Structured Precision Matrix Modelling for Speech Recognition

Khe Chai Sim

Churchill College
and
Cambridge University Engineering Department

6th April 2006

Dissertation submitted to the University of Cambridge
for the degree of Doctor of Philosophy

Declaration

This dissertation is the result of my own work and includes nothing which is the outcome of the work done in collaboration, except where stated. It has not been submitted in whole or part for a degree at any other university.

The length of this thesis including footnotes and appendices is approximately 53,000 words.

Summary

The most extensively and successfully applied acoustic model for speech recognition is the Hidden Markov Model (HMM). In particular, a multivariate Gaussian Mixture Model (GMM) is typically used to represent the output density function of each HMM state. For reasons of efficiency, the covariance matrix associated with each Gaussian component is assumed diagonal and the probability of successive observations is assumed independent given the HMM state sequence. Consequently, the *spectral* (intra-frame) and *temporal* (inter-frame) correlations are poorly modelled. This thesis investigates ways of improving these aspects by extending the standard HMM. Parameters for these extended models are estimated discriminatively using the Minimum Phone Error (MPE) criterion. The performance of these models is investigated and benchmarked against the state-of-the-art CUHTK evaluation systems.

The first part of this thesis examines various precision matrix approximation schemes to improve the modelling of spectral correlation. Previous work have found that techniques such as the semi-tied covariances, extended maximum likelihood linear transform and subspace for precision and mean, outperform the diagonal covariance matrix model. In this thesis, a unified framework of basis superposition is formulated to describe these precision matrix models in a consistent manner. This framework emphasises the importance of finding a well-defined set of bases to capture the common precision matrix structures in the system, therefore reducing the redundancies in the system to yield a compact model representation. Furthermore, the minimum phone error discriminative training technique is also applied to train these precision matrix models in a large vocabulary continuous speech recognition system.

Another limitation of HMMs is the *conditional independence assumption*, which makes it a poor *temporal correlation* model. This is typically improved by using a trajectory model. In the second part of this thesis, a semi-parametric trajectory model is introduced where the mean vectors and precision matrices in the system are modelled as a function of the observation sequence. When the mean vector alone is modelled, this model is the same as the fMPE technique, originally proposed as a discriminative training technique of the features. In addition, a novel pMPE technique is also proposed to incorporate non-stationarity to the precision matrix structures. Combining fMPE and pMPE yields a simple form of semi-parametric trajectory model, with time varying mean vector and diagonal precision matrices. The performance of fMPE and pMPE and the interaction between these techniques will be examined, primarily based on the Mandarin conversational telephone speech transcription task.

Keywords: precision matrix modelling, speech recognition, large vocabulary continuous speech recognition, discriminative training, hidden Markov models, Gaussian mixture models, trajectory model

Acknowledgement

First of all, I would like to extend my utmost gratitude to my supervisor, Mark Gales, for his countless valuable guidance, motivation and criticism throughout the work of this thesis. By providing the right balance of suggestions and freedom as well as the wonderful discussions on all aspects of this work, he has made this work not just possible, but more importantly, enjoyable!

I would also like to thank Steve Young and Phil Woodland for leading the speech group of the Machine Intelligence Laboratory at Cambridge University. Furthermore, this research work could not have been possible without the wonderful HTK software. For that, I must thank Steve Young, Phil Woodland, Julian Odell, Mark Gales, Gunnar Evermann, Anna Langley and all those who have contributed toward developing and maintaining HTK.

I am also very grateful to Phil Woodland for kindly providing funding support for my degree and many international workshops and conferences through the DARPA funded Effective, Affordable and Reusable Speech-to-text (EARS) program[1]. It was a privilege to have worked with the EARS team here in the Machine Intelligence Laboratory (formerly known as the Speech, Vision and Robotics) group. I would also like to thank Phil Woodland and Mark Gales for giving me the opportunity to participate in many EARS workshops, through which I have greatly benefited from interaction with many other research members from the world leading speech groups.

I owe my thanks to the EARS team members and other members of the group who have helped me in various ways and make the group an interesting place to work in. There are too many individuals to acknowledge, but I must thank, in no particular order, Gunnar Evermann, Do Yeong Kim, Bin Jia, Thomas Hain, Xunying Liu, Kai Yu, Ho Yin Chan, Lan Wang, Hui Ye and David Mrva. I must also thank Anna Langley and Patrick Gosling for their excellent job in providing and maintaining the computing facilities.

Finally, I owe my biggest thank to my family for their endless support and encouragement over the years. In particular, I would like to thank my wife who has always believed in me in everything I do!

[1]This work was supported by DARPA grant MDA972-02-1-0013. The thesis does not necessarily reflect the position or the policy of the US Government and no official endorsement should be inferred.

Glossary

ASR	Automatic Speech Recognition
ATIS	Air Travel Information System
BN	Broadcast News
BW	Baum Welch
CDHMM	Continuous Density Hidden Markov Model
CER	Character Error Rate
CMLE	Conditional Maximum Likelihood Estimation
CMLLR	Constrained Maximum Likelihood Linear Regression
CMN	Cepstral Mean Normalisation
CN	Confusion Network
CNC	Confusion Network Combination
CTS	Conversational Telephone Speech
CVN	Cepstral Variance Normalisation
EBW	Extended Baum Welch
EM	Expectation Maximisation
EMLLT	Extended Maximum Likelihood Linear Transformation
FAHMM	Factor Analysed HMM
FD	Frame Discrimination
FMLLR	Feature Maximum Likelihood Linear Regression
GD	Gender Dependent
GI	Gender Independent
GMM	Gaussian Mixture Model
HDA	Heteroscedastic Discriminant Analysis
HMM	Hidden Markov Model
HLDA	Heteroscedastic Linear Discriminant Analysis
HTK	HMM Toolkit

LDA	Linear Discriminant Analysis
LVCSR	Large Vocabulary Continuous Speech Recognition
MAPLR	Maximum a Posteriori Linear Regression
MDI	Minimum Discrimination Information
MFCC	Mel Frequency Cepstral Coefficients
MLE	Maximum Likelihood Estimation
MLLT	Maximum Likelihood Linear Transform
MLLR	Maximum Likelihood Linear Regression
MMI	Maximum Mutual Information
MoG	Mixture of Gaussians
MCE	Minimum Classification Error
MPE	Minimum Phone Error
MWE	Minimum Word Error
PDF	Probability Density Function
PLP	Perceptual Linear Prediction
PMM	Precision Matrix Model
PoE	Product of Experts
PoG	Product of Gaussians
QEP	Quadratic Eigenvalue Problem
RM	Resource Management
ROVER	Recogniser Output Voting Error Reduction
SAT	Speaker Adaptive Training
SD	Speaker Dependent
SI	Speaker Independent
SPAM	Subspace for Precision and Mean
STC	Semi-tied Covariances
STFT	Short Time Fourier Transform
SVD	Singular Value Decomposition
VTLN	Vocal Tract Length Normalisation
WER	Word Error Rate
WSJ	Wall Street Journal

Notations

The following notations have been used throughout this work:

General Notations:

s	a *scalar* variable is denoted by a lowercase letter with plain face
$\boldsymbol{v}$	a *vector* is denoted by a lowercase letter with **bold** face
v_j	the jth element of vector $\boldsymbol{v}$
$\boldsymbol{M}$	a *matrix* is denoted by an uppercase letter with **bold** face
$\boldsymbol{M}_i$	the ith row of matrix $\boldsymbol{M}$
M_{ij}	the (i, j)th element of matrix $\boldsymbol{M}$
$\boldsymbol{M}'$	transpose of matrix $\boldsymbol{M}$
$\boldsymbol{M}^{-1}$	inverse of matrix $\boldsymbol{M}$

Standard HMM Parameters Notations:

$\boldsymbol{o}_t$	a $d \times 1$ observation vector at time t
$\mathcal{O}_1^T$	a set of observation vectors, $\boldsymbol{o}_1, \boldsymbol{o}_2, \ldots, \boldsymbol{o}_T$
q_t	the HMM state at time t
Q_1^T	a sequence of HMM states, $q_1, q_2, \ldots, q_T$
$\boldsymbol{\theta}$	denotes a set of model parameters in general
$\boldsymbol{\theta}^{\mathrm{am}}$	the set of parameters for acoustic model
$\boldsymbol{\theta}^{\mathrm{lm}}$	the set of parameters for language model
$\boldsymbol{\mu}_{sm}$	a $d \times 1$ mean vector for component m in state s
$\boldsymbol{\Sigma}_{sm}$	a $d \times d$ covariance matrix for component m in state s
$\boldsymbol{P}_{sm}$	a $d \times d$ precision matrix for component m in state s
μ_{smj}	the jth element of the mean vector for component m in state s
σ^2_{smj}	the jth diagonal element of the covariance matrix for component m in state s
ψ_{smj}	the jth diagonal element of the precision matrix for component m in state s

Precision Matrix Modelling Parameters Notations:

$\boldsymbol{S}_b$	the bth basis matrix
$\boldsymbol{a}_b$	the bth basis *row* vector
λ_{smb}	the bth basis coefficient of component m in state s

Notations used for the MLLR adaptation:

$\boldsymbol{\zeta}_t$	the augmented observation vector used in constrained MLLR adaptation
$\boldsymbol{\xi}_m$	the augmented mean vector used in MLLR mean adaptation
$\boldsymbol{B}^{\mathtt{r}}$	the observation transformation matrix associated with the rth regression class
$\boldsymbol{A}^{\mathtt{r}}$	the mean transformation matrix associated with the rth regression class
$\boldsymbol{b}^{\mathtt{r}}$	the mean bias vector associated with the rth regression class
$\boldsymbol{H}^{\mathtt{r}}$	the covariance transformation matrix associated with the rth regression class
$\boldsymbol{X}^{\mathtt{r}}$	the augmented transformation matrix (linear transformation and bias) associated with the rth regression class

Semi-parametric Trajectory Model Parameters Notations:

$\boldsymbol{\mu}_{smt}$	a $d \times 1$ mean vector for component m in state s at time t
$\boldsymbol{\Sigma}_{smt}$	a $d \times d$ covariance matrix for component m in state s at time t
$\boldsymbol{P}_{smt}$	a $d \times d$ precision matrix for component m in state s at time t
$\boldsymbol{C}_t$	the observation transformation matrix at time t
$\boldsymbol{d}_t$	the observation bias vector at time t
$\boldsymbol{A}_t$	the mean transformation matrix at time t
$\boldsymbol{b}_t$	the mean bias vector at time t
$\boldsymbol{Z}_t$	the precision matrix transformation at time t
$\boldsymbol{B}^{(\mathtt{i})}$	the observation transformation matrix associated with the ith centroid
$\boldsymbol{A}^{(\mathtt{i})}$	the mean transformation matrix associated with the ith centroid
$\boldsymbol{b}^{(\mathtt{i})}$	the mean bias vector associated with the ith centroid
$\boldsymbol{Z}^{(\mathtt{i})}$	the precision matrix transformation associated with the ith centroid

Notations used for parameters estimations:

$\mathcal{R}^{[\mathrm{ml,mpe}]}$	ML or MPE objective function
$\mathcal{Q}^{[\mathrm{ml,mpe}]}$	ML or MPE auxiliary function
$\mathcal{L}^{[\mathrm{ml,mpe}]}$	ML or MPE log likelihood function
$\gamma_{sm}^{[\mathrm{ml,mpe}]}(t)$	ML or MPE posterior of component m in state s at time t
$\beta_{sm}^{[\mathtt{ml,mpe}]}$	ML or MPE occupancy counts for component m in state s
$\boldsymbol{x}_{sm}^{[\mathtt{ml,mpe}]}$	first order ML or MPE statistics for component m in state s
$\boldsymbol{Y}_{sm}^{[\mathtt{ml,mpe}]}$	second order ML or MPE statistics for component m in state s
$\boldsymbol{W}_{sm}^{[\mathtt{ml,mpe}]}$	Covariance matrix ML or MPE statistics for component m in state s

Contents

List of Tables

List of Figures

1

Introduction

Automatic Speech Recognition (ASR) is the process of converting speech waveform automatically into sequence of words (texts). It has many real world applications ranging from dictation software for medical transcription and desktop dictation on personal computers (e.g. IBM's *ViaVoice*[1] and Dragon's *NaturallySpeaking*)[2]; automated call centres (e.g. flight information enquiry, cinema ticket booking and weather enquiry systems); embedded systems such as voice dialling on mobile phones. For many of these applications, an ASR system often has to cope with uncertainties due to speaker and environmental variabilities. A call centre ASR system, for example, accepts calls from a wide range of customers with different speaking styles, rates and accents, on telephone lines with different conditions (landlines or mobile phones), and different background noise. In addition, even speaker specific desktop dictation systems have to handle *intra-speaker* variabilities because human beings cannot produce identical waveforms speaking the same words at different instances. These variabilities can be modelled using a probabilistic model. The most successful and popular ASR systems to date are based on Hidden Markov Models (HMMs).

Continuous Density HMMs (CDHMMs) are by far the most popular acoustic models used in speech recognition systems. In CDHMMs, the output Probability Density Function (PDF) is usually represented by a multivariate Gaussian Mixture Model (GMM) [57]. A multivariate Gaussian distribution is parameterised in terms of a mean vector and covariance matrix. The inter-variable correlations are captured by the covariance matrices, which are symmetric and positive-definite. Complete modelling of the correlations using full covariance matrices poses two critical issues: the large number of model parameters involved; and the increase in likelihood computational cost. These issues are more apparent for systems with a large number of Gaussian components. A Large Vocabulary Continuous Speech Recognition (LVCSR) [133]

[1] www-3.ibm.com/software/speech

[2] http://www.dragontalk.com/NATURAL.htm

system may have more than 100,000 Gaussian components. The conventional approach to circumvent these problems is to use a diagonal covariance matrix approximation. Hence, the inter-dimensional correlations are ignored and implicitly modelled. Recently, advanced approximation schemes were found to yield improved performance, including both covariance matrix and precision (inverse covariance) matrix approximations. The former is generally less efficient in terms of likelihood calculation due to the need to perform matrix inversion. For this reason, this work began by investigating several forms of precision matrix models within a generic framework of basis superposition, including the Semi-Tied Covariance (STC) [32, 34], (also known as Maximum Likelihood Linear Transform or MLLT [52, 53]), Extended Maximum Likelihood Linear Transformation (EMLLT) [85, 86] and Subspace for Precision and Mean (SPAM) [7] models. The major part of the work goes to the investigation of Minimum Phone Error (MPE) [95, 100] discriminative training of these models for LVCSR systems. Speaker adaptation and adaptive training of these models will also be investigated to achieve state-of-the-art performance.

In standard HMMs, the statistics within HMM states are *piece-wise* constant. This assumption does not work well with speech data in general. To overcome this limitation, various extensions to HMMs have been widely investigated by many researchers to model the trajectory of the feature vectors within the acoustic space. For examples, the trajectory models [43, 47, 48, 128], trajectory HMMs [118, 119], vector linear predictor [130], buried Markov model [13], segmental HMMs [41, 42], stochastic segment models [88, 89] and switching linear dynamical systems [104]. A trajectory model can be generalised as a standard HMM with time varying parameters, in particular the Gaussian mean vectors and covariance matrices. In the second part of this thesis, a semi-parametric trajectory model will be introduced. In this model, the time varying mean vectors and covariance matrices are modelled as a semi-parametric function of the observation sequence. The form of semi-parametric function used in this work, when applied to the mean vectors, yields the fMPE technique [96, 97]. This technique was reported to yield a further relative gain of approximately 10% over the MPE alone systems. In this work, a novel approach which models the time varying precision matrices using the same semi-parametric function will be introduced. This technique is known as pMPE [112]. In Chapter 7, fMPE and pMPE will be described in greater details as a unified semi-parametric trajectory model.

1.1 Statistical Speech Recognition

Research to the speech technology began in 1936 at the AT&T's Bell Labs. In 1939, Bell Labs demonstrated "Voder", an electronic speech synthesiser operated by a human in the 1939 World's Fair in New York. During the same event, some of the earliest speech recognition attempts were also publicised. From the 40's, the U.S. Department of Defense (DoD) began to invest in the speech recognition technology. In the early 70's, Lenny Baum of Princeton University, together

with a team from ARPA (Advanced Research Projects Agency), applied HMM for speech recognition. This mathematical pattern-matching strategy was eventually adopted by many of the leading speech recognition companies including IBM [61], Bell System [71], Dragon System [10], Philips and others. In 1971, DARPA (Defense Advanced Research Projects Agency) established the Speech Understanding Research (SUR) to develop a continuous speech understanding system. Since 1985, the National Institute of Standards and Technology (NIST)[3] conducted performance assessments and benchmark testings for speech recognition tasks. Several databases were collected for these purposes, including the earlier Resource Management (RM) [90, 93], Air Travel Information System (ATIS) [92] and Wall Street Journal (WSJ) tasks as well as the more recent Conversational Telephone Speech (CTS) and Broadcast News (BN) data sets. These databases are widely used by many research institutes to conduct speech recognition experiments. A summary of the history of NIST's benchmark tests are given in [91]

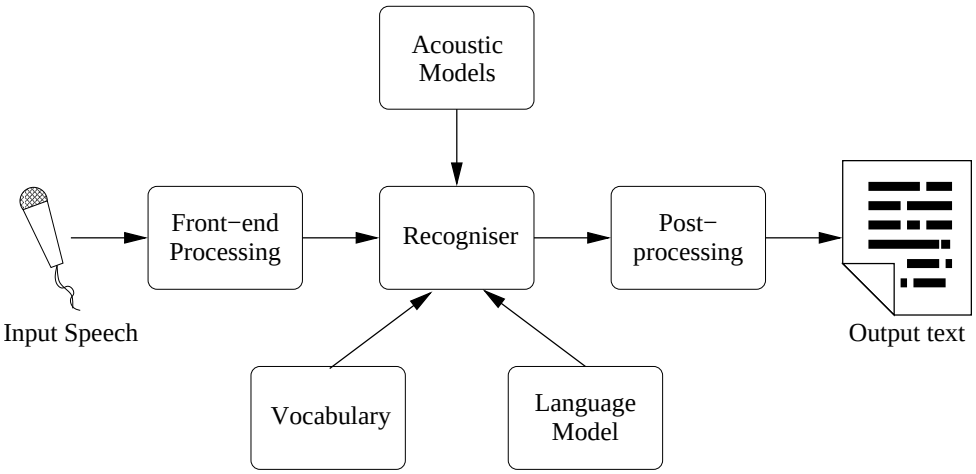

Figure 1.1 *A Generic Speech Recognition System*

Figure 1.1 illustrates a generic speech recognition system. This system accepts input speech waveforms, which are pre-processed to extract suitable features. These features are then fed into a recogniser to generate a sequence of word hypotheses. The recogniser consists of a set of acoustic models, a language model and the vocabulary (lexicon). Each of the blocks shown in Figure 1.1 will be described briefly in the following.

1.1.1 Front-end Processing

Speech waveforms are usually sampled by hardware devices (e.g. a PC sound card) into digital signals. Digital signals are represented in binary format (bits stream). So, speech signals are given by finite samples across time (time *discretisation*) and each sample is represented by discrete values (value *quantisation*). Speech samples in digital form are typically characterised by

[3] http://www.nist.gov

the sampling frequency and sample size (e.g. a telephone speech is typically 8kHz and 16bits). An example speech waveform is shown in Figure 1.2

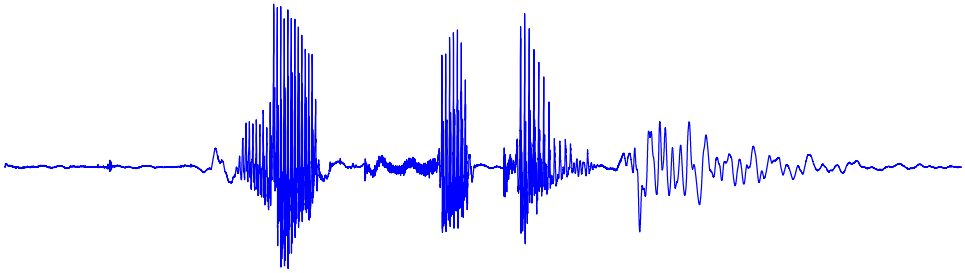

Figure 1.2 *An example speech waveform*

Short Time Fourier Transform (STFT) [60] is commonly applied to the speech waveforms to achieve a more compact representation. To do this, a Discrete Fourier Transform (DFT) [60] is applied to a window[4] of the speech signal to obtain a speech *frame*. The window is then slid forward in time to produce a series of speech frames. These are conveniently presented as a *spectogram*. This process is illustrated in Figure 1.3. The typical window length is 25 msec

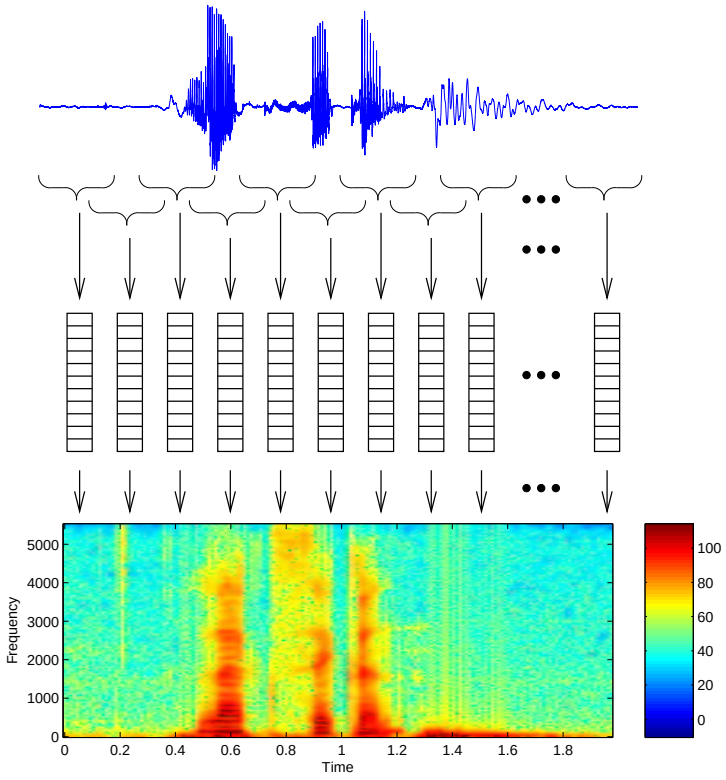

Figure 1.3 *Short Time Fourier Transform on a speech waveform to obtain a spectogram*

(milliseconds) and the window shift is 10 msec. Thus, there are 100 frames per second and

[4] A Hamming window is commonly used as a tapering function to reduce aliasing (the discontinuities at the frame edges). This process is also known as apodisation.

successive frames are overlapped by 15 msec to give a smooth transition. Each of these speech frames is *filtered* by a series of triangular bandpass filters to form a vector of *filter bank* coefficients. Two commonly used features based on the filter bank coefficients are the Mel Frequency Cepstral Coefficients (MFCC) [18] and Perceptral Linear Prediction (PLP) [55] coefficients. For MFCC, the bandpass filters are located according to the *Mel* scale, which is related to the linear frequency scale as follows:

$$f_{\texttt{mel}} = 1125 \log \left(1 + \frac{f_{\texttt{Hz}}}{625}\right) \tag{1.1}$$

where $f_{\texttt{mel}}$ is the warped frequency of $f_{\texttt{Hz}}$ on the Mel scale. The resulting filter bank coefficients are further transformed using the Discrete Cosine Transform (DCT) to reduce the spectral correlation between filter bank coefficients. The DCT transform is given by

$$o_{ti}^{\texttt{mfcc}} = \sqrt{\frac{2}{B}} \sum_{b=1}^{B} \log(o_{tb}^{\texttt{fb}}) \cos\left(\frac{i(b-0.5)\pi}{B}\right) \tag{1.2}$$

where $o_{tb}^{\texttt{fb}}$ and $o_{ti}^{\texttt{mfcc}}$ denote the bth filter bank coefficient and the ith Mel frequency cepstral coefficient respectively. For PLP frontends, the *Bark-frequency* scale is employed for frequency warping. This scale is given by

$$f_{\texttt{bark}} = \log \left(\frac{f_{\texttt{Hz}}}{600} + \sqrt{1 + \left(\frac{f_{\texttt{Hz}}}{600}\right)^2}\right) \tag{1.3}$$

Critical band filters are then used to compute the filter bank coefficients. Equal-loudness, pre-emphasis and intensity loudness power law are also applied. Finally, linear prediction (LP) coefficients are computed and transformed to the cepstral domain to obtain the PLP coefficients.

Usually, 12 coefficients (MFCC or PLP) are computed, along with the normalised log energy term or the zeroth order cepstral coefficient to yield a 13 dimensional *static* feature. *Dynamic* features [28] are often appended to improve the recognition performance. The first order dynamic features (also known as the delta coefficients) may be computed as follows:

$$\Delta \boldsymbol{o}_t = \frac{\sum_{d=1}^{D} d(\boldsymbol{o}_{t+d} - \boldsymbol{o}_{t-d})}{2\sum_{d=1}^{D} d^2} \tag{1.4}$$

where a window of $2D+1$ frames are used. The above equation may also be applied to the delta coefficients to obtain the delta-delta coefficients and so on. These higher order coefficients are appended to the static coefficients to form the feature vector for speech recognition. Typically, up to the second (or third order) dynamic coefficients are used which gives a total of 39 (or 52) dimensional feature vectors. This is particularly useful for HMM-based systems due to the *conditional independence* assumption (more details will follow in the next chapter).

Furthermore, to avoid having too many model parameters, diagonal covariance matrices are used to model the Gaussian PDFs of the acoustic features. This assumption is often

accompanied by some kind of projection schemes to achieve both dimension reduction and feature space decorrelation. For reasons of efficiency, linear projection schemes such as the Linear Discriminant Analysis (LDA) [105], Heteroscedastic Discriminant Analysis (HDA) [44] and Heteroscedastic LDA (HLDA) [68] are the popular choices.

1.1.2 Recogniser

Given a speech utterance, represented by the appropriate front-end (MFCC or PLP), the recogniser is required to obtain the most probable word sequence. Most systems are based on the HMM acoustic models in combination with n-gram language model. Also, a vocabulary along with the lexicon has to be defined. In other words, a recogniser will only produce words which are defined in the vocabulary. In a probabilistic framework, a recogniser seeks to uncover the most likely word sequence, $\hat{\mathcal{W}}_1^N$, given the observations and the underlying model, i.e.

$$\hat{\mathcal{W}}_1^N = \arg\max_{\mathcal{W}_1^N} P(\mathcal{W}_1^N|\mathcal{O}_1^T, \boldsymbol{\theta}) \tag{1.5}$$

where $\mathcal{W}_1^N$ and $\mathcal{O}_1^T$ denote the N-word sequence and the T-frame observation sequence respectively. $\boldsymbol{\theta}$ denotes the underlying model set. By using the Bayes' theorem and some simple manipulations, equation (1.5) can be rewritten as

$$\hat{\mathcal{W}}_1^N = \arg\max_{\mathcal{W}_1^N} P(\mathcal{O}_1^T|\mathcal{W}_1^N, \boldsymbol{\theta}^{\text{am}}) P(\mathcal{W}_1^N|\boldsymbol{\theta}^{\text{lm}}) \tag{1.6}$$

where $\boldsymbol{\theta}^{\text{am}}$ and $\boldsymbol{\theta}^{\text{lm}}$ denote the acoustic and language model parameters respectively.

A speech recognition system can operate in two different modes. The first mode is the *isolated* word recognition, where the speech samples are presented to the recogniser word by word (i.e. the word boundaries are known). The recogniser in this case is simple but requires the users to pause momentarily between words. This is unnatural and not suitable for many real life applications. Most applications operate in the second mode, *continuous speech* recognition. However, a continuous speech recogniser is more complex and requires more computational power, particularly for a large vocabulary system with a high order n-gram language model.

1. *Acoustic Models:*
   An acoustic model is used to represent a unit of speech represented by its acoustic characteristics. The choice of the speech units to be modelled depends on the applications. There is a trade-off between the choice of the speech units and the size of the resulting acoustic model set. For small and medium vocabulary isolated word recognition, a word or sub-word level representation may be used. However, for a large vocabulary system, a phone representation is more suitable. Often, context dependent models are also used to capture the *co-articulation* effects in speech.

2. *Language Models:*
   A general form of a stochastic language model may be used to model the sentence probabilities as follows:

$$P(\mathcal{W}_1^N|\boldsymbol{\theta}^{\mathrm{lm}}) = P(\mathrm{w}_1|\boldsymbol{\theta}^{\mathrm{lm}}) \prod_{i=2}^{N} P(\mathrm{w}_i|\mathcal{W}_1^{i-1}, \boldsymbol{\theta}^{\mathrm{lm}}) \tag{1.7}$$

   where w_i represents the ith word of the word sequence, $\mathcal{W}_1^N$. The most common language model is the n-gram due to its simple integration into the HMM-based recogniser. An n-gram model simply approximate the probability of a word is dependent on the $n-1$ most recent history. Mathematically,

$$P(\mathrm{w}_i|\mathcal{W}_1^{i-1}, \boldsymbol{\theta}^{\mathrm{lm}}) \approx P(\mathrm{w}_i|\mathrm{w}_{i-1}, \mathrm{w}_{i-2}, \ldots, \mathrm{w}_{i-n+1}, \boldsymbol{\theta}^{\mathrm{lm}}) \tag{1.8}$$

   Typical forms of n-gram used are the unigram ($n = 1$), bigram ($n = 2$) and trigram ($n = 3$).

3. *Vocabulary/Lexicon:*
   The vocabulary is the set of possible words from which the recogniser selects the best word sequence. Therefore, the recogniser only produces words which are included in the vocabulary. Words which are outside the vocabulary set are termed Out-Of-Vocabulary (OOV) words. OOV word rate is measured against a corpus of text that represents the domain within which the recogniser will operate. The size of the vocabulary directly impacts the performance of the recognition system. Increasing the vocabulary size reduces the OOV rate and hence improves the recognition performance, but at the same time increases the search space, making the recogniser run slower. Typically, speech recognition tasks are broadly categorised into three major groups, according to the vocabulary sizes:

   - small vocabulary ($<$ 1k words)
   - medium vocabulary (1k – 10k words)
   - large vocabulary ($>$ 10k words)

   A lexicon, on the other hand, defines the mapping between the words and the acoustic unit representations. For examples, the mapping between the word `lexicon` and its phoneme representation is given by

`lexicon` $\Rightarrow$ **l eh k s ih k aa n**

Figure 1.4: *Phoneme representation of the word "lexicon"*

   For word level acoustic models, the mapping is trivial.

1.1.3 Post-processing

Post-processing is an optional stage. There are many forms of post processing, aiming at producing outputs which can be used by other tasks (e.g. machine translation, speech understanding etc) or simply making the outcome more useful to the end users. For examples, certain applications may decide to convert the text to a more useful format, such as HTML for web browsing or adding punctuations and capitalisation for report generation and so on. Recently, NIST conducted a series of Rich Transcription evaluations [81] whose goal is to produce transcriptions which are more readable by humans and more useful for machines. Down stream processing of the output generated from a speech recogniser can be an independent research topic by itself.

1.2 Organisation of Thesis

The rest of this thesis is organised as follows.

Chapter 2 gives an overview of HMM-based speech recognition systems. The mathematical formulation of HMMs is provided giving particular emphasis in its use in statistical pattern classification and speech recognition. Specifically, the evaluation, decoding and training algorithms for HMMs are briefly described. Finally, this chapter also describes the various commonly employed techniques in state-of-the-art HMM-based LVCSR systems.

Chapter 3 motivates the importance of correlation modelling for multivariate Gaussian distributions in statistical pattern classification. This chapter also describes several forms of covariance and precision approximation schemes. A generic framework of basis superposition for precision matrix modelling is introduced. This framework subsumes several existing precision matrix models, including the Semi-tied Covariance (STC) [32, 34], Extended Maximum Likelihood Linear Transform (EMLLT) [85, 86] and Subspace for Precision And Mean (SPAM) [7].

Chapter 4 presents the maximum likelihood parameter estimation for various precision matrix models. The re-estimation formulae are derived within the Expectation Maximisation (EM) framework. Implementation issues of these models are also detailed, focusing on optimising the memory and computational requirements for LVCSR systems.

Chapter 5 discusses parameter estimation in a discriminative training framework. In contrast to the previous work on MMI training of EMLLT and SPAM models, this chapter is devoted to the discussion of Minimum Phone Error (MPE) training of various precision matrix models. The implementation issues related to the MPE training process are also discussed.

Chapter 6 concerns the adaptation and adaptive training techniques with these more com-

plex precision matrix models. Linear transformation based techniques such as Maximum Likelihood Linear Regression (MLLR) [40, 70] techniques will be described. Estimation of the MLLR transformation matrices for precision matrix models does not have a closed-form solution and numerical optimisation schemes were used in the past. This chapter introduces an efficient iterative row-by-row update for the MLLR mean and Constrained MLLR (CMLLR) adaptations. Implementation issues to improve the memory and computational requirements are discussed.

Chapter 7 investigates a semi-parametric trajectory model. This model is formulated such that the Gaussian parameters are modelled with temporal shifting of the mean vectors and temporal scaling of the diagonal precision matrix elements based on the posteriors of a set of centroids. The former yields the fMPE technique introduced by Povey [97]. The latter is a novel approach to modelling the temporal aspects of the precision matrix structures, called pMPE [112].

Chapter 8 and 9 present the experimental results for various techniques introduced in this thesis. Performance are evaluated on a number of different tasks using different languages. Specifically, the Conversational Telephone Speech (CTS) English and Mandarin; and Broadcast News (BN) English transcription tasks are used.

2

Hidden Markov Model Speech Recognition

2.1 Overview

HMMs [102] are the most popular and successful statistical acoustic models for speech recognition. This chapter will describe the mathematical formulation of the HMM and how it can be used as the acoustic model for speech recognition. The HMM is a finite state transducer comprising a variable number of discrete states. The standard HMM makes the following assumptions:

- **Instantaneous first-order transition:**
  The probability of making a transition to the next state q_{t+1} is independent of the historical states, $Q_1^{t-1} = \{q_1, q_2, \ldots, q_{t-1}\}$, given the current state, q_t.
- **Conditional independence assumption:**
  The probability of observing $\boldsymbol{o}_t$ at time t is independent of the historical observations, $\mathcal{O}_1^{t-1}$, and the states, Q_1^{t-1} given the current state, q_t.

Thus, the generative model of the standard HMM is given by

$$q_{t+1} \sim P(q_{t+1}|q_t) \tag{2.1}$$
$$\boldsymbol{o}_t \sim p(\boldsymbol{o}_t|q_t) \tag{2.2}$$

Figure 2.1: *A generative hidden Markov model*

where q_t and $\boldsymbol{o}_t$ are the state and observation vector respectively at time t. The parameters of a HMM with N discrete states are given by $\boldsymbol{\theta}(\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{b})$ where $\boldsymbol{\pi} = \{\pi_i : 1 \leq i \leq N\}$ are the initial probabilities of the states, $\boldsymbol{A} = \{a_{ij} : 1 \leq i, j \leq N\}$ are the transition probabilities between two

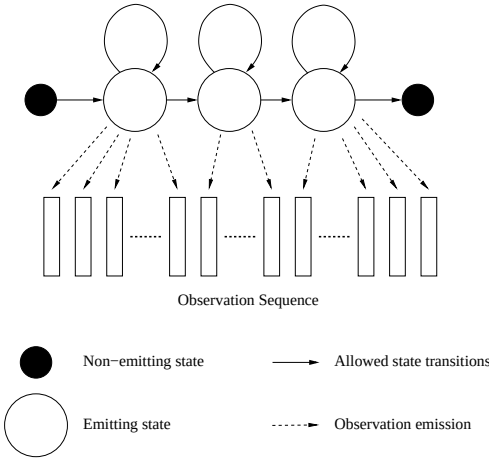

Figure 2.2 *A left-to-right hidden Markov model*

states and $\boldsymbol{b} = \{b_j(\boldsymbol{o}_t) : 1 \leq j \leq N\}$ are the observation probabilities of the states. A typical *left-to-right* HMM topology used in a speech recognition system is illustrated in Figure 2.2. Two special *non-emitting* states are used to represent the left-to-right topology. By having the non-emitting start state[1], the initial probabilities are simply $\pi_1 = 1$ and $\pi_i = 0$ for $i \neq 1$. The arrow joining two states indicates the permissible transitions in the given direction. The HMM in Figure 2.2 assumes that the speech signals being modelled can be logically divided into three segments, within each, the signals are considered i.i.d (independent and identically distributed) and *piece-wise stationary*.

In order to use the HMMs in speech recognition, one needs to be able to:

- *evaluate* the likelihood of the model given the observations
- *decode* the most likely state sequence given the observations
- *estimate* the HMM parameters to maximise the objective function.

