y^{(i)-1} [\mathbf{G}_n^{(i)}]_d + [\mathbf{G}_n^{(i)}]_d^T \tilde{\mathbf{\Sigma}}_y^{(i)-1} \mathbf{\Omega}_i \frac{\partial \tilde{\mathbf{\Sigma}}_y^{(i)-1}}{\partial \sigma^2_{n,d}} [\mathbf{G}_n^{(i)}]_d$$
$$= -\underbrace{[\mathbf{G}_n^{(i)}]_d^T \tilde{\mathbf{\Sigma}}_y^{(i)-1} [\mathbf{G}_n^{(i)}]_d}_{\kappa_{id}} \underbrace{[\mathbf{G}_n^{(i)}]_d^T \tilde{\mathbf{\Sigma}}_y^{(i)-1} \mathbf{\Omega}_i \tilde{\mathbf{\Sigma}}_y^{(i)-1} [\mathbf{G}_n^{(i)}]_d}_{\beta_{id}} \tag{C.15}$$
$$- \underbrace{[\mathbf{G}_n^{(i)}]_d^T \tilde{\mathbf{\Sigma}}_y^{(i)-1} \mathbf{\Omega}_i \tilde{\mathbf{\Sigma}}_y^{(i)-1} [\mathbf{G}_n^{(i)}]_d}_{\beta_{id}} \underbrace{[\mathbf{G}_n^{(i)}]_d^T \tilde{\mathbf{\Sigma}}_y^{(i)-1} [\mathbf{G}_n^{(i)}]_d}_{\kappa_{id}}$$
$$= -2\kappa_{id}\beta_{id}.$$

By summing the two parts we obtain

$$\frac{\partial^2 Q(\cdot)}{\partial(\sigma^2_{n,d})^2} = -\frac{1}{2} \sum_{i=1}^{I} \left(2\kappa_{id}\beta_{id} - \gamma_i \kappa^2_{id} \right), \tag{C.16}$$

and the Hessian of the logarithm of the variance can be estimated as

$$\begin{aligned}\frac{\partial^2 Q(\cdot)}{\partial(\tilde{\sigma}^2_{n,d})^2} &= \frac{\partial}{\partial\tilde{\sigma}^2_{n,d}}\left(\frac{\partial Q(\cdot)}{\partial\tilde{\sigma}^2_{n,d}}\right)\\ &= \frac{\partial Q(\cdot)}{\partial\sigma^2_{n,d}}\sigma^2_{n,d} + \frac{\partial^2 Q(\cdot)}{\partial(\sigma^2_{n,d})^2}\sigma^2_{n,d}\sigma^2_{n,d}\\ &= -\frac{1}{2}\sum_{i=1}^{I}\left(\gamma_i\kappa_{id} - \beta_{id}\right)\sigma^2_{n,d} - \frac{1}{2}\sum_{i=1}^{I}\left(2\kappa_{id}\beta_{id} - \gamma_i\kappa^2_{id}\right)\sigma^2_{n,d}\sigma^2_{n,d}. \end{aligned} \tag{C.17}$$

After obtaining the gradient (C.12) and Hessian (C.17) of the logarithm of the variance $\tilde{\sigma}^2_{n,d}$ we can update the estimate similar to (C.18) as

$$\hat{\tilde{\sigma}}^2_{n,d} = \tilde{\sigma}^2_{n,d} - \zeta\left(\frac{\partial^2 Q(\cdot)}{\partial(\tilde{\sigma}^2_{n,d})^2}\right)^{-1}\left(\frac{\partial Q(\cdot)}{\partial\tilde{\sigma}^2_{n,d}}\right). \tag{C.18}$$

Then we use equation (C.11) to compute the original variance.

Bibliography

Acero, A. (1990). *Acoustic and Enviromental Robustness in Automatic Speech Recognition*. PhD thesis, Carnegie Mellon University.

Anastasakos, Y., McDonough, J., Schwartz, R., and Makhoul, J. (1996). A compact model for speaker-adaptive training. In *ICSLP Proceedings*.

Aradilla, G., Bourlard, H., and Doss, M. (2008). Using KL-based acoustic models in a large vocabulary recognition task. In *Proc. INTERSPEECH*.

Arrowood, J. and Clements, M. (2002). Using observation uncertainty in HMM decoding. In *Proc. ICSLP*.

Aubert, X. (2002). An overview of decoding techniques for large vocabulary continuous speech recognition. *Computer Speech & Language*, 16(1):89–114.

Axelrod, S., Goel, V., Gopinath, R., Olsen, P., and Visweswariah, K. (2005). Subspace constrained gaussian mixture models for speech recognition. *Speech and Audio Processing, IEEE Transactions on*, 13(6):1144–1160.

Barker, J. (1975). The DRAGON system–An overview. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 23(1):24–29.

Barzilai, J. and Borwein, J. (1988). Two point step size gradient methods. *IMAJ. Numer. Anal.*, 8:141–148.

Baum, L. (1972). An equality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. *Inequalities*, 3(1–8).

Baum, L., Petrie, T., Soules, G., and Weiss, N. (1970). A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The annals of mathematical statistics*, pages 164–171.

Bishop, C. (2006). *Pattern Recognition and Machine Learning*. Springer New York:.

Burget, L., Schwarz, P., Agarwal, M., Akyazi, P., Feng, K., Ghoshal, A., Glembek, O., Goel, N., Karafiát, M., Povey, D., Rastrow, A., Rose, R., and Thomas, S. (2010). Multilingual acoustic modeling for speech recognition based on subspace Gaussian mixture models. In *Proc. IEEE ICASSP*, pages 4334–4337.

Byrne, W., Beyerlein, P., Huerta, J., Khudanpur, S., Marthi, B., Morgan, J., Peterek, N., Picone, J., and Wang, W. (2000). Towards language independent acoustic modeling. In *Proc. ICASSP*, pages 1029–1032. IEEE.

Çetin, O., Magimai-Doss, M., Livescu, K., Kantor, A., King, S., Bartels, C., and Frankel, J. (2007). Monolingual and crosslingual comparison of tandem features derived from articulatory and phone MLPs. In *Proc. ASRU*, pages 36–41. IEEE.

Chen, S., Donoho, D., and Saunders, M. (2001). Atomic decomposition by basis pursuit. *SIAM review*, pages 129–159.

Dahl, G., Yu, D., Deng, L., and Acero, A. (2012). Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):30–42.

Davis, S. and Mermelstein, P. (1980). Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 28(4):357–366.

Dempster, A., Laird, N., and Rubin, D. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38.

Deng, L., Acero, A., Plumpe, M., and Huang, X. (2000). Large-vocabulary speech recognition under adverse acoustic environments. In *Proc. ICSLP*.

Deng, L., Droppo, J., and Acero, A. (2004). 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 Transactions on Speech and Audio Processing*, 12(2):133–143.

Deng, L., Droppo, J., and Acero, A. (2005). Dynamic compensation of HMM variances using the feature enhancement uncertainty computed from a parametric

model of speech distortion. *IEEE Transactions on Speech and Audio Processing*, 13(3):412–421.

Donoho, D. (2006). Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306.

Droppo, J. and Acero, A. (2008). Environmental robustness. In Benesty, J., Sondhi, M., and Huang, Y., editors, *Handbook of Speech Processing*, pages 653–679. Springer.

Droppo, J., Acero, A., and Deng, L. (2002). Uncertainty decoding with SPLICE for noise robust speech recognition. In *Proc. ICASSP*. IEEE.

Du, J. and Huo, Q. (2011). A feature compensation approach using high-order vector Taylor series approximation of an explicit distortion model for noisy speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(8):2285–2293.

Dutilleul, P. (1999). The MLE algorithm for the matrix normal distribution. *Journal of Statistical Computation and Simulation*, 64(2):105–123.