These three aspects will be discussed in the following sections.

[1]Some literatures describe HMMs without the non-emitting start state. Instead, an initial probability is assigned to each emitting state directly

2.1.1 Evaluation

To evaluate an HMM given the observation sequence, $\mathcal{O}_1^T$, we need to be able to compute $p(\mathcal{O}_1^T|\boldsymbol{\theta})$. This is given by

$$p(\mathcal{O}_1^T|\boldsymbol{\theta}) = \sum_{Q_1^T} p(\mathcal{O}_1^T, Q_1^T|\boldsymbol{\theta}) = \sum_{Q_1^T} \prod_{t=1}^{T} P(q_t, q_{t-1}|\boldsymbol{\theta}) p(\boldsymbol{o}_t, q_t|\boldsymbol{\theta}) \tag{2.3}$$

For simplicity, the initial state probabilities are denoted as $\pi_i = P(q_1 = i|q_0)$. Equation (2.3) can be calculated efficiently using a forward recursion:

$$\begin{aligned}
\alpha_j(t) &= p(\mathcal{O}_1^t, q_t = j|\boldsymbol{\theta}) \\
&= p(\boldsymbol{o}_t|q_t = j, \boldsymbol{\theta}) \sum_{i=1}^{N} P(q_t = j|q_{t-1} = i, \boldsymbol{\theta}) p(\mathcal{O}_1^{t-1}, q_{t-1} = i|\boldsymbol{\theta}) \\
&= p(\boldsymbol{o}_t|q_t = j, \boldsymbol{\theta}) \sum_{i=1}^{N} P(q_t = j|q_{t-1} = i, \boldsymbol{\theta}) \alpha_i(t-1) \\
&= b_j(\boldsymbol{o}_t) \sum_{i=1}^{N} a_{ij} \alpha_i(t-1)
\end{aligned} \tag{2.4}$$

where $\alpha_i(()t)$ is the probability of being in state j at time t, after observing the observation sequence $\mathcal{O}_1^T 1t$ given the HMM parameters and the initial conditions are given by

$$\alpha_j(0) = \begin{cases} 1 & \text{for } j = 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

since the HMMs have to begin in the start state, $(j = 1)$. Also, since the HMMs have to end in the exit state, $p(\mathcal{O}_1^T|\boldsymbol{\theta}) = \alpha_N(T)$.

2.1.2 Viterbi Decoding

To decode an observation sequence using HMM is to find the best state sequence which emitted those observations, *i.e.*

$$\hat{Q}_1^T = \arg\max_{Q_1^T} P(\mathcal{O}_1^T, Q_1^T|\boldsymbol{\theta}) \tag{2.6}$$

where the optimum state sequence, $\hat{Q}_1^T$, can be found using the Viterbi algorithm. This algorithm can be efficiently performed using the following recursion:

$$\begin{aligned}
v_j(t) &= p(\mathcal{O}_1^t, Q_1^{t-1}, q_t = j|\boldsymbol{\theta}) \\
&= \max_{1 \le i \le N} \left[P(q_t = j|q_{t-1} = i, \boldsymbol{\theta}) v_i(t-1) \right] P(\boldsymbol{o}_t|q_t = j, \boldsymbol{\theta}) \\
&= \max_{1 \le i \le N} \left[a_{ij} v_i(t-1) \right] b_j(\boldsymbol{o}_t)
\end{aligned} \tag{2.7}$$

where $v_j(t)$ is the probability of the most likely partial state sequence, Q_1^{t-1}, which has generated the partial observation sequence, $\mathcal{O}_1^t$ and ends in state j at time t. The initial conditions are given by

$$v_j(0) = \begin{cases} 1 & \text{for } j = 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.8}$$

To obtain the best state sequence, the previous state which has generated the best partial observation sequence, $\mathcal{O}_1^t$ and ends in state j at time t has to be recorded for all $1 \leq t \leq T$ and $1 \leq j \leq N$. Let this be denoted by $q_j^{\max}(t)$, where

$$\begin{aligned} q_j^{\max}(t) &= \underset{1 \leq i \leq N}{\arg\max} \left[P(q_t = j | q_{t-1} = i, \boldsymbol{\theta}) v_i(t-1) \right] \\ &= \underset{1 \leq i \leq N}{\arg\max} \left[a_{ij} v_i(t-1) \right] \end{aligned} \tag{2.9}$$

The best final state is given by $\hat{q}_T = v_N(T)$ and the entire best state sequence can be retrieved by *backtracking* from $\hat{q}_T$ using the following backward recursion:

$$\hat{q}_t = q_{\hat{q}_{t+1}}^{\max}(t+1) \tag{2.10}$$

The best sequence is then given by $\hat{Q}_1^T = \{\hat{q}_1, \hat{q}_2, \ldots, \hat{q}_T\}$.

2.1.3 Maximum Likelihood Parameters Estimation

Estimation of the HMM parameters is the hardest among the three. In this section, Maximum Likelihood (ML) criterion will be used to estimate the HMM parameters. The optimum parameters are given by maximising the following *log likelihood* function:

$$\mathcal{R}^{\text{ml}}(\boldsymbol{\theta}) = \log p(\mathcal{O}_1^T | \boldsymbol{\theta}) \tag{2.11}$$

Direct optimisation of the ML objective function in equation (2.11) is difficult. However, an efficient Baum-Welch algorithm[2] [11] can be applied to maximise equation (2.11) using an iterative approach. In this method, an auxiliary function (also known as the Q-function), $\mathcal{Q}^{\text{ml}}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}})$, is defined as

$$\begin{aligned} \mathcal{Q}^{\text{ml}}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) &= \mathbb{E}_{\boldsymbol{\theta}} \left[\log p(\mathcal{O}_1^T, Q_1^T | \hat{\boldsymbol{\theta}}) \right] \\ &= \sum_{Q_1^T} P(Q_1^T | \mathcal{O}_1^T, \boldsymbol{\theta}) \log p(\mathcal{O}_1^T, Q_1^T | \hat{\boldsymbol{\theta}}) \\ &= \sum_{Q_1^T} \frac{P(Q_1^T, \mathcal{O}_1^T | \boldsymbol{\theta})}{P(\mathcal{O}_1^T | \boldsymbol{\theta})} \left(\log p(Q_1^T, \hat{\boldsymbol{\theta}}^{\text{trans}}) + \log p(\mathcal{O}_1^T | Q_1^T, \hat{\boldsymbol{\theta}}^{\text{obs}}) \right) \end{aligned} \tag{2.12}$$

[2] Baum-Welch algorithm is also known as the Forward-backward algorithm, which is a form of Expectation Maximisation (EM) algorithm [19]

where $\hat{\boldsymbol{\theta}}^{\text{trans}}$ and $\hat{\boldsymbol{\theta}}^{\text{obs}}$ are the model parameters associated with the state transition probabilities and the observation emission probabilities respectively. $\mathbb{E}_{\boldsymbol{\theta}}$ denotes the expectation over Q_1^T, given the observation sequence, $\mathcal{O}_1^T$, based on the current parameter set, $\boldsymbol{\theta}$. This auxiliary function satisfies the following inequality

$$\mathcal{R}^{\text{ml}}(\hat{\boldsymbol{\theta}}) - \mathcal{R}^{\text{ml}}(\boldsymbol{\theta}) \geq \mathcal{Q}^{\text{ml}}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) - \mathcal{Q}^{\text{ml}}(\boldsymbol{\theta}, \boldsymbol{\theta}) \tag{2.13}$$

which is obtained by applying the Jensen's inequality [19]. The inequality in equation (2.13) shows that increasing the auxiliary function guarantees an increase in the objective function unless a *local* maximum is reached. Furthermore, equation (2.12) shows that the transition and observation parameters can be maximised separately based on each of the summation terms in the equation. Thus, the iterative Baum-Welch algorithm can be summarised as follows:

```
Initialise θ;
while [increase in Q^ml(θ, θ̂) > threshold]
do
   E-step: Compute Q^ml(θ, θ̂);
   M-step: Estimate θ := arg max_θ̂ Q^ml(θ, θ̂);
done
```

Figure 2.3: *The Baum-Welch algorithm*

The E-step of the algorithm requires the calculation of $p(Q_1^T, \mathcal{O}_1^T|\boldsymbol{\theta})$ for all possible state transitions and the observation vectors. This can be done efficiently using both the forward and backward recursion [3]. Similar to the forward probabilities defined in equation 2.4, a *backward* probability can also be defined in a similar fashion:

$$\begin{aligned}
\beta_j(t) &= p(\mathcal{O}_{t+1}^T|q_t = j, \boldsymbol{\theta}) \\
&= \sum_{i=1}^{N} p(\boldsymbol{o}_{t+1}|q_t = i, \boldsymbol{\theta}) P(q_t = i|q_{t-1} = j, \boldsymbol{\theta}) p(\mathcal{O}_{t+2}^T|q_{t-1} = i, \boldsymbol{\theta}) \\
&= \sum_{i=1}^{N} p(\boldsymbol{o}_{t+1}|q_t = i, \boldsymbol{\theta}) P(q_t = i|q_{t-1} = j, \boldsymbol{\theta}) \beta_i(t+1) \\
&= \sum_{i=1}^{N} b_i(\boldsymbol{o}_{t+1}) a_{ji} \beta_i(t+1)
\end{aligned} \tag{2.14}$$

The initial conditions are given by

$$\beta_j(T) = \begin{cases} \sum_{i=1}^{N} a_{ij} & \text{for } j = N \\ 0 & \text{otherwise} \end{cases} \tag{2.15}$$

[3] This is why the algorithm is named Forward-backward algorithm.

since the model must end in the exit state ($j = N$). This allows us to compute two quantities which are essential for the estimation of the HMM parameters. These quantities are:

$$\gamma^{\text{ml}}_{(i,j)}(t) = \frac{P(q_{t-1} = i, q_t = j, \mathcal{O}_1^T|\boldsymbol{\theta})}{p(\mathcal{O}_1^T|\boldsymbol{\theta})} = \frac{\alpha_i(t-1)a_{ij}b_j(\boldsymbol{o}_t)\beta_j(t)}{\alpha_N(T)} \tag{2.16}$$

$$\gamma^{\text{ml}}_{j}(t) = \frac{p(q_t = j, \mathcal{O}_1^T|\boldsymbol{\theta})}{p(\mathcal{O}_1^T|\boldsymbol{\theta})} = \frac{\alpha_j(t)\beta_j(t)}{\alpha_N(T)} \tag{2.17}$$

$\gamma^{\text{ml}}_{(i,j)}(t)$ represents the posterior probability of making a transition from state i at time $t-1$ to state j at time t. Meanwhile, $\gamma^{\text{ml}}_{j}(t)$ represents the posterior probability of being in state j at time t.

Given $\gamma^{\text{ml}}_{(i,j)}(t)$ and $\gamma^{\text{ml}}_{j}(t)$, the auxiliary function is then maximised to yield the optimum model parameters. For standard HMMs, the auxiliary function may be optimised efficiently with a closed form solution. However, when more complex acoustic modelling techniques are employed, the auxiliary function may not have a closed-form solution or is simply not trivial to solve. Under these circumstances, a modified algorithm known as the Generalised Expectation Maximisation (GEM) [19] algorithm may be used, where the parameters are estimated in the M-step such that the auxiliary function is *improved* (not necessary to find the maximum). Every GEM iteration also guarantees an improvement in the objective function (from equation 2.13). In the following sections, the estimation formulae for the transition probabilities, discrete HMM output probabilities and the Gaussian Mixture Model (GMM) parameters for Continuous Density HMMs (CDHMMs) will be derived using the EM approach.

2.1.3.1 Estimation of the transition probabilities

Considering the terms in equation (2.12) which are dependent on the transition parameters, the auxiliary function can be rewritten as

$$\begin{aligned}\mathcal{Q}^{\text{ml}}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) &= K_{\text{trans}} + \sum_{Q_1^T} \frac{p(Q_1^T, \mathcal{O}_1^T|\boldsymbol{\theta})}{p(\mathcal{O}_1^T|\boldsymbol{\theta})} \log p(Q_1^T, \hat{\boldsymbol{\theta}}^{\text{trans}}) \\ &= K_{\text{trans}} + \sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{j=1}^{N} \frac{p(q_{t-1} = i, q_t = j, \mathcal{O}_1^T|\boldsymbol{\theta})}{p(\mathcal{O}_1^T|\boldsymbol{\theta})} \log a_{ij}\end{aligned} \tag{2.18}$$

where K_{trans} denotes terms independent of the transition parameters. Equation (2.18) needs to be maximised with respect to the transition parameters, $\{a_{ij} : 1 \leq i, j \leq N\}$, subject to the following constraints:

$$\sum_{j=1}^{N} a_{ij} = 1 \quad \text{for } 1 \leq i \leq N \tag{2.19}$$

The analytic solution can be found by using the Lagrange multipliers. The update of the transition parameters is thus given by

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T} \gamma^{\text{ml}}_{(i,j)}(t)}{\sum_{t=1}^{T}\sum_{k=1}^{N} \gamma^{\text{ml}}_{(i,k)}(t)} \tag{2.20}$$

2.1.3.2 Estimation of the Discrete HMMs parameters

For a discrete HMM, the observation vectors are represented by k discrete symbols, $\{\tilde{o}_k : 1 \leq k \leq K\}$, using a Vector-Quantisation (VQ) table. Thus the probability mass function is thus given by $p(\boldsymbol{o}_t = \tilde{o}_k | q_t = j, \boldsymbol{\theta}) = b_j(k)$. These probabilities can be estimated by maximising the following auxiliary function

$$\begin{aligned} \mathcal{Q}^{\mathrm{ml}}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) &= K_{\mathrm{obs}} + \sum_{Q_1^T} \frac{p(Q_1^T, \mathcal{O}_1^T|\boldsymbol{\theta})}{p(\mathcal{O}_1^T|\boldsymbol{\theta})} \log p(\mathcal{O}_1^T|Q_1^T, \hat{\boldsymbol{\theta}}^{\mathrm{obs}}) \\ &= K_{\mathrm{obs}} + \sum_{j=1}^{N} \sum_{k=1}^{K} \sum_{t \in \boldsymbol{O}_t = \tilde{o}_k} \frac{p(q_t = j, \mathcal{O}_1^T|\boldsymbol{\theta})}{p(\mathcal{O}_1^T|\boldsymbol{\theta})} \log b_j(k) \end{aligned} \tag{2.21}$$

subject to the following sum-to-one constraint to form a valid probability mass function

$$\sum_{k=1}^{K} b_j(k) = 1 \quad \text{for } 1 \leq j \leq N \tag{2.22}$$

K_{obs} subsumes terms independent of the observation probability parameters. Again, using the Lagrange multipliers, the discrete probabilities are given by

$$\hat{b}_j(k) = \frac{\sum_{t \in \boldsymbol{O}_t = \tilde{o}_k} \gamma_j^{\mathrm{ml}}(t)}{\sum_{t=1}^{T} \gamma_j^{\mathrm{ml}}(t)} \tag{2.23}$$

2.1.3.3 Estimation of the Continuous Density HMMs (CDHMMs) parameters

Continuous density HMMs (CDHMMs) are more popular and they have been shown to yield better performance compared to a discrete HMM system. The probability density function (pdf) of the state dependent observation distribution is typically represented by a Gaussian Mixture Model (GMM). Alternative forms of pdf have also been investigated, for examples the Laplacian distribution or the Gamma distribution. Here, the use of GMMs as the pdf will be described since all the work in this thesis is based on HMM systems with GMM distributions.

A GMM models the output probability of the observation, $\boldsymbol{o}_t$, given the HMM state, s, as

$$b_s(\boldsymbol{o}_t) = \sum_{m=1}^{M} c_{sm} \mathcal{N}(\boldsymbol{o}_t; \boldsymbol{\mu}_{sm}, \boldsymbol{\Sigma}_{sm}) \tag{2.24}$$

where $\mathcal{N}(\boldsymbol{o}_t; \boldsymbol{\mu}_{sm}, \boldsymbol{\Sigma}_{sm})$ represents a Gaussian distribution given by

$$\mathcal{N}(\boldsymbol{o}_t; \boldsymbol{\mu}_{sm}, \boldsymbol{\Sigma}_{sm}) = \frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}_{sm}|}} \exp\left\{ -\frac{1}{2} (\boldsymbol{o}_t - \boldsymbol{\mu}_{sm})' \boldsymbol{\Sigma}_{sm}^{-1} (\boldsymbol{o}_t - \boldsymbol{\mu}_{sm}) \right\} \tag{2.25}$$

with the component prior, c_{sm}, mean vector, $\boldsymbol{\mu}_{sm}$, and covariance matrix, $\boldsymbol{\Sigma}_{sm}$ for component m of state j. The GMM parameters, c_{sm}, $\boldsymbol{\mu}_{sm}$ and $\boldsymbol{\Sigma}_{sm}$ can be estimated by maximising

$$\mathcal{Q}^{\mathrm{ml}}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) = K_{\mathrm{obs}} + \sum_{s=1}^{N} \sum_{t=1}^{T} \frac{p(q_t = s, \mathcal{O}_1^T|\boldsymbol{\theta})}{p(\mathcal{O}_1^T|\boldsymbol{\theta})} \log b_s(\boldsymbol{o}_t) \tag{2.26}$$

subject to the constraints

$$\sum_{m=1}^{M} c_{sm} = 1 \qquad \text{and} \qquad \int_{-\infty}^{\infty} b_s(\boldsymbol{o}_t) d\boldsymbol{o}_t = 1 \tag{2.27}$$

for $1 \leq s \leq N$. Using constrained maximisation with the Lagrange multipliers, the update formulae for the GMM parameters are given by

$$\hat{c}_{sm} = \frac{\sum_{t=1}^{T} \gamma_{sm}^{\mathrm{ml}}(t)}{\sum_{t=1}^{T} \sum_{k=1}^{M} \gamma_{sk}^{\mathrm{ml}}(t)} \tag{2.28}$$

$$\hat{\boldsymbol{\mu}}_{sm} = \frac{\sum_{t=1}^{T} \gamma_{sm}^{\mathrm{ml}}(t) \boldsymbol{o}_t}{\sum_{t=1}^{T} \gamma_{sm}^{\mathrm{ml}}(t)} \tag{2.29}$$

$$\hat{\boldsymbol{\Sigma}}_{sm} = \frac{\sum_{t=1}^{T} \gamma_{sm}^{\mathrm{ml}}(t) \left(\boldsymbol{o}_t - \hat{\boldsymbol{\mu}}_{sm}\right) \left(\boldsymbol{o}_t - \hat{\boldsymbol{\mu}}_{sm}\right)'}{\sum_{t=1}^{T} \gamma_{sm}^{\mathrm{ml}}(t)} \tag{2.30}$$

where the probability of being in state s and component m at time t is given by

$$\gamma_{sm}^{\mathrm{ml}}(t) = \frac{p(q_t = s, k_t = m, \mathcal{O}_1^T | \boldsymbol{\theta})}{p(\mathcal{O}_1^T | \boldsymbol{\theta})} = \sum_{i=1}^{N} \frac{\alpha_i(t-1) a_{ij} c_{sm} b_s m(\boldsymbol{o}_t) \beta_s(t)}{\alpha_N(T)} \tag{2.31}$$

k_t denotes the component at time t. $b_{sm}(\boldsymbol{o}_t) = \mathcal{N}\left(\boldsymbol{o}_t; \boldsymbol{\mu}_{sm}, \boldsymbol{\Sigma}_{sm}\right)$ is the Gaussian distribution of component m of state s.

2.2 State-of-the-art LVCSR systems

State-of-the-art large vocabulary continuous speech recognition (LVCSR) systems employ statistical CDHMMs and n-gram language models. These systems are complex and employ many advanced techniques to improve the recognition performance and to reduce the run time of the systems. The total number of distinct states in an LVCSR system is large, ranging from thousands to tens of thousands. Furthermore, virtually all CDHMM systems use multivariate GMMs of high dimensionality (typically 39 or 52) as the output probability distribution. Therefore, there may be in total more than 100,000 Gaussian components in the system. The elements of the feature vector are assumed to be *uncorrelated* so that each Gaussian component has a diagonal covariance matrix structure. This not only reduces the total number of free parameters in the system by a significant amount, the likelihood computation of the Gaussian components is also more efficient. The former is crucial to ensure robust parameters estimation in order to avoid *over-training* (or over-fitting) issues associated with the data-sparseness problems. The latter is also important to obtain efficient run time for the recogniser. In the following, several commonly employed state-of-the-art techniques for LVCSR systems will be described briefly.

2.2.1 Context Dependent Models and Decision Tree State Clustering

Many speech recognition systems use HMMs to model the phones of speech. Due to the *co-articulation*[4] effects, context-dependent phone models are typically used. A context can be a single phone to the left or (and) to the right of the current phone. Table 2.1 summarises several

Context-dependent models	# of contexts	
	left	right
left biphone	1	0
right biphone	0	1
triphone	1	1
quinphone	2	2
septaphone	3	3

Table 2.1 *Different types of context dependent phone models*

types of context dependent models used in the speech recognition systems. Among all, triphone models are the most popular. The lexicon of the word "lexicon"

`lexicon` ⇒ **l+eh l-eh+k eh-k+s k-s+ih s-ih+k ih-k+aa k-aa+nn k- n**

Figure 2.4: *Triphone representation of the word "lexicon"*

Recent development of the state-of-the-art systems reveals that is it possible to build LVCSR systems with higher context dependencies within a reasonable runtime limits (10xRT and 20xRT). For examples, the Cambridge University Engineering Department (CUED) recent LVCSR systems [22, 23, 24, 37, 38, 65, 65, 66, 67] used quinphone models with the HTK (HMM Toolkit) software [134]. Besides, the IBM RT04 system also uses the septaphone HMMs.

Since the number of triphones in English is large (more than 100,000), there will be many *unseen* triphones and triphones with very few examples in the training data. To ensure the robustness of system trainability, HMM states with similar output distributions may be grouped together so that each distinct triphone receives a reasonable amount of training data and the *unseen* triphones may be clustered to an appropriately trained triphone. State clustering is commonly achieved using a decision tree [8, 135]. A top-down binary split decision tree is depicted in Figure 2.5. A decision tree is built for each centre phone. Expert knowledge may be incorporated by asking some linguistic related questions about its contexts at each node. The decision tree state clustering algorithm starts by grouping all the context-dependent models with the same centre phone at the root node of the tree. The best question which results in the largest decrease in entropy or increase in likelihood is chosen to split the root node. This step is applied

[4] Co-articulation is the effect where the sound of a phone is influenced by the neighbouring phones in connected speech.

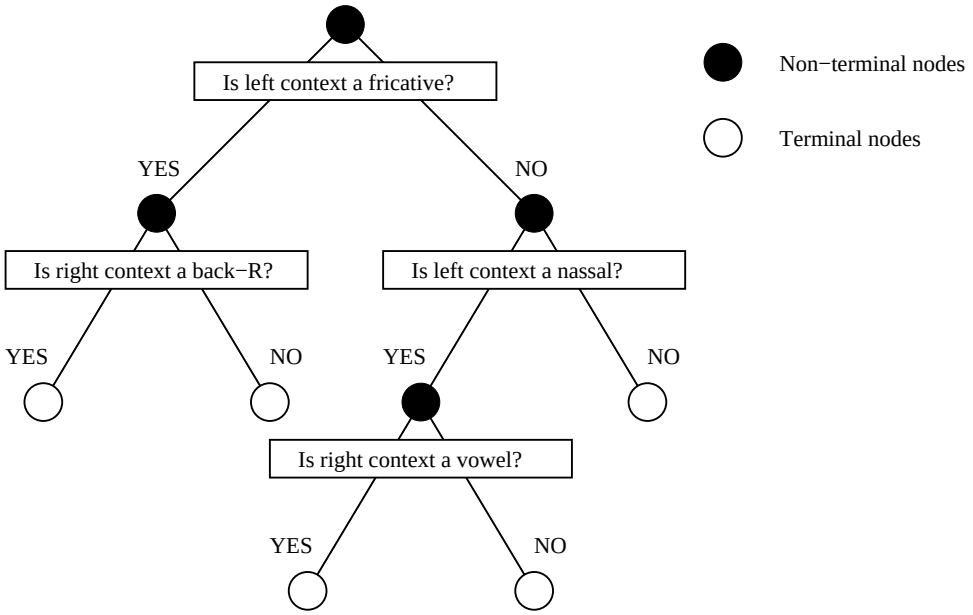

Figure 2.5 *A top-down binary split decision tree for state clustering.*

recursively to the terminal nodes of the tree until the number of examples associated with each terminal node is below a prescribed threshold.

2.2.2 Discriminative Training

Traditionally, model parameters are estimated based on a maximum likelihood estimator (MLE). The MLE solution can be formally written as

$$\hat{\boldsymbol{\theta}}^{\mathrm{ml}} = \arg\max_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}|\mathcal{O}_1^T) \tag{2.32}$$

where $\mathcal{L}(\boldsymbol{\theta}|\mathcal{O}_1^T)$ is the likelihood of the model parameter set, $\boldsymbol{\theta}$ given the complete set of observation vectors, $\mathcal{O}_1^T$. Thus, the ML estimator finds a set of parameters such that the model best fit the observation data. However, it strongly depends on the assumption that the observed data was in fact generated by the proposed model. In speech recognition, HMMs are commonly employed with the observation density function represented by a GMM. However, HMMs may not be the correct model for the speech data. Furthermore, the observation distribution may not be appropriately modelled using GMM. Moreover, according to the iterative Baum-Welch parameter estimation method using the maximum likelihood criterion as described in Section 2.1.3, the model parameters associated with each HMM in the system are estimated independent of the other HMMs once the auxiliary function has been computed. If the underlying model assumption was incorrect, the newly estimated parameters may also increase its likelihood given the observations generated by other HMMs. Clearly, this does not improve the discrimination power of the system. To overcome this problem, a new form of training paradigm is required, one

which is more closely related to the primary objective of the speech recognition task – to reduce the recognition error rate. Recall that in speech recognition and other statistical classification tasks, the aim is to select the best model (or sequence of models) from a large set of competing models. Thus, the model parameters should be estimated by maximising the ability to discriminate the correct models from the ones that potentially lead to mis-classifications. Parameter estimation based on this form of training scheme is known as *discriminative* training.

In the past two decades, several forms of discriminative training criteria have been proposed and were found to outperform the conventional maximum likelihood training scheme. One of the earliest discriminative training schemes was the Maximum Mutual Information (MMI) scheme proposed by *Bahl et al.* , 1986 [9]. MMI estimates the HMM parameters such that the mutual information between the models and the training data is maximised. This is equivalent to maximising the *a posteriori* probability that each training utterance was generated by the corresponding model. Following that, further research on MMI training for speech recognition had been carried out by many other researchers. The early MMI training was applied to isolated word recognition by Bahl [9] and Brown [14]. Merialdo [78], Normandin [82, 84] and *Kapadia et al.* [64] also successfully applied MMI training to small vocabulary continuous speech recognition. Next, the MMI training was extended to the Large Vocabulary Continuous Speech Recognition (LVCSR) by *Normandin et al.* [83] (1994), *Valtchev et al.* [123] (1996) and *Woodland and Povey* [131, 132] (2000). The MMI training paradigm is very closely related to the Conditional Maximum Likelihood Estimation (CMLE) [79, 80]. As pointed out in [122], the MMI and CMLE methods are slightly different in that the former includes the language model probability of the training utterances in the objective function. However, almost all MMI training experiments were based on a fixed language model which means that the MMI and CMLE schemes yields the same parameter estimation. In addition to MMI training, many other discriminative training paradigms for speech recognition have been investigated in the past. Examples of the earlier research on discriminative training include the Minimum Discrimination Information (MDI) [20, 21], the H-criterion [49], Minimum Classification Error (MCE) [27, 62, 75] and the Frame Discrimination (FD) [98, 99] schemes. More recent development of discriminative training methods include the Minimum Phone Error (MPE) [100] and Minimum Word Error (MWE) [95] criteria. In particular, MPE has been found to yield consistent gain over MMI for LVCSR systems [95]. In Chapter 5, discriminative training of precision matrix models using MPE criterion will be presented.

2.2.3 Speaker Adaptive Training and Adaptation

Adaptation techniques are of great importance to speech recognition systems. These techniques are used to reduce the *mismatch* in the systems. There are various kinds of mismatch in speech recognition systems. For example, the acoustic data used to train the system may be collected

from a set of speakers which are different from the actual user of the system. Besides, the environmental conditions may also differ. The mismatch between training and testing conditions may deteriorate the system performance by a considerable amount. Furthermore, there can also be mismatch *within* the training data itself. Such mismatch are due to the *non-homogeneity* of the training data. As discussed earlier in Chapter 1, it is often difficult to build many condition-specific models to cope with the wide range of operating conditions. Thus, a common approach is to build a general (condition independent) system using training data covering the range of conditions of interest. As a result, the training data may be highly non-homogeneous. These non-homogeneity can be reduced by employing the *adaptive training* schemes.

Speaker adaptation and adaptive training techniques are employed in most speech recognition systems so that a speaker independent (SI) system can be built and adapted to unknown users during deployment. The most commonly used technique is the Maximum Likelihood Linear Regression (MLLR) [40, 70] scheme. Maximum a Posteriori Linear Regression (MAPLR) techniques [16, 17] have also been applied to improve the performance of MLLR in the case of data sparseness. MLLR is a technique of (linearly) transforming the parameters of a general model to reflect (adapt to) a specific condition using only a small amount of adaptation data. To cope with the small amount of adaptation data, *linear regression* method is used to transform a group of parameters. Grouping of these parameters into *regression classes* is often done using a *top-down* splitting or a *bottom-up* clustering (agglomerative) scheme to generate a regression class tree [29]. Top-down clustering schemes allow expert knowledge to be incorporated into the tree generation process (e.g. a *speech-silence* partition). The linear transformation matrices are then estimated based on the statistics accumulated at the deepest node of the regression class tree having sufficient data (above a certain threshold). Figure 2.6 depicts an example regression class tree used for adaptation. In this example, there are 9 regression classes, partitioned into *speech* (`c1, c2, c3, c4, c5`) and *silence* (`c6, c7, c8, c9`). These regression classes, together with the other non-terminal nodes, form a regression tree These nodes are used to determine the sufficient data available for each regression class. The adaptation data used to estimate the transformation matrix for each regression class is shown with the transform links (dashed arrows). So, regression classes, `c1`, `c4` and `c8` have sufficient data to estimate their respective transformation matrices. `c2` requires data from both `c1` and `c2` for transform estimation. Similarly, `c3`, `c5` and `c7` require data from `[c1, c2, c3]`, `[c4, c5]` and `[c6, c7, c8, c9]` respectively. Finally, `c9` can only be estimated robustly using data from all regression classes.

The MLLR technique has been successfully applied to the Gaussian parameters (mean and variance) of the CDHMM systems. These techniques are known as the MLLR mean [70] and MLLR variance [31] adaptation schemes. Constrained MLLR (CMLLR) [31] adaptation is a special form of MLLR adaptation where the mean and variance of the same regression class share the same linear transformation matrix. This turns out to be equivalent to applying a linear transformation to the feature vectors for each regression class. CMLLR are often employed in Speaker

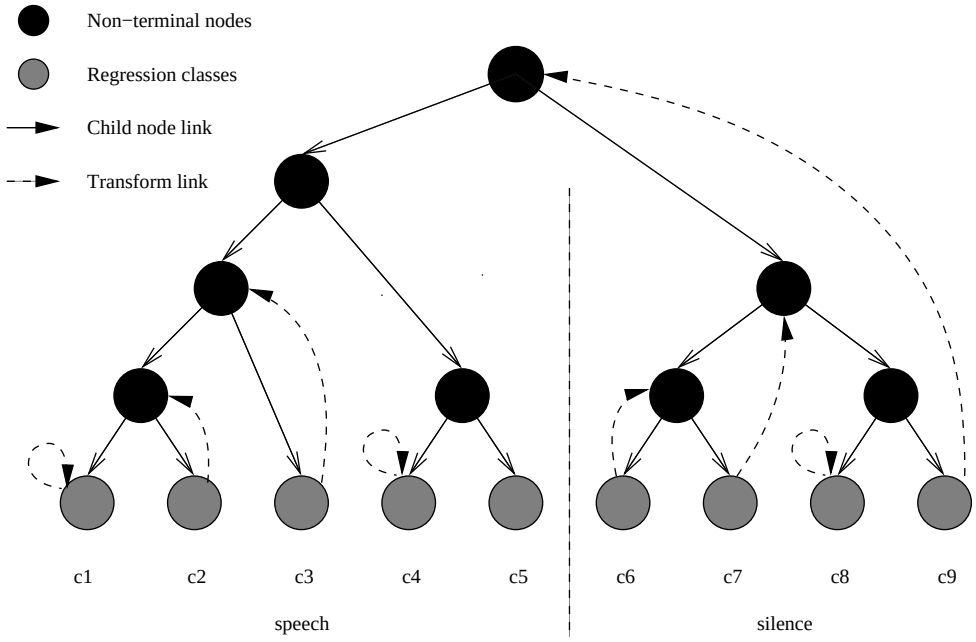

Figure 2.6 *An example regression class tree*

Adaptive Training (SAT) [2] due to its efficient storage requirement (one linear transform for each regression class as oppose to two: one for the mean vectors and one for the covariance matrices). An extension to the CMLLR technique which incorporates dimension reduction to the feature space, FMLLR, was proposed in [107]. This technique was found to outperform the CMLLR technique when used in the corresponding projected space.

2.2.4 Decoding Strategies

State-of-the-art systems typically employ advanced decoding strategies to improve the basic Viterbi decoding. Several common techniques are listed below:

- **Pronunciation Probabilities**

  Lexicon defines the mapping between the words in the vocabulary and the corresponding subword representation. This is an important aspect for phone-based systems where the mapping between a word and its lexicon is governed by the pronunciation. Variation in the pronunciation may occur as a result of several factors. For instance, it may vary depending on its context in a sentence. Besides, it may also be caused by the different accents or dialects of the speakers. Pronunciation variations can be accounted for using a probabilistic model where probabilities are assigned to each pronunciation variant of the words. In some cases, selecting the most probable pronunciation (single pronunciation system) may yield better performance than a multiple pronunciations system with pronunciation

probabilities. Typically, adding the pronunciation probability information may improve the system's performance by 5%–10% relative.

- **Multi-pass Decoding**

  Typically, more advanced systems have the disadvantage of running slower. Multi-pass decoding strategy repeatedly refines the search space in multiple stages so that more powerful but slower systems can be incorporated in the later stages. The refined search space is often a compact representation of multiple hypotheses from the previous stage in the format of word lattices (network). For example, the use of a 4-gram language model is prohibitively slow when used in a single-pass full decoding system. In a two-stage decoding setup, a trigram language model is used to generate a lattice of hypotheses. The lattice is then expanded to include 4-gram context information and rescored with the 4-gram language model to obtain better hypotheses. In Chapter 8, several multi-pass decoding setup will be described to illustrate the use of a discriminatively trained precision matrix model, SPAM, in a multi-pass and multi-branch framework.

- **Confusion Network Decoding**

  The decoding (recognition) process of a speech recognition system is associated with finding the most likely word sequence given the speech. This is often based on the likelihood scores. Alternatively, best word sequence can be obtain by considering the confidence measures [25]. Confidence measure may be calculated by looking at the confusion between multiple hypotheses typically in the form of a confusion network (CN). CN decoding [25] has been found to yield improvements in the range of 5–10% relative improvements compared to the standard likelihood-based decoding.

- **Systems Combination**

  Instead of a sequential multi-pass decoding strategy, multiple systems can perform recognition in parallel and the final output from all the systems are combined together to (hopefully) give a better performance than the best individual system. The ROVER (Recognizer Output Voting Error Reduction) [26] technique developed by NIST is a method of combining multiple 1-best transcriptions to reduce the recognition error rate. Alternatively, Confusion Network Combination (CNC) method [25, 76] can be used to combine multiple hypotheses from multiple systems to yield better performance. Experimental results on CNC combination of precision matrix models with other standard HMM systems will be discussed in Chapter 8.

2.3 Limitations of HMMs for Speech Recognition

Recall from Sections 2.1.1 to 2.1.3 that the use of HMMs in speech recognition involves the calculation of the likelihood function, which is of the form

$$p(\mathcal{O}_1^T|\boldsymbol{\theta}) = \sum_{Q_1^T} p(\mathcal{O}_1^T, Q_1^T|\boldsymbol{\theta}) = \sum_{Q_1^T} p(\mathcal{O}_1^T|Q_1^T, \boldsymbol{\theta}) P(Q_1^T|\boldsymbol{\theta}) \tag{2.33}$$

where the summation is defined over all possible state sequence, Q_1^T. To allow efficient manipulation of the above expression, two important assumptions (given at the beginning of this chapter) are made about the speech waveform, which decompose $P(Q_1^T|\boldsymbol{\theta})$ and $p(\mathcal{O}_1^T|Q_1^T, \boldsymbol{\theta})$ as follows:

1. **Instantaneous first-order transition:**

$$P(Q_1^T|\boldsymbol{\theta}) = \prod_{t=1}^{T} P(q_t|q_{t-1}, \boldsymbol{\theta}) \tag{2.34}$$

   where $P(q_1|q_0, \boldsymbol{\theta}) = P(q_1|\boldsymbol{\theta})$ is the initial probability of state, q_1.