Faubel, F., McDonough, J., and Klakow, D. (2010). On expectation maximization based channel and noise estimation beyond the vector Taylor series expansion. In *Proc. ICASSP*, pages 4294–4297. IEEE.

Figueiredo, M., Nowak, R., and Wright, S. (2007). Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *IEEE Journal on selected topics in Signal Processing*, 1(4):586–597.

Flego, F. and Gales, M. (2009). Discriminative adaptive training with vts and jud. In *Automatic Speech Recognition & Understanding, 2009. ASRU 2009. IEEE Workshop on*, pages 170–175. IEEE.

Flego, F. and Gales, M. (2011). Factor analysis based VTS and JUD noise estimation and compensation. In *Proc. ICASSP*, pages 4792–4795. IEEE.

Flego, F. and Gales, M. (2012). Factor analysis based VTS discriminative adaptive training. In *Proc. ICASSP*, pages 4669–4672. IEEE.

Frey, B., Deng, L., Acero, A., and Kristjansson, T. (2001). ALGONQUIN: Iterating Laplace's method to remove multiple types of acoustic distortion for robust speech

recognition. In *Seventh European Conference on Speech Communication and Technology*.

Furui, S. (1986). Speaker-independent isolated word recognition using dynamic features of speech spectrum. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 34(1):52–59.

Gales, M. (1995). *Model-based techniques for noise robust speech recognition*. PhD thesis, Cambridge University.

Gales, M. (1996). The generation and use of regression class trees for MLLR adaptation. Technical report, Univeristy of Cambridge, Department of Engineering.

Gales, M. (1998a). Cluster adaptive training for speech recognition. In *Proc. ICSLP*.

Gales, M. (1998b). Maximum likelihood linear transformations for HMM-based speech recognition. *Computer Speech and Language*, 12:75–98.

Gales, M. and Flego, F. (2010). Discriminative classifiers with adaptive kernels for noise robust speech recognition. *Computer Speech & Language*, 24(4):648–662.

Gales, M. and Van Dalen, R. (2007). Predictive linear transforms for noise robust speech recognition. In *Proc. ASRU*, pages 59–64. IEEE.

Gales, M. and Young, S. (2008). The application of hidden markov models in speech recognition. *Foundations and Trends in Signal Processing*, 1(3):195–304.

Gauvain, J. and Lee, C. (1994). Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains. *IEEE transactions on speech and audio processing*, 2(2):291–298.

Ghalehjegh, S. H. and Rose, R. C. (2013). Phonetic subspace adaptation for automatic speech recognition. In *Proc. ICASSP*.

Ghoshal, A., Povey, D., Agarwal, M., Akyazi, P., Burget, L., Feng, K., Glembek, O., Goel, N., Karafiát, M., Rastrow, A., et al. (2010). A novel estimation of feature-space mllr for full-covariance models. In *Proc. ICASSP*, pages 4310–4313. IEEE.

Goel, N., Thomas, S., Agarwal, M., Akyazi, P., Burget, L., Feng, K., Ghoshal, A., Glembek, O., Karafiát, M., Povey, D., Rastrow, A., Rose, R., and Schwarz, P. (2010). Approaches to automatic lexicon learning with limited training examples. In *proc. ICASSP*, pages 5094–5097. IEEE.

Gopinath, R., Gales, M., Gopalakrishnan, P., Balakrishnan-Aiyer, S., and Picheny, M. (1995). Robust speech recognition in noise—Performance of the IBM continuous speech recogniser on the ARPA noise spoke task. In *Proc. ARPA Workshops Spoken Lang. Syst. Technol.*, pages 127–130.

Gupta, A. and Nagar, D. (1999). *Matrix Variate Distributions*, volume 104 of *Monographs and Surveys in Pure and Applied Mathematics*. Chapman & Hall/CRC.

Gustafsson, F. and Hendeby, G. (2012). Some relations between extended and unscented Kalman filters. *IEEE Transactions on Signal Processing*, 60(2):545–555.

Hastie, T., Tibshirani, R., and Friedman, J. (2005). *The elements of statistical learning: data mining, inference and prediction*. Springer.

Hermansky, H. (1990). Perceptual linear predictive (plp) analysis of speech. *The Journal of the Acoustical Society of America*, 87:1738.

Hermansky, H., Ellis, D., and Sharma, S. (2000). Tandem connectionist feature extraction for conventional HMM systems. In *Proc. ICASSP*, pages 1635–1638. IEEE.

Hu, Y. and Huo, Q. (2006). An HMM compensation approach using unscented transformation for noisy speech recognition. *Chinese Spoken Language Processing*, pages 346–357.

Hu, Y. and Huo, Q. (2007). Irrelevant variability normalization based HMM training using VTS approximation of an explicit model of environmental distortions. In *Proc. INTERSPEECH*.

Huang, X. and Jack, M. (1989). Semi-continuous hidden markov models for speech signals. *Computer Speech & Language*, 3(3):239–251.

Imseng, D., Bourlard, H., and Garner, P. (2012). Using KL-divergence and multilingual informaiton to improve ASR for under-resourced languages. In *Proc. ICASSP*, pages 4869–4872. IEEE.

Imseng, D., Rasipuram, R., and Doss, M. (2011). Fast and flexible Kullback-Leibler divergence based acoustic modelling for non-native speech recognition. In *Proc. ASRU*, pages 348–353. IEEE.

Jelinek, F. (1976). Continuous speech recognition by statistical methods. *Proceedings of the IEEE*, 64(4):532–556.

Julier, S. and Uhlmann, J. (2004). Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422.

Kalgaonkar, K., Seltzer, M., and Acero, A. (2009). Noise robust model adaptation using linear spline interpolation. In *Proc. ASRU*, pages 199–204.

Kalinli, O., Seltzer, M., Droppo, J., and Acero, A. (2010). Noise adaptive training for robust automatic speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(8):1889–1091.

Katz, S. (1987). Estimation of probabilities from sparse data for the language model component of a speech recognizer. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 35(3):400–401.

Kenny, P. (2005). Joint factor analysis of speaker and session variability: theory and algorithms. Technical report, CRIM-06/08-13.

Kim, D. and Gales, M. (2011). Noisy constrained maximum-likelihood linear regression for noise-robust speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(2):315–325.

Kim, D., Kwan Un, C., and Kim, N. (1998). Speech recognition in noisy environments using first-order vector Taylor series. *Speech Communication*, 24(1):39–49.

Kneser, R. and Ney, H. (1995). Improved backing-off for m-gram language modeling. In *Proc. ICASSP*, pages 181–184. IEEE.

Kohler, J. (1996). Multi-lingual phoneme recognition exploiting acoustic-phonetic similarities of sounds. In *Proc. ICSLP*, pages 2195–2198.

Kuhn, R., Junqua, J., Nguyen, P., and Niedzielski, N. (2000). Rapid speaker adaptation in eigenvoice space. *IEEE Transactions on Speech and Audio Processing*, 8(6):695–707.

Lal, P. (2011). *Cross-lingual Automatic Speech Recognition using Tandem Features*. PhD thesis, The University of Edinburgh.

Le, V. and Besacier, L. (2005). First steps in fast acoustic modeling for a new target language: application to Vietnamese. In *Proc. ICASSP*, pages 821–824. IEEE.

Lee, K. (1988). On large-vocabulary speaker-independent continuous speech recognition. *Speech communication*, 7(4):375–379.

Leggetter, C. and Woodland, P. (1995). Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models. *Computer speech and language*, 9(2):171.

Li, J., Deng, L., Yu, D., Gong, Y., and Acero, A. (2009). A unified framework of HMM adaptation with joint compensation of additive and convolutive distortions. *Computer Speech & Language*, 23(3):389–405.