2. **Conditional independence assumption:**

$$p(\mathcal{O}_1^T|Q_1^T, \boldsymbol{\theta}) = \prod_{t=1}^{T} p(\boldsymbol{o}_t|q_t, \boldsymbol{\theta}) \tag{2.35}$$

   where the state dependent output distribution, $p(\boldsymbol{o}_t|q_t, \boldsymbol{\theta})$, is commonly represented by a Gaussian mixture model. Furthermore, the covariance matrix of each Gaussian component is often assumed a diagonal structure (ignoring spectral correlations) to further reduce the computational cost.

In particular, from the "conditional independence assumption", observations are assumed to be conditionally independent given the state that generated the observation. Thus, the temporal correlation between successive observations is ignored and the output distribution associated with an HMM state is constant. Existing ways to overcome this limitation include the use of switching linear dynamical systems [104], stochastic segment models [88, 89], polynomial segment models, buried Markov models [13] and trajectory HMM [118, 119], . All these models have a common aim of relaxing the "conditional independence assumption" by allowing the state output distribution to vary with time. This time variation is achieved by adding dependency on the observation sequence, $\mathcal{O}_1^T$, either directly or using latent variables. The model parameters for the state output probability are now time dependent and equation (2.35) may be rewritten as

$$p(\mathcal{O}_1^T|Q_1^T, \boldsymbol{\theta}_t) = \prod_{t=1}^{T} p(\boldsymbol{o}_t|\boldsymbol{\theta}_t) \tag{2.36}$$

The time dependent parameter set, $\boldsymbol{\theta}_t$, is expressed as a function of the observation sequence, $\mathcal{O}_1^T$, state sequence, Q_1^T, and the time, t, *i.e.*

$$\boldsymbol{\theta}_t = f_\Phi\left(\mathcal{O}_1^T, Q_1^T, t\right) \tag{2.37}$$

The form of function, $f_\Phi(.)$, with parameters, $\boldsymbol{\Phi}$, defines the type of model used. In the following sections, several trajectory and segmental models will be described based on the time varying parameters formulation given by equations (2.36) and (2.37).

2.3.1 Explicit Temporal Correlation Modelling

One of the earliest work on explicit time correlation modelling was carried out by Wellekens [128]. In this work, correlation between adjacent frames is explicitly modelled by modifying equation (2.36) as

$$p(\mathcal{O}_1^T|Q_1^T, \boldsymbol{\theta}) = p(\boldsymbol{o}_1|q_1, \boldsymbol{\theta}) \prod_{t=2}^{T} p(\boldsymbol{o}_t|\boldsymbol{o}_{t-1}, q_t, q_{t-1}, \boldsymbol{\theta}) \tag{2.38}$$

where the conditional probability distribution of $\boldsymbol{o}_t$ is given by

$$p(\boldsymbol{o}_t|\boldsymbol{o}_{t-1}, q_t, q_{t-1}, \boldsymbol{\theta}) = \frac{p(\boldsymbol{o}_t, \boldsymbol{o}_{t-1}|q_t, q_{t-1}, \boldsymbol{\theta})}{p(\boldsymbol{o}_{t-1}|q_{t-1}, \boldsymbol{\theta})} \tag{2.39}$$

The joint distribution of $\boldsymbol{o}_t$ and $\boldsymbol{o}_{t-1}$ in the numerator of the *r.h.s.* of equation (2.39) is given by

$$p(\boldsymbol{o}_t, \boldsymbol{o}_{t-1}|q_t = s, q_{t-1} = u, \boldsymbol{\theta}) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{o}_{t-1} \\ \boldsymbol{o}_t \end{bmatrix}; \begin{bmatrix} \boldsymbol{\mu}_u \\ \boldsymbol{\mu}_s \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{uu} & \boldsymbol{\Sigma}_{us} \\ \boldsymbol{\Sigma}_{su} & \boldsymbol{\Sigma}_{ss} \end{bmatrix}\right) \tag{2.40}$$

and the conditional distribution of $\boldsymbol{o}_{t-1}$ given q_{t-1} is given by

$$p(\boldsymbol{o}_{t-1}|q_{t-1} = u, \boldsymbol{\theta}) = \mathcal{N}\left(\boldsymbol{o}_{t-1}; \boldsymbol{\mu}_u, \boldsymbol{\Sigma}_{uu}\right) \tag{2.41}$$

$\boldsymbol{\mu}_s$ and $\boldsymbol{\Sigma}_{ss}$ denote the mean and covariance matrix of state s and $\boldsymbol{\Sigma}_{su}$ is the covariance between state s and u. Therefore, equation (2.39) is also a Gaussian distribution of the form

$$p(\boldsymbol{o}_t|\boldsymbol{o}_{t-1}, q_t = s, q_{t-1} = u, \boldsymbol{\theta}) = \mathcal{N}\left(\boldsymbol{o}_t; \boldsymbol{\mu}_{st|u}, \boldsymbol{\Sigma}_{s|u}\right) \tag{2.42}$$

where the conditional mean vector and covariance matrix may be expressed as

$$\boldsymbol{\mu}_{st|u} = \boldsymbol{\mu}_s + \boldsymbol{\Sigma}_{su}\boldsymbol{\Sigma}_{uu}^{-1}(\boldsymbol{o}_{t-1} - \boldsymbol{\mu}_u) \tag{2.43}$$

$$\boldsymbol{\Sigma}_{s|u} = \boldsymbol{\Sigma}_{ss} - \boldsymbol{\Sigma}_{su}\boldsymbol{\Sigma}_{uu}^{-1}\boldsymbol{\Sigma}_{us} \tag{2.44}$$

Note that, $\boldsymbol{\mu}_{st|u}$, the conditional mean of state s given state u is now dependent on the previous observation vector, $\boldsymbol{o}_{t-1}$. This is one form of time varying parameters described by equation (2.37). However, the cross covariance matrix between two states, $\boldsymbol{\Sigma}_{su}$, has to be estimated

for every possible pair of states in the system. This increases the number of model parameters dramatically. To further simplify the above model, the dependency on the previous state in equation (2.39) may be dropped, such that

$$\boldsymbol{\mu}_u \rightarrow \tilde{\boldsymbol{\mu}} \qquad \boldsymbol{\Sigma}_{su} \rightarrow \tilde{\boldsymbol{\Sigma}}_s \qquad \boldsymbol{\Sigma}_{uu} \rightarrow \tilde{\boldsymbol{\Sigma}} \tag{2.45}$$

This yields

$$p(\boldsymbol{o}_t|\boldsymbol{o}_{t-1}, q_t = s, \boldsymbol{\theta}) = \mathcal{N}\left(\boldsymbol{o}_t; \boldsymbol{\mu}_{st}, \boldsymbol{\Sigma}_s\right) \tag{2.46}$$

where the conditional mean vector and covariance matrix are now given by

$$\boldsymbol{\mu}_{st} = \boldsymbol{\mu}_s + \tilde{\boldsymbol{\Sigma}}_s \tilde{\boldsymbol{\Sigma}}^{-1} (\boldsymbol{o}_{t-1} - \tilde{\boldsymbol{\mu}}) \tag{2.47}$$

$$\boldsymbol{\Sigma}_s = \boldsymbol{\Sigma}_{ss} - \tilde{\boldsymbol{\Sigma}}_s \tilde{\boldsymbol{\Sigma}}^{-1} \tilde{\boldsymbol{\Sigma}}_s' \tag{2.48}$$

Another form of trajectory model is the vector linear prediction (VLP) model [130], where the state output probability of $\boldsymbol{o}_t$ is conditionally independent of other parameters given the current state, q_t, and observation dependencies, $\mathcal{H}_t$. Therefore, equation (2.39) is altered such that

$$p(\mathcal{O}_1^T|Q_1^T, \boldsymbol{\theta}) = \prod_{t=1}^{T} p(\boldsymbol{o}_t|\mathcal{H}_t, q_t, \boldsymbol{\theta}) \tag{2.49}$$

where $\mathcal{H}_t \subset \mathcal{O}_1^T$. The state distribution is thus re-expressed in terms of the observation history as

$$p(\boldsymbol{o}_t|\mathcal{H}_t, q_t = s, \boldsymbol{\theta}) = \mathcal{N}\left(\boldsymbol{o}_t; \boldsymbol{\mu}_{st}, \boldsymbol{\Sigma}_s\right) \tag{2.50}$$

where the state mean vector varies with time according to the form

$$\boldsymbol{\mu}_{st} = \boldsymbol{\mu}_s^{(0)} + \sum_{p=1}^{P} \boldsymbol{A}_s^{(p)} \left(\boldsymbol{o}_{t+\tau_p} - \boldsymbol{\mu}_s^{(\tau_p)}\right) \tag{2.51}$$

where P is the number of predictors. The mean vector given the HMM state is dependent on the observation history, $\mathcal{H}_t = \{\boldsymbol{o}_{t+\tau_p} : 1 \leq p \leq P, 1 \leq t + \tau_p \leq T\}$. This form of model is again a specific form of a time varying parameter formulation given by equation (2.37). Note that the mean vector of the VLP model given in the above equation is similar to the expression of the conditional mean for an explicit temporal correlation model given in equation (2.47). Equation (2.51) is identical to equation (2.47) when only one predictor is used ($P = 1$) such that $\tau_1 = -1$ and

$$\boldsymbol{\mu}_s^{(\tau_1)} = \tilde{\boldsymbol{\mu}} \quad \text{and} \quad \boldsymbol{A}_s^{(1)} = \tilde{\boldsymbol{\Sigma}}_s \tilde{\boldsymbol{\Sigma}}^{-1} \tag{2.52}$$

Therefore, increasing the number of linear predictor (P) enhances the model's capability of explicitly modelling the temporal correlations.

A more general form of temporal correlation modelling scheme was introduced by Bilmes [13], known as the Buried Markov Model (BMM). This model is similar to the above VLP model, except that the observation dependencies are specifically selected (discriminatively using a maximum entropy criterion) from any of the elements of the entire observation sequence. In [13], a

Gaussian-mixture BMM was described where the state output density function is modelled by a mixture of Gaussian of the form

$$p(\boldsymbol{o}_t|\boldsymbol{h}_t, q_t = s, \boldsymbol{\theta}) = \sum_{m=1}^{M}\sum_{v=1}^{V} P(m|s,v)P(v|\boldsymbol{h}_t)\mathcal{N}\left(\boldsymbol{o}_t; \boldsymbol{\mu}_{smvt}, \boldsymbol{\Sigma}_{smv}\right) \tag{2.53}$$

where $\boldsymbol{h}_t$ is a column vector defining the entire collection of dependencies variables any element of $\boldsymbol{o}_t$ might use and v denotes the class of h_t. M and V are the number of components and classes respectively. $P(m|s,v)$, the prior of component m, given the state s and class v, is a discrete probability table and $P(v|\boldsymbol{h}_t)$ is the probability of class v given the continuous vector $\boldsymbol{h}_t$. This formulation also yields a time varying Gaussian mean vector given by

$$\boldsymbol{\mu}_{smvt} = \boldsymbol{A}_{smv}\boldsymbol{h}_t + \boldsymbol{b}_{smv} \tag{2.54}$$

where $\boldsymbol{A}_{smv}$ and $\boldsymbol{b}_{smv}$ are model parameters that can be estimated efficiently using the EM approach [13]. This expression is dependent on time via the vector of dependency variables, $\boldsymbol{h}_t$. Note that if $\boldsymbol{h}_t$ contains elements from $\boldsymbol{o}_t$ itself, both *spectral* and *temporal* correlations are modelled. Furthermore, equation (2.53) may be rewritten as

$$p(\boldsymbol{o}_t|\boldsymbol{h}_t, q_t = s, \boldsymbol{\theta}) = \sum_{m=1}^{M}\sum_{v=1}^{V} c_{smvt}\mathcal{N}\left(\boldsymbol{o}_t; \boldsymbol{\mu}_{smvt}, \boldsymbol{\Sigma}_{smv}\right) \tag{2.55}$$

where $c_{smvt} = p(m|s,v)p(v|\boldsymbol{h}_t)$ denotes the time varying component weights.

2.3.2 Stochastic Segment Models

Segmental models represent another class of model which allows the trajectory within a segment of speech to be modelled. Here, the Stochastic Segment Model (SSM) [88, 89] will be described. An SSM models the trajectory within a speech segment via the use of *latent* variables, which may be formulated within a state-space framework [104]. Speech segments are assumed independent and therefore the state posteriors are not propagated across segment boundaries. The generative model of SSM can be expressed as

$$\begin{aligned} q_{t+1} &\sim P(q_{t+1}|q_t) \\ \boldsymbol{x}_t &= \begin{cases} \sim \mathcal{N}\left(\boldsymbol{\mu}_{q_t}^{(0)}, \boldsymbol{\Sigma}_{q_t}^{(0)}\right) & q_t \neq q_{t-1} \\ \boldsymbol{A}_{q_t}\boldsymbol{x}_{t-1} + \boldsymbol{w}_{q_t} & q_t = q_{t-1} \end{cases} \\ \boldsymbol{o}_t &= \boldsymbol{C}_{q_t}\boldsymbol{x}_t + \boldsymbol{v}_{q_t} \end{aligned}$$

Figure 2.7: *A generative Stochastic Segment Model*

where $\boldsymbol{v}_{q_t}$ and $\boldsymbol{w}_{q_t}$ are the noise vectors associated with the observations and latent variables respectively. The distribution of these noise vectors may be represented by Gaussian Mixture

Models (GMMs). Therefore, the state output probability distribution for an SSM is a GMM which depends on the underlying latent variables, $\boldsymbol{x}_t$ and $\boldsymbol{x}_{t-1}$, *i.e.*

$$p(\boldsymbol{o}_t|q_t, \boldsymbol{\theta}_t) = p(\boldsymbol{o}_t|q_t, \boldsymbol{x}_t, \boldsymbol{x}_{t-1}, \boldsymbol{\theta}) \tag{2.56}$$

The posterior probabilities associated with the latent variables, $\boldsymbol{x}_t$, may be obtained from the E-step of the Kalman filter or smoother techniques.

2.3.3 Switching Linear Dynamical Systems

The Switching Linear Dynamical System (SLDS) [104] is an extension to the above SSM model, where the trajectory of the speech data is modelled across segments. The generative model of an SLDS is given by

$$\begin{aligned} q_{t+1} &\sim P(q_{t+1}|q_t) \\ \boldsymbol{x}_t &= \boldsymbol{A}_{q_t}\boldsymbol{x}_{t-1} + \boldsymbol{w}_{q_t} \\ \boldsymbol{o}_t &= \boldsymbol{C}_{q_t}\boldsymbol{x}_t + \boldsymbol{v}_{q_t} \end{aligned}$$

Figure 2.8: *A generative Switching Linear Dynamical System*

As before, the state probability distribution can be written as a GMM whose parameters vary with time according to the underlying latent state evolution.

Various models have been described which attempt to relax the *conditional independence assumption* made by the standard HMM formulation. Unfortunately, to date, these models have had little success in improving the performance of large vocabulary continuous speech recognition systems. Despite the difference in model formulation (trajectory models, segment models, switching linear dynamical system *etc.*), these models can all be expressed as a non-stationary state output Gaussian mixture model. This non-stationarity may be viewed as a generic function of the observation sequence, as described in equation (2.37).

In Chapter 7, a discriminative semi-parametric trajectory model will be presented. This model represents the Gaussian mean vectors and covariance matrices as time varying parameters. This time dependent parameters are modelled as a function of the location of the current observation (and the neighbouring observations) in the acoustic space, which is represented by a series of centroids. Model parameters are discriminatively estimated using the Minimum Phone Error (MPE) criterion [95].

One form of temporally varying mean vector is obtained by applying a time dependent bias to the static Gaussian mean. This time dependent bias is a weighted contribution from the

bias vectors associated with each centroid (to be estimated discriminatively). The contribution weights are calculated as the posteriors of the observation (and neighbouring observations) given the centroids. The resulting model yields an fMPE model [96, 97]. On the other hand, the variance of each dimension may also be scaled by a positive time dependent factor to yield a temporally varying covariance matrix. This model is known as pMPE [112]. Similar to fMPE, the time dependent scale factor is a weighted contribution from the centroid specific scales where the weights are given by the posteriors of the observations given the centroids.

3

Covariance and Precision Matrix Modelling

To date, most state-of-the-art speech recognition systems are based on HMMs with multivariate continuous output density function to model the acoustic units of speech data. One major issue with using multivariate continuous density HMMs (CDHMMs) is how to *accurately* and *efficiently* model the correlation of the feature vectors. Two forms of correlation exist:

1. ***inter-frame* correlation**: temporal correlation between feature vectors from successive frames.
2. ***intra-frame* correlation**: spatial correlation between the feature elements within each speech frame

In standard systems, both types of correlation are typically ignored so that efficient training and decoding schemes can be used. Inter-frame correlation is represented only by the underlying state sequence and the observations are assumed conditionally independent given the state sequence, as described in Chapter 2. This allows the Baum-Welch training algorithm (see Section 2.1.3) and the Viterbi decoding scheme (see Section 2.1.2) to be applied *efficiently*. Many improvements have been investigated to relax the independence assumption, as described in the previous chapter. These include the trajectory models [13, 118, 119, 128, 130], segmental models [41, 42, 88, 89] and the switching linear dynamical systems [104]. Further discussion on this will be presented in Chapter 7. The remaining of this chapter concentrates on the discussion of intra-frame correlation.

3.1 Likelihood Calculation

The likelihood calculation is an important aspect of a speech recognition system. In both training and decoding, the likelihood of a Gaussian component given each observation, $\boldsymbol{o}_t$, for all the *active* components at time t needs to be computed. For LVCSR systems with typically more than 100,000 Gaussian components, it is crucial to formulate the acoustic model such that the likelihood computation is as efficient as possible.

Modelling all the intra-frame correlations requires full covariance matrices to be used. In this case, the log-likelihood of Gaussian component m in state s (with mean vector $\boldsymbol{\mu}_{sm}$ and full covariance matrix $\boldsymbol{\Sigma}_{sm}$) given the observation vector, $\boldsymbol{o}_t$ at time t may be expressed as

$$\mathcal{L} = \log p(\boldsymbol{o}_t|\boldsymbol{\mu}_{sm}, \boldsymbol{\Sigma}_{sm}) = -\frac{1}{2}\left\{d\log(2\pi) + \log|\boldsymbol{\Sigma}_{sm}| + (\boldsymbol{o}_t - \boldsymbol{\mu}_{sm})' \boldsymbol{\Sigma}_{sm}^{-1} (\boldsymbol{o}_t - \boldsymbol{\mu}_{sm})\right\} \quad (3.1)$$

The computation of the above expression is dominated by the final term in the curly brackets, which requires $\mathcal{O}(d(d+1))$ operations[1]. As the feature dimensionality increases, the likelihood computational cost increases dramatically. For LVCSR systems with high dimensionality (d has a typical value of 39) and many Gaussian components (more than 100,000), the likelihood computation of a full covariance matrix system becomes impractical. This problem is usually alleviated by ignoring the intra-frame correlation. This is achieved by using diagonal covariance matrix approximation. This simplifies the log likelihood expression in equation (3.1) as

$$\mathcal{L} = \log p(\boldsymbol{o}_t|\boldsymbol{\mu}_{sm}, \boldsymbol{\Sigma}_{sm}) = -\frac{1}{2}\sum_{j=1}^{d}\left\{\log(2\pi) + \log\sigma_{smj}^2 + \frac{(o_{tj} - \mu_{smj})^2}{\sigma_{smj}^2}\right\} \quad (3.2)$$

where μ_{smj} and σ_{smj}^2 denote the jth element of the mean vector and diagonal covariance matrix respectively. Once again, the cost of calculating the above expression is dominated by the computation of the final term in the curly brackets[2], which requires two operations for each term and in total $\mathcal{O}(2d)$ operations. This is significantly smaller compared to the full covariance matrix case (78 *versus* 1560 operations for $d = 39$). An issue is by how much the system performance is degraded by using diagonal covariance matrix approximation? Also, are there ways to improve the system performance by adopting other approximation schemes while maintaining the efficiency in terms of likelihood computation? In the following sections, these issues will be addressed by first motivating the importance of covariance matrix modelling using a simple 2-class classification problem and then presenting several methods of improving the performance of a diagonal covariance matrix system. Next, *structured approximations* of covariance and precision matrices will be discussed. In general, the latter is more efficient as the likelihood function can be expressed directly in terms of the precision matrix (see equation 3.1). Covariance matrix modelling usually incurs an additional cost of matrix inversion in likelihood calculation. Hence,

[1] The cost associated of computing the offset between the observation and the mean vectors, $\boldsymbol{o}_t - \boldsymbol{\mu}_{sm}$, is ignored.

[2] The sum of the first two terms may be pre-computed since they are independent of the observation vectors.

this chapter will concentrate on various forms of precision matrix modelling schemes. In particular, a unifying framework of basis superposition for efficient precision matrix modelling (PMM) will be introduced.

3.2 Effect of Covariance Matrix

As mentioned in Chapter 1, there are inherent *uncertainties* associated with speech data. These variabilities are captured by the covariance matrix of the random samples. These random samples are commonly represented by a Gaussian distribution, which may be described by its mean vector, $\boldsymbol{\mu}$ and covariance matrix, $\boldsymbol{\Sigma}$, as follows:

$$\mathcal{N}(\boldsymbol{o}_t; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}|}} \exp\left\{-\frac{1}{2}(\boldsymbol{o}_t - \boldsymbol{\mu})' \boldsymbol{\Sigma}^{-1} (\boldsymbol{o}_t - \boldsymbol{\mu})\right\} \tag{3.3}$$

where $\boldsymbol{o}_t$ is a $d \times 1$ observation vector at time t and

$$\boldsymbol{\mu} = \mathbb{E}[\boldsymbol{o}_t] \tag{3.4}$$

$$\boldsymbol{\Sigma} = \mathbb{E}[(\boldsymbol{o}_t - \boldsymbol{\mu})(\boldsymbol{o}_t - \boldsymbol{\mu})'] = \mathbb{E}[\boldsymbol{o}_t \boldsymbol{o}_t'] - \boldsymbol{\mu}\boldsymbol{\mu}' \tag{3.5}$$

$\mathbb{E}[\boldsymbol{x}] = \int p(\boldsymbol{x}) \boldsymbol{x} d\boldsymbol{x}$ denotes the expected value of a random vector $\boldsymbol{x}$ with a distribution function $p(\boldsymbol{x})$. Equation (3.5) shows that $\boldsymbol{\Sigma}$ is a *semi-positive-definite* symmetric matrix whose leading diagonal elements are the variances of each dimension and the off-diagonal elements are the covariances in between dimensions:

$$\text{Var}(o_{tj}) = \sigma_j^2 \tag{3.6}$$

$$\text{Cov}(o_{tj}, o_{tk}) = \sigma_{jk} \tag{3.7}$$

where the covariance between dimensions j and k, σ_{jk}, are related to the correlation coefficients, ρ_{jk} by

$$\rho_{jk} = \frac{\sigma_{jk}}{\sqrt{\sigma_j^2 \sigma_k^2}} \quad \text{and} \quad -1 \leq \rho_{jk} \leq 1 \tag{3.8}$$

The leading diagonal elements of a correlation matrix is always one ($\rho_{jj} = 1$) and $\rho_{jk} = 0$ indicates that o_{tj} and o_{tk} are uncorrelated.

The inverse of a covariance matrix is known as the precision matrix. While the elements of a covariance matrix capture the variance and correlation information, a precision matrix contains the *conditional dependence* information. Thus, if the (i, j)th element of a precision matrix is zero, the ith and jth random variables are conditionally independent. (Refer to Appendix C for more details).

To illustrate the need for intra-frame correlation, a simple problem is used. Consider a 2-class classification problem, where the samples are represented by a 2-dimensional feature

vector. The samples from the two classes, A and B, are normally distributed with distributions $\mathcal{N}(\boldsymbol{\mu}_A, \boldsymbol{\Sigma}_A)$ and $\mathcal{N}(\boldsymbol{\mu}_B, \boldsymbol{\Sigma}_B)$ respectively. The associated Gaussian parameters are given by

$$\boldsymbol{\mu}_A = \begin{bmatrix} 0 \\ -1 \end{bmatrix}; \quad \boldsymbol{\mu}_B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}; \quad \boldsymbol{\Sigma}_A = \begin{bmatrix} 1.50 & 0.00 \\ 0.00 & 1.00 \end{bmatrix}; \quad \boldsymbol{\Sigma}_B = \begin{bmatrix} 1.00 & 0.60 \\ 0.60 & 1.00 \end{bmatrix} \tag{3.9}$$

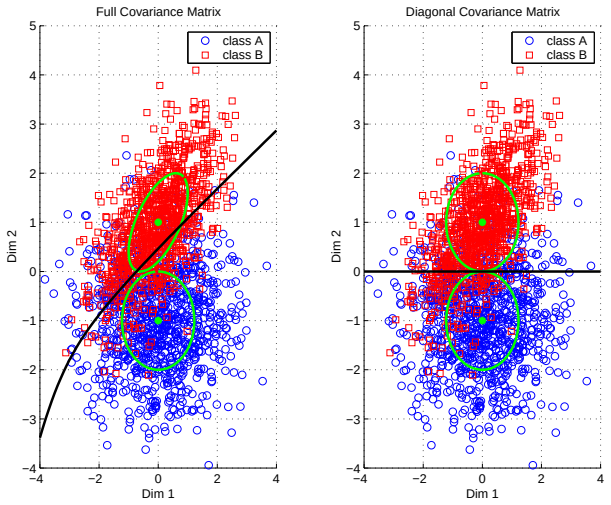

Figure 3.1 *2-dimensional sample space of a 2-class classification problem. Samples from each class are represented by a Gaussian distribution with full covariance matrix (left) and diagonal covariance matrix (right)*

Figure 3.1 shows the sample space of this problem together with the Gaussian model and the optimal Bayes' decision boundaries. The left and right figures correspond to the Gaussian models with full and diagonal covariance matrices respectively. The optimum decision boundary divides the sample space into two regions, one for each class, such that the misclassification is minimum. The points on this boundary have equal likelihood given class A or class B, i.e.

$$\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_A, \boldsymbol{\Sigma}_A) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_B, \boldsymbol{\Sigma}_B) \tag{3.10}$$

Rearranging this using equation (3.3) yields a *quadratic* decision boundary given by

$$\boldsymbol{x}' \left(\boldsymbol{\Sigma}_A^{-1} - \boldsymbol{\Sigma}_B^{-1}\right) \boldsymbol{x} - 2 \left(\boldsymbol{\mu}_A' \boldsymbol{\Sigma}_A^{-1} - \boldsymbol{\mu}_B' \boldsymbol{\Sigma}_B^{-1}\right) \boldsymbol{x} = \left(\boldsymbol{\mu}_B' \boldsymbol{\Sigma}_B^{-1} \boldsymbol{\mu}_B - \boldsymbol{\mu}_A' \boldsymbol{\Sigma}_A^{-1} \boldsymbol{\mu}_A + 2 \log \frac{|\boldsymbol{\Sigma}_B|}{|\boldsymbol{\Sigma}_A|}\right) \tag{3.11}$$

Note that when there is a global covariance matrix is used for all the classes ($\boldsymbol{\Sigma}_A = \boldsymbol{\Sigma}_B$), the decision boundary becomes a straight line (or a hyperplane in a multivariate case). Furthermore, the decision boundary for the case of using diagonal and full covariance matrices is also considerably different (see Figure 3.1). In the above example, the classification errors with and without the correlation information are 14.45% and 16.40% respectively. This illustrates the importance of modelling the inter-dimensional correlations, more so for a high dimensional feature space.

3.3 Implicit Correlation Modelling using GMMs

Diagonal covariance matrix approximation is commonly applied to obtain efficient likelihood computation. However, this implies that the intra-frame correlations are completely ignored and may result in degradation of system performance as discussed in the previous section. A common solution to this problem is to *implicitly* model the spectral correlations using a Gaussian Mixture Model (GMM). A GMM distribution is given by equation (3.3). Given a sufficiently large number of Gaussian components, a GMM is able to model *almost any* distributions. As described at the beginning of this chapter, a GMM distribution is typically used in a CDHMM-based speech recognition system to capture the *non-Gaussian* attributes (multi-modal, skewed, heavy-tail etc) of the true distribution. Often, a diagonal covariance matrix structure is assumed for each Gaussian component to retain the simplicity and efficiency of its likelihood computation. Under such a circumstance, a GMM can also model the intra-frame correlations *implicitly* to some extend. To see how this is the case, consider the correlation 2-dimensional space in Figure 3.2. The left figure shows a single full covariance matrix Gaussian distribution model for

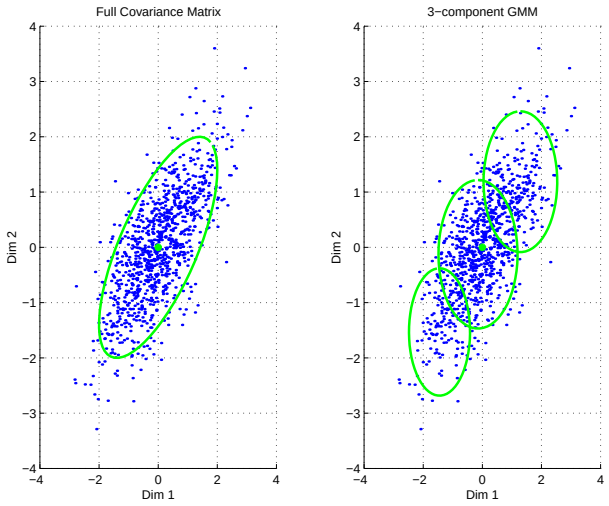

Figure 3.2 *2-dimensional correlated sample space. The samples are represented by a Gaussian distribution (left) and a 3-component GMM (right).*

the correlated samples and the right figure shows a 3-component GMM distribution for the same samples. Diagonal covariance matrices are used for the three Gaussian components of the GMM distribution. These Gaussian components are positioned such that they lie along the principal component of the samples to reflect the inter-dimensional correlations. For high dimensional data, correlation modelling using GMM with diagonal covariance matrices is generally computationally more efficient and requires a smaller amount of parameters, depending on the number of components used.

3.4 Covariance Matrix Approximation

Ideally, it would be useful to incorporate as much correlation information in the covariance matrices as possible so that the probability distribution can better represent the random data. Unfortunately, speech recognition systems involve a large number of Gaussian components (typically greater than 100,000 for LVCSR) and the dimensionality of the feature space is high (typically $d = 39$). To model the inter-dimensional correlations completely (using a full covariance matrix structure), the total number of parameters required is $\frac{d}{2}(d+1) = 780$. This is twenty times larger than a diagonal covariance matrix structure. Due to the large number of Gaussian components, the number of free parameters in the system increases dramatically and more data is required to obtain reliable estimates for these full covariance matrices. Furthermore, the complexity of the likelihood computation, according to equation (3.1), is $\mathcal{O}(d(d+1))$ for a full covariance matrix system compared to $\mathcal{O}(2d)$ when a diagonal covariance matrix structure is used (see Section 3.1). Hence, the use of full covariance matrix structure is not practical[3] due to the large number of free parameters involved and the prohibitively high computational costs. This motivates the use of more efficient covariance modelling schemes, which can be defined as satisfying the following criteria:

- *accurate* modelling;
- *compact* model representation;
- *efficient* likelihood calculation.

It has been empirically found that modelling the covariance matrix structure yields slightly better performance compared to precision matrix modelling using the same approximation. However, more importantly, covariance matrix models generally incur a higher likelihood computational cost due to the need to perform matrix inversion. Due to this reason, more powerful and efficient precision matrix models can be derived which will be discussed later in this chapter. One simple example is using a diagonal or block-diagonal approximation. Alternatively, more advanced technique that incorporate factor analysis into the HMM framework (FAHMM) [103] has also been found to yield improved modelling. These methods will be briefly described here.

[3] IBM has successfully trained a full covariance matrix LVCSR system in their RT04 system [117]. To ensure robustness and tractability, diagonal covariance matrix smoothing was used in training and some form of Gaussian selection scheme was employed. However, the system still requires a large execution memory (up to 2Gb based on conversation with D. Povey)

3.4.1 Diagonal/Block-diagonal covariance matrix approximation

Diagonal and block-diagonal covariance matrices are modelled as

$$\boldsymbol{\Sigma}_{sm}^{\texttt{diag}} = \begin{bmatrix} \sigma_{sm1}^2 & 0 & \cdots & 0 \\ 0 & \sigma_{sm2}^2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \sigma_{smd}^2 \end{bmatrix} \quad \text{and} \quad \boldsymbol{\Sigma}_{sm}^{\texttt{block}} = \begin{bmatrix} \boldsymbol{\Sigma}_{sm}^{\texttt{static}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma}_{sm}^{\Delta} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\Sigma}_{sm}^{\Delta\Delta} \end{bmatrix} \tag{3.12}$$

The use of diagonal covariance matrices assumes that the elements of the feature vectors are uncorrelated. This assumption significantly reduce the number of model parameters and the likelihood computational cost. When the block-diagonal approximation is used, it is intuitive to divide the variables into blocks of the static coefficients, first (Δ) and second ($\Delta\Delta$) derivatives. However, such an approximation is based on prior expert knowledge, without making use of the training data. Is it a reasonable approximation? Many techniques can be used to make the independence and block-independence assumptions safer. For instance, the feature vectors can be chosen such that the variables are as uncorrelated as possible. The Mel Frequency Cepstral Coefficients (MFCC) [18] and the Perceptual Linear Prediction (PLP) [55] coefficients are commonly used because these parameters are less correlated compared to the filter bank coefficients or the Linear Prediction Coefficients (LPC). Further decorrelation can also be achieved using the feature transformation techniques such as Linear Discriminant Analysis (LDA) [105] and Heteroscedastic LDA (HLDA) [68].

3.4.2 Factor analysis and factor-analysed HMMs

Factor analysis is a technique where high dimensional data are modelled using a small set of *latent* variables. n common underlying independent factors that affect the data are found. Combining with the d independent unique factors associated to the original variables, the data vectors can be expressed as

$$\boldsymbol{o} = \boldsymbol{C}\boldsymbol{x} + \boldsymbol{v} \tag{3.13}$$

where $\boldsymbol{x}$ is a $n \times 1$ vector of common factors (latent variable), $\boldsymbol{v}$ is a $d \times 1$ vector of unique factors associated to the original variables and $\boldsymbol{C}$ is a $d \times n$ matrix of factor loadings. $\boldsymbol{x}$ and $\boldsymbol{v}$ are random vectors with distributions $\mathcal{N}(\boldsymbol{\mu}^{(x)}, \boldsymbol{\Sigma}^{(x)})$ and $\mathcal{N}(\boldsymbol{\mu}^{(o)}, \boldsymbol{\Sigma}^{(o)})$ respectively. Usually, the mean of $\boldsymbol{o}$ is modelled directly in the d-dimensional observed space so that $\boldsymbol{\mu}^{(x)} = 0$. Furthermore, the covariance matrix of the latent variables may be assumed an identity matrix, $\boldsymbol{\Sigma}^{(x)} = \boldsymbol{I}$, since any covariance matrix can be subsumed by $\boldsymbol{C}$. Thus, the resulting mean and covariance matrix of $\boldsymbol{o}$ is expressed as

$$\boldsymbol{\mu} = \boldsymbol{\mu}^{(o)} \quad \text{and} \quad \boldsymbol{\Sigma} = \boldsymbol{C}\boldsymbol{C}' + \boldsymbol{\Sigma}^{(o)} \tag{3.14}$$

$\boldsymbol{\Sigma}^{(o)}$ should not be set to be a full covariance matrix, otherwise the use of the latent variables becomes redundant. Typically, $\boldsymbol{\Sigma}^{(o)}$ takes the diagonal matrix form and the correlation information is captured by the factor loading matrix, $\boldsymbol{C}$. An issue associated with this is how many common factors to extract? One way is to examine the eigenvalues of the correlation matrix and extract those whose eigenvalues are greater than a certain threshold as the common factors. The total number of free parameters for this form of covariance matrix model is $d(n+1)$.