Li, J., Seltzer, M., and Gong, Y. (2012). Improvements to VTS feature enhanecement. In *Proc. ICASSP*, pages 4677–4680. IEEE.

Li, J., Yu, D., Gong, Y., and Deng, L. (2010). Unscented transform with online distortion estimation for HMM adaptation. In *Proc. INTERSPEECH*.

Liao, H. (2007). *Uncertainty decoding for noise robust speech recognition*. PhD thesis, University of Cambridge.

Liao, H. and Gales, M. (2005). Joint uncertainty decoding for noise robust speech recognition. In *Proc. INTERSPEECH*. Citeseer.

Liao, H. and Gales, M. (2007). Adaptive training with joint uncertainty decoding for robust recognition of noisy data. In *Proc. ICASSP*, volume 4, pages IV–389. IEEE.

Lu, L., Chin, K., Ghoshal, A., and Renals, S. (2012a). Noise compensation for subspace Gaussian mixture models. In *Proc. INTERSPEECH*.

Lu, L., Chin, K., Ghoshal, A., and Renals, S. (2013a). Joint uncertainty decoding for noise robust subspace Gaussian mixture models. *IEEE Transactions on Audio, Speech, and Language Processing*.

Lu, L., Ghoshal, A., and Renals, S. (2011a). Regularized subspace Gaussian mixture models for cross-lingual speech recognition. In *Proc. IEEE ASRU*.

Lu, L., Ghoshal, A., and Renals, S. (2011b). Regularized subspace Gaussian mixture models for speech recognition. *IEEE Signal Processing Letters*, 18(7):419–422.

Lu, L., Ghoshal, A., and Renals, S. (2012b). Joint uncertainty decoding with unscented transforms for noise robust subspace Gaussian mixture models. In *Proc. SAPA-SCALE Workshop*.

Lu, L., Ghoshal, A., and Renals, S. (2012c). Maximum a posteriori adaptation of subspace Gaussian mixture models for cross-lingual speech recognition. In *Proc. ICASSP*.

Lu, L., Ghoshal, A., and Renals, S. (2013b). Cross-lingual subspace Gaussain mixture models for low-resource speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing (submitted)*.

Lu, L., Ghoshal, A., and Renals, S. (2013c). Noise adaptive training for subspace Gaussian mixture models. In *Proc. INTERSPEECH*.

Milner, B. (1993). A comparison of front-end configurations for robust speech recognition. In *Acoustics, Speech, and Signal Processing, 1993. ICASSP-93., 1993 IEEE International Conference on*, volume 1, pages I–I. IEEE.

Mohri, M., Pereira, F., and Riley, M. (2002). Weighted finite-state transducers in speech recognition. *Computer Speech & Language*, 16(1):69–88.

Moreno, P., Raj, B., and Stern, R. (1996). A vector Taylor series approach for environment-independent speech recognition. In *Proc. ICASSP*, volume 2, pages 733–736. IEEE.

Olsen, P. and Gopinath, R. (2002). Modeling inverse covariance matrices by basis expansion. In *Proc. ICASSP*.

Omar, M. (2007). Regularized feature-based maximum likelihood linear regression for speech recognition. In *Proc. INTERSPEECH*.

Paul, D. and Baker, J. (1992). The design for the Wall Street Journal-based CSR corpus. In *Second International Conference on Spoken Language Processing*.

Plahl, C., Schluter, R., and Ney, H. (2011). Cross-lingual portability of Chinese and English neural network features for French and German LVCSR. In *Proc. ASRU*, pages 371–376. IEEE.

Povey, D. (2009). A tutorial-style introduction to subspace Gaussian mixture models for speech recognition. Technical report, MSR-TR-2009-111, Microsoft Research.

Povey, D., Burget, L., Agarwal, M., Akyazi, P., Kai, F., Ghoshal, A., Glembek, O., Goel, N., Karafiát, M., Rastrow, A., Rose, R., Schwarz, P., and Thomas, S. (2011a).

The subspace Gaussian mixture model—A structured model for speech recognition. *Computer Speech & Language*, 25(2):404–439.

Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlıcek, P., Qian, Y., Schwarz, P., Silovský, J., Semmer, G., and Veselý, K. (2011b). The Kaldi speech recognition toolkit. In *Proc. ASRU*.

Povey, D., Karafiát, M., Ghoshal, A., and Schwarz, P. (2011c). A symmetrization of the subspace Gaussian mixture model. In *Proc. ICASSP*, pages 4504–4507. IEEE.

Povey, D. and Saon, G. (2006). Feature and model space speaker adaptation with full covariance gaussians. In *Proc. INTERSPEECH*.

Qian, Y., Xu, J., Povey, D., and L, J. (2011). Strategies for using MLP based features with limited target-language training data. In *Proc. ASRU*, pages 354–358. IEEE.

Rabiner, L. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Ragni, A. and Gales, M. (2011). Derivative kernels for noise robust ASR. In *Proc. ASRU*, pages 119–124. IEEE.

Riedhammer, K., Bocklet, T., Ghoshal, A., and Povey, D. (2012). Revisiting semi-continuous hidden markov models. In *Proc. ICASSP*, pages 4721–4724. IEEE.

Sainath, T., Carmi, A., Kanevsky, D., and Ramabhadran, B. (2010a). Bayesian compressive sensing for phonetic classification. In *Proc. ICASSP*, pages 4370–4373.

Sainath, T., Ramabhadran, B., Nahamoo, D., Kanevsky, D., and Sethy, A. (2010b). Sparse representation features for speech recognition. In *Proc. INTERSPEECH*, pages 2254–2257.

Saon, G. and Chien, J. (2011). Bayesian sensing hidden Markov models for speech recognition. In *Proc. ICASSP*, pages 5056–5059.

Saon, G., Zweig, G., and Padmanabhan, M. (2001). Linear feature space projections for speaker adaptation. In *Proc. ICASSP*, pages 325–328. IEEE.

Schultz, T. (2002). GlobalPhone: a multilingual speech and text database developed at Karlsruhe University. In *Proc. ICLSP*, pages 345–348.

Schultz, T. and Waibel, A. (1997). Fast bootstrapping of LVCSR systems with multilingual phoneme sets. In *Proc. Eurospeech*, pages 371–374. Citeseer.

Schultz, T. and Waibel, A. (1998). Multilingual and crosslingual speech recognition. In *Proc. DARPA Workshop on Broadcast News Transcription and Understanding*. Citeseer.

Schultz, T. and Waibel, A. (2001). Language-independent and language-adaptive acoustic modeling for speech recognition. *Speech Communication*, 35(1):31–52.

Seltzer, M. and Acero, A. (2011). Separating Speaker and Environmental Variability Using Factored Transforms. In *Proc. INTERSPEECH*.

Serafini, T., Zanghirati, G., and Zanni, L. (2005). Gradient projection methods for quadratic programs and applications in training support vector machines. *Optimization Methods and Software*, 20(2-3):353–378.

Sim, K. (2009). Discriminative product-of-expert acoustic mapping for cross-lingual phone recognition. In *Proc. ASRU*, pages 546–551. IEEE.

Sim, K. and Li, H. (2008). Robust phone set mapping using decision tree clustering for cross-lingual phone recognition. In *Proc. ICASSP*, pages 4309–4312. IEEE.

Singh, R., Raj, B., and Stern, R. (2000). Automatic generation of phone sets and lexical transcriptions. In *Proc. ICASSP*, pages 1691–1694. IEEE.

Siniscalchi, S., Lyu, D., Svendsen, T., and Lee, C. (2012). Experiments on cross-language attribute detection and phone recognition with minimal target-specific training data. *IEEE transactions on audio, speech, and language processing*, 20(3):875–887.

Siohan, O., Chesta, C., and Lee, C. (2001). Joint maximum a posteriori adaptation of transformation and HMM parameters. *IEEE Transactions on Speech and Audio Processing*, 9(4):417–428.