Factor analysis has been successfully implemented in the HMM-based speech recognition systems. This technique was first applied to speech recognition by *Saul et al.* [108, 109]. This method was generalised by *Rosti et al.* as the Factor-analysed HMM (FAHMM)[103]. FAHMM is a generative model that belongs to the generalised linear Gaussian model [103]. It can be described using the following state-space formulation:

$$\boldsymbol{x}_t = \boldsymbol{u}_t \qquad \boldsymbol{u}_t \sim \sum_{n=1}^{N} c_{sn}^{(x)} \mathcal{N}\left(\boldsymbol{\mu}_{sn}^{(x)}, \boldsymbol{\Sigma}_{sn}^{(x)}\right) \tag{3.15}$$

$$\boldsymbol{o}_t = \boldsymbol{C}\boldsymbol{x}_t + \boldsymbol{v}_t \qquad \boldsymbol{v}_t \sim \sum_{m=1}^{M} c_{sm}^{(o)} \mathcal{N}\left(\boldsymbol{\mu}_{sm}^{(o)}, \boldsymbol{\Sigma}_{sm}^{(o)}\right) \tag{3.16}$$

Figure 3.3: *A generative Factor-analysed HMM model*

where $\boldsymbol{x}_t$ and $\boldsymbol{o}_t$ are the state and observation vectors respectively at time t. The state and observation distributions are represented by Gaussian Mixture Models (GMMs) with parameters $\left\{c_{sn}^{(x)}, \boldsymbol{\mu}_{sn}^{(x)}, \boldsymbol{\Sigma}_{sn}^{(x)}\right\}$ and $\left\{c_{sm}^{(o)}, \boldsymbol{\mu}_{sm}^{(o)}, \boldsymbol{\Sigma}_{sm}^{(o)}\right\}$, where s denotes the HMM state. Therefore, these parameters are dependent on the underlying HMM state sequence, Q_1^T. $\boldsymbol{C}$ is the state-observation matrix (c.f. factor loading matrix in factor analysis), shared by all the Gaussian components. This yields the following form of FAHMM covariance structure:

$$\boldsymbol{\Sigma}_{smn} = \boldsymbol{C}\boldsymbol{\Sigma}_{sn}^{(x)}\boldsymbol{C}' + \boldsymbol{\Sigma}_{sm}^{(o)} \tag{3.17}$$

where s, n and m represent the HMM state, state space component and observation space component respectively. Although the FAHMM provides a compact covariance matrix representation, it is computationally expensive in decoding due to the inversion of the covariance matrix in likelihood calculation. On the other hand, modelling the precision (inverse covariance) matrices alleviates the cost of inversion, resulting in a more efficient modelling technique.

3.5 Precision Matrix Approximations

In recent years, techniques which model the precision (inverse covariance) matrices have become increasingly popular. Methods such as Semi-tied Covariances (STC) [32, 34], Extended

MLLT (EMLLT) [85], Subspace for Precision and Mean (SPAM) [7] and Mixture of Inverse Covariances (MIC) [124, 125] have been successfully implemented in the HMM-based speech recognition. Furthermore, the HLDA [68] feature projection scheme can also be considered as a special form of precision matrix modelling technique [110]. In the following, a unifying framework of basis superposition will be described for the above techniques. The following two chapters will discuss the Maximum Likelihood (Chapter 4) and Minimum Phone Error (Chapter 5) training of these models on LVCSR systems.

The major issues with using full covariance matrices is the large number of free parameters which are difficult to estimate reliably and the high likelihood computational cost. Therefore, a compact model representation is required such that the total number of model parameters is reduced and the underlying structure of the model can be exploited to yield efficient likelihood calculation. One way of achieving a compact model is by *sharing* some of the parameters over many Gaussian components in the system. The form of parameter sharing can be chosen carefully such that part of the likelihood calculation costs can also be shared over these Gaussian components to yield an efficient model. This can be realised by approximating the precision matrix as a linear basis superposition structure. In the basis superposition formulation, precision matrices are modelled as a linear superposition of a set of symmetric matrices, $\{\boldsymbol{S}_i; 1 \leq i \leq n\}$; weighted by a corresponding set of coefficients $\{\lambda_{smi}; 1 \leq i \leq n\}$. This can be expressed as

$$\boldsymbol{P}_{sm} = \sum_{i=1}^{n} \lambda_{smi} \boldsymbol{S}_i = \sum_{i=1}^{n} \lambda_{smi} \left(\sum_{r=1}^{R_i} \lambda_{ir} \boldsymbol{a}'_{ir} \boldsymbol{a}_{ir} \right) \tag{3.18}$$

where $\boldsymbol{P}_{sm}$ denotes the precision matrix for component m in state s, n is the *basis order* and R_i is the rank of $\boldsymbol{S}_i$. The right hand side of equation (3.18) indicates that the basis matrix, $\boldsymbol{S}_i$, can be decomposed into R_i rank-1 matrices, $\boldsymbol{a}'_{ir}\boldsymbol{a}_{ir}$, where $\boldsymbol{a}_{ir}$ is the rth basis *row* vector for the ith basis matrix. This basis decomposition forms the fundamental concept of the basis superposition precision matrix modelling framework. The aim is to extract common basic structures from a large number of precision matrices and model these using the basis matrices, $\boldsymbol{S}_i$ (shared by Gaussian components). This reduces the number of component specific parameters (λ_{smi}) required and yields a compact model representation. The basis superposition framework also provides a flexibility of adjusting the basis order, n, to vary the model complexity.

3.5.1 Compact Model Representation

The primary advantage of a basis superposition formulation is its compact model representation. From equation (3.18), the parameters, $\{\boldsymbol{S}_i, \lambda_{smi}\}$, can be divided into two sets:

- **"Global" parameters**: a set of basis matrices $\boldsymbol{S}_i$

- **"Component" parameters**: a set of basis coefficients λ_{smi}

For some of the precision matrix approximations, the basis matrices are chosen to be rank-1 ($R = 1$) symmetric for all basis, then $\boldsymbol{S}_i = \boldsymbol{a}_i'\boldsymbol{a}_i$ and

$$\boldsymbol{P}_{sm} = \sum_{i=1}^{n} \lambda_{smi}\boldsymbol{a}_i'\boldsymbol{a}_i = \boldsymbol{A}'\boldsymbol{\Lambda}_{sm}\boldsymbol{A} \tag{3.19}$$

where the basis *row* vector, $\boldsymbol{a}_i$, is the ith row of $\boldsymbol{A}$. $\boldsymbol{\Lambda}_{sm}$ is a diagonal matrix with diagonal elements λ_{smi}. In the case, where there is a large number of precision matrices to be modelled, the above parameterisation leads to a compact model representation through the sharing of basis matrices. In other words, the basis matrices form a set common basic symmetric structures extracted from a large set of precision matrices. A set of coefficients (superposition weights) are then assigned to each individual component to yield the desired precision matrices. This is the well-known concept of *basis decomposition*. The basis superposition technique is applicable to both covariance and precision matrix approximations. However, it is generally more efficient to model the precision matrices in terms of the likelihood calculation cost. Efficient likelihood calculation for precision matrix approximation schemes will be discussed next.

3.5.2 Efficient Likelihood Calculation

One of the attractive attributes of precision matrix modelling is its computational efficiency during decoding. This is mainly due to the fact that the likelihood expression is directly related to the precision matrix (*c.f.* equation (3.1)):

$$\mathcal{L} = \log p(\boldsymbol{o}_t|\boldsymbol{\mu}_{sm}, \boldsymbol{P}_{sm}) = K + \frac{1}{2}\Big\{\log|\boldsymbol{P}_{sm}| - (\boldsymbol{o}_t - \boldsymbol{\mu}_{sm})'\boldsymbol{P}_{sm}(\boldsymbol{o}_t - \boldsymbol{\mu}_{sm})\Big\} \tag{3.20}$$

where $\log p(\boldsymbol{o}_t|\boldsymbol{\mu}_{sm}, \boldsymbol{P}_{sm})$ is the log likelihood of the model parameters given the observations at time t, $\boldsymbol{o}_t$. The efficiency is clearly illustrated using the basis superposition framework. Substituting equation (3.18) into equation (3.20) yields

$$\begin{aligned}
\mathcal{L} = \log p(\boldsymbol{o}_t|\boldsymbol{\mu}_{sm}, \boldsymbol{P}_{sm}) &= K + \frac{1}{2}\left\{\log|\boldsymbol{P}_{sm}| - \sum_{i=1}^{n}\lambda_{smi}(\boldsymbol{o}_- - \boldsymbol{\mu}_{sm})'\boldsymbol{S}_i(\boldsymbol{o}_- - \boldsymbol{\mu}_{sm})\right\} \\
&= K + \frac{1}{2}\left\{\log|\boldsymbol{P}_{sm}| - \sum_{i=1}^{n}\lambda_{smi}\boldsymbol{o}_t'\boldsymbol{S}_i\boldsymbol{o}_t \right. \\
&\qquad \left. - \boldsymbol{\mu}_{sm}'\boldsymbol{P}_{sm}\boldsymbol{\mu}_{sm} + 2\sum_{i=1}^{n}\lambda_{smi}\boldsymbol{\mu}_{sm}'\boldsymbol{S}_i\boldsymbol{o}_t\right\}
\end{aligned} \tag{3.21}$$

The terms in equation (3.21) can be divided into two parts:

- **Model dependent**: $\log|\boldsymbol{P}_{sm}|$ and $\boldsymbol{\mu}_{sm}'\boldsymbol{P}_{sm}\boldsymbol{\mu}_{sm}$

- **Observation dependent**: $\boldsymbol{o}_t'\boldsymbol{S}_i\boldsymbol{o}_t$ and $\boldsymbol{S}_i\boldsymbol{o}_t$

The model dependent terms can be precomputed and cached once the model parameters have been estimated. On the other hand, the terms dependent on the observation vectors can be cached and reused for all the Gaussian components. This yields a significant reduction in computational cost.

Furthermore, when the basis matrices are rank-1 symmetric matrices, likelihood calculation can be achieved more efficiently. In this case, the likelihood expression in equation (3.20) can be rewritten as

$$\begin{aligned}\mathcal{L} = \log p(\boldsymbol{o}_t|\boldsymbol{\mu}_{sm}, \boldsymbol{P}_{sm}) &= K + \frac{1}{2}\Big\{\log|\boldsymbol{P}_{sm}| - (\boldsymbol{o}_t - \boldsymbol{\mu}_{sm})'\boldsymbol{A}'\boldsymbol{\Lambda}_{sm}\boldsymbol{A}(\boldsymbol{o}_t - \boldsymbol{\mu}_{sm})\Big\} \\ &= K + \frac{1}{2}\Big\{\log|\boldsymbol{P}_{sm}| - (\boldsymbol{o}_t'\boldsymbol{A}' - \boldsymbol{\mu}_{sm}'\boldsymbol{A}')\boldsymbol{\Lambda}_{sm}(\boldsymbol{A}\boldsymbol{o}_t - \boldsymbol{A}\boldsymbol{\mu}_{sm})\Big\} \\ &= K + \frac{1}{2}\Big\{\log|\boldsymbol{P}_{sm}| - (\tilde{\boldsymbol{o}}_t' - \tilde{\boldsymbol{\mu}}_{sm}')\boldsymbol{\Lambda}_{sm}(\tilde{\boldsymbol{o}}_t - \tilde{\boldsymbol{\mu}}_{sm})\Big\} \qquad (3.22)\end{aligned}$$

where $\tilde{\boldsymbol{o}}_t = \boldsymbol{A}\boldsymbol{o}_t$ and $\tilde{\boldsymbol{\mu}}_{sm} = \boldsymbol{A}\boldsymbol{\mu}_{sm}$ are the "projected" observation and mean vectors respectively. The model dependent terms are $\log|\boldsymbol{P}_{sm}|$ and $\tilde{\boldsymbol{\mu}}_{sm}$ and the observation term to be cached is the transformed observation, $\tilde{\boldsymbol{o}}_t$. Two models that have this form are STC and EMLLT models, where the matrix $\boldsymbol{A}$ can be viewed as a feature transformation matrix. Once the observation and mean vectors have been transformed, the likelihood expression simplifies to the form of diagonal covariance system with the 'variances'[4] given by $1/\lambda_{smi}$. The likelihood computation is thus linearly proportional to the basis order, n.

3.6 Basis Superposition – A Unifying Framework

In the literature, various forms of precision matrix modelling techniques have been applied to HMM-based speech recognition systems. In this section, a general basis superposition framework is introduced to unify various existing precision matrix models. Table 3.1 compares different

Precision Matrix Model	Basis Type	Basis Order, n
STC	rank-1 symmetric	d
EMLLT	rank-1 symmetric	$d < n \leq \frac{d}{2}(d+1)$
Hybrid-EMLLT	rank-R symmetric	$d \leq nR \leq \frac{d}{2}(d+1)$
SPAM	any symmetric	$d \leq nd \leq \frac{d}{2}(d+1)$

Table 3.1 *Precision matrix models as basis decomposition techniques*

[4] As described in Section 3.6.2, this is merely the variance contribution which corresponds to the basis row vector, $\boldsymbol{a}_i$

precision matrix models within the basis superposition framework. These models are differentiated by the different basis order and rank. For example, when rank-1 basis matrices are used ($R = 1$), equation (3.1) becomes a STC model [34] when $n = d$ and an EMLLT model [85, 86] when ($d < n \leq \frac{d}{2}(d+1)$). Furthermore, the Hybrid-EMLLT [127] model consists of rank-R basis matrices, where $1 < R < d$. Typically, R is chosen to be small (between 2 and 4) so that the basis vector update for the EMLLT model can be used. Finally, SPAM [7] is the most general form of precision matrix model among those listed in Table 3.1. The basis matrices of this model are arbitrary symmetric matrices. Among these models, SPAM is the most powerful model. The following sections describe the specific attributes of these standard models.

3.6.1 Semi-tied covariance

Semi-tied covariance (STC) [34] (also known as Maximum Likelihood Linear Transform or MLLT [53]), is an example of a precision matrix model. When a *global* transformation matrix [5] is used, the precision matrix can be expressed as

$$\boldsymbol{P}_{sm} = \boldsymbol{A}'\boldsymbol{\Lambda}_{sm}\boldsymbol{A} = \sum_{i=1}^{d} \lambda_{smi}\boldsymbol{a}_i'\boldsymbol{a}_i \tag{3.23}$$

where $\boldsymbol{A}$ is a $d \times d$ global transformation matrix and $\boldsymbol{\Lambda}_{sm}$ is a diagonal matrix whose leading diagonal elements, $\lambda_{smj} = 1/\sigma^2_{smj}$, are the inverse variances in the *transformed* space due to the matrix $\boldsymbol{A}$. If $\boldsymbol{\Sigma}_{sm}$ is the covariance matrix in the original space, then the covariance matrix in the *transformed* space becomes $\boldsymbol{A}\boldsymbol{\Sigma}_{sm}\boldsymbol{A}'$; the corresponding precision matrix is given by $\boldsymbol{A}'^{-1}\boldsymbol{P}_{sm}\boldsymbol{A}^{-1} = \boldsymbol{\Lambda}_{sm}$, using equation (3.23).

The total number of free parameters in this model is $d(M + d)$. For LVCSR systems ($M >> d$), so the total number of parameters is approximately the same as the case of diagonal covariance matrix (Md). Here, the number of parameters associated with the transformation matrix, $\boldsymbol{A}$, is negligible compared to the total number of model parameters of the system. In fact, the global STC system can be viewed as a diagonal covariance matrix system with an additional global feature transformation. The STC formulation allows the transformation matrix, $\boldsymbol{A}$ and the transformed variances, σ^2_{smj} to be estimated efficiently using an iterative closed-form update formulae (see Chapter 4 for more details).

[5] STC can also operate in a multiple-transform mode. This will be discussed further in Section 3.7

3.6.2 Extended MLLT

Extended MLLT (EMLLT) [85] is an extension to the STC (MLLT) model where the precision matrix is given by

$$\boldsymbol{P}_{sm} = \boldsymbol{A}'\boldsymbol{\Lambda}_{sm}\boldsymbol{A} = \sum_{i=1}^{n} \lambda_{smi}\boldsymbol{a}_i'\boldsymbol{a}_i \tag{3.24}$$

where $\boldsymbol{a}_i$ is the ith row of the $n \times d$ matrix, $\boldsymbol{A}$, and λ_{smi} is the ith leading diagonal element of the $n \times n$ diagonal matrix $\boldsymbol{\Lambda}_{sm}$. The precision matrix is also a weighted sum of n rank-1 symmetric matrices. The number of rank-1 bases is constrained by $d \leq n \leq \frac{d}{2}(d+1)$. The lower bound is required to obtain at least d independent $\boldsymbol{a}_i$ to ensure a positive-definite precision matrices. The upper bound is the total number of free parameters for a full covariance matrix, beyond which redundancies will be introduced into the EMLLT model. Thus, n can be viewed as a parameter that can be adjusted to alter the complexity of the model, from a STC model ($n = d$) to a full covariance matrix model ($n = \frac{d}{2}(d+1)$).

It is interesting to note that for EMLLT, some of the basis coefficients are allowed to take negative values while still resulting in a positive-definite precision matrix. In fact, to ensure positive-definiteness, there should be at least d positive basis coefficients whose basis vectors are linearly independent. This implies that there can be up to $n-d$ negative basis coefficients. Since the basis row vectors, $\boldsymbol{a}_i$, form a transformation matrix, $\boldsymbol{A}$ and the basis coefficients, λ_{smi}, can be viewed as the variance along the projection given by $\boldsymbol{a}_i$. Since some of these basis coefficients are negative, does this mean the EMLLT model is capable of modelling negative 'variances'? The answer in *no*.

Variance is by definition a positive quantity. To account for the *negative* 'variances' in an EMLLT model, let p_{ii} be the precision (inverse variance) along the projection given by $\boldsymbol{a}_i$. So,

$$p_{ii} = \boldsymbol{a}_i\boldsymbol{P}_{sm}\boldsymbol{a}_i' \tag{3.25}$$

Substituting the EMLLT precision matrix expression in equation (3.24) yields

$$\begin{aligned} p_{ii} &= \boldsymbol{a}_i'\left(\boldsymbol{A}'\boldsymbol{\Lambda}_{\boldsymbol{sm}}\boldsymbol{A}\right)\boldsymbol{a}_i' \\ &= \sum_{j=1}^{n} \lambda_{smj}(\boldsymbol{a}_i'\boldsymbol{a}_j)^2 \end{aligned} \tag{3.26}$$

Clearly, $p_{ii} = \lambda_{smi}$ if and only if

$$(\boldsymbol{a}_i'\boldsymbol{a}_j)^2 = \begin{cases} 1 & i = j \\ 0 & \text{otherwise} \end{cases} \tag{3.27}$$

This condition is satisfied only when $\boldsymbol{A}$ is orthonormal. However, for STC and EMLLT, $\boldsymbol{A}$ is not constrained to be orthonormal. So, according to equation (3.26), the *true* projected variance is actually given by the *joint* contributions from all the basis vectors. Thus, λ_{smi} can be regarded

as the *contribution* from a particular basis and a 'negative' λ_{ii} simply means a *negative* precision contribution associated to the basis vectors $\boldsymbol{a}_i$. Note that the overall projected precisions (and hence variances) have to be positive because the precision matrix, $\boldsymbol{P}_{sm}$, is positive-definite.

3.6.3 Subspace for precision and mean

Subspace for Precision and Mean (or SPAM) [7] is a model that constrains the mean vector and the precision matrix to be within a subspace of the space spanned by a symmetric matrix (a space with $\frac{d}{2}(d+1)$ degrees of freedom). A special form of SPAM model, which leaves the mean vectors unconstrained [7], is simply a precision matrix model. This formulation is similar to the Mixture of Inverse Covariances (MIC) [124, 125]. In [5], this form of SPAM model is categorised as a Precision Constrained Gaussian Mixture Model (PCGMM). It is a further generalisation of the EMLLT model where the rank-1 constraint on the basis matrices is relaxed so that the bases can be any arbitrary symmetric matrices. Under this formulation, the precision matrix is given by

$$\boldsymbol{P}_{sm} = \sum_{i=1}^{n} \lambda_{smi} \boldsymbol{S}_i = \sum_{i=1}^{n} \lambda_{smi} \left(\sum_{r=1}^{R_i} \lambda_{ir} \boldsymbol{a}'_{ir} \boldsymbol{a}_{ir} \right) \tag{3.28}$$

where $\boldsymbol{S}_i = \sum_{r=1}^{R_i} \lambda_{ir} \boldsymbol{a}'_{ir} \boldsymbol{a}_{ir}$ is the decomposition of $\boldsymbol{S}_i$ into R_i rank-1 matrices. This allows $\boldsymbol{S}_i$ to take any arbitrary symmetric matrix. If λ_{ir} is constrained to be positive and $R_i = d$, the resulting symmetric matrices will be positive-definite. However, this is not a strict requirement. As indicated by equation (3.28), a SPAM precision matrix model can also be viewed as a *nested* sum of $N = \sum_{i=1}^{N} R_i$ rank-1 matrices, a special form of the EMLLT model. Similar to the EMLLT model, N is bounded by $d \leq N \leq \frac{d}{2}(d+1)$. Thus, the number of rank-1 matrices associated with each precision matrix should be at least d to ensure positive-definiteness and at most $\frac{d}{2}(d+1)$ where the rank-1 matrices span the entire space of a symmetric matrix. Note that when at least one of the basis matrices, $\boldsymbol{S}_i$, is positive-definite ($\lambda_{ir} > 0$ and $R_i = d$), n may be reduced to 1, including only the positive-definite basis matrix, while maintaining the positive-definiteness of the resulting precision matrices, $\boldsymbol{P}_{sm}$.

3.6.4 Hybrid EMLLT

Another generalisation of the EMLLT model is known as the Hybrid EMLLT model [127] model. This model is also very similar to the SPAM model, except that the rank of the basis matrices are restricted to be R. R can take any integer value between 1 and d. The former yields the EMLLT model while the latter yields the SPAM model. So, the hybrid EMLLT precision matrix can be

written as

$$\boldsymbol{P}_{sm} = \sum_{i=1}^{n} \lambda_{smi} \boldsymbol{S}_i = \sum_{i=1}^{n} \lambda_{smi} \left(\sum_{r=1}^{R} \lambda_{ir} \boldsymbol{a}'_{ir} \boldsymbol{a}_{ir} \right) \tag{3.29}$$

The only difference between equations (3.28) and (3.29) is the rank of the basis matrices. It was shown in [127] that the rank of the basis matrices can be reduced to between 2–4 without significant lost in terms of word error rate (WER) performance compared to a SPAM model.

3.6.5 Heteroscedastic LDA (HLDA)

Linear projection schemes are typically employed to improve the performance of a diagonal covariance matrix system, including the Linear Discriminant Analysis (LDA) [105], Heteroscedastic Discriminant Analysis (HDA) [44] and Heteroscedastic Linear Discriminant Analysis (HLDA) [68]. These methods achieve both dimension reduction and *feature decorrelation*. LDA computes a global linear transform such that the ratio of the projected between class variance and the within class variance is maximised. The within class covariance matrices are assumed to be the same and is calculated as the average covariance matrix of all the components. HLDA extends LDA by having component specific within class covariance matrices. An HLDA projection matrix may be expressed as

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}^{\mathtt{H}} \\ \boldsymbol{A}^{\mathtt{N}} \end{bmatrix} \tag{3.30}$$

where $\boldsymbol{A}^{\mathtt{H}}$ and $\boldsymbol{A}^{\mathtt{N}}$ are the $n \times d$ HLDA projection and $(d-n) \times d$ nuisance projection respectively. If the mean and covariance matrix of component m in state s in the original feature space are given by $\boldsymbol{\mu}_{sm}$ and $\boldsymbol{\Sigma}_{sm}$ respectively, the corresponding mean and covariance matrix in the transformed space are given by

$$\bar{\boldsymbol{\mu}}_{sm} = \boldsymbol{A}\boldsymbol{\mu}_{sm} \tag{3.31}$$

$$\bar{\boldsymbol{\Sigma}}_{sm} = \boldsymbol{A}\boldsymbol{\Sigma}_{sm}\boldsymbol{A}' \tag{3.32}$$

The covariance matrix in the transformed space, $\bar{\boldsymbol{\Sigma}}_{sm}$, is assumed diagonal. The mean and covariance matrix in the nuisance space are shared *globally* by all the Gaussian components in the system. Therefore,

$$\bar{\boldsymbol{\mu}}_{sm} = \begin{bmatrix} \boldsymbol{A}^{\mathtt{H}}\boldsymbol{\mu}_{sm} \\ \boldsymbol{\mu} \end{bmatrix} \tag{3.33}$$

$$\bar{\boldsymbol{\Sigma}}_{sm} = \begin{bmatrix} \boldsymbol{A}^{\mathtt{H}}\boldsymbol{\Sigma}_{sm}\boldsymbol{A}^{\mathtt{H}'} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\Sigma} \end{bmatrix} \tag{3.34}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are the global mean and covariance matrix of the nuisance space. From equation (3.32), the precision matrix of component m in state s in the original space is given by

$$\boldsymbol{P}_{sm} = \boldsymbol{A}'\boldsymbol{\Lambda}_{sm}\boldsymbol{A} = \sum_{i=1}^{n} \lambda_{smi}\boldsymbol{a}_i'\boldsymbol{a}_i + \sum_{i=n+1}^{d} \lambda_i\boldsymbol{a}_i'\boldsymbol{a}_i \tag{3.35}$$

where

$$\boldsymbol{\Lambda}_{sm} = \bar{\boldsymbol{\Sigma}}_{sm}^{-1} = \begin{pmatrix} \boldsymbol{\Lambda}_{sm}^{\mathtt{H}} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Lambda}^{\mathtt{N}} \end{pmatrix} \tag{3.36}$$

Note that this equation may be justified using the basis superposition framework. It is very similar to STC with a global transform where the variances in the projected nuisance space are shared by all the components in the system. Therefore, the first n basis coefficients are component specific while the final $d-n$ coefficients are tied over all components. As a result, the second term in equation (3.35) is independent of the HMM states, s, and the GMM components, m. The HLDA parameters may be estimated efficiently using the STC's iterative row-by-row update scheme [34].

In conventional HLDA, the nuisance dimension mean vector is also global. Hence, only the useful dimension parameters are retained because the nuisance parameters will only introduce a constant offset when calculating the likelihood. Instead of viewing this as a linear projection scheme, HLDA can be used *purely* as a precision matrix model by allowing the mean in the nuisance space to be component specific. Therefore, no constraint is applied to the mean vector and may be modelled in the original feature space. The use of HLDA as precision matrix model was introduced in [110]. For convenience, this form of model is called HLDA-PMM to distinguish this from the conventional HLDA model.

3.6.6 Factored Semi-tied Covariance

Factored Semi-tied Covariance (or factored STC) is an extension to the STC model where each component in the system selects d basis vectors from a larger set of $n > d$ basis vectors. This is also equivalent to an EMLLT model where $n - d$ basis coefficients for each component are selectively tied to zero. Therefore, only d basis vectors are superimposed to form the final precision matrix of each Gaussian component. The expression for a factored STC precision matrix may be written as

$$\boldsymbol{P}_{sm} = \sum_{i \in \mathcal{I}_{sm}} \lambda_{smi}\boldsymbol{a}_i'\boldsymbol{a}_i \tag{3.37}$$

where $\mathcal{I}_{sm}$ denotes the set of indices of the selected basis vectors for component m in state s. Although the basis row vectors, $\boldsymbol{a}_i$, are not component specific, the component dependent selection indicated by $\mathcal{I}_{sm}$ allows the *effective* basis vectors to vary from component to component. However, the determination of the selection sets, $\mathcal{I}_{sm}$, is computationally expensive. Note that if the selection sets are tied within groups of Gaussian components such that

$$\cap_{g=1}^{G} \mathcal{I}_g = \emptyset \quad \text{and} \quad \cup_{g=1}^{G} \mathcal{I}_g = \{1, 2, \ldots, n\} \tag{3.38}$$

where G is the total number of Gaussian groups and $\mathcal{I}_g$ is the selection set for the gth Gaussian group, the factored STC system degenerates to a multiple-transform STC system.

3.7 Multiple Bases Systems

The formulation of the basis superposition structure to achieve a compact model representation relies on a key concept of *parameter tying*. In machine learning, parameter tying is often employed to reduce the total number of free parameters in a system to reduce the risk of *overfitting*. This is likely to occur when there is insufficient training data. In a speech recognition system, parameter tying can be applied to different sets of parameters. This is best illustrated using the hierarchical structure in Figure 3.4. Each node in the hierarchy indicates a possible

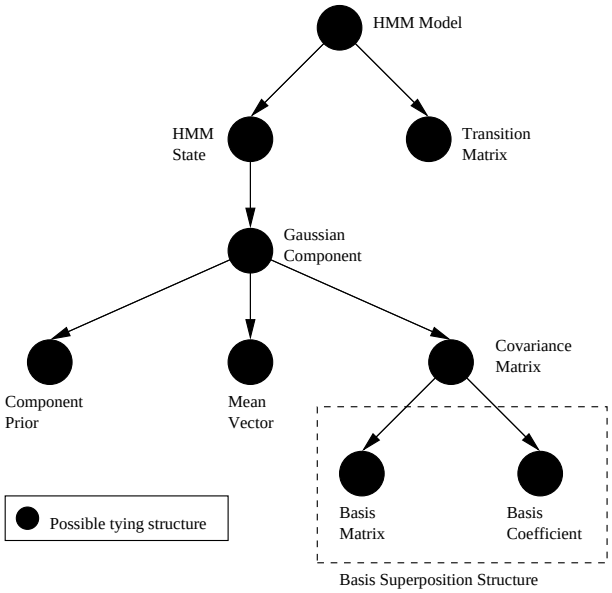

Figure 3.4 *A hierarchical parameter tying structure for an HMM system.*

tying structure. At the top most level, the HMM models can be tied. Moving down the hierarchy, the HMM states, transition matrices and Gaussian components can also be tied. Furthermore, it is also possible to tie the Gaussian mean vector and covariance matrices separately. Higher level parameter tying schemes has been introduced in Section 2.2.1. This section will concentrate on the discussion the tying of covariance matrices and its underlying structure.

The simplest form of tying is to share covariance matrices within clusters of Gaussian components so that the number of covariance matrices in the system is reduced. This kind of system is known as a *tied variance* system. A tied variance system can be built by clustering the covariance matrices of a system. Alternatively, if the GMMs in the system is trained using an iterative mixture-splitting approach, a tied-variance system can be constructed conveniently

by not splitting the covariance matrices, *i.e.* the split Gaussian components share the same covariance matrix.

In Section 3.5, the basis superposition framework was introduced as a compact precision matrix representation. A single set of basis matrices is used to capture the underlying basic structures of all the Gaussian components (a *global* tying scheme). This form of tying is illustrated in Figure 3.5 (left). In this figure, the horizontal black strips represent the basis row

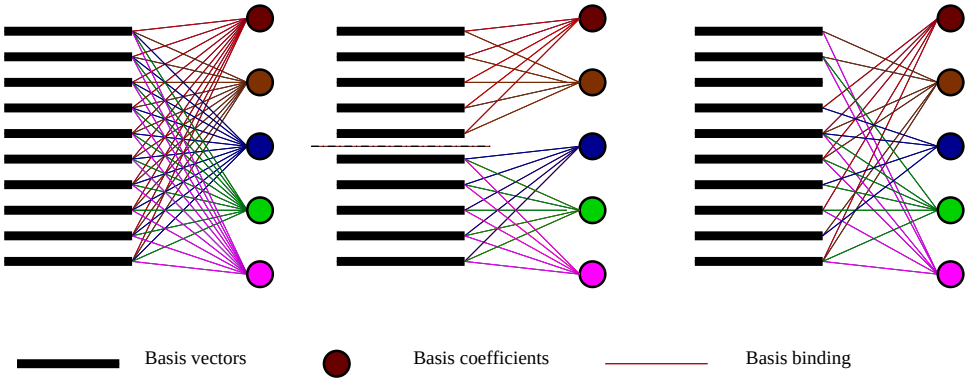

Figure 3.5 *Various multiple bases configurations: global bases (left), multiple bases with hard binding (middle) and multiple bases with soft binding.*

vectors while the coloured circles represent the component specific basis coefficients (different colour indicates different component). In this simple configuration, there is a complete binding between the basis vectors and the basis coefficients of all the components, *i.e.* no tying of basis coefficients. The global STC, EMLLT and SPAM models fall within this category.

Due to the large number of Gaussian components in an LVCSR system, a higher basis order is required to obtain a good representation. Unfortunately, increasing the basis order increases both the memory and computational requirements (see Sections 3.5.2 and 4.3.1). Alternatively, the components can be partitioned into clusters, as shown in Figure 3.5 (middle). In this example, the components are grouped into two clusters: the top cluster contains two components while the bottom cluster contains three. The basis binding is complete within each cluster but no binding is allowed across clusters. This form of binding is known as *hard* binding. Now, each cluster contains a smaller number of Gaussian components. Extracting basis from each cluster of Gaussian components should yield a more accurate basis information. Recall that for rank-1 basis matrices, the basis vectors form the rows of the transformation matrix. Thus, tying the basis matrices at a cluster level leads to a multiple transformation scheme where each transformation is associated with each cluster of Gaussian components. A good summarisation of multiple projections schemes is given in [33]. Multiple Linear Transforms (MLT) [45] models such as multiple STC and EMLLT transforms models are examples of multiple bases systems with *hard* binding.

Another form of basis binding, known as the *soft* binding is shown in Figure 3.5 (right). This configuration allows an arbitrary binding between the basis vectors and basis coefficients. In other words, some of the basis coefficients are tied to zero to remove some of the basis bindings. Therefore, each component *selects* an individual subset of the basis vectors to form the final precision matrix. The factored semi-tied covariances (factored-STC) [35] as described previously in Section 3.6.6 is an example of the multiple bases systems with *soft* binding. Instead of tying some of the basis coefficients to zero, one may also tie these parameters globally to yield a multiple HLDA projections scheme [73]. This kind of tying has been found to lead to good recognition performance [73]. In addition, this model can also be used to control the model complexity by having different number of retained dimensions for each cluster.

Both methods are similar to the STC and EMLLT models with the exception that the number of basis matrices in the system is larger than the basis coefficients defined for each Gaussian components. An additional set of parameters is defined to associate each basis coefficient with a basis matrix. The multiple bases systems described in this work is a restricted version of factored-STC and MLT in that the association between basis coefficients and basis matrices are pre-defined by a Gaussian clustering process and each cluster of Gaussian components shares the same set of basis matrices (*hard* binding).

To formulate a multiple bases system, let the set of Gaussian components, $\mathcal{C}$, be clustered into N groups, where $\mathcal{C} = \{c1, c2, \ldots, cN\}$. The basis superposition expression in equation (3.18) can be rewritten as

$$\boldsymbol{P}_{sm} = \sum_{i=1}^{n} \lambda_{smi} \boldsymbol{S}_i^{c(sm)} \tag{3.39}$$

where $c(sm) \subseteq \mathcal{C}$ denotes the cluster to which component m in state s belongs to. There are many ways to perform Gaussian clustering. One way is to use a regression class tree [29] and the terminal nodes of the tree corresponds to the clusters of Gaussian components. This approach is adopted in this thesis.

Effectively, the precision matrix of each component in the system is still composed of n bases. So, the required memory to store the projected statistics (as described in Section 4.3.1) is similar to those using global bases, i,e, proportional to n. However, due to the use of multiple bases, the required memory to store the observation cache (see Section 3.5.2) will grow linearly with the number of Gaussian clusters, N, since the terms to be cached are dependent on the basis matrices. Fortunately, the overhead cost incurred is relatively small compared to the likelihood calculation for all the Gaussian components in the system.

3.8 Relationship With Other Frameworks

In Section 3.6.5, HLDA was compared from two distinct perspective: 1) as a global feature projection scheme (conventional HLDA); 2) as a 'pure' precision matrix model (HLDA-PMM). A clear distinction was made between these where basis coefficients of the nuisance dimensions are tied for the latter case to yield a compact precision matrix approximation structure, while leaving the mean vectors unconstrained. In the following, two existing frameworks of modelling both the mean vectors and precision matrices will be described, namely the Subspace Constrained GMMs (SCGMM) [5] and the Product of Gaussians (PoG) [36]. The resulting precision matrix structure from this frameworks is similar to the basis superposition formulation. By applying suitable constraints to these framework, a pure precision matrix model may be obtained.

3.8.1 Subspace Constrained GMMs

Recent research work, closely related to precision matrix modelling, is the Subspace Constrained Gaussian Mixture Models (SCGMMs) [5]. SCGMMs extends the basis superposition formulation to model both the mean vectors and precision matrices. Therefore, many existing precision matrix models described earlier, including the STC, EMLLT and SPAM, are also special cases of a SCGMM. This model expresses the Gaussian probability distribution as a generic exponential family distribution and the parameters associated with this exponential family distribution are constrained to lie within a subspace. A Gaussian distribution, when expressed as a member of the exponential family distribution, takes the following form [5]:

$$\begin{aligned}\mathcal{N}\left(\boldsymbol{o}_t;\boldsymbol{\mu}_{sm},\boldsymbol{\Sigma}_{sm}\right) &= \frac{1}{\sqrt{(2\pi)^d|\boldsymbol{\Sigma}_{sm}|}}\exp\left\{-\frac{1}{2}\left(\boldsymbol{o}_t-\boldsymbol{\mu}_{sm}\right)'\boldsymbol{\Sigma}_{sm}^{-1}\left(\boldsymbol{o}_t-\boldsymbol{\mu}_{sm}\right)\right\} \\ &= \exp\Big\{\boldsymbol{g}(\boldsymbol{\mu}_{sm},\boldsymbol{P}_{sm})'\boldsymbol{f}(\boldsymbol{o}_t)+K(\boldsymbol{\mu}_{sm},\boldsymbol{P}_{sm})\Big\} \qquad (3.40)\end{aligned}$$

where

$$\boldsymbol{f}(\boldsymbol{o}_t) = \begin{bmatrix} -\frac{1}{2}\,\mathrm{vec}(\boldsymbol{o}_t\boldsymbol{o}_t') \\ \boldsymbol{o}_t \end{bmatrix} \qquad (3.41)$$

$$\boldsymbol{g}(\boldsymbol{\mu}_{sm},\boldsymbol{P}_{sm}) = \begin{bmatrix} \mathrm{vec}(\boldsymbol{P}_{sm}) \\ \boldsymbol{\mu}_{sm} \end{bmatrix} \qquad (3.42)$$

$$K(\boldsymbol{\mu}_{sm},\boldsymbol{P}_{sm}) = \frac{1}{2}\Big\{\log|\boldsymbol{P}_{sm}| - d\log(2\pi) - \boldsymbol{\mu}_{sm}'\boldsymbol{P}_{sm}^{-1}\boldsymbol{\mu}_{sm}\Big\} \qquad (3.43)$$

$\mathrm{vec}(.)$ denotes an operator which vectorises the upper triangular elements of a symmetric matrix and d is the dimensionality of the feature. One may view $\boldsymbol{f}(\boldsymbol{o}_t)$ and $\boldsymbol{g}(\boldsymbol{\mu}_{sm},\boldsymbol{\Sigma}_{sm})$ as the effective observation sufficient statistics and parameter vectors respectively for the exponential distribution. Constraints are applied to $\boldsymbol{g}(\boldsymbol{\mu}_{sm},\boldsymbol{\Sigma}_{sm})$ such that it is in an affine subspace of $\mathbb{R}^{\frac{d}{2}(d+3)}$, *i.e.*

$$\boldsymbol{g}(\boldsymbol{\mu}_{sm}, \boldsymbol{\Sigma}_{sm}) = \boldsymbol{g}_0 + \sum_{i=1}^{n} \lambda_{smi} \boldsymbol{g}_i \tag{3.44}$$

The equation resembles the basis superposition formulation in equation (3.18). $\boldsymbol{g}_0$ may be chosen such that the precision matrix portion in equation (3.42) yields a positive-definite matrix. By having appropriate constraints on $\boldsymbol{g}(\boldsymbol{\mu}_{sm}, \boldsymbol{\Sigma}_{sm})$, SCGMMs may be used to formulate existing precision matrix models such as STC, EMLLT and SPAM. See [5] for more details. However, SCGMM is more than a precision matrix approximation technique as it also defines a subspace for the mean vector. Gains were reported in [5] using the SCGMM models over SPAM with unconstrained mean vectors in a small vocabulary task using a grammar based language model. The gain diminished as the basis order, n, was increased.

3.8.2 Product of Experts Systems

An interesting attribute of the basis superposition framework is its close relation to the Product of Experts (PoE) framework [56]. The most common form of the PoE framework is the Product of Gaussians (PoG) framework [1, 36, 129] where the experts are each a Gaussian distribution. Since the product of Gaussian distributions is itself a Gaussian, the effective distribution represented by a PoG framework is also a Gaussian distribution. Consider a PoG with S experts:

$$p(\boldsymbol{o}_t|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{K} \prod_{s=1}^{S} \mathcal{N}(\boldsymbol{o}_t; \boldsymbol{\mu}_s, \boldsymbol{\Sigma}_s) = \mathcal{N}(\boldsymbol{o}_t; \boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{3.45}$$

where K is the normalisation constant to yield a valid distribution. $\boldsymbol{\mu}_s$ and $\boldsymbol{\Sigma}_s$ are the mean and covariance matrix of the sth Gaussian expert respectively. The effective mean and covariance matrix are given by

$$\boldsymbol{\mu} = \boldsymbol{\Sigma} \left(\sum_{s=1}^{S} \boldsymbol{\Sigma}_s^{-1} \boldsymbol{\mu}_s \right) \tag{3.46}$$

$$\boldsymbol{\Sigma} = \left(\sum_{s=1}^{S} \boldsymbol{\Sigma}_s^{-1} \right)^{-1} \tag{3.47}$$

Expressing equation (3.47) in terms of the precision matrices yields

$$\boldsymbol{P} = \boldsymbol{\Sigma}^{-1} = \sum_{s=1}^{S} \boldsymbol{\Sigma}_s^{-1} = \sum_{s=1}^{S} \boldsymbol{P}_s \tag{3.48}$$

Comparing this with equation (3.18), it immediately becomes clear that basis superposition is in fact an example of a Product of Gaussians system. Each basis matrix corresponds to the precision structure of an expert. In the formulation of the basis superposition framework, the basis matrices are not required to be positive definite. Therefore, each expert in the PoG system

may not be a valid distribution. This is not important provided the resulting precision matrix is positive definite.

One special form of the PoG framework is the Product of Gaussian Pancake (PoGP) [129]. The precision matrix of each Gaussian expert in a PoGP system takes the follow special 'pancake' form

$$\boldsymbol{P}_s = \boldsymbol{I} + \boldsymbol{a}_s' \boldsymbol{a}_s \tag{3.49}$$

where the *pancake* is formed by stretching the spherical structure ($\boldsymbol{I}$) along the direction denoted by the row vector $\boldsymbol{a}_s$. The resulting precision matrix is therefore given by

$$\boldsymbol{P} = S\boldsymbol{I} + \sum_{s=1}^{S} \boldsymbol{a}_s' \boldsymbol{a}_s \tag{3.50}$$

This expression is very similar to the EMLLT formulation given by equation (3.24) with the exception that the PoGP framework consists of an additional expert with a full rank spherical precision matrix. The relationship between EMLLT and PoGP is also detailed in [36].

Another form of PoE described in [36] is the Product of Mixture of Gaussians (PoMoG). This is generalisation to the PoG framework such that the experts are represented by Mixture of Gaussians (MoG). Equation (3.45) may be rewritten as (borrowing the notations from [36])

$$\begin{aligned} p(\boldsymbol{o}_t|\boldsymbol{\mu}, \boldsymbol{\Sigma}) &\propto \prod_{s=1}^{S} \left(\sum_{m=1}^{M_s} c_{sm} \mathcal{N}\left(\boldsymbol{o}_t; \boldsymbol{\mu}_{sm}, \boldsymbol{\Sigma}_{sm}\right) \right) \\ &= \sum_{m_1=1}^{M_1} \cdots \sum_{m_S=1}^{M_S} \prod_{s=1}^{S} c_{sm_s} \mathcal{N}\left(\boldsymbol{o}_t; \boldsymbol{\mu}_{sm}, \boldsymbol{\Sigma}_{sm}\right) \\ &= \sum_{\mathbf{m}} \bar{c}_{\mathbf{m}} \mathcal{N}\left(\boldsymbol{o}_t; \boldsymbol{\mu}_{\mathbf{m}}, \boldsymbol{\Sigma}_{\mathbf{m}}\right) \end{aligned} \tag{3.51}$$

where $\mathbf{m} = [m_1 \quad m_2 \quad \dots \quad m_S]$ denotes the *meta components* in the *producted space* and m_s identifies the component from stream s. It is interesting to note that the PoMoG framework yields an effective distribution which is also a MoG, with a total of $\prod_{s=1}^{S} M_s$ meta components. The effective mean and covariance matrix of each meta component is given by

$$\boldsymbol{\mu}_{\mathbf{m}} = \boldsymbol{\Sigma}_{\mathbf{m}} \left(\sum_{s=1}^{S} \boldsymbol{\Sigma}_{sm_s}^{-1} \boldsymbol{\mu}_{sm_s} \right) \tag{3.52}$$

$$\boldsymbol{\Sigma}_{\mathbf{m}} = \left(\sum_{s=1}^{S} \boldsymbol{\Sigma}_{sm_s}^{-1} \right)^{-1} \tag{3.53}$$

Once again, equation (3.53) shows that the precision matrix of each meta component is a summation of the precision matrices of S components, one from each stream.

3.9 Summary

This chapter discussed the aspects of covariance and precision matrix modelling schemes for speech recognition. In general, precision matrix approximation schemes were found to yield more efficient likelihood computation compared to covariance matrix approximation methods. In particular, a unifying framework of basis superposition was introduced to describe existing precision matrix modelling techniques, including the Semi-Tied Covariances (STC), Extended Maximum Likelihood Linear Transform (EMLLT) and Subspace for Precision and Mean (SPAM) models. This is followed by the discussion of multiple bases systems which allows more compact model representation to be achieved. Finally, related work on precision matrix modelling for speech recognition was also discussed. For example, feature transformation and projection schemes (*e.g.* LDA, HDA and HLDA), Subspace Constrained GMMs (SCGMMs) and Product of Gaussians (PoG) may be viewed as precision matrix models. Actually, these techniques are more than a precision matrix model as they also model the structure of the mean vectors in the system.

4

Maximum Likelihood Estimation of Precision Matrix Model Parameters

Maximum Likelihood (ML) is the most widely used parametric estimation criterion. ML estimation of the standard HMM parameters using the efficient EM algorithm [19] was discussed in Section 2.1.3. This chapter will describe the maximum likelihood estimation of various precision matrix models within the unified framework of basis superposition introduced earlier in Chapter 3. The parameter estimation formulae are derived within the Expectation Maximisation (EM) framework. For several models, the resulting auxiliary function does not have a closed form solution, in which case the Generalised Expectation Maximisation (GEM) algorithm (see Section 2.1.3) is adopted.

4.1 Maximum Likelihood Estimation Formulae

The maximum likelihood criterion is given by

$$\mathcal{R}^{\text{ml}}(\boldsymbol{\theta}) = \log p(\mathcal{O}_1^T|\boldsymbol{\theta}) = \sum_{Q_1^T} \left\{ \log p(\mathcal{O}_1^T|Q_1^T, \boldsymbol{\theta}) + \log P(Q_1^T|\boldsymbol{\theta}) \right\} \tag{4.1}$$

where $\sum_{Q_1^T}$ sums through all possible state sequence of length T. This criterion can be optimised efficiently using the Baum Welch algorithm (see Section 2.1.3). This is achieved by iteratively maximising the auxiliary function in equation (2.26), which can be rewritten in terms of the Gaussian parameters as follows:

$$\mathcal{Q}^{\text{ml}}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) = K - \frac{1}{2} \sum_{s=1}^{S} \sum_{m=1}^{M} \sum_{t=1}^{T} \gamma_{sm}^{\text{ml}}(t) \left\{ \log |\boldsymbol{\Sigma}_{sm}| - (\boldsymbol{o}_t - \boldsymbol{\mu}_{sm})' \boldsymbol{\Sigma}_{sm}^{-1} (\boldsymbol{o}_t - \boldsymbol{\mu}_{sm}) \right\} \tag{4.2}$$

where S denotes the number of states in the system, M is the number of Gaussian component per state and T is the length of the observation sequence. $\gamma_{sm}^{\text{ml}}(t)$ denotes the probability of being in component m of state s at time t. $\boldsymbol{\mu}_{sm}$ and $\boldsymbol{\Sigma}_{sm}$ represent the mean vector and covariance

matrices respectively of component m in state s. The above auxiliary function is maximised by setting the Gaussian mean and covariance matrix as

$$\boldsymbol{\mu}_{sm} = \frac{\boldsymbol{x}_{sm}^{\mathtt{ml}}}{\beta_{sm}^{\mathtt{ml}}} \quad \text{and} \quad \boldsymbol{\Sigma}_{sm} = \frac{\boldsymbol{W}_{sm}^{\mathtt{ml}}}{\beta_{sm}^{\mathtt{ml}}} \tag{4.3}$$

where the required statistics are given by

$$\boldsymbol{x}_{sm}^{\mathtt{ml}} = \sum_{t=1}^{T} \gamma_{sm}^{\mathrm{ml}}(t) \boldsymbol{o}_t \tag{4.4}$$

$$\boldsymbol{W}_{sm}^{\mathtt{ml}} = \sum_{t=1}^{T} \gamma_{sm}^{\mathrm{ml}}(t) \left(\boldsymbol{o}_t - \boldsymbol{\mu}_{sm}\right)\left(\boldsymbol{o}_t - \boldsymbol{\mu}_{sm}\right)' \tag{4.5}$$

$$\beta_{sm}^{\mathtt{ml}} = \sum_{t=1}^{T} \gamma_{sm}^{\mathrm{ml}}(t) \tag{4.6}$$

For a diagonal covariance matrix system, the optimum parameters are given by the diagonal elements of $\boldsymbol{\Sigma}_{sm}$. Thus, the jth element of the diagonal covariance matrix is given by

$$\sigma_{smj}^2 = \frac{\sum_{t=1}^{T} \gamma_{sm}^{\mathrm{ml}}(t) \left(o_{tj} - \mu_{smj}\right)^2}{\beta_{sm}^{\mathtt{ml}}} \tag{4.7}$$

where o_{tj} and μ_{smj} are the jth element of $\boldsymbol{o}_t$ and $\boldsymbol{\mu}_{sm}$ respectively. Therefore, only the diagonal elements of $\boldsymbol{W}_{sm}^{\mathtt{ml}}$ are required.

When the precision matrices are expressed in terms of a basis superposition formulation, the estimation of the model parameters is more complicated. The precision matrix expressed in a basis superposition form is given by (*c.f.* equation (3.18))

$$\boldsymbol{P}_{sm} = \boldsymbol{\Sigma}_{sm}^{-1} = \sum_{i=1}^{n} \lambda_{smi} \boldsymbol{S}_i \tag{4.8}$$

where $\boldsymbol{S}_i$ and λ_{smi} are the ith basis matrix and basis coefficient respectively. Substituting this expression into the auxiliary function in equation (4.2) yields

$$\begin{aligned} \mathcal{Q}^{\mathrm{ml}}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) &= K + \frac{1}{2} \sum_{s=1}^{S} \sum_{m=1}^{M} \sum_{t=1}^{T} \gamma_{sm}^{\mathrm{ml}}(t) \left\{ \log \left|\boldsymbol{P}_{sm}\right| - \left(\boldsymbol{o}_t - \boldsymbol{\mu}_{sm}\right)' \boldsymbol{P}_{sm} \left(\boldsymbol{o}_t - \boldsymbol{\mu}_{sm}\right) \right\} \\ &= K + \frac{1}{2} \sum_{s=1}^{S} \sum_{m=1}^{M} \left\{ \beta_{sm}^{\mathtt{ml}} \log \left| \sum_{i=1}^{n} \lambda_{smi} \boldsymbol{S}_i \right| - \mathrm{Tr} \left[\left(\sum_{i=1}^{n} \lambda_{smi} \boldsymbol{S}_i \right) \boldsymbol{W}_{sm}^{\mathtt{ml}} \right] \right\} \end{aligned} \tag{4.9}$$

There is no simple closed-form solutions for $\boldsymbol{S}_i$ and λ_{smi} which maximise this auxiliary function. Therefore, the Generalised EM algorithm will be employed with the aim of *improving* (rather than maximising) the auxiliary function in equation (4.9) upon each iteration. To simplify the estimation procedure, the optimisation of $\boldsymbol{S}_i$ and λ_{smi} are considered separately, *i.e.* , when updating $\boldsymbol{S}_i$, λ_{smi} is held constant and *vice versa*.

4.1.1 Estimation of basis matrices

Optimising equation (4.9) with respect to $\boldsymbol{S}_i$ is very difficult. The optimal value of $\boldsymbol{S}_i$ which maximises the auxiliary function corresponds to the point where the first derivative of the auxiliary function with respect to $\boldsymbol{S}_i$ is zero. Differentiating equation (4.9) with respect to $\boldsymbol{S}_i$ yields

$$\frac{\partial \mathcal{Q}^{\text{ml}}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}})}{\partial \boldsymbol{S}_i} = \frac{1}{2} \sum_{s=1}^{S} \sum_{m=1}^{M} \left\{ \beta_{sm}^{\texttt{ml}} \frac{\frac{\partial}{\partial \boldsymbol{S}_i} |\sum_{i=1}^{n} \lambda_{smi} \boldsymbol{S}_i|}{|\sum_{i=1}^{n} \lambda_{smi} \boldsymbol{S}_i|} - \left(\lambda_{smi} \boldsymbol{W}_{sm}^{\texttt{ml}}\right) \right\} \tag{4.10}$$

There is no simple closed form solution of $\boldsymbol{S}_i$ such that equation (4.10) equals zero. Furthermore, equation (4.10) cannot be evaluated analytically due to the term $\frac{\partial}{\partial \boldsymbol{S}_i} |\sum_{i=1}^{n} \lambda_{smi} \boldsymbol{S}_i|$. Hence, numerical optimisation packages are used to obtain the new estimate of $\boldsymbol{S}_i$ [5, 7, 125]. As $\boldsymbol{S}_i$ is in general *globally* shared by all the Gaussian components in the systems, the estimation process has been found to have a poor convergence property [7]. Moreover, it was reported in [5] that the gain from estimating $\boldsymbol{S}_i$ using ML training is relatively small compared to the improvement obtained from updating the basis coefficients, λ_{smi}.

Hence, in this work, the basis matrices of SPAM models are initialised once and are not re-estimated in the subsequent training iterations. However, for STC and EMLLT models where the basis matrices take a special rank-1 symmetric matrix form, *i.e.*

$$\boldsymbol{S}_i = \boldsymbol{a}_i' \boldsymbol{a}_i \tag{4.11}$$

the estimation of each basis row vector, $\boldsymbol{a}_i$, can be achieved more efficiently using an iterative update approach. The update formulae for STC and EMLLT models will be discussed in Sections 4.2.1 and 4.2.2 respectively.

4.1.2 Estimation of basis coefficients

The basis coefficients are estimated such that the auxiliary function in equation (4.9) is maximised. This is achieved when the gradient of the auxiliary function with respect to the basis coefficients equals zero. Differentiating equation (4.9) with respect to λ_{smi} yields

$$\frac{\partial \mathcal{Q}^{\text{ml}}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}})}{\partial \lambda_{smi}} = \frac{1}{2} \left\{ \beta_{sm}^{\texttt{ml}} + \text{Tr}\left[\boldsymbol{S}_i \left(\boldsymbol{\Sigma}_{sm} - \boldsymbol{W}_{sm}^{\texttt{ml}} \right) \right] \right\} \tag{4.12}$$

using the identity in equation (A-6) in the appendix where

$$\frac{\partial \log |\sum_{i=1}^{n} \lambda_{smi} \boldsymbol{S}_i|}{\partial \lambda_{smi}} = \text{Tr}\left[\boldsymbol{S}_i \left(\sum_{i=1}^{n} \lambda_{smi} \boldsymbol{S}_i \right)^{-1} \right] = \text{Tr}\left[\boldsymbol{S}_i \boldsymbol{\Sigma}_{sm} \right] \tag{4.13}$$

Although there is no simple closed form solution of λ_{smi} for which equation 4.12 becomes zero, the expression of the gradient with respect to λ_{smi} can be computed analytically as given by equation 4.12. Therefore, a gradient descent scheme may be used.

Alternatively, the estimation of the basis coefficients can be realised more efficiently using the Polak-Ribeire conjugate-gradient method [94]. Consider a vector of all the basis coefficients associated with component m in state s:

$$\boldsymbol{\lambda}_{sm} = [\lambda_{sm1} \quad \lambda_{sm2} \quad \dots \quad \lambda_{smn}]' \tag{4.14}$$

and let $\boldsymbol{d}_{sm}$ be the *conjugate gradient* for $\boldsymbol{\lambda}_{sm}$. The new estimate of the basis coefficients is found by performing a line search along the direction of $\boldsymbol{d}_{sm}$ which yields an optimum auxiliary function value. Thus, the new estimate of the basis coefficients is given by

$$\hat{\boldsymbol{\lambda}}_{sm} = \boldsymbol{\lambda}_{sm} + \Delta_{sm}\boldsymbol{d}_{sm} \tag{4.15}$$

where the parameter to be optimised, Δ_{sm}, is a scalar quantity. First, the auxiliary function for component m in state s is defined as

$$\mathcal{Q}_{sm}^{\text{ml}} = K_{sm} + \frac{1}{2}\beta_{sm}^{\texttt{ml}}\left\{\log|\boldsymbol{P}_{sm}| - \text{Tr}(\boldsymbol{P}_{sm}\boldsymbol{W}_{sm}^{\texttt{ml}})\right\} \tag{4.16}$$

such that

$$\mathcal{Q}^{\text{ml}}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) = \sum_{s=1}^{S}\sum_{m=1}^{M}\mathcal{Q}_{sm}^{\text{ml}} \tag{4.17}$$

The change in the auxiliary function value for component m in state s due to Δ_{sm} is given by

$$\Delta\mathcal{Q}(\Delta_{sm}) = \mathcal{Q}_{sm}^{\text{ml}} - \hat{\mathcal{Q}}_{sm}^{\text{ml}} = \frac{\beta_{sm}^{\texttt{ml}}}{2}\left\{\sum_{j=1}^{d}\log\left(1 + \Delta_{sm}w_{smj}\right) - \Delta_{sm}\sum_{i=1}^{n}d_{smi}\,\text{Tr}(\boldsymbol{W}_{sm}^{\texttt{ml}}\boldsymbol{S}_i)\right\} \tag{4.18}$$

where $\mathcal{Q}_{sm}^{\text{ml}}$ and $\hat{\mathcal{Q}}_{sm}^{\text{ml}}$ are the component auxiliary functions for the old and new model parameters respectively. w_{smj} is the jth eigenvalue of $\boldsymbol{P}_{sm}^{-\frac{1}{2}}\boldsymbol{R}_{sm}\boldsymbol{P}_{sm}^{-\frac{1}{2}}$ and d_{smi} is the ith element of the direction vector, $\boldsymbol{d}_{sm}$, for component m and state s. $\boldsymbol{R}_{sm}$, the effective basis matrix along the direction, $\boldsymbol{d}_{sm}$, is given by

$$\boldsymbol{R}_{sm} = \sum_{i=1}^{n}d_{smi}\boldsymbol{S}_i \tag{4.19}$$

So, the updated basis coefficients are given by

$$\hat{\lambda}_{smi} = \lambda_{smi} + \Delta_{sm}d_{smi} \tag{4.20}$$

Derivations of the conjugate gradient formula and the change in the component auxiliary function in equation (4.18) are given in greater detail in Appendix F.

Line search is a *scalar* (one dimensional) optimisation problem, which can be performed very quickly and efficiently using a simple *bisection* method. Note that

$$1 + \Delta_{sm}w_{smj} > 0 \tag{4.21}$$

for equation (4.18) to be valid. Since w_{smj} can be positive or negative, a valid range of value for Δ_{sm} can be defined as

$$\Delta_{sm}^{\texttt{min}} \leq \Delta_{sm} \leq \Delta_{sm}^{\texttt{max}} \tag{4.22}$$

where

$$\Delta_{sm}^{\min} = \min\left(-\frac{1}{w_{sm}^{\max}}, 0\right) \quad \text{and} \quad \Delta_{sm}^{\max} = \max\left(-\frac{1}{w_{sm}^{\min}}, 0\right) \tag{4.23}$$

$w_{sm}^{\min}$ and $w_{sm}^{\max}$ are the minimum and maximum values of w_{smj} for $j = 1, 2, \ldots, n$. The bisection algorithm is summarised as follows:

```
Initialise Δ^min_sm and Δ^min_sm using equation (4.23)
Compute ΔQ(Δ^min_sm) and ΔQ(Δ^min_sm)
Repeat
     Set Δ_sm = (Δ^min_sm + Δ^max_sm)/2 and compute ΔQ(Δ_sm)
     If ΔQ(Δ^min_sm) > ΔQ(Δ^max_sm)
          Set Δ^max_sm = Δ_sm and recompute ΔQ(Δ^max_sm)
     Else
          Set Δ^min_sm = Δ_sm and recompute ΔQ(Δ^min_sm)
     End If
Until |Δ^max_sm − Δ^min_sm| < threshold
```

Figure 4.1: *The bisection line search algorithm for basis coefficient estimation*

The above algorithm can be repeated many time until a good estimate is obtained.

4.2 Efficient Estimation for Special Cases

In general, the basis matrix and basis coefficient estimation schemes described in Sections 4.1.1 and 4.1.2 respectively may be used to train any precision matrix models which are subsumed by the basis superposition framework. However, if the basis matrices are rank-1 symmetric matrices, more efficient estimation methods are available. In the following, the efficient iterative update schemes for STC and EMLLT models are considered. Derivation of these update formulae can also be found in the respective literatures [32, 34, 85, 86]. For the completeness of this thesis, essential derivations are given in Appendix D and E.

4.2.1 Semi-tied Covariance

Semi-tied covariance (STC) [30, 34] can be described as a basis superposition model using d rank-1 basis matrices. The transformation matrix associated with this model is square and the basis coefficients are positive. The parameters associated with the STC models can be estimated

efficiently using the following update formulae

$$\lambda_{smi} = \frac{1}{\boldsymbol{a}_i \boldsymbol{W}_{sm}^{\texttt{ml}} \boldsymbol{a}_i'} \tag{4.24}$$

$$\boldsymbol{a}_i = \boldsymbol{c}_i \boldsymbol{G}_i^{-1} \sqrt{\frac{\beta}{\boldsymbol{c}_i \boldsymbol{G}_i^{-1} \boldsymbol{c}_i'}} \tag{4.25}$$

where $\boldsymbol{c}_i$ is the row vector of cofactors of $\boldsymbol{A}$ corresponding to the row $\boldsymbol{a}_i$ and

$$\boldsymbol{G}_i = \sum_{s=1}^{S} \sum_{m=1}^{M} \beta_{sm}^{\texttt{ml}} \lambda_{smi} \boldsymbol{W}_{sm}^{\texttt{ml}} \tag{4.26}$$

$$\beta = \sum_{s=1}^{S} \sum_{m=1}^{M} \beta_{sm}^{\texttt{ml}} \tag{4.27}$$

The basis vectors and coefficients can be updated iteratively in an *alternating* fashion using equations (4.24) and (4.25) respectively.

4.2.2 Extended MLLT

The Extended MLLT (EMLLT) model [85] differs from the STC model only in terms of the basis order. An EMLLT precision matrix is formed by superimposing n rank-1 basis matrices, where n is greater than the dimensionality of the feature vectors. As a result, the transformation matrix associated with an EMLLT model is rectangular. Due to this reason, the term $\log |\boldsymbol{A}' \boldsymbol{\Lambda}_{sm} \boldsymbol{A}|$ cannot be easily differentiated with respect to each of the basis vectors, $\boldsymbol{a}_i$. In contrast to the STC case, it cannot be decomposed into determinants of the individual matrices. No closed form solution exists for updating the basis vectors. Thus, a generic gradient-based optimisation algorithm has to be used to update the basis vectors. Newton optimisation method is a popular example. Given the current estimate of the basis vector, $\boldsymbol{a}_i$, the new estimate, $\hat{\boldsymbol{a}}_i$, can be written as

$$\hat{\boldsymbol{a}}_i = \boldsymbol{a}_i + \eta \bar{\boldsymbol{f}}_{sm} \bar{\boldsymbol{H}}_i^{-1} \tag{4.28}$$

where η denotes the learning rate. The gradient vector and Hessian matrix are given by

$$\bar{\boldsymbol{f}}_{sm} = \sum_{s=1}^{S} \sum_{m=1}^{M} \beta_{sm}^{\texttt{ml}} \lambda_{smi} \boldsymbol{a}_i \left(\boldsymbol{\Sigma}_{sm} - \boldsymbol{W}_{sm}^{\texttt{ml}} \right) \tag{4.29}$$

$$\bar{\boldsymbol{H}}_i = \sum_{s=1}^{S} \sum_{m=1}^{M} \beta_{sm}^{\texttt{ml}} \lambda_{smi} \left\{ \boldsymbol{\Sigma}_{sm} \left(1 - \boldsymbol{a}_i \boldsymbol{\Sigma}_{sm} \boldsymbol{a}_i' \right) - \boldsymbol{W}_{sm}^{\texttt{ml}} - \lambda_{smi} \boldsymbol{\Sigma}_{sm} \boldsymbol{a}_i' \boldsymbol{a}_i \boldsymbol{\Sigma}_{sm} \right\} \tag{4.30}$$

respectively. The basis coefficients, on the other hand, may be updated using the following formula

$$\hat{\lambda}_{smi} = \lambda_{smi} + \left(\frac{1}{\boldsymbol{a}_i \boldsymbol{W}_{sm}^{\texttt{ml}} \boldsymbol{a}_i'} - \frac{1}{\boldsymbol{a}_i \boldsymbol{\Sigma}_{sm} \boldsymbol{a}_i'} \right) \tag{4.31}$$

Full derivation of equations (4.29), (4.30) and (4.31) is given in Appendix E. The update rule given in equation (4.31) is known as the additive update [85]. This method yields both positive and negative coefficients. Alternatively, a multiplicative update [85] may be used, which is given by

$$\hat{\lambda}_{smi} = \lambda_{smi} \left(\frac{\boldsymbol{a}_i \boldsymbol{\Sigma}_{sm} \boldsymbol{a}_i'}{\boldsymbol{a}_i \boldsymbol{W}_{sm}^{\mathtt{ml}} \boldsymbol{a}_i'} \right) \tag{4.32}$$

The multiplicative update yields only positive basis coefficients. The resulting model is less flexible because it does not allow negative variance contributions. Thus, the additive coefficient update has been found to outperform the multiplicative update [85]. The update formulae for the EMLLT model can be easily extended to update the hybrid EMLLT model [127].

4.3 Implementation Issues

This section addresses the implementation issues of various precision matrix models, paying particular attention to building LVCSR systems. Many of these models have been successfully applied to LVCSR systems [6, 34, 58]. This paper emphasises issues such as memory requirement, computational feasibility and training robustness in LVCSR systems. System efficiency may be adversely affected if these issues are not addressed properly.

4.3.1 Memory Issues

One issue with implementing precision matrix models on LVCSR systems is the large amount of memory required for the statistics accumulation. Recall that the sufficient statistics for updating a precision matrix model parameters are given by equations (4.4), (4.5) and (4.6). The memory requirement is dominated by the full covariance statistics, $\boldsymbol{W}_{sm}^{\mathtt{ml}}$, one for each component m in every state s. $\boldsymbol{W}_{sm}^{\mathtt{ml}}$ is a symmetric matrix with $\frac{d}{2}(d+1)$ free parameters. For a large vocabulary system with 9000 states, 36 components per state and 39-dimensional feature vector, the total amount of memory required to store all the full covariance statistics is just under 1Gb [1].

Given a good set of basis (through good initialisation schemes to be discussed in Section 4.3.2 or a few ML training iterations), the precision matrix models can be refined by simply updating the basis coefficients alone. This is more efficient in terms of memory requirement because the sufficient statistics can be reduced to a more compact form known as the *projected* statistics, $\tilde{\boldsymbol{y}}_{sm}^{\mathtt{ml}}$. $\tilde{\boldsymbol{y}}_{sm}^{\mathtt{ml}}$ is an n-dimensional vector whose ith element is given by

$$\tilde{y}_{smi}^{\mathtt{ml}} = \mathrm{Tr}\left(\boldsymbol{S}_i \boldsymbol{W}_{sm}^{\mathtt{ml}}\right) \tag{4.33}$$

[1] Assuming that each parameter requires 4 bytes (32 bits), the total amount of memory required is computed as $4 \times \frac{d}{2}(d+1) \times M \times S$ bytes

for $i = 1, 2, \ldots, n$. $\tilde{y}^{\mathtt{ml}}_{smi}$ is a scalar term associated with each basis matrix, $\boldsymbol{S}_i$. Therefore, the total amount of memory required is proportional to the basis order, n, rather than $\frac{d}{2}(d+1)$ for the full covariance statistics, $\boldsymbol{W}^{\mathtt{ml}}_{sm}$. This dramatically reduces the total memory requirement for large vocabulary systems.

Hence, equation (4.12) may be rewritten in terms of the projected statistics as

$$\frac{\partial \mathcal{Q}^{\mathrm{ml}}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}})}{\partial \lambda_{smi}} = \frac{1}{2}\left\{\beta^{\mathtt{ml}}_{sm} + \mathrm{Tr}\left(\boldsymbol{S}_i \boldsymbol{\Sigma}_{sm}\right) - \tilde{y}^{\mathtt{ml}}_{smi}\right\} \tag{4.34}$$

and the projected statistics required to update the basis coefficients can be expanded using equation (4.5) such that

$$\begin{aligned}
\tilde{y}^{\mathtt{ml}}_{smi} &= \mathrm{Tr}\left(\boldsymbol{S}_i \boldsymbol{W}^{\mathtt{ml}}_{sm}\right) \\
&= \mathrm{Tr}\left(\boldsymbol{S}_i \frac{\sum_{t=1}^{T} \gamma_m(t)(\boldsymbol{o}_t - \boldsymbol{\mu}_{sm})(\boldsymbol{o}_t - \boldsymbol{\mu}_{sm})'}{\beta_m}\right) \\
&= \frac{\sum_{t=1}^{T} \gamma_m(t)(\boldsymbol{o}_t - \boldsymbol{\mu}_{sm})' \boldsymbol{S}_i (\boldsymbol{o}_t - \boldsymbol{\mu}_{sm})}{\beta_m} \\
&= \frac{\sum_{t=1}^{T} \gamma_m(t)(\boldsymbol{o}_t' \boldsymbol{S}_i \boldsymbol{o}_t - 2\boldsymbol{\mu}_{sm}' \boldsymbol{S}_i \boldsymbol{o}_t + \boldsymbol{\mu}_{sm}' \boldsymbol{S}_i \boldsymbol{\mu}_{sm})}{\beta_m}
\end{aligned} \tag{4.35}$$

Note that the terms $\boldsymbol{o}_t' \boldsymbol{S}_i \boldsymbol{o}_t$ and $\boldsymbol{S}_i \boldsymbol{o}_t$ have already been computed and cached when calculating the likelihood (see Section 3.5.2). Moreover, the term $\boldsymbol{\mu}_{sm}' \boldsymbol{S}_i \boldsymbol{\mu}_{sm})$ depends only on the model parameters and can therefore be pre-computed (independent of the amount of training data). The cost of computing the projected statistics is of the order $\mathcal{O}(d)$ for each basis due to the computation of the scalar product between $\boldsymbol{\mu}_{sm}$ and $\boldsymbol{S}_i \boldsymbol{o}_t$. The expression of the change in the component auxiliary function needed by the line search method in equation (4.18) can be rewritten as

$$\mathcal{Q}^{\mathrm{ml}}_{sm} - \hat{\mathcal{Q}}^{\mathrm{ml}}_{sm} = \frac{\beta^{\mathtt{ml}}}{2}\left\{\sum_{j=1}^{d} \log\left(1 + \Delta_{sm} w_{smj}\right) - \Delta_{sm} \sum_{i=1}^{n} \bar{f}_{smi} \tilde{y}^{\mathtt{ml}}_{sm}\right\} \tag{4.36}$$

where the terms in the equation hold the same meaning as before. This makes it tractable, in terms of memory requirement, to build systems with a large number of Gaussian components.

For STC and EMLLT models, where the basis matrices are of rank one, the *projected* statistics can be further simplified as

$$\tilde{y}^{\mathtt{ml}}_{smi} = \mathrm{Tr}(\boldsymbol{S}_i \boldsymbol{W}^{\mathtt{ml}}_{sm}) = \boldsymbol{a}_i \boldsymbol{W}^{\mathtt{ml}}_{sm} \boldsymbol{a}_i' \tag{4.37}$$

Thus the basis coefficient update for STC and EMLLT in equations (4.24) and (4.31) can be expressed in terms of the projected statistics, $\tilde{y}^{\mathtt{ml}}_{smi}$, as

$$\hat{\lambda}_{smi} = \frac{1}{\tilde{y}^{\mathtt{ml}}_{smi}} \tag{4.38}$$

$$\hat{\lambda}_{smi} = \lambda_{smi} + \left(\frac{1}{\tilde{y}^{\mathtt{ml}}_{smi}} - \frac{1}{\boldsymbol{a}_i \boldsymbol{\Sigma}_{sm} \boldsymbol{a}_i'}\right) \tag{4.39}$$

respectively. . Besides, note that

$$\begin{aligned} \tilde{y}_{smi}^{\mathtt{ml}} &= \boldsymbol{a}_i \boldsymbol{W}_{sm}^{\mathtt{ml}} \boldsymbol{a}_i' \\ &= \boldsymbol{a}_i \left(\frac{\sum_{t=1}^{T} \gamma_{sm}^{\mathrm{ml}}(t)(\boldsymbol{o}_t - \boldsymbol{\mu}_{sm})(\boldsymbol{o}_t - \boldsymbol{\mu}_{sm})'}{\beta^{\mathtt{ml}}} \right) \boldsymbol{a}_i' \\ &= \frac{\sum_{t=1}^{T} \gamma_{sm}^{\mathrm{ml}}(t)(\tilde{o}_{ti} - \tilde{\mu}_{smi})^2}{\beta^{\mathtt{ml}}} \end{aligned} \tag{4.40}$$

for $i = 1, 2, \ldots, n$. $\tilde{o}_{ti} = \boldsymbol{a}_i \boldsymbol{o}_t$ and $\tilde{\mu}_{smi} = \boldsymbol{a}_i \boldsymbol{\mu}_{sm}$ are the projected observation and mean vectors associated with the projection vector, $\boldsymbol{a}_i$. As discussed in Section 3.5.2, the values of $\tilde{o}_{ti}$ and $\tilde{\mu}_{smi}$ have already been pre-computed and cached for efficient likelihood computation. Thus, the overall cost of computing the projected statistics of component m in state s for STC and EMLLT is proportional to n. For STC, $n = d$ and the cost of computing the sufficient statistics is the same as the diagonal covariance matrix model.

4.3.2 Basis Initialisations

Section 3.5 described several forms of structured precision matrix approximation in terms of basis superposition. Initialisation is important as a good initial set of basis allows fast convergence and avoids hitting a poor local maximum during parameters estimation process. This is especially true for the EMLLT and SPAM models, where the update of basis vectors/matrices is slow. For STC, a trivial initialisation of setting the transformation matrix as an identity matrix may be used, which simply leads to a diagonal covariance matrix system. The initialisation schemes for EMLLT and SPAM will be discussed as follows.

4.3.2.1 Initialisation Schemes for EMLLT

As discussed in Section 4.2.2, there is no closed form solution to update the basis vectors. Standard optimisation packages such as Newton optimisation must be used to estimate them. In general, the use of the Newton optimisation method is computationally expensive and has slow convergence property if the auxiliary function is highly non-quadratic. An initial set of basis vectors is required in order to evaluate the gradient vector and Hessian matrix in equations (4.29) and (4.30). It is not possible to initialise the basis vectors to be zero vectors as doing so leads to an invalid stationary point[2]. The initial set of non-zero basis vectors has to contain at least d independent vectors to yield positive-definite precision matrices.

Several initialisation schemes based on stacking multiple transforms together will be introduced. The resulting EMLLT transform and the corresponding coefficient matrix can be ex-

[2]This is not a feasible solution as the resulting precision matrix will be zero

pressed as

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}^{(1)} \\ \boldsymbol{A}^{(2)} \end{bmatrix} \quad \text{and} \quad \boldsymbol{\Lambda}_{sm} = \begin{bmatrix} \boldsymbol{\Lambda}_{sm}^{(1)} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Lambda}_{sm}^{(2)} \end{bmatrix} \tag{4.41}$$

where $\boldsymbol{A}^{(1)}$ and $\boldsymbol{A}^{(2)}$ are $n_1 \times d$ and $n_2 \times d$ matrices and the corresponding coefficient matrices, $\boldsymbol{\Lambda}_{sm}^{(1)}$ and $\boldsymbol{\Lambda}_{sm}^{(2)}$ are $n_1 \times n_1$ and $n_2 \times n_2$ diagonal matrices respectively. Table 4.1 tabulates several

Initialisation Scheme	Descriptions	Initial model
Uninformative (UI)	$\boldsymbol{A}^{(1)}$: identity matrix $\boldsymbol{A}^{(2)}$: random (see Appendix B for an example) $\boldsymbol{\Lambda}_{sm}^{(1)}$: inverse diagonal covariance matrix $\boldsymbol{\Lambda}_{sm}^{(2)}$: zero matrix	DIAGC
STC+EYE	$\boldsymbol{A}^{(1)}$: global STC transform $\boldsymbol{A}^{(2)}$: identity matrix $\boldsymbol{\Lambda}_{sm}^{(1)}$: STC basis coefficient matrix $\boldsymbol{\Lambda}_{sm}^{(2)}$: zero matrix	STC
STC+HLDA	$\boldsymbol{A}^{(1)}$: global STC transform $\boldsymbol{A}^{(2)}$: HLDA projection matrix $\boldsymbol{\Lambda}_{sm}^{(1)}$: STC basis coefficient matrix $\boldsymbol{\Lambda}_{sm}^{(2)}$: zero matrix	STC
STC-SS	$\boldsymbol{A}^{(1)}$: STC transform for speech models $\boldsymbol{A}^{(2)}$: STC transform for silence models $\boldsymbol{\Lambda}_{sm}^{(1)}$: STC basis coefficient matrix (speech models) $\boldsymbol{\Lambda}_{sm}^{(2)}$: zero matrix (speech models)	STC (SS)

Table 4.1 *EMLLT initialisation schemes*

initialisation schemes. The second and third columns describe the properties of each initialisation scheme and the equivalent initial model. These initialisation schemes will be described below:

- **UI initialisation:**
  The UI method initialises the model to be a diagonal covariance matrix model. The first d basis vectors form an identity matrix with the corresponding coefficients set as the inverse diagonal variances. The remaining basis vectors may be randomly initialised to any non-zero vectors with the corresponding coefficients set to zero. This form of initialisation is less efficient because the set of basis vectors are suboptimal. As a result, the gradient descent method will eventually converge at a poor local maximum.

- **STC+EYE initialisation:**
  The STC basis vectors provide a better set of initial vectors. For the STC+EYE initialisation,

an identity matrix is appended to a global STC transform. The corresponding coefficients are set to be the STC coefficients and zeros for the STC and identity transforms respectively. This initialisation method yields an STC model.

- **STC+HLDA initialisation:**
  Alternatively, the identity matrix of the STC+EYE method may be replaced by a HLDA projection matrix, giving the STC+HLDA scheme. There is an added flexibility when appending HLDA matrix which allows the basis order to be adjusted by using projection matrix of different sizes. Similar to the STC+EYE method, this method yields an initial STC model.

- **STC+SS initialisation:**
  Finally, an EMLLT transform can be initialised by stacking multiple STC transforms together. These STC transforms may be trained with respect to different clusters of Gaussian components. For example, the STC-SS method uses the speech and silence as the two clusters of Gaussian components. This results in a multiple transforms STC model as the initial model.

4.3.2.2 Initialisation Schemes for SPAM

In order to update the basis coefficients for SPAM models, the set of matrices, $\boldsymbol{S}_i$, must be selected such that the initial precision matrices, $\boldsymbol{P}_{sm}$, are positive definite. In practice, this can be achieved by ensuring that at least one of the basis is positive definite. In [7], it was suggested that one of the basis matrices (the nth basis matrix is used here) is initialised as a positive definite symmetric matrix such that it is the weighted average precision matrix over all the Gaussian components.

$$\boldsymbol{S}_n = \frac{\sum_{s=1}^{S}\sum_{m=1}^{M} c_{sm}\boldsymbol{P}_{sm}}{\sum_{s=1}^{S}\sum_{m=1}^{M} c_{sm}} \tag{4.42}$$

Positive-definiteness can now be guaranteed by initialising $\lambda_{smn} = 1$ and the rest of the basis coefficients set to zero. The remaining basis matrices can be initialised to any arbitrary symmetric matrix. However, according to [7], it is useful to initialise the set of basis matrices, $\{\boldsymbol{S}_i\}$, as the symmetric matrices associated to the top $n-1$ singular vectors of the matrix

$$\boldsymbol{V} = \frac{\sum_{s=1}^{S}\sum_{m=1}^{M} c_{sm}\boldsymbol{v}_m\boldsymbol{v}_m'}{\sum_{s=1}^{S}\sum_{m=1}^{M} c_{sm}} \tag{4.43}$$

where $\boldsymbol{v}_m = \text{vec}\left([\boldsymbol{W}_{sm}^{\texttt{ml}}]^{-1}\right)$. On large systems, it was found that the basis matrix initialisation given by equation (4.43) is not robust because the full covariance statistics for each component may not be estimated reliably due to data sparseness problem. Instead, the state-level inverse covariance statistics is used to produce a more reliable set of basis matrices.

$$\boldsymbol{V} = \frac{\sum_{s=1=1}^{S} c_s\boldsymbol{v}_s\boldsymbol{v}_s'}{\sum_{s=1=1}^{S} c_s} \tag{4.44}$$

where

$$\boldsymbol{v}_s = \text{vec}\left(\left\{\sum_{m=1}^{M_s} c_{sm}\boldsymbol{W}_{sm}^{\texttt{ml}}\right\}^{-1}\right) \tag{4.45}$$

$$c_s = \sum_{m=1}^{M_s} c_{sm} \tag{4.46}$$

and S is the total number of states in the HMM system and M_s is the number of Gaussian component for state s. The update of the basis matrices is highly constrained by the positive-definiteness of the precision matrices. As a result, the convergence is slow and the gain from updating the basis matrices is small. In practice, the basis matrices are fixed as the initial values and only the basis coefficients are updated for subsequent training iterations. The exact update procedure for the basis matrices is described in [127]. Recently, an alternative way of initialising the bases is also described in [87] for the SCGMMs. As SPAM is a special case of SCGMMs (see Section 3.8.1), this method may also be used to initialise the SPAM model.

4.3.3 Variance Flooring

In situations of data sparseness, which is common in LVCSR systems, a variance floor required to prevent over-fitting. It imposes a lower bound to the variances (diagonal elements of the covariance matrix). The form of the variance floor used in HTK [134] may be expressed as

$$\bar{\sigma}_{smj}^2 = \max\left(\sigma_{smj}^2, \sigma_j^{(\texttt{vf})2}\right) \tag{4.47}$$

where σ_{smj}^2 and $\bar{\sigma}_{smj}^2$ are the estimated and floored variances respectively. $\sigma_j^{(\texttt{vf})2}$ denotes the variance floor. In HTK, the variance floor elements, $\sigma_j^{(\texttt{vf})2}$, are computed as a proportion of the weighted average within state variances. Hence,

$$\sigma_j^{(\texttt{vf})2} = \frac{\alpha \sum_{s=1}^{S} \beta_s \sigma_j^{(s)2}}{\sum_{s=1}^{S} \beta_s} \tag{4.48}$$

where $\sigma_j^{(s)2}$ and β_s are the within state variance and occupancy count respectively. α is a scaling factor which is typically set as 0.1 (10%). This is the approach adopted in this work.

STC models consists of a set of independent basis vectors. The resulting basis coefficients have to be positive to ensure valid precision matrix modelling. Recall that the basis coefficients is indeed the inverse variances in the projected space[3]. Naturally, variance flooring is readily applicable to the basis coefficients using the projected variance floor. Hence,

$$1/\hat{\lambda}_{smi} = \max\left(1/\lambda_{smi}, \sigma_j^{(\texttt{vf})2} a_{ii}^2\right) \tag{4.49}$$

[3]Strictly speaking, the basis coefficients are the inverse variance contributions. Because STC basis vectors are independent, these inverse variance contributions are also independent

where $\sigma_j^{(\mathtt{vf})2} a_{ii}^2$ is the ith variance floor element in the transformed space and a_{ii} is the ith diagonal element of $\boldsymbol{A}$.

Application of variance floor for EMLLT models is not so straightforward due to the existence of negative coefficients. However, instead of applying variance floor to the estimated parameters, variance floor can be applied to the underlying covariance statistics in the similar way as would have been done for the variance parameters. In the case where full covariance statistics, $\boldsymbol{W}_{sm}^{\mathtt{ml}}$, are accumulated, flooring is done as follows:

$$\bar{y}_{sm}^{\mathtt{ml}}(i,i) = \max\left(\tilde{y}_{sm}^{\mathtt{ml}}(i,i), \sigma_i^{(\mathtt{vf})2}\right) \tag{4.50}$$

where $\bar{y}_{sm}^{\mathtt{ml}}(i,i)$ and $\tilde{y}_{sm}^{\mathtt{ml}}(i,i)$ are the leading diagonal elements of the full covariance statistics, before and after flooring respectively. In other words, if full covariance statistics are accumulated, variance floor is applied to the leading diagonal elements. On contrary, for updating the basis coefficients alone, only the transformed statistics, $\boldsymbol{a}_i \boldsymbol{W}_{sm}^{\mathtt{ml}} \boldsymbol{a}_i'$, are accumulated. To apply flooring onto these transformed statistics, the variance floor elements need to be transformed, too. Hence, the equation governing variance flooring of transformed statistics is given by

$$\bar{y}_{smi}^{\mathtt{ml}} = \max\left(\boldsymbol{a}_i \boldsymbol{W}_{sm}^{\mathtt{ml}} \boldsymbol{a}_i, \sigma_i^{(\mathtt{vf})2} a_{ii}^2\right) \tag{4.51}$$

where $\bar{y}_{smi}^{\mathtt{ml}}$ is floored projected statistics associated with the projection vector $\boldsymbol{a}_i$.

Unfortunately, the use of full rank basis matrices for SPAM model does not allow trivial variance flooring when only the projected statistics, $\mathrm{Tr}\left(\boldsymbol{W}_{sm}^{\mathtt{ml}} \boldsymbol{S}_i\right)$, are accumulated. However, if one of the basis matrices is initialised to be positive-definite ($\boldsymbol{S}_n$) [7], the coefficient corresponding to $\boldsymbol{S}_1$ may be increased gradually until the final precision matrices satisfy the variance floor condition. Instead of having a lower bound (floor) on the leading diagonal elements of the covariance matrices, an equivalent upper bound may be applied to the leading diagonal elements of the precision matrices. Let $p_{ii}^{(ub)} = 1/\left(\sigma_i^{(\mathtt{vf})2} a_{ii}^2\right)$ be the upper bound for the diagonal elements of the precision matrices. The following satisfies the equivalent variance flooring condition:

$$\bar{p}_{ii} = \max\left(p_{ii}, p_{ii}^{(ub)}\right) \tag{4.52}$$

where $\bar{p}_{ii}$ and p_{ii} are the leading diagonal elements of the precision matrix before and after 'ceiling' respectively. Since the precision matrix expression for the SPAM model is given by

$$\boldsymbol{P}_{sm} = \sum_{i=1}^{n-1} \lambda_{smi} \boldsymbol{S}_i + \lambda_{smn} \boldsymbol{S}_n \tag{4.53}$$

variance floor condition may be achieved by finding the minimum value of λ_{smn} that satisfies

$$\bar{p}_{ii} \geq p_{ii}^{(ub)} \tag{4.54}$$

for all i. However, this approach is computationally inefficient. In practice, the parameters of the SPAM model for a particular Gaussian component will not be updated if the corresponding occupancy count, $\beta_{sm}^{\mathtt{ml}}$, falls below a certain threshold. This threshold helps to ensure that the resulting parameter estimates are robust.

5

Discriminative Training of Precision Matrix Models

The conventional way of estimating model parameters using the Maximum Likelihood (ML) criterion relies greatly on the correctness of the model assumptions. Despite the extensive use of HMMs in speech recognition, a series of assumptions are made about the speech data, many of which are known to be poor. In Section 2.3, some of these shortcomings have been discussed. To account for the poor model assumptions, the model parameters are more appropriately learned by minimising the *Bayesian risk*. This approach has motivated an alternative training paradigm known as *discriminative training* (see Section 2.2.2). This chapter will describe Minimum Phone Error (MPE) training as a *Minimum Bayes Risk* (MBR) [63] estimation and employ this training criterion to estimate the parameters of precision matrix models described in Chapter 3.

5.1 The Minimum Phone Error (MPE) Criterion

From the Bayesian viewpoint, model parameters are considered as random variables. *Bayesian risk* is then defined as an expected loss of an estimator,

$$\mathcal{R}^{\text{mbr}}(\boldsymbol{\theta}) = \sum_{u=1}^{U} \sum_{h \in H_u} p(h|\mathcal{O}_1^T, \boldsymbol{\theta}) l(h, \hat{h}) \tag{5.1}$$

where $p(h|\mathcal{O}_1^T, \boldsymbol{\theta})$ denotes the posterior probability of a hypothesis, h, for sentence s, given the observation sequence, $\mathcal{O}_1^T$ and the model parameters, $\boldsymbol{\theta}$. $l(h, \hat{h})$ is the *loss function* of h given the reference, $\hat{h}$. The summation in equation (5.1) is defined over all possible hypotheses, H_u, for each of the training sentences, $s = 1, 2, \ldots, S$, where S denotes the total number of training sentences. Therefore, a MBR estimator (or simply Bayesian estimator) finds the new model parameters, $\hat{\boldsymbol{\theta}}^{\text{mbr}}$, such that the Bayesian risk in equation (5.1) is minimised, *i.e.*

$$\hat{\boldsymbol{\theta}}^{\text{mbr}} = \arg\min_{\boldsymbol{\theta}} \mathcal{R}^{\text{mbr}}(\boldsymbol{\theta}) \tag{5.2}$$

Many forms of loss function, $l(h, \hat{h})$, may be defined, each one yields a different discriminative training criterion. In this work, the Minimum Phone Error (MPE) criterion will be described. The loss function for MPE is measured by the *raw phone errors* between the hypothesis and the reference at *sentence level*, *i.e.*

$$l(h, \hat{h}) = \text{PhoneError}(h, \hat{h}) = N - \text{PhoneAcc}(h, \hat{h}) \tag{5.3}$$

where $\text{PhoneError}(h, \hat{h})$ and $\text{PhoneAcc}(h, \hat{h})$ measure the number of incorrect and correct phones respectively in h given the reference, $\hat{h}$. N denotes the total number of phones in $\hat{h}$ plus the number of insertions in h. From the above equation, minimising the expected loss function (phone error) is equivalent to maximising the expected phone accuracy. Therefore, the MPE objective function to be *maximised* may be expressed as

$$\mathcal{R}^{\text{mpe}}(\boldsymbol{\theta}) = \sum_{u=1}^{U} \sum_{h \in H_u} p(h|\mathcal{O}_1^T, \boldsymbol{\theta}) \text{PhoneAcc}(h, \hat{h}) \tag{5.4}$$

where the posterior probability of h given the observation sequence and the model parameters is given by

$$p(h|\mathcal{O}_1^T, \boldsymbol{\theta}) = \frac{p(\mathcal{O}_1^T|h, \boldsymbol{\theta})P(h)}{\sum_{g=H_u} p(\mathcal{O}_1^T|g, \boldsymbol{\theta})P(g)} \tag{5.5}$$

$p(\mathcal{O}_1^T|h, \boldsymbol{\theta})$ is the likelihood of h given the observation sequence and the model set and $P(h)$ is the language model probability of hypothesis h. In practice, the posterior probability, $p(h|\mathcal{O}_1^T, \boldsymbol{\theta})$, is small for many h. Therefore, H_u, is usually approximated by a smaller set of competing sentences which may be obtained by performing a recognition on the training utterance using the current model, $\boldsymbol{\theta}$, and a simple language model[1]. The resulting hypotheses may be stored as an N-best list or in a lattice structure. The latter is preferred due to its compact representation. Ideally, $\text{PhoneAcc}(h_u, \hat{h}_u)$ equals the number of correct phones minus the number insertions[2] of hypothesis h_u compared to the reference sentence, $\hat{h}_u$. In the case where competing sentences are represented in a lattice format, $\text{PhoneAcc}(h_u, \hat{h}_u)$ is calculated as the sum of the phone accuracies, $\text{PhoneAcc}(q)$, over all phone arcs, q in the lattice. The exact calculation of the phone accuracies requires alignment of the hypothesis and reference phone sequence, and hence is computationally expensive. An approximation was proposed in [95] such that

$$\text{PhoneAcc}(q) = \max_z \left\{ \begin{array}{ll} -1 + 2e(q, z) & \text{if } z = q \\ -1 + e(q, z) & \text{otherwise} \end{array} \right\} \tag{5.6}$$

where $e(q, z)$ is the proportion of the duration of z overlapped by q. Equation (5.6) has a range between -1 and 1. With perfect alignment, $\text{PhoneAcc}(q)$ gives 1, 0 and -1 for a correct phone, substitution/deletion and insertion respectively. This approximation is adopted in this work.

[1] A unigram or bigram language model is typically used to improve generalisation [95]

[2] Deletions and substitutions are accounted for in the number of correct phones.

5.2 Optimising the MPE Criterion

The MPE objective function in equation (5.4) is difficult to optimise directly. In the ML case, an auxiliary function (Q-function) exists as the *strict* lower bound to the objective function such that an increase in the auxiliary function guarantees to increase the objective function, or

$$\mathcal{R}^{\mathrm{ml}}(\hat{\boldsymbol{\theta}}) - \mathcal{R}^{\mathrm{ml}}(\boldsymbol{\theta}) \geq \mathcal{Q}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) - \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}) \tag{5.7}$$

where $\boldsymbol{\theta}$ and $\hat{\boldsymbol{\theta}}$ are the current and updated model parameter set respectively. The above inequality is the result of the well-known Baum-Eagon inequality [11]. However, the Baum-Eagon inequality is not applicable to the discriminative objective functions which are rational functions (a ratio between the *numerator* and *denominator* polynomial functions with positive coefficients). Earlier work on MMI discriminative training are based on direct optimisation of the objective function using hill climbing methods. In 1989, *Gopalakrishnan et al.* introduced the Extended Baum-Welch algorithm [50, 51] which extended the Baum-Eagon inequality to rational functions. This allows discriminative training to be carried out in an efficient iterative manner similar to the Baum-Welch algorithm for ML training. In [95, 100, 101], *Povey et al.* introduced an alternative approach of deriving the MPE update formulae for the Gaussian parameters of HMMs using the so called *weak-sense* auxiliary function. A weak-sense auxiliary function, $\mathcal{Q}^{\mathrm{mpe}}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}})$, is defined to have the same derivative with respect to the model parameters as that of the objective function, when evaluated at the current estimates of the model parameters, *i.e.*

$$\left.\frac{\partial \mathcal{Q}^{\mathrm{mpe}}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}})}{\partial \hat{\boldsymbol{\theta}}}\right|_{\hat{\boldsymbol{\theta}}=\boldsymbol{\theta}} = \left.\frac{\partial \mathcal{R}^{\mathrm{mpe}}(\hat{\boldsymbol{\theta}})}{\partial \hat{\boldsymbol{\theta}}}\right|_{\hat{\boldsymbol{\theta}}=\boldsymbol{\theta}} \tag{5.8}$$

The form of auxiliary function suggested in [100] takes the following form

$$\mathcal{Q}^{\mathrm{mpe}}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) = \mathcal{Q}^{\mathrm{n}}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) - \mathcal{Q}^{\mathrm{d}}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) + \mathcal{R}^{\mathrm{s}}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) \tag{5.9}$$

where $\mathcal{Q}^{\mathrm{n}}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}})$ and $\mathcal{Q}^{\mathrm{d}}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}})$ are the numerator and denominator terms respectively, which take the same form as the ML auxiliary function in equation (2.26), except that the sufficient statistics are now derived based on the numerator and denominator 'counts' for state s and component m, $\gamma^{\mathrm{n}}_{sm}(t)$ and $\gamma^{\mathrm{d}}_{sm}(t)$ respectively (cf. equations (4.4) to (4.6)). The next section will describe how the values of $\gamma^{\mathrm{n}}_{sm}(t)$ and $\gamma^{\mathrm{d}}_{sm}(t)$ are computed. $\mathcal{R}^{\mathrm{s}}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}})$ is a smoothing function which, as suggested in [100], takes the form

$$\mathcal{R}^{\mathrm{s}}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) = K + \frac{1}{2}\sum_{s=1}^{S}\sum_{m=1}^{M} D_{sm}\Big\{\log|\boldsymbol{P}_{sm}| - \mathrm{Tr}(\boldsymbol{P}_{sm}\boldsymbol{\Sigma}_{sm})\Big\} \tag{5.10}$$

where $\boldsymbol{\Sigma}_{sm}$ is the current estimate of the full covariance matrix and D_{sm} is a component-dependent constant that controls the amount of $\boldsymbol{\Sigma}_{sm}$ to be smoothed onto the covariance statistics. The smoothing function has a zero differential with respect to the model parameters evaluated at the current parameter set so that the condition in equation (5.8) is satisfied.

The weak-sense auxiliary function in equation (5.9) is not a *strict* lower bound of the MPE objective function in equation (5.4). Therefore, maximising the weak-sense auxiliary function *does not* guarantee an improvement in the objective function. However, due to the condition in equation (5.8), optimising equation (5.9) will update the model parameters in the direction along the gradient of the objective function at the current model parameters. Hence, weak-sense auxiliary function optimisation may be viewed as a form of *gradient-based* optimisation scheme. The learning rate may be controlled by adjusting the smoothing constant, D_{sm}. A larger D_{sm} value yields a lower learning rate. It is shown later in Section 5.3.1 that a lower bound is applied to D_{sm} such that the resulting covariance estimate is positive definite and the effective component occupancy count $(\beta^{\mathtt{n}}_{sm} - \beta^{\mathtt{d}}_{sm} + D_{sm})$ is positive.

5.3 Discriminative Training of Precision Matrix Models

Due to the success of discriminative training schemes in speech recognition, several forms of MMI trained precision matrix models have been published recently. *Goel et al.* , 2003 [46] presented the MMI estimation of the SPAM models with a small vocabulary system. MMI estimation of SCGMMs and EMLLT models was also studied in [4]. *McDonough et al.* [77] employed MMI trained STC models in speaker-adapted training (SAT). *Tsakalidis et al.* [121] introduced Discriminative Likelihood Linear Transform (DLLT), a method that discriminatively train the MLLT parameters based on the MMI criterion. The consistent improvement of MPE training on large scale diagonal covariance matrix systems compared to the MMI discriminative criterion [100] motivates the investigation of MPE training of precision matrix models on LVCSR systems. The rest of this chapter is devoted to the discussion of MPE training of various precision matrix models described in Chapter 3 [113].

5.3.1 MPE Estimation of Precision Matrix Model Parameters

This section presents the derivation of the MPE estimation formulae for basis superposition precision matrix models given by equation 3.18. The estimation formulae will be derived by maximising the weak-sense auxiliary function given in equation (5.9). Recall from Chapter 4 that the ML auxiliary function to be maximised in ML training is given by (from equation 4.9)

$$\mathcal{Q}^{\mathrm{ml}}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) = K + \frac{1}{2} \sum_{s=1}^{S} \sum_{m=1}^{M} \left\{ \beta^{\mathtt{ml}}_{sm} \log \left| \sum_{i=1}^{n} \lambda_{smi} \boldsymbol{S}_i \right| - \mathrm{Tr} \left[\left(\sum_{i=1}^{n} \lambda_{smi} \boldsymbol{S}_i \right) \boldsymbol{W}^{\mathtt{ml}}_{sm} \right] \right\} \tag{5.11}$$

The sufficient statistics, $\boldsymbol{\Theta}_{sm}^{\mathtt{ml}} = \{\beta_{sm}^{\mathtt{ml}}, \boldsymbol{x}_{sm}^{\mathtt{ml}}, \boldsymbol{Y}_{sm}^{\mathtt{ml}}\}$, are given by

$$\beta_{sm}^{\mathtt{ml}} = \sum_{t=1}^{T} \gamma_{sm}^{\mathrm{ml}}(t) \tag{5.12}$$

$$\boldsymbol{x}_{sm}^{\mathtt{ml}} = \sum_{t=1}^{T} \gamma_{sm}^{\mathrm{ml}}(t) \boldsymbol{o}_t \tag{5.13}$$

$$\boldsymbol{Y}_{sm}^{\mathtt{ml}} = \sum_{t=1}^{T} \gamma_{sm}^{\mathrm{ml}}(t) \boldsymbol{o}_t \boldsymbol{o}_t' \tag{5.14}$$

and the full covariance statistics can be expressed in terms of the above zeroth, first and second moment statistics as

$$\boldsymbol{W}_{sm}^{\mathtt{ml}} = \frac{\left(\boldsymbol{Y}_{sm}^{\mathtt{ml}} - \boldsymbol{x}_{sm}^{\mathtt{ml}} \boldsymbol{\mu}_{sm} - \boldsymbol{\mu}_{sm} \boldsymbol{x}_{sm}^{\mathtt{ml}\,\prime} + \beta_{sm}^{\mathtt{ml}} \boldsymbol{\mu}_{sm} \boldsymbol{\mu}_{sm}'\right)}{\beta_{sm}^{\mathtt{ml}}} \tag{5.15}$$

Given the set of parameters, $\boldsymbol{\theta}$, the above ML auxiliary function can be rewritten in terms of the ML statistics $\boldsymbol{\Theta}_{sm}^{\mathtt{ml}}$ as

$$\mathcal{Q}^{\mathrm{ml}}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) = \mathcal{G}(\boldsymbol{\Theta}_{sm}^{\mathtt{ml}}) \tag{5.16}$$

where

$$\mathcal{G}(\boldsymbol{\Theta}_{sm}^{\mathtt{ml}}) = K + \frac{1}{2} \sum_{s=1}^{S} \sum_{m=1}^{M} \beta_{sm}^{\mathtt{ml}} \left\{ \log |\boldsymbol{P}_{sm}| - \mathrm{Tr}\left(\boldsymbol{P}_{sm} \boldsymbol{W}_{sm}^{\mathtt{ml}}\right) \right\} \tag{5.17}$$

In a similar way, the weak-sense auxiliary function in equation (5.9) can also be expressed in terms of sufficient statistics

$$\mathcal{Q}^{\mathrm{mpe}}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) = \mathcal{G}(\boldsymbol{\Theta}_{sm}^{\mathtt{n}}) - \mathcal{G}(\boldsymbol{\Theta}_{sm}^{\mathtt{d}}) + \mathcal{G}(\boldsymbol{\Theta}_{sm}^{\mathtt{s}}) \tag{5.18}$$

using the function $\mathcal{G}(.)$ defined in equation (5.17). $\boldsymbol{\Theta}^{\mathtt{n}}$ and $\boldsymbol{\Theta}^{\mathtt{d}}$ denote the sufficient statistics for numerator $(\{\beta_{sm}^{\mathtt{n}}, \boldsymbol{x}_{sm}^{\mathtt{n}}, \boldsymbol{Y}_{sm}^{\mathtt{n}}\})$ and denominator $\left(\{\beta_{sm}^{\mathtt{d}}, \boldsymbol{x}_{sm}^{\mathtt{d}}, \boldsymbol{Y}_{sm}^{\mathtt{d}}\}\right)$ respectively. To compute these sufficient statistics, $\gamma_{sm}^{\mathrm{n}}(t)$ and $\gamma_{sm}^{\mathrm{d}}(t)$ are required. One can think of these quantities as the posterior of component m in state s at time t for the numerator and denominator statistics respectively. According to [95], they are calculated from the numerator and denominator phone lattices[3] as

$$\gamma_{sm}^{\mathrm{n}}(t) = \sum_{q: s_q \leq t \leq e_q} \gamma_{qsm}(t) \max\left(0, \gamma_q^{\mathrm{mpe}}(t)\right) \tag{5.19}$$

$$\gamma_{sm}^{\mathrm{d}}(t) = \sum_{q: s_q \leq t \leq e_q} \gamma_{qsm}(t) \max\left(0, -\gamma_q^{\mathrm{mpe}}(t)\right) \tag{5.20}$$

where s_q and e_q are the start and end times of arc q and $\gamma_{qsm}(t)$ are the occupation probabilities for the Gaussian conditional on arc q which are computed by performing a forward backward over the arc q. $\gamma_q^{\mathrm{mpe}}(t)$ is defined as

$$\gamma_q^{\mathrm{mpe}}(t) = \frac{\partial \mathcal{R}^{\mathrm{mpe}}}{\partial \log p(\mathcal{O}_1^T | q)} \tag{5.21}$$

[3] Each arc in the phone lattice contains the identity of the phone as well as its start and end times

This is the differential of the objective function with respect to the log likelihood of arc q ($\log p(\mathcal{O}_1^T|q)$).

The set of parameters, $\boldsymbol{\Theta}_{sm}^{\mathtt{s}}$, which corresponds to the smoothing function (5.10), $\mathcal{R}^{\mathtt{s}}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) = \mathcal{G}(\boldsymbol{\Theta}_{sm}^{\mathtt{s}})$, is given by

$$\begin{aligned} \beta_{sm}^{\mathtt{s}} &= D_{sm} && (5.22)\\ \boldsymbol{x}_{sm}^{\mathtt{s}} &= D_{sm}\boldsymbol{\mu}_{sm} && (5.23)\\ \boldsymbol{Y}_{sm}^{\mathtt{s}} &= D_{sm}(\boldsymbol{\Sigma}_{sm} + \boldsymbol{\mu}_{sm}\boldsymbol{\mu}_{sm}') && (5.24) \end{aligned}$$

Note that the set of parameters associated with the smoothing function, $\boldsymbol{\Theta}_{sm}^{\mathtt{s}} = \{\beta_{sm}^{\mathtt{s}}, \boldsymbol{x}_{sm}^{\mathtt{s}}, \boldsymbol{Y}_{sm}^{\mathtt{s}}\}$, is only dependent on the current model parameters, $\boldsymbol{\theta}$ and is independent of the training data observations. The final form of equation 5.18 can also be conveniently expressed as a function $\mathcal{G}(.)$

$$\mathcal{Q}^{\mathrm{mpe}}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) = \mathcal{G}(\boldsymbol{\Theta}_{sm}^{\mathtt{n}}) - \mathcal{G}(\boldsymbol{\Theta}_{sm}^{\mathtt{d}}) + \mathcal{G}(\boldsymbol{\Theta}_{sm}^{\mathtt{s}}) = \mathcal{G}(\boldsymbol{\Theta}_{sm}^{\mathtt{mpe}}) \quad (5.25)$$

in terms of the sufficient statistics collectively as $\boldsymbol{\Theta}_{sm}^{\mathtt{mpe}}$, where

$$\begin{aligned} \beta_{sm}^{\mathtt{mpe}} &= \beta_{sm}^{\mathtt{n}} - \beta_{sm}^{\mathtt{d}} + \beta_{sm}^{\mathtt{s}} && (5.26)\\ \boldsymbol{x}_{sm}^{\mathtt{mpe}} &= \boldsymbol{x}_{sm}^{\mathtt{n}} - \boldsymbol{x}_{sm}^{\mathtt{d}} + \boldsymbol{x}_{sm}^{\mathtt{s}} && (5.27)\\ \boldsymbol{Y}_{sm}^{\mathtt{mpe}} &= \boldsymbol{Y}_{sm}^{\mathtt{n}} - \boldsymbol{Y}_{sm}^{\mathtt{d}} + \boldsymbol{Y}_{sm}^{\mathtt{s}} && (5.28)\\ \boldsymbol{W}_{sm}^{\mathtt{mpe}} &= \boldsymbol{W}_{sm}^{\mathtt{n}} - \boldsymbol{W}_{sm}^{\mathtt{d}} + \boldsymbol{W}_{sm}^{\mathtt{s}} && (5.29) \end{aligned}$$

Maximising the weak-sense auxiliary function with respect to the mean vector and the full covariance matrix parameters yields the MPE update formulae for the standard HMM parameters as

$$\hat{\boldsymbol{\mu}}_{sm}^{\mathtt{mpe}} = \frac{\boldsymbol{x}_{sm}^{\mathtt{mpe}}}{\beta_{sm}^{\mathtt{mpe}}} \quad \text{and} \quad \hat{\boldsymbol{\Sigma}}_{sm}^{\mathtt{mpe}} = \frac{\boldsymbol{W}_{sm}^{\mathtt{mpe}}}{\beta_{sm}^{\mathtt{mpe}}} = \frac{\boldsymbol{Y}_{sm}^{\mathtt{mpe}}}{\beta_{sm}^{\mathtt{mpe}}} - \boldsymbol{\mu}_{sm}\boldsymbol{\mu}_{sm}' \quad (5.30)$$

Note the similarities between equations (5.30) and (4.3). In fact, the MPE update formulae for various precision matrix models are the same as those described in Chapter 4 for ML training, with the ML sufficient statistics replace by the corresponding MPE statistics given by equations (5.26) to (5.29). In the following, the MPE update of the basis coefficients for STC, EMLLT and SPAM will be given.

To obtain the MPE update formula for the STC and EMLLT basis coefficients, the ML statistics ($\boldsymbol{W}_{sm}^{\mathtt{ml}}$) in equation (4.31) is replaces by the MPE statistics ($\boldsymbol{W}_{sm}^{\mathtt{mpe}}$). This yields

$$\hat{\lambda}_{smi} = \lambda_{smi} + \left(\frac{1}{\boldsymbol{a}_i \boldsymbol{W}_{sm}^{\mathtt{mpe}} \boldsymbol{a}_i'} - \frac{1}{\boldsymbol{a}_i \boldsymbol{\Sigma}_{sm} \boldsymbol{a}_i'} \right) \quad (5.31)$$

where λ_{smi} and $\boldsymbol{\Sigma}_{sm}$ are the current estimates of the basis coefficient and full covariance matrix respectively.

The MPE update for the SPAM basis coefficients may be obtained from equation (4.18) in a similar way to the STC/EMLLT case above. The resulting auxiliary function to be maximised becomes

$$\mathcal{Q}^{\text{sm}} - \hat{\mathcal{Q}}^{\text{sm}} = \frac{\beta_{sm}^{\texttt{mpe}}}{2} \left\{ \sum_{j=1}^{d} \log\left(1 + \Delta_{sm} w_{smj}\right) - \Delta_{sm} \sum_{i=1}^{n} d_{smi} \operatorname{Tr}(\boldsymbol{W}_{sm}^{\texttt{mpe}} \boldsymbol{S}_i) \right\} \tag{5.32}$$

where Δ_{sm}, w_{smj} and $\bar{\boldsymbol{f}}_{sm}(sm)$ have the same definition as those in equation (4.18).

5.3.2 Determination of the Smoothing Constant, D_{sm}

As previously mentioned, the smoothing constant D_{sm} may be adjusted to control the learning rate. However, a more important role of D_{sm} is to ensure that:

1. the new precision matrix estimate, $\hat{\boldsymbol{\Sigma}}_{sm}$, is positive definite;
2. the effective occupancy count, $\beta_{sm}^{\texttt{mpe}}$, is positive.

To find the minimum value of D_{sm} required to satisfy the first condition above, the new estimate covariance matrix estimate is rewritten in terms of D_{sm}, by combining equations (5.30) to yield

$$\hat{\boldsymbol{\Sigma}}_{sm}^{\texttt{mpe}} = \frac{\boldsymbol{B}_2 D_{sm}^2 + \boldsymbol{B}_1 D_{sm} + \boldsymbol{B}_0}{\beta_{sm}^{\texttt{c}} + D_{sm}} \tag{5.33}$$

where

$$\boldsymbol{B}_2 = \boldsymbol{\Sigma}_{sm} \tag{5.34}$$

$$\boldsymbol{B}_1 = \boldsymbol{Y}_{sm}^{\texttt{c}} + \beta_{sm}^{\texttt{c}} \left(\boldsymbol{\Sigma}_{sm} + \boldsymbol{\mu}_{sm}\boldsymbol{\mu}_{sm}'\right) - \left(\boldsymbol{\mu}_{sm}\boldsymbol{x}_{sm}^{\texttt{c}\prime} + \boldsymbol{x}_{sm}^{\texttt{c}}\boldsymbol{\mu}_{sm}'\right) \tag{5.35}$$

$$\boldsymbol{B}_0 = \beta_{sm}^{\texttt{c}} \boldsymbol{Y}_{sm}^{\texttt{c}} - \boldsymbol{x}_{sm}^{\texttt{c}} \boldsymbol{x}_{sm}^{\texttt{c}\prime} \tag{5.36}$$

and the *combined* statistics of the numerator and denominator statistics are given by

$$\beta_{sm}^{\texttt{c}} = \beta_{sm}^{\texttt{n}} - \beta_{sm}^{\texttt{d}} \tag{5.37}$$

$$\boldsymbol{x}_{sm}^{\texttt{c}} = \boldsymbol{x}_{sm}^{\texttt{n}} - \boldsymbol{x}_{sm}^{\texttt{d}} \tag{5.38}$$

$$\boldsymbol{Y}_{sm}^{\texttt{c}} = \boldsymbol{Y}_{sm}^{\texttt{n}} - \boldsymbol{Y}_{sm}^{\texttt{d}} \tag{5.39}$$

The smallest D_{sm} value required to yield a positive-definite $\hat{\boldsymbol{\Sigma}}_{sm}^{\texttt{mpe}}$ is given by the largest positive eigenvalues of the Quadratic Eigenvalue Problem (QEP) of equation (5.33) [46]. Let this be denoted by $D_{sm}^{\texttt{qep}}$. Furthermore, another lower bound, $E\beta_{sm}^{\texttt{d}}$, is applied to the smoothing constant to ensure that $\beta_{sm}^{\texttt{mpe}}$ is positive. Therefore, the actual smoothing constant value, D_m, is calculated as

$$D_{sm} = \max\left(2D_{sm}^{\texttt{qep}}, E\beta_{sm}^{\texttt{d}}\right) \tag{5.40}$$

$E = 2$ was empirically found to lead to good test-set performance [100].

If one is interested only in updating the basis coefficients, solving the above QEP is unnecessary. Imposing the positivity constraint on the projected statistics, $\boldsymbol{a}_i \boldsymbol{W}_{sm}^{\texttt{mpe}} \boldsymbol{a}_i'$, is sufficient to yield positive-definite precision matrices for STC and EMLLT models provided that the initial precision matrices are positive-definite. This is clearly evident given that equation (4.31) only depends on the projected statistics. Furthermore, a proof for this is also given in [85] for EMLLT models. The projected statistics can be expressed in terms of $\beta_{sm}^{\texttt{c}}$, $\boldsymbol{x}_{sm}^{\texttt{c}}$ and $\boldsymbol{W}_{sm}^{\texttt{c}}$ in the following quadratic form

$$\boldsymbol{a}_i \boldsymbol{W}_{sm}^{\texttt{mpe}} \boldsymbol{a}_i' = \frac{b_2^{(i)} D_{sm}^2 + b_1^{(i)} D_{sm} + b_0^{(i)}}{\beta_{sm}^{\texttt{c}} + D_{sm}} \tag{5.41}$$

where

$$b_2^{(i)} = \boldsymbol{a}_i \boldsymbol{\Sigma}_{sm} \boldsymbol{a}_i' \tag{5.42}$$

$$b_1^{(i)} = \boldsymbol{a}_i (\boldsymbol{W}_{sm}^{\texttt{c}} + \beta_{sm}^{\texttt{c}} \boldsymbol{\Sigma}_{sm}) \boldsymbol{a}_i' + \beta_{sm}^{\texttt{c}} \left(\boldsymbol{a}_i \boldsymbol{\mu}_{sm}\right)^2 - 2 \left(\boldsymbol{a}_i \boldsymbol{\mu}_{sm}\right) \left(\boldsymbol{a}_i \boldsymbol{x}_{sm}^{\texttt{c}}\right) \tag{5.43}$$

$$b_0^{(i)} = \beta_{sm}^{\texttt{c}} \boldsymbol{a}_i \boldsymbol{W}_{sm}^{\texttt{c}} \boldsymbol{a}_i' - \left(\boldsymbol{a}_i \boldsymbol{x}_{sm}^{\texttt{c}}\right)^2 \tag{5.44}$$

Thus, the value of D_{sm} can be determined by finding the largest positive root of the quadratic equation (5.41) rather than solving a QEP of equation (5.33).

Unlike STC and EMLLT models where the basis matrices are rank-1, the 'projected' statistics, $\mathrm{Tr}(\boldsymbol{W}_{sm}^{\texttt{mpe}} \boldsymbol{S}_i)$, associated with the basis matrices, $\boldsymbol{S}_i$ of the SPAM model can not be used to infer the positive-definiteness of the resulting precision matrices. Instead of obtaining the exact smoothing constant value by solving the QEP for equation (5.33), this value can be approximated by using a *pseudo* transformation matrix, $\boldsymbol{A}^*$. The transformed space is assumed to have negligible correlation such that the QEP is once again simplified to n independent quadratic equations as for the STC and EMLLT models. Thus, two sets of statistics are required: one for determining the smoothing constant, D_{sm} $\left(\boldsymbol{a}_i^* \boldsymbol{W}_{sm}^{\texttt{mpe}} \boldsymbol{a}_i^{*'}\right)$ and the other one for estimating the model parameters, $\mathrm{Tr}(\boldsymbol{W}_{sm}^{\texttt{mpe}} \boldsymbol{S}_i)$. To obtain a good approximation for the smoothing constant, $\boldsymbol{A}^*$ should be chosen such that the transformed space is as uncorrelated as possible. It is reasonable to select the STC transform as the *pseudo* transformation matrix. In the case where STC transform is unavailable, an identity matrix may be used. This was found to be a good approximation [111] and is used in this work.

5.4 I-smoothing and Maximum a Posteriori (MAP)

I-smoothing is an interpolation technique proposed by *Povey et al.* [100] that incorporates prior information over the Gaussian parameters to control the convergence of the MPE training process. Incorporation of I-smoothing is achieved by augmenting the weak-sense auxiliary function

in equation (5.9) with an I-smoothing function as follows:

$$\mathcal{Q}^{\text{mpe}}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) = \mathcal{Q}^{\text{n}}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) - \mathcal{Q}^{\text{d}}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) + \mathcal{R}^{\text{s}}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) + \log\left(p(\boldsymbol{\theta})\right) \tag{5.45}$$

where the I-smoothing function is given by

$$\log\left(p(\boldsymbol{\theta})\right) = K + \frac{\tau_I^{\texttt{prior}}}{2} \sum_{s=1}^{S} \sum_{m=1}^{M} \left\{ \log|\boldsymbol{P}_{sm}| - \text{Tr}\left(\boldsymbol{P}_{sm} \boldsymbol{\Sigma}_{sm}^{\texttt{prior}}\right) \right\} \tag{5.46}$$

where $\boldsymbol{\mu}_{sm}^{\texttt{prior}}$ and $\boldsymbol{\Sigma}_{sm}^{\texttt{prior}}$ denote the prior mean vector and covariance matrix respectively and $\tau_I^{\texttt{prior}}$ is the I-smoothing constant. The second order statistics associated with the prior is given by

$$\boldsymbol{Y}_{sm}^{\texttt{prior}} = \boldsymbol{\Sigma}_{sm}^{\texttt{prior}} + \boldsymbol{\mu}_{sm}^{\texttt{prior}} \boldsymbol{\mu}_{sm} + \boldsymbol{\mu}_{sm} \boldsymbol{\mu}_{sm}^{\texttt{prior}'} - \boldsymbol{\mu}_{sm} \boldsymbol{\mu}_{sm}' \tag{5.47}$$

The smoothing function has the same differential as the objective function at the current model parameters so equation (5.45) remains as a weak-sense auxiliary function. The prior can be regarded as the log likelihood of $\tau_I^{\texttt{prior}}$ data points with the mean, $\boldsymbol{\mu}_{sm}^{\texttt{prior}}$, and covariance matrix, $\boldsymbol{\Sigma}_{sm}^{\texttt{prior}}$, of the prior. Incorporating I-smoothing is easy by rewriting the MPE statistics in equations (5.26) to (5.29) as

$$\boldsymbol{x}_{sm}^{\texttt{mpe}} = \boldsymbol{x}_{sm}^{\texttt{n}} - \boldsymbol{x}_{sm}^{\texttt{d}} + \boldsymbol{x}_{sm}^{\texttt{s}} + \tau_I^{\texttt{prior}} \boldsymbol{\mu}_{sm}^{\texttt{prior}} \tag{5.48}$$

$$\boldsymbol{Y}_{sm}^{\texttt{mpe}} = \boldsymbol{Y}_{sm}^{\texttt{n}} - \boldsymbol{Y}_{sm}^{\texttt{d}} + \boldsymbol{Y}_{sm}^{\texttt{s}} + \tau_I^{\texttt{prior}} \boldsymbol{Y}_{sm}^{\texttt{prior}} \tag{5.49}$$

$$\beta_{sm}^{\texttt{mpe}} = \beta_{sm}^{\texttt{n}} - \beta_{sm}^{\texttt{d}} + \beta_{sm}^{\texttt{s}} + \tau_I^{\texttt{prior}} \tag{5.50}$$

It is simple to see that as the I-smoothing constant, $\tau_I^{\texttt{prior}}$, tends to infinity, the resulting estimation formulae tend to those of the prior estimates. The prior information used for I-smoothing may be `dynamic` or *static*. In the following, two forms of dynamic priors will be presented, namely the dynamic ML and dynamic MMI prior. A static prior, which yields a MAP update, is also given.

A dynamic ML prior is based on the ML statistics. The prior mean and covariance matrix are given by the dynamic ML estimates respectively, which can be expressed in terms of the ML sufficient statistics as follows:

$$\boldsymbol{\mu}_{sm}^{\texttt{prior}} = \hat{\boldsymbol{\mu}}_{sm}^{\texttt{ml}} = \frac{\boldsymbol{x}_{sm}^{\texttt{ml}}}{\beta_{sm}^{\texttt{ml}}} \qquad \text{and} \qquad \boldsymbol{\Sigma}_{sm}^{\texttt{prior}} = \hat{\boldsymbol{\Sigma}}_{sm}^{\texttt{ml}} = \frac{\boldsymbol{W}_{sm}^{\texttt{ml}}}{\beta_{sm}^{\texttt{ml}}} \tag{5.51}$$

Hence, in addition to the numerator and denominator statistics, the ML statistics are also required when a dynamic ML prior is used for I-smoothing.

Alternatively, a dynamic MMI prior may also be used for I-smoothing. This is achieved by setting the prior mean and covariance matrix as the corresponding the MMI estimates. The MMI update formulae for mean and covariance matrix take similar forms to the MPE update formulae in equations (5.30) [95], which is given by

$$\hat{\boldsymbol{\mu}}_{sm}^{\texttt{mmi}} = \frac{\boldsymbol{x}_{sm}^{\texttt{mmi}} + D_{sm} \boldsymbol{\mu}_{sm}}{\beta_{sm}^{\texttt{mmi}} + D_{sm}} \tag{5.52}$$

$$\hat{\boldsymbol{\Sigma}}_{sm}^{\texttt{mmi}} = \frac{\boldsymbol{Y}_{sm}^{\texttt{mmi}} + D_{sm}(\boldsymbol{\Sigma}_{sm} + \boldsymbol{\mu}_{sm} \boldsymbol{\mu}_{sm}')}{\beta_{sm}^{\texttt{mmi}} + D_{sm}} - \boldsymbol{\mu}_{sm} \boldsymbol{\mu}_{sm}' \tag{5.53}$$

where $\beta_{sm}^{\texttt{mmi}}$, $\boldsymbol{x}_{sm}^{\texttt{mmi}}$ and $\boldsymbol{Y}_{sm}^{\texttt{mmi}}$ are the corresponding MMI statistics. Thus, the dynamic MMI prior is given by

$$\boldsymbol{\mu}_{sm}^{\texttt{prior}} = \hat{\boldsymbol{\mu}}_{sm}^{\texttt{mmi}} \qquad \text{and} \qquad \boldsymbol{\Sigma}_{sm}^{\texttt{prior}} = \hat{\boldsymbol{\Sigma}}_{sm}^{\texttt{mmi}} \tag{5.54}$$

A dynamic ML prior I-smoothing may also be employed to the MMI estimates of the mean and covariance matrix in equations (5.52) and (5.53) respectively. Hence, it is possible to the smooth the MPE estimates with a dynamic MMI prior which is in turn smoothed by a dynamic ML prior. The effect is similar to smoothing the MPE estimates with both dynamic ML and MMI priors. The use of dynamic MMI prior has been found experimentally to yield slightly improved performance compared to using a dynamic ML prior [106].

In some cases, dynamic priors may not be obtained robustly. This may be the case when building a gender dependent MPE model. A *Maximum a Posteriori* (MAP) estimation scheme is usually employed to improve robustness when building gender dependent (GD) models. Incorporating MAP estimation for MPE training can be achieved by using the I-smoothing technique with *static* priors [101]. A suitable static prior for GD MAP would be the corresponding mean vectors and covariance matrices from the gender independent model. A static prior is used by setting

$$\boldsymbol{\mu}_{sm}^{\texttt{prior}} = \boldsymbol{\mu}_{sm}^{\texttt{static}} \qquad \text{and} \qquad \boldsymbol{\Sigma}_{sm}^{\texttt{prior}} = \boldsymbol{\Sigma}_{sm}^{\texttt{static}} \tag{5.55}$$

where $\boldsymbol{\mu}_{sm}^{\texttt{static}}$ and $\boldsymbol{\Sigma}_{sm}^{\texttt{static}}$ are the prior mean and covariance matrix respectively. The corresponding second order 'statistics' is given by

$$\boldsymbol{Y}_{sm}^{\texttt{prior}} = \boldsymbol{\Sigma}_{sm}^{\texttt{static}} + \boldsymbol{\mu}_{sm}^{\texttt{static}} \boldsymbol{\mu}_{sm}^{\texttt{static}'} \tag{5.56}$$

6

Adaptation of Precision Matrix Models

In Section 2.2.3, the importance of adaptation and adaptive training techniques has been briefly discussed. This chapter will focus on the discussion of the MLLR framework and its application to various structured precision matrix models described earlier in Chapter 3. In MLLR, the linear transformation matrices, $\boldsymbol{A}^{\mathrm{r}}$ for the mean vectors, and $\boldsymbol{H}^{\mathrm{r}}$ for the covariance matrices, are defined, one for each regression class, $r = 1, 2, \ldots, R$, where R is the total number of regression classes. The adapted mean vector and covariance matrix can be written as

$$\tilde{\boldsymbol{\mu}}_{sm} = \boldsymbol{A}^{\mathrm{r}}\boldsymbol{\mu}_{sm} + \boldsymbol{b}^{\mathrm{r}} = \boldsymbol{X}^{\mathrm{r}}\boldsymbol{\xi}_{sm} \tag{6.1}$$

$$\tilde{\boldsymbol{\Sigma}}_{sm} = \boldsymbol{H}^{\mathrm{r}}\boldsymbol{\Sigma}_{sm}\boldsymbol{H}^{\mathrm{r}\prime} \tag{6.2}$$

where the augmented mean vector, $\boldsymbol{\xi}_{sm}$, and the augmented transformation matrix for regression class r, $\boldsymbol{X}^{\mathrm{r}}$, are given by

$$\boldsymbol{\xi}_{sm} = \begin{bmatrix} \boldsymbol{\mu}_{sm} \\ 1 \end{bmatrix} \tag{6.3}$$

$$\boldsymbol{X}^{\mathrm{r}} = \begin{bmatrix} \boldsymbol{A}^{\mathrm{r}} & \boldsymbol{b}^{\mathrm{r}} \end{bmatrix} \tag{6.4}$$

An alternative covariance matrix transformation is also given in [33] as

$$\tilde{\boldsymbol{\Sigma}}_{sm} = \boldsymbol{L}_{sm}\boldsymbol{H}^{\mathrm{r}}\boldsymbol{L}'_{sm} \tag{6.5}$$

where $\boldsymbol{L}_{sm}$ is the Choleski factor of the original covariance matrix, $\boldsymbol{\Sigma}_{sm}$ ($\boldsymbol{\Sigma}_{sm} = \boldsymbol{L}_{sm}\boldsymbol{L}'_{sm}$). Although the estimation of $\boldsymbol{H}^{\mathrm{r}}$ is simple, the computational cost associated with recognition is high [40]. Thus, equation (6.2) is more commonly used to transform the covariance matrix.

The transformation parameters can be estimated by maximising the ML auxiliary function in equation (2.26), replacing the mean vector and covariance matrices with those given in equations (6.1) and (6.2) respectively. The resulting auxiliary function for a given regression class,

r, omitting the `ml` superscript, is given by

$$\mathcal{Q}^{\mathtt{r}}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) = K_r + \sum_{(s,m)\in\mathcal{C}_r} \left\{ \beta_{sm} \log |\boldsymbol{Z}^{\mathtt{r}}| - \frac{1}{2} \mathrm{Tr}(\boldsymbol{Z}^{\mathtt{r}} \boldsymbol{P}_{sm} \boldsymbol{Z}^{\mathtt{r}\prime} \boldsymbol{W}^{\mathtt{r}}_{sm}) \right\} \tag{6.6}$$

where K_r subsume terms independent of the adaptation parameters of regression class r, $\mathcal{C}_r$ denotes the set of components in regression class r, $\boldsymbol{Z}^{\mathtt{r}} = (\boldsymbol{H}^{\mathtt{r}})^{-1}$ is the linear transformation of the precision matrices and

$$\boldsymbol{W}^{\mathtt{r}}_{sm} = \sum_{t=1}^{T} \gamma_{sm}(t)(\boldsymbol{o}_t - \boldsymbol{X}^{\mathtt{r}}\boldsymbol{\xi}_{sm})(\boldsymbol{o}_t - \boldsymbol{X}^{\mathtt{r}}\boldsymbol{\xi}_{sm})' \tag{6.7}$$

$$\beta_{sm} = \sum_{t=1}^{T} \gamma_{sm}(t) \tag{6.8}$$

The adaptation parameters, $\boldsymbol{X}^{\mathtt{r}}$ and $\boldsymbol{Z}^{\mathtt{r}}$ can be estimated by maximising equation (6.6). The estimation formulae of these parameters are derived in [31, 70] for both *diagonal* and *full* covariance matrix systems. The update formulae for MLLR mean, variance and constrained MLLR adaptation methods will be discussed further in Sections 6.1, 6.2 and 6.3 respectively. The MLLR techniques is particularly attractive for *diagonal* covariance matrix systems due to the independence assumption between feature elements. As a result, the linear transformation matrices can be estimated row-by-row efficiently and independently. Unfortunately, for the case of full covariance matrix systems, the transformation matrices cannot be updated row-by-row independently. The closed-form solution for updating the entire transformation matrix is computationally expensive and may not be robust as the numerical precision required by the calculations may exceed the limit of the machines as the feature dimensionality increases. In previous work on SAT training of EMLLT [58] and SPAM [6] models, the numerical stability issue was overcome by using numerical optimisation techniques. In the following, efficient adaptation methods for the full covariance matrix models as well as various structured precision matrix models will be described. In particular, an efficient row-by-row iterative update approach for MLLR mean and CMLLR adaptation is presented. The update formulae are derived within the standard Expectation Maximisation (EM) framework.

6.1 MLLR Mean Adaptation

In MLLR mean adaptation, the adapted mean is given by a linear transformation of the original mean followed by a translation (linear offset), as given by equation (6.1). The transformation matrix, $\boldsymbol{A}^{\mathtt{r}}$, and the translation vector, $\boldsymbol{b}^{\mathtt{r}}$, are updated by maximising the auxiliary function in equation (6.6). The parameters can be expressed in a more compact form of augmented transformation matrix, $\boldsymbol{X}^{\mathtt{r}}$, as given by equation (6.4). This auxiliary function, expressed in

terms of $\boldsymbol{X}^{\mathtt{r}}$, becomes,

$$\mathcal{Q}^{\mathtt{r}}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) = K_r - \frac{1}{2} \sum_{(s,m)\in\mathcal{C}_r} \sum_{t=1}^{T} \gamma_{sm}(t) \left(\boldsymbol{o}_t - \boldsymbol{X}^{\mathtt{r}}\boldsymbol{\xi}_{sm}\right)' \boldsymbol{P}_{sm} \left(\boldsymbol{o}_t - \boldsymbol{X}^{\mathtt{r}}\boldsymbol{\xi}_{sm}\right) \tag{6.9}$$

where $\boldsymbol{\xi}_{sm}$ is the corresponding augmented mean vector. Differentiating this with respect to $\boldsymbol{X}^{\mathtt{r}}$ and equating to zero yields [40]

$$\mathrm{vec}(\boldsymbol{X}^{\mathtt{r}}) = (\boldsymbol{G}^r)^{-1} \mathrm{vec}(\boldsymbol{K}^r) \tag{6.10}$$

where

$$\boldsymbol{G}^r = \sum_{(s,m)\in\mathcal{C}_r} \boldsymbol{V}^r_{sm} \otimes \boldsymbol{D}^r_{sm} \tag{6.11}$$

$$\boldsymbol{K}^r = \sum_{t=1}^{T} \sum_{(s,m)\in\mathcal{C}_r} \gamma_{sm}(t) \boldsymbol{P}_{sm} \boldsymbol{o}_t \boldsymbol{\xi}'_{sm} \tag{6.12}$$

and

$$\boldsymbol{V}^r_{sm} = \sum_{t=1}^{T} \gamma_{sm}(t) \boldsymbol{P}_{sm} \tag{6.13}$$

$$\boldsymbol{D}^r_{sm} = \boldsymbol{\xi}_{sm} \boldsymbol{\xi}'_{sm} \tag{6.14}$$

$\mathrm{vec}(.)$ is a vectorisation operator that converts a matrix to a vector ordered in terms of the rows and $\otimes$ is the Kronecker product. In the case of diagonal covariance matrix systems ($\boldsymbol{P}_{sm}$ is a diagonal matrix), $\boldsymbol{V}^r_{sm}$ is diagonal and the matrix $\boldsymbol{G}^r$ has a block diagonal structure as a result of the Kronecker product,

$$\boldsymbol{G}^r = \begin{bmatrix} \boldsymbol{G}^{r(1)} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{G}^{r(2)} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \boldsymbol{0} \\ \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{G}^{r(d)} \end{bmatrix} \quad \text{and} \quad \boldsymbol{K}^r = \begin{bmatrix} \boldsymbol{k}^{r(1)} \\ \boldsymbol{k}^{r(2)} \\ \vdots \\ \boldsymbol{k}^{r(d)} \end{bmatrix} \tag{6.15}$$

where the sufficient statistics are given by

$$\boldsymbol{G}^{r(j)} = \sum_{(s,m)\in\mathcal{C}_r} \psi_{smj} \beta_{sm} \boldsymbol{\xi}_{sm} \boldsymbol{\xi}'_{sm} \tag{6.16}$$

$$\boldsymbol{k}^{r(j)} = \sum_{(s,m)\in\mathcal{C}_r} \psi_{smj} \left(\sum_{t=1}^{T} \gamma_{sm}(t) o_{tj} \right) \boldsymbol{\xi}'_{sm} \tag{6.17}$$

$$\beta_{sm} = \sum_{t=1}^{T} \gamma_{sm}(t) \tag{6.18}$$

The block-diagonal structure of $\boldsymbol{G}^r$ simplifies equation (6.10) to row-wise independent update

$$\boldsymbol{x}^{\mathtt{r}}_j = (\boldsymbol{G}^{r(j)})^{-1} \boldsymbol{k}^{r(j)} \tag{6.19}$$

where $\boldsymbol{x}^{\mathtt{r}}_j$ is the jth row of $\boldsymbol{X}^{\mathtt{r}}$.

6.1.1 Iterative Row-by-row Update

Clearly, any form of correlation modelling schemes will result in a *non-block-diagonal* structure for $\boldsymbol{G}^r$, which is a symmetric matrix of size $d^2 \times d^2$. Standard inversion routine requires $\mathcal{O}(d^6)$ operations which is computationally expensive for practical applications. Inversion of such a huge matrix may also be numerically unstable. To overcome is problem, the transformation matrices are updated in an iterative row-by-row fashion. Each row is updated by keeping the remaining rows constant. So, maximising equation (6.9) with respect to $\boldsymbol{x}_j^{\mathtt{r}}$ and equating to zero yields

$$\boldsymbol{x}_j^{\mathtt{r}} = (\boldsymbol{G}^{r(j,j)})^{-1}\boldsymbol{k}^{r(j)} \tag{6.20}$$

where

$$\boldsymbol{G}^{r(j,k)} = \sum_{(s,m)\in\mathcal{C}_r} \psi_{sm}(j,k)\beta_{sm}\boldsymbol{\xi}_{sm}\boldsymbol{\xi}_{sm}' \tag{6.21}$$

$$\boldsymbol{k}^{r(j)} = \sum_{(s,m)\in\mathcal{C}_r} \boldsymbol{p}_{sm}(j)\left(\sum_{t=1}^{T}\gamma_{sm}(t)\boldsymbol{o}_t\right)\boldsymbol{\xi}_{sm}' - \sum_{k=1,k\neq j}^{d}\boldsymbol{x}_k^{\mathtt{r}}\boldsymbol{G}^{r(j,k)} \tag{6.22}$$

$\psi_{sm}(j,k)$ and $\boldsymbol{p}_{sm}(j)$ denote the (j,k) element and the jth row of $\boldsymbol{P}_{sm}$ respectively. Not surprising, equations (6.19) and (6.20) are similar since both methods are row-by-row updates. The main difference between these equations is due to the non-zero correlations in the latter. As a result, the update of the rows is no longer independent. The dependencies are clearly shown by the last term on the right hand side of equation (6.22). In fact, when $\boldsymbol{P}_{sm}$ is diagonal, equations (6.21) and (6.22) simplifies as equations (6.16) and (6.17) respectively, as expected.

Since the update of the rows of the transformation matrix depends on other rows, an initial estimate of $\boldsymbol{W}^r$ is required and an iterative approach is adopted. In the next section, various initialisation schemes are given.

6.1.2 Approximation Schemes

Although $\boldsymbol{W}^r$ can be initialised as an identity matrix, a better starting value may be found by using some kind of diagonal approximation of the precision matrix, $\boldsymbol{P}_{sm}$ such that $\psi_{sm}(j,k) = 0$ for $j \neq k$. Hence, the system simplifies to a diagonal covariance system and the update formula in equation (6.19) may be used. There are three different ways of approximating the precision matrix, $\boldsymbol{P}_{sm}$, as a diagonal matrix. They are:

- **Least Squares Approximation**: $\boldsymbol{P}_{sm} = \boldsymbol{I}$
- **Diagonal Covariance Matrix Approximation**: $\boldsymbol{P}_{sm} = (\mathrm{diag}(\boldsymbol{\Sigma}_{sm}))^{-1}$

- **Diagonal Precision Matrix Approximation**: $\boldsymbol{P}_{sm} = \text{diag}(\boldsymbol{P}_{sm})$

Among the three, diagonal precision approximation yields the best initialisation since equations (6.21) and (6.22) are directly expressed in terms of $\boldsymbol{P}_{sm}$. Interestingly, if the number of subsequent row-by-row iterations is reduced to zero, the initialisation methods become an approximation scheme for the transform estimation. In fact, the results presented later in Chapter 8 indicates that subsequent row-by-row iterations yield very little gain in terms of likelihood and the diagonal precision matrix approximation itself gives good estimates.

6.2 MLLR Variance Adaptation

Variance transformation is achieved using equation (6.2). In the case where the original covariance matrices are diagonal, an efficient iterative row-by-row solution exists [33]. The update for the jth row of the transformation matrix of the precision matrix, $\boldsymbol{Z}^{\mathtt{r}}$ is given by

$$\boldsymbol{z}_j^{\mathtt{r}} = \boldsymbol{c}_j^r \boldsymbol{G}^{r(j)} \sqrt{\frac{\beta}{\boldsymbol{c}_j^r \boldsymbol{G}^{r(j)-1} \boldsymbol{c}_j^{r'}}} \tag{6.23}$$

where $\boldsymbol{c}_j^r$ is the row of cofactors corresponding to the jth row of $\boldsymbol{z}_j^{\mathtt{r}}$ and

$$\beta = \sum_{(s,m)\in\mathcal{C}_r} \beta_{sm} \tag{6.24}$$

Unfortunately, when the covariance matrices are not diagonal, there is no efficient closed-form solution for the update of $\boldsymbol{Z}^{\mathtt{r}}$ and numerical approaches have to be sought. The computational cost of these approaches are typically high which prohibits its use in practical speech recognition systems. However, for basis superposition precision matrix models, the superposition structures may be exploited to yield a variance adaptation method which is computationally more tractable. Recall from Chapter 3 that a basis superposition precision matrix can be expressed as

$$\boldsymbol{P}_{sm} = \sum_{i=1}^{n} \lambda_{smi} \boldsymbol{S}_i \tag{6.25}$$

Hence, one way of adapting $\boldsymbol{P}_{sm}$ is to adapt the basis matrices, $\boldsymbol{S}_i$. Since $\boldsymbol{S}_i$ is globally shared by all Gaussian components, the estimation of $\boldsymbol{S}_i$ requires only a small amount of adaptation data. Hence, the same basis update formulae given in Chapter 4 can be used. This form of variance adaptation is particularly interesting when the basis matrices are of *rank one* (STC and EMLLT), where the precision matrices are given by

$$\boldsymbol{P}_{sm} = \boldsymbol{A}\boldsymbol{\Lambda}_{sm}\boldsymbol{A} \tag{6.26}$$

Comparing this with equation (6.2), the variance adaptation of a diagonal covariance matrix system is in fact estimating a new set of *bases* that reflects the target condition. Therefore, the update formulae for the STC bases and the variance transformation matrices are similar. However, for more complicated precision matrix model such as SPAM, the update of the basis matrices relies on numerical schemes which is computationally expensive. In the following section, an efficient CMLLR adaptation scheme for structured precision matrices will be described. This method constrains the transformation matrices for the mean and covariance matrix to be the same. This, to some extend, achieves variance adaptation *implicitly*.

6.3 Constrained MLLR (CMLLR) Adaptation

A special case of adaptation scheme is when both the mean and variance adaptations share the same transformation matrix, i.e. $\boldsymbol{A}^{\mathrm{r}} = \boldsymbol{H}^{\mathrm{r}}$. In this case, it is equivalent to transforming the observation vector with the transformation matrix, $\boldsymbol{B}^{\mathrm{r}} = (\boldsymbol{H}^{\mathrm{r}})^{-1}$. Thus, CMLLR can also be viewed as a feature-based speaker normalisation [33] technique where speaker-dependent feature transforms are estimated. For the same reason, CMLLR is also known as feature MLLR (FMLLR). So, an adapted observation vector can be expressed as

$$\tilde{\boldsymbol{o}}_t = \boldsymbol{B}^{\mathrm{r}}\boldsymbol{o}_t + \boldsymbol{b}^{\mathrm{r}} = \boldsymbol{X}^{\mathrm{r}}\boldsymbol{\zeta}_t \tag{6.27}$$

where the augmented transformation matrix $\boldsymbol{X}^{\mathrm{r}}$ takes the same form as equation (6.4) and

$$\boldsymbol{\zeta}_t = \begin{bmatrix} \boldsymbol{o}_t \\ 1 \end{bmatrix} \tag{6.28}$$

The auxiliary function in equation (6.4), can be rewritten, using equation (6.27), as

$$\begin{aligned} \mathcal{Q}^{\mathrm{r}}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) \quad = \quad & K_r + \sum_{(s,m)\in\mathcal{C}_r} \Big[\beta_{sm} \log |\boldsymbol{Z}^{\mathrm{r}}| \\ & -\frac{1}{2} \sum_{(s,m)\in\mathcal{C}_r} \sum_{t=1}^{T} \gamma_{sm}(t) \left(\tilde{\boldsymbol{o}}_t - \boldsymbol{\mu}_{sm}\right)' \boldsymbol{P}_{sm} \left(\tilde{\boldsymbol{o}}_t - \boldsymbol{\mu}_{sm}\right) \Big] \end{aligned} \tag{6.29}$$

The new parameters is found by maximising equation (6.29). This occurs when its gradient with respect to $\boldsymbol{X}^{\mathrm{r}}$ equals to zero. There is no simple closed-form solution to $\boldsymbol{X}^{\mathrm{r}}$ which maximises equation (6.29) directly. When $\boldsymbol{P}_{sm}$ is a diagonal matrix, an iterative row-by-row update is derived in [33]. For other form of precision matrix models, numerical optimisation schemes were adopted to adapt EMLLT [58] and SPAM [6] models. However, the actual computational costs involved were not discussed. In the following, an iterative scheme, similar to the one for diagonal covariance matrix systems, described in [33] will be detailed for a full covariance matrix system. The update formula is derived within the EM framework and is guaranteed to improve the objective function. This update is also applicable to any form of covariance and precision matrix models by performing adaptation on the resulting covariance matrices.

6.3.1 Iterative Row-by-row Update

To maximise equation (6.29) in a row-by-row fashion, the partial differential with respect to each row of $\boldsymbol{X}^{\mathtt{r}}$ is computed, keeping the rest of the rows constant. Equating this differential to zero yields the row-by-row update formula as

$$\boldsymbol{x}_j^{\mathtt{r}} = \left(\alpha\tilde{\boldsymbol{c}}_j + \boldsymbol{k}^{r(j)}\right)\boldsymbol{G}^{r(j,j)-1} \tag{6.30}$$

where $\tilde{\boldsymbol{c}}_j = [\boldsymbol{c}_j \;\; 0]$ is the augmented vector of cofactors and

$$\boldsymbol{G}^{r(j,k)} = \sum_{(s,m)\in\mathcal{C}_r} \psi_{sm}(j,k)\sum_{t=1}^{T}\gamma_{sm}(t)\boldsymbol{\zeta}_t\boldsymbol{\zeta}_t' \tag{6.31}$$

$$\boldsymbol{k}^{r(j)} = \sum_{(s,m)\in\mathcal{C}_r} \boldsymbol{p}_{sm}(j)\boldsymbol{\mu}_{sm}\left(\sum_{t=1}^{T}\gamma_{sm}(t)\boldsymbol{\zeta}_t'\right) - \sum_{k=1,k\neq j}^{d}\boldsymbol{x}_k^{\mathtt{r}}\boldsymbol{G}^{r(j,k)} \tag{6.32}$$

Again, observing that $\boldsymbol{G}^{r(j,k)} = 0$ when $\psi_{sm}(j,k) = 0$ for $j \neq k$, equation (6.32) simplifies to the update for the diagonal covariance matrix case [33]. In addition to the term $\tilde{\boldsymbol{c}}_j$, the update of one row is also dependent on the other rows through the second term on the right hand side of equation (6.32).

6.3.2 Approximation using a Diagonal Covariance Matrix System

Unlike the case of MLLR mean, the diagonal precision matrix approximation does not work for constrained MLLR because the estimated transforms operates on both the mean vectors and the precision matrices. However, the CMLLR transforms estimation process for any other form of precision matrix models can be approximated using a diagonal covariance matrix model. For a good approximation, this model should be the initial model used to train the precision matrix models.

6.4 Sufficient Statistics for Structured Precision Matrix Models

So far, a *computationally* efficient row-by-row update approach has been introduced for both MLLR mean and CMLLR adaptations. This row-by-row update scheme is very similar to the approach used in the diagonal covariance matrix system. However, the memory required for the sufficient statistics is considerably higher for a general full covariance matrix model. Recall that the sufficient statistics are given by

- $\boldsymbol{G}^{r(j,k)}$ – a $d \times d$ symmetric matrix for $1 \leq j \leq d$ and $1 \leq k \leq j$

- $\boldsymbol{k}^{r(j)}$ – a d-dimensional vector for $1 \leq j \leq d$
- β^{r} – a scalar quantity

for each regression class, r. These statistics are given by equations (6.21) and (6.22) for MLLR mean and equations (6.31) and (6.32) for CMLLR. Among these statistics, $\boldsymbol{G}^{r(j,k)}$, which is a symmetric matrix, dominates the required memory. The total number of the $\boldsymbol{G}^{r(j,k)}$ matrices to be stored is $\frac{d}{2}(d+1)$ and each matrix contains $\frac{d}{2}(d+1)$ terms. So, the total required memory will be of the order of $\mathcal{O}(d^4)$. Note, from equations (6.21) and (6.31), that the only term which depends on j and k is $\psi_{sm}(j,k)$, the (j,k)th element of the precision matrix, $\boldsymbol{P}_{sm}$. Hence, the the precision matrix is modelled by some form of structured approximation, this underlying model structure can be exploited to reduce the memory requirement. Consider the generic form of basis superposition:

$$\psi_{sm}(j,k) = \sum_{i=1}^{n} \lambda_{smi} s_i(j,k) \tag{6.33}$$

So, instead on accumulating the statistics in the original symmetric matrix space, one can simply accumulate statistics within the subspace spanned by the basis matrices, in much the same way as the use of *projected* statistics for the model parameters update described in Chapter 4. Define the statistics associated with each basis matrix as $\boldsymbol{G}^{r(i)}$ such that

$$\boldsymbol{G}^{r(j,k)} = \sum_{i=1}^{n} s_i(j,k) \boldsymbol{G}^{r(i)} \quad \text{and} \quad \boldsymbol{G}^{r(i)} = \sum_{(s,m)\in\mathcal{C}_r} \lambda_{smi} \boldsymbol{G}_{sm}$$

where $\boldsymbol{G}_{sm}$ are given by

$$\boldsymbol{G}_{sm}^{\texttt{mllr}} = \beta_{sm} \boldsymbol{\xi}_{sm} \boldsymbol{\xi}_{sm}' \tag{6.34}$$

$$\boldsymbol{G}_{sm}^{\texttt{cmllr}} = \sum_{t=1}^{T} \gamma_{sm}(t) \boldsymbol{\zeta}_t \boldsymbol{\zeta}_t' \tag{6.35}$$

for MLLR mean and CMLLR respectively. $s_i(j,k)$ denotes the (j,k)th element of the ith basis matrix, $\boldsymbol{S}_i$ and $1 \leq i \leq n$. So, instead of storing $\frac{d}{2}(d+1)$ terms of $\boldsymbol{G}^{r(j,k)}$, only n terms of $\boldsymbol{G}^{r(i)}$ are needed. Thus, the required memory is reduced from the order $\mathcal{O}(d^4)$ to $\mathcal{O}(nd^2)$.

6.5 Discussions

This chapter has described the adaptation techniques for precision matrix models within the MLLR framework. Adaptation can be applied to the mean vectors (MLLR mean), covariance matrices (MLLR variance) and the feature vectors (CMLLR or FMLLR). Usually, the models to be adapted have diagonal covariance matrices, in which case, the adaptation process can be carried out efficiently using a *row-by-row* update fashion. This is the result of the independence

assumption to avoid the "curse of dimensionality". However, in one way or another, a covariance or precision matrix model attempts to model the inter-dimensional correlations explicitly. Consequently, the rows of the transforms cannot be updated independently. Previous work on adapting the EMLLT and SPAM models has relied on numerical optimisation schemes to estimate the adaptation transforms. This chapter has shown that an iterative row-by-row update approach similar to the case of diagonal covariance matrix systems can also be applied to full covariance matrix systems. This approach is almost as efficient as the diagonal covariance matrix system computationally. Because the update of the rows depends on other rows, *iterative* estimation scheme are adopted. When there is *complete* dependency between all the dimensions (as in the case of full covariance matrix), the required memory is considerably larger. However, for structured precision matrix approximation schemes, correlations are modelled only within a subspace spanned by the basis matrices of the model. Hence, the sufficient statistics can be accumulated in a more compact form. In other words, adaptation of structured precision matrix models can be achieved efficiently both in terms of computational and memory requirements. This is particularly important for LVCSR systems where these requirements are the major factors that affects the system performance.

7

Temporally Varying Precision Matrix Models

7.1 Introduction

In Chapter 3, various forms of precision matrix modelling techniques have been described within a basis superposition framework. This chapter will extend the basis superposition formulation to allow the model parameters to vary with time. Such formulation is motivated from a trajectory model viewpoint, as briefly described in Section 2.3. Recall, from the *conditional independence assumption*, that the observation probability at time t is conditionally independent of all variables given the current state, q_t, *i.e.*

$$p(\boldsymbol{o}_t|\mathcal{O}_1^{t-1}, Q_1^t, q_t = s, \boldsymbol{\theta}) \approx p(\boldsymbol{o}_t|q_t = s, \boldsymbol{\theta}) \tag{7.1}$$

where $\mathcal{O}_1^T$ and Q_1^T denote the observation and state sequences. $\boldsymbol{o}_t$ and q_t are the observation and state at time t. Typically, this probability distribution is modelled using a Gaussian Mixture Model (GMM):

$$p(\boldsymbol{o}_t|q_t = s, \boldsymbol{\theta}) = \sum_{m=1}^{M} c_{sm} \mathcal{N}\left(\boldsymbol{o}_t, \boldsymbol{\mu}_{sm}, \boldsymbol{\Sigma}_{sm}\right) \tag{7.2}$$

where c_{sm}, $\boldsymbol{\mu}_{sm}$ and $\boldsymbol{\Sigma}_{sm}$ denote the weight, mean and covariance matrix respectively for component m in state s. The above distribution is stationary (does not vary with time) given the HMM state. In Section 2.3, several ways of overcoming this limitation have been described, such as the trajectory models [13, 118, 119, 130], segmental models [41, 42, 88, 89] and switching linear dynamical systems [104]. These models can be conveniently described as having time varying state output probability distribution, where the time variation is model specific. For a GMM state output distribution, the time varying Gaussian parameters may be expressed as a general function of the observation sequence, $\mathcal{O}_1^T$, state sequence, Q_1^T, and the time, t, as

follows:

$$\boldsymbol{\mu}_{smt} = f_{\theta_\mu}\left(\mathcal{O}_1^T, Q_1^T, t\right) \tag{7.3}$$

$$\boldsymbol{\Sigma}_{smt} = f_{\theta_\Sigma}\left(\mathcal{O}_1^T, Q_1^T, t\right) \tag{7.4}$$

$$c_{smt} = f_{\theta_c}\left(\mathcal{O}_1^T, Q_1^T, t\right) \tag{7.5}$$

where f_{θ_μ}, f_{θ_Σ} and f_{θ_c} are the functions which describe the time variation of the mean, covariance matrix and component weights respectively. Therefore, equations (7.1) and (7.2) become

$$p(\boldsymbol{o}_t|\mathcal{O}_1^{t-1}, Q_1^t, q_t = s, \boldsymbol{\theta}) = p(\boldsymbol{o}_t|\boldsymbol{\theta}_t) = \sum_{m=1}^{M} c_{sm}\mathcal{N}\left(\boldsymbol{o}_t, \boldsymbol{\mu}_{smt}, \boldsymbol{\Sigma}_{smt}\right) \tag{7.6}$$

In the next section, a discriminative semi-parametric trajectory model will be described, where the form of f_{θ_μ} and f_{θ_Σ} will be introduced for modelling the time varying mean and covariance matrix respectively. This is followed by the derivation of the parameter estimation formulae in Section 7.3.

7.2 Semi-parametric Trajectory Model

A semi-parametric trajectory model can be formulated by expressing equations (7.3) and (7.4) as follows:

$$\boldsymbol{\mu}_{smt} = \boldsymbol{A}_t\boldsymbol{\mu}_{sm} + \boldsymbol{b}_t \tag{7.7}$$

$$\boldsymbol{P}_{smt} = \boldsymbol{Z}_t\boldsymbol{P}_{sm}\boldsymbol{Z}_t{}' \tag{7.8}$$

where $\boldsymbol{A}_t$ and $\boldsymbol{Z}_t$ are the time dependent linear transformations for the mean vector and precision matrix respectively. $\boldsymbol{b}_t$ denotes a time dependent bias vector for the mean. When the linear transformations are set as identity matrices ($\boldsymbol{A}_t = \boldsymbol{Z}_t = \boldsymbol{I}$) and the bias vector is set as a zero vector ($\boldsymbol{b}_t = \boldsymbol{0}$), the above expressions degenerate to the mean and precision matrix of a standard HMM system. Therefore, the above form of trajectory model can be viewed as applying a time varying affine transformation to the component mean vectors and precision matrices in the system. The form of time varying transformation parameters will be represented in semi-parametric way as described in the next section.

7.2.1 A Semi-parametric Representation

Modelling of the time variation is an important aspect for trajectory models. In this work, a *semi-parametric* representation will be considered. First, a series of centroids is defined to represent the regions of interest in the acoustic feature space. Associated with the ith centroid, the following parameters are defined:

- $\boldsymbol{A}^{(\mathtt{i})}$: a linear transformation matrix for the mean vector
- $\boldsymbol{Z}^{(\mathtt{i})}$: a linear transformation matrix for the precision matrix
- $\boldsymbol{b}^{(\mathtt{i})}$: a bias vector for the mean vector

The corresponding time varying affine transformations discussed above will be modelled as a *weighted contribution* from all the centroids:

$$\boldsymbol{A}_t = \boldsymbol{I} + \sum_{i=1}^{n} h_i(t)\boldsymbol{A}^{(\mathtt{i})} \tag{7.9}$$

$$\boldsymbol{b}_t = \sum_{i=1}^{n} h_i(t)\boldsymbol{b}^{(\mathtt{i})} \tag{7.10}$$

$$\boldsymbol{Z}_t = \boldsymbol{I} + \sum_{i=1}^{n} h_i(t)\boldsymbol{Z}^{(\mathtt{i})} \tag{7.11}$$

where $h_i(t)$ denotes the contribution weights from the i centroid at time t and n is the total number of centroids.

Each centroid is modelled using a Gaussian component where the Gaussian mean and covariance matrix denote the location and the uncertainty of the centroid in the acoustic space. Let g_i represent the ith centroid represented by the Gaussian component $\mathcal{N}(\boldsymbol{o}_t, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ such that the likelihood of g_i given an observation, $\boldsymbol{o}_t$, is given by

$$p(\boldsymbol{o}_t|g_i) = \frac{1}{\sqrt{(2\pi)^d|\boldsymbol{\Sigma}_i|}} \exp\left\{-\frac{1}{2}(\boldsymbol{o}_t - \boldsymbol{\mu}_i)' \boldsymbol{\Sigma}_i^{-1}(\boldsymbol{o}_t - \boldsymbol{\mu}_i)\right\} \tag{7.12}$$

The weights, $h_i(t)$ is then computed as the posterior probability of g_i given $\boldsymbol{o}_t$,

$$h_i(t) = P(g_i|\boldsymbol{o}_t) = \frac{p(\boldsymbol{o}_t|g_i)P(g_i)}{\sum_{j=1}^{n} p(\boldsymbol{o}_t|g_j)P(g_j)} \tag{7.13}$$

where $P(g_i)$, the prior probability of g_i, is assumed to be uniformly distributed in this work. Consider a two-dimensional example in Figure 7.1. The centroids may be considered as the Vector-Quantisation (VQ) codebook representing the acoustic space. The posterior probabilities, $P(g_i|\boldsymbol{o}_t)$, would then be the probabilistic quantisation of $\boldsymbol{o}_t$. Thus, the interpolation formulae given in equations (7.9), (7.10) and (7.11) can be interpreted as the weighted contribution from the transformations associated with each centroid given the position of the observation in the acoustic space. This formulation is analogous to the way the output probabilities are computed for semi-continuous HMMs, which leads to the interpretation of the above trajectory model as a semi-parametric model.

Figure 7.2 depicts the visualisation of the semi-parametric trajectory model using a two-dimensional example. The interpolation weights are computed as a probabilistic VQ feature at

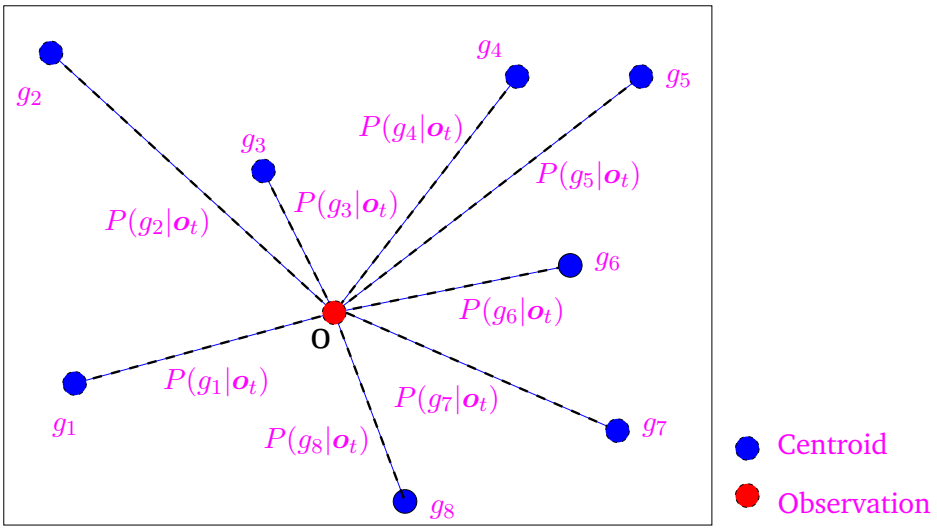

Figure 7.1 *Obtaining interpolation weights from the posterior of a set of centroids given the observation sequence*

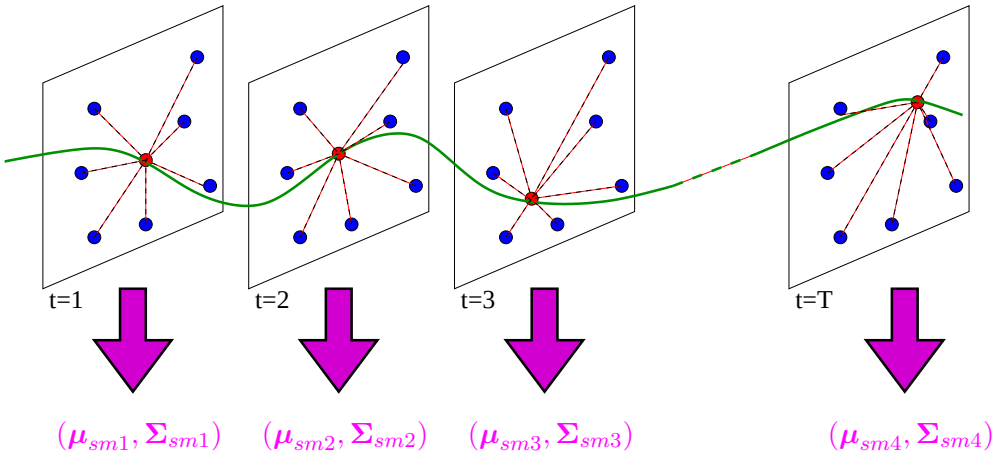

Figure 7.2 *A semi-parametric representation of the Gaussian parameters*

each time t (see Figure 7.1) which tracks the observation as a smoothed trajectory. Interpolation using these time-dependent weights yields a trajectory of the Gaussian parameters, $\boldsymbol{\mu}_{smt}$ and $\boldsymbol{P}_{smt}$, conditioned upon the observation sequence, as given by equations (7.7) and (7.8) respectively.

7.2.2 Time Varying Feature Transformation

Instead of applying time varying transforms to the Gaussian parameters, one may also apply a time varying transform to the feature vectors as follows:

$$\hat{\boldsymbol{o}}_t = \boldsymbol{C}_t \boldsymbol{o}_t + \boldsymbol{d}_t \tag{7.14}$$

where $\boldsymbol{o}_t$ and $\hat{\boldsymbol{o}}_t$ are the original and transformed observation vectors respectively. This is equivalent to setting the case where the linear transformation matrices for the mean vector and covariance matrices are the same. The feature transforms ($\boldsymbol{C}_t$ and $\boldsymbol{d}_t$) are related to the mean and covariance matrix transforms ($\boldsymbol{A}_t$, $\boldsymbol{Z}_t$ and $\boldsymbol{b}_t$) as follows:

$$\boldsymbol{C}_t = \boldsymbol{Z}_t = \boldsymbol{A}_t^{-1} \quad \text{and} \quad \boldsymbol{d}_t = -\boldsymbol{C}_t \boldsymbol{b}_t \tag{7.15}$$

It is interesting to note that equation (7.14) relates directly to the fMPE model [97] when $\boldsymbol{C}_t = \boldsymbol{A}_t = \boldsymbol{I}$. This is true when $\boldsymbol{A}^{(\mathrm{i})} = 0$ for all i. Thus, the effective mean vector at each time t is simply given by the component mean, $\boldsymbol{\mu}_{sm}$, shifted by a time-dependent bias, $\boldsymbol{b}_t$. In [97], fMPE was presented by *Povey et al.* as a way of discriminative training features by adding a time dependent bias to the original feature. This time dependent bias was obtained by projecting a high dimensional vector of posteriors $\boldsymbol{h}_t$ (a vector whose elements are given by $h_i(t)$) to the original feature space. This method then learns the projection matrix (whose columns are given by the negative[1] of the mean biases $\boldsymbol{b}^{(\mathrm{i})}$ in this case) by maximising the Minimum Phone Error (MPE) criterion. Following the introduction of fMPE technique, further refinements were made by Povey in [96]. One of the improvements made to the originally proposed fMPE technique was augmenting the high dimensional feature vector of posteriors such that

$$\boldsymbol{h}_t = \begin{bmatrix} \bar{\boldsymbol{h}}_{1t} \\ \bar{\boldsymbol{h}}_{2t} \\ \vdots \\ \bar{\boldsymbol{h}}_{nt} \end{bmatrix} \tag{7.16}$$

where $\bar{\boldsymbol{h}}_{it}$ is a $(d+1)$-dimensional vector given by

$$\bar{\boldsymbol{h}}_{it} = \begin{bmatrix} 1 \\ (\boldsymbol{o}_t - \boldsymbol{\mu}_i) \end{bmatrix} h_i(t) \tag{7.17}$$

[1] The negative sign is due to the fact that adding a bias to the mean vector is equivalent to subtracting it from the observation vector

Using the notation in [97] where $\boldsymbol{M}$ denotes the fMPE projection matrix, the feature transformation with the new posterior feature, $\boldsymbol{h}_t$, in equation (7.16) is given by

$$\hat{\boldsymbol{o}}_t = \boldsymbol{o}_t + \boldsymbol{M}\boldsymbol{h}_t \tag{7.18}$$

$$= \boldsymbol{o}_t + \begin{bmatrix} \boldsymbol{M}_1 & \boldsymbol{M}_2 & \cdots & \boldsymbol{M}_n \end{bmatrix} \begin{bmatrix} \bar{\boldsymbol{h}}_{1t} \\ \bar{\boldsymbol{h}}_{2t} \\ \vdots \\ \bar{\boldsymbol{h}}_{nt} \end{bmatrix} \tag{7.19}$$

$$= \boldsymbol{o}_t + \sum_{i=1}^{n} \boldsymbol{M}_i \bar{\boldsymbol{h}}_{it} \tag{7.20}$$

$$= \boldsymbol{o}_t + \sum_{i=1}^{n} \boldsymbol{M}_i \begin{bmatrix} 1 \\ (\boldsymbol{o}_t - \boldsymbol{\mu}_i) \end{bmatrix} h_i(t) \tag{7.21}$$

$$\tag{7.22}$$

The above expression may be rewritten in terms of $\boldsymbol{C}_t$ and $\boldsymbol{d}_t$ such that

$$\hat{\boldsymbol{o}}_t = \boldsymbol{o}_t + \sum_{i=1}^{n} \boldsymbol{M}_i \begin{bmatrix} 1 \\ (\boldsymbol{o}_t - \boldsymbol{\mu}_i) \end{bmatrix} h_i(t) \tag{7.23}$$

$$= \boldsymbol{o}_t + \sum_{i=1}^{n} h_i(t) \left(\boldsymbol{C}^{(i)} \boldsymbol{o}_t + \boldsymbol{d}^{(i)} \right) \tag{7.24}$$

$$= \boldsymbol{C}_t \boldsymbol{o}_t + \boldsymbol{d}_t \tag{7.25}$$

where

$$\boldsymbol{C}_t = \sum_{i=1}^{n} h_i(t) \left(\boldsymbol{I} + \boldsymbol{C}^{(i)} \right) \tag{7.26}$$

$$\boldsymbol{d}_t = \sum_{i=1}^{n} h_i(t) \boldsymbol{d}^{(i)} \tag{7.27}$$

$$\boldsymbol{M}_i = \begin{bmatrix} \left(\boldsymbol{I} + \boldsymbol{C}^{(i)} \right) \boldsymbol{d}^{(i)} & \boldsymbol{C}^{(i)} \end{bmatrix} \tag{7.28}$$

Hence, the modified posterior feature given by equation (7.16) is equivalent to applying a time dependent full linear transform, $\boldsymbol{C}_t$ to the observation vector, in addition to the time varying bias vector, $\boldsymbol{d}_t$.

Moreover, it is worth pointing out that using a full transformation matrices for equations (7.7) and (7.8) is not practical as this incurs a high computational cost in applying the transformation at each time to each Gaussian component in the system (typical LVCSR systems comprises more than 100,000 Gaussian components). This problem may be alleviated by using diagonal transforms. In other words, transformations are applied per dimension independently. Then, equations (7.7) and (7.8) may be expressed as scaling and shifting of the mean and diagonal precision matrix elements for each dimension:

$$\mu_{smtj} = a_{tjj} \mu_{smj} + b_{tj} \tag{7.29}$$

$$\psi_{smtj} = z_{tjj}^2 \psi_{smj} \tag{7.30}$$

where μ_{smtj} and b_{tj} are the jth element of $\boldsymbol{\mu}_{smt}$ and $\boldsymbol{b}_t$ respectively. ψ_{smtj}, a_{tjj} and z_{tjj} denote the jth diagonal element of $\boldsymbol{P}_{smt}$, $\boldsymbol{A}_t$ and $\boldsymbol{Z}_t$ respectively. μ_{smj} and ψ_{smj} denote the jth element of the *time independent* mean and precision matrix respectively for component m in state s. Temporally varying precision scaling is known as pMPE[2] [112]. In the following, the semi-parametric trajectory model parameters estimation will be presented based on the use of an identity matrix for the mean transformation, $\boldsymbol{A}_t$ and a diagonal transform for the precision matrix, $\boldsymbol{Z}_t$. The latter transformation will be referred to as the pMPE model.

7.3 Parameters Estimation

The parameterisation of the semi-parametric trajectory model can be broadly divided into those associated with the standard HMMs ($\boldsymbol{\theta}^{\mathrm{h}}$) and those associated with the centroids ($\boldsymbol{\theta}^{\mathrm{c}}$). In the remaining of this discussion, $\boldsymbol{\theta}^{\mathrm{h}}$ and $\boldsymbol{\theta}^{\mathrm{c}}$ will be referred to as the *static* and *dynamic* parameters respectively to emphasise the latter as the parameters that capture the temporally varying attributes of the trajectory model. This section derives the estimation formulae for these parameters using the MPE criterion described earlier in Chapter 5. Recall that the MPE objective function given in equation (5.4) is a measure of the expected phone accuracy of recognising the training data given the HMM model. For convenience, this objective function will be repeated here:

$$\mathcal{R}^{\mathrm{mpe}}(\boldsymbol{\theta}) = \sum_{u=1}^{U} \sum_{h \in H_u} p(h|\mathcal{O}_1^T, \boldsymbol{\theta}) \mathrm{PhoneAcc}(h, \hat{h}) \tag{7.31}$$

where $\boldsymbol{\theta}$ now encompasses both $\boldsymbol{\theta}^{\mathrm{h}}$ and $\boldsymbol{\theta}^{\mathrm{c}}$. As previously mentioned in Chapter 5, it is difficult to optimisation this objective function directly. Again, the weak-sense auxiliary function will be used. Recall from Section 5.2 that the weak-sense auxiliary function is given by equation (5.9). Here, the weak-sense auxiliary function will be used without the smoothing function term ($\mathcal{R}^{\mathrm{s}}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}})$) and it will be rewritten, for the purpose of making the subsequent derivations simpler, as

$$\mathcal{Q}^{\mathrm{mpe}}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) = \sum_{t=1}^{T} \sum_{s=1}^{S} \sum_{m=1}^{M} \gamma_{sm}^{\mathrm{mpe}}(t) \log p(\boldsymbol{o}_t|\boldsymbol{\theta}) \tag{7.32}$$

where the log likelihood of component m in state s is given by,

$$\log p(\boldsymbol{o}_t|\boldsymbol{\theta}) = K_{sm} + \frac{1}{2} \sum_{j=1}^{d} \left\{ \log \psi_{smtj} - \psi_{smtj} \left(o_{tj} - \mu_{smtj} \right)^2 \right\} \tag{7.33}$$

K_{sm} subsumes terms independent of the model parameters. T is the total number of training speech frames, M is the total number of Gaussian components per state and S is the total number of states in the system. $\gamma_{sm}^{\mathrm{mpe}}(t)$ may be regarded as the 'MPE posterior' for component m in state s at time t, which is calculated for standard MPE training [95]. Maximising the above

[2] In this work, only diagonal covariance matrix systems are considered for pMPE

weak-sense auxiliary function with respect to all the model parameters ($\boldsymbol{\theta}^{\mathrm{h}}$ and $\boldsymbol{\theta}^{\mathrm{c}}$) is not trivial. Hence, these two sets of model parameters will be updated separately, each time keeping the other parameter set constant.

7.3.1 Static Parameters Estimation

First, consider the update of the static parameters given that the dynamic parameters are held constant. The weak-sense auxiliary function to be optimised in equation (5.9) is rewritten in terms of the trajectory model parameters as

$$\mathcal{Q}^{\mathrm{mpe}}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) = K + \frac{1}{2} \sum_{s=1}^{S} \sum_{m=1}^{M} \sum_{t=1}^{T} \sum_{j=1}^{d} \gamma_{sm}^{\mathrm{mpe}}(t) \left\{ \log |\psi_{smtj}| - \psi_{smtj} \left(o_{tj} - \mu_{smtj} \right)^2 \right\} \tag{7.34}$$

The new parameters are found such that the differential of the auxiliary function with respect to the parameters at the new estimates equals to zero. Thus,

$$\begin{aligned} \frac{\partial \mathcal{Q}^{\mathrm{mpe}}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}})}{\partial \mu_{smj}} &= \sum_{t=1}^{T} \left(\frac{\partial \mathcal{Q}^{\mathrm{mpe}}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}})}{\partial \mu_{smtj}} \frac{\partial \mu_{smtj}}{\partial \mu_{smj}} \right) \\ &= \sum_{t=1}^{T} \gamma_{sm}^{\mathrm{mpe}}(t) \psi_{smtj} \left(o_{tj} - \mu_{smtj} \right) \end{aligned} \tag{7.35}$$

$$\begin{aligned} \frac{\partial \mathcal{Q}^{\mathrm{mpe}}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}})}{\partial \psi_{smj}} &= \sum_{t=1}^{T} \left(\frac{\partial \mathcal{Q}^{\mathrm{mpe}}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}})}{\partial \psi_{smtj}} \frac{\partial \psi_{smtj}}{\partial \psi_{smj}} \right) \\ &= \frac{1}{2} \sum_{t=1}^{T} \gamma_{sm}^{\mathrm{mpe}}(t) \left\{ \frac{1}{\psi_{smtj}} - \left(o_{tj} - \mu_{smtj} \right)^2 \right\} z_{tjj}^2 \end{aligned} \tag{7.36}$$

Equating these to zero yields the update formulae for the jth element of the mean and variance as

$$\mu_{smj} = \frac{x_{smj}^{\mathtt{mpe}}}{\tilde{\beta}_{smj}^{\mathtt{mpe}}} \qquad \text{and} \qquad \sigma_{smj}^2 = \frac{1}{\psi_{smj}} = \frac{w_{smj}^{\mathtt{mpe}}}{\beta_{sm}^{\mathtt{mpe}}} \tag{7.37}$$

where the sufficient statistics are given by

$$\beta_{sm}^{\mathtt{mpe}} = \sum_{t=1}^{T} \gamma_{sm}^{\mathrm{mpe}}(t) \tag{7.38}$$

$$\tilde{\beta}_{smj}^{\mathtt{mpe}} = \sum_{t=1}^{T} \gamma_{sm}^{\mathrm{mpe}}(t) z_{tjj}^2 \tag{7.39}$$

$$x_{smj}^{\mathtt{mpe}} = \sum_{t=1}^{T} \gamma_{sm}^{\mathrm{mpe}}(t) z_{tjj}^2 \left(o_{tj} - b_{tj} \right) \tag{7.40}$$

$$w_{smj}^{\mathtt{mpe}} = \sum_{t=1}^{T} \gamma_{sm}^{\mathrm{mpe}}(t) z_{tjj}^2 \left(o_{tj} - b_{tj} - \mu_{smj} \right)^2 \tag{7.41}$$

Note that $\beta_{sm}^{\mathtt{mpe}}$ is the already accumulated in the standard HMM parameters update. $x_{smj}^{\mathtt{mpe}}$ and $w_{smj}^{\mathtt{mpe}}$ are the jth element of the mean and covariance matrix statistics given by equations (5.27) and (5.29) respectively, with the exception that the component posterior is scaled by z_{tjj}^2 and the observation is shifted by b_{tj} for each dimension j. The additional statistics required is the d-dimensional $\tilde{\beta}_{smj}^{\mathtt{mpe}}$.

7.3.2 Dynamic Parameters Estimation

Having estimated the static parameters, the dynamic parameters may be estimated by keeping the static parameters constant. Here, the update of the centroid specific bias, $b_j^{(i)}$, and scaling factor, $z_j^{(i)}$, for the jth element of the mean vector and precision matrix will be described. Due to the large number of posteriors (ranging from thousands to hundreds of thousands), it is not feasible to accumulate the full second order statistics. Thus, a simple gradient optimisation approach proposed in [97] will be used. An important quantity to be calculated is the gradient of the weak-sense auxiliary function with respect to the dynamic parameters, $b_j^{(i)}$ and $z_j^{(i)}$ for all i. These gradients are given by

$$\frac{d\mathcal{Q}^{\mathrm{mpe}}}{db_j^{(i)}} = \sum_{t=1}^{T}\sum_{s=1}^{S}\sum_{m=1}^{M}\frac{d\mathcal{Q}_{smt}^{\mathrm{mpe}}}{db_j^{(i)}} \quad \text{and} \quad \frac{d\mathcal{Q}^{\mathrm{mpe}}}{dz_j^{(i)}} = \sum_{t=1}^{T}\sum_{s=1}^{S}\sum_{m=1}^{M}\frac{d\mathcal{Q}_{smt}^{\mathrm{mpe}}}{dz_j^{(i)}} \tag{7.42}$$

where $\mathcal{Q}_{smt}^{\mathrm{mpe}} = \beta_{sm}^{\mathtt{mpe}}\mathcal{L}^{\mathrm{sm}}(\boldsymbol{o}_t)$ and

$$\frac{d\mathcal{Q}_{smt}^{\mathrm{mpe}}}{db_j^{(i)}} = \frac{\partial\mathcal{Q}_{smt}^{\mathrm{mpe}}}{\partial b_j^{(i)}} + \frac{\partial\mathcal{Q}_{smt}^{\mathrm{mpe}}}{\partial\mu_{smj}}\frac{\partial\mu_{smj}}{\partial b_j^{(i)}} + \frac{\partial\mathcal{Q}_{smt}^{\mathrm{mpe}}}{\partial\sigma_{smj}^2}\frac{\partial\sigma_{smj}^2}{\partial b_j^{(i)}} \tag{7.43}$$

$$\frac{d\mathcal{Q}_{smt}^{\mathrm{mpe}}}{dz_j^{(i)}} = \frac{\partial\mathcal{Q}_{smt}^{\mathrm{mpe}}}{\partial z_j^{(i)}} + \frac{\partial\mathcal{Q}_{smt}^{\mathrm{mpe}}}{\partial\mu_{smj}}\frac{\partial\mu_{smj}}{\partial z_j^{(i)}} + \frac{\partial\mathcal{Q}_{smt}^{\mathrm{mpe}}}{\partial\sigma_{smj}^2}\frac{\partial\sigma_{smj}^2}{\partial z_j^{(i)}} \tag{7.44}$$

Equations (7.43) and (7.44) represent the *complete* differential of $\mathcal{Q}_{smt}^{\mathrm{mpe}}$ with respect to $b_j^{(i)}$ and $z_j^{(i)}$ respectively. In addition to finding the direction that maximises $\mathcal{Q}_{smt}^{\mathrm{mpe}}$, the last two terms in the right hand side of equations (7.43) (referred to as the *indirect* differentials in [97]) and (7.44) also take into account the fact that the global shifting and scaling of the mean should be reflected by updating the static parameters. The partial differentials in the above equations are given by

$$\frac{\partial\mathcal{Q}_{smt}^{\mathrm{mpe}}}{\partial b_j^{(i)}} = \frac{h_i(t)\gamma_{sm}^{\mathrm{mpe}}(t)(\boldsymbol{o}_t - \mu_{smtj})}{\sigma_{smj}^2} \tag{7.45}$$

$$\frac{\partial\mathcal{Q}_{smt}^{\mathrm{mpe}}}{\partial z_j^{(i)}} = \frac{h_i(t)\gamma_{sm}^{\mathrm{mpe}}(t)(1 - \psi_{smtj}(o_{tj} - \mu_{smtj})^2)}{\psi_{smtj}} \tag{7.46}$$

$$\frac{\partial\mathcal{Q}_{smt}^{\mathrm{mpe}}}{\partial\mu_{smj}} = \frac{\left(x_{smj}^{\mathtt{n}} - x_{smj}^{\mathtt{d}}\right)}{\sigma_{smj}^2} \tag{7.47}$$

$$\frac{\partial\mathcal{Q}_{smt}^{\mathrm{mpe}}}{\partial\sigma_{smj}^2} = \frac{(w_{smj}^{\mathtt{n}} - w_{smj}^{\mathtt{d}})/\sigma_{smj}^2 - \beta_{sm}^{\mathtt{mpe}}}{2\sigma_{smj}^2} \tag{7.48}$$

where $x^{\mathtt{n}}_{smj}$ and $w^{\mathtt{n}}_{smj}$ are the jth element of the MPE sufficient numerator statistics $\boldsymbol{x}^{\mathtt{n}}_{sm}$ and $\boldsymbol{W}^{\mathtt{n}}_{sm}$ respectively and similarly for the denominator statistics $x^{\mathtt{d}}_{smj}$ and $w^{\mathtt{d}}_{smj}$ (see Section 5.3.1). The actual forms of the remaining differentials $\frac{\partial \mu_{smj}}{\partial b^{(i)}_j}$, $\frac{\partial \sigma^2_{smj}}{\partial