Sivaram, G., Nemala, S., Elhilali, M., Tran, T., and Hermansky, H. (2010). Sparse coding for speech recognition. In *Proc. ICASSP*, pages 4346–4349.

Slobada, T. and Waibel, A. (1996). Dictionary learning for spontaneous speech recognition. In *Proc. ICSLP*, pages 2328–2331.

Stolcke, A., Grézl, F., Hwang, M., Lei, X., Morgan, N., and Vergyri, D. (2006). Cross-domain and cross-language portability of acoustic features estimated by multilayer perceptrons. In *Proc. ICASSP*, pages 321–324. IEEE.

Swietojanski, P., Ghoshal, A., and Renals, S. (2012). Unsupervised Cross-lingual knowledge transfer in DNN-based LVCSR. In *Proc. IEEE Workshop on Spoken Language Technology*.

Thomas, S., Ganapathy, S., and Hermansky, H. (2010). Cross-lingual and multi-stream posterior features for low resource LVCSR systems. In *Proc. INTERSPEECH*, pages 877–880.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288.

van Dalen, R. and Gales, M. (2009). Extended VTS for noise-robust speech recognition. In *Proc. ICASSP*, pages 3829–3832. IEEE.

Viterbi, A. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *Information Theory, IEEE Transactions on*, 13(2):260–269.

Wang, Y. and Gales, M. (2011). Speaker and noise factorisation on the Aurora 4 task. In *Proc. ICASSP*, pages 4584–4587. IEEE.

Woodland, P., Gales, M., and Pye, D. (1996). Improving environmental robustness in large vocabulary speech recognition. In *Proc. ICASSP*, volume 1, pages 65–68. IEEE.

Woodland, P., Odell, J., Valtchev, V., and Young, S. (1994). Large vocabulary continuous speech recognition using HTK. In *Proc. ICASSP*, pages 125–128.

Xu, H. and Chin, K. (2009a). Comparison of estimation techniques in joint uncertainty decoding for noise robust speech recognition. In *Proc. INTERSPEECH*, pages 2403–2406.

Xu, H. and Chin, K. (2009b). Joint uncertainty decoding with the second order approximation for noise robust speech recognition. In *Proc. ICASSP*, pages 3841–3844. IEEE.

Xu, H., Gales, M., and Chin, K. (2011). Joint uncertainty decoding with predictive methods for noise robust speech recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 19(6):1665–1676.

Young, S. et al. (2002). *The HTK Book*. Cambridge University Engineering Department.

Young, S., Evermann, G., Gales, M., Hain, T., Kershaw, D., Liu, X., Moore, G., Odell, J., Ollason, D., Povey, D., et al. (2006). The htk book (for htk version 3.4).

Young, S., Odell, J., and Woodland, P. (1994). Tree-based state tying for high accuracy acoustic modelling. In *Proceedings of the workshop on Human Language Technology*, page 312. Association for Computational Linguistics.

Yu, D., Deng, L., Droppo, J., Wu, J., Gong, Y., and Acero, A. (2008). Robust speech recognition using a cepstral minimum-mean-square-error-motivated noise suppressor. *Audio, Speech, and Language Processing, IEEE Transactions on*, 16(5):1061–1070.

Zen, H., Braunschweiler, N., Buchholz, S., Gales, M., Knill, K., Krstulovic, S., and Latorre, J. (2012). Statistical parametric speech synthesis based on speaker and language factorization. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 20(6):1713–1724.

Zhao, X., Dong, Y., Zhao, J., Lu, L., Liu, J., and Wang, H. (2009). Variational bayesian joint factor analysis for speaker verification. In *Proc. ICASSP*, pages 4049–4052. IEEE.

Zhao, Y. and Juang, B. (2010). A comparative study of noise estimation algorithms for VTS-based robust speech recognition. In *Proc. INTERSPEECH*.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320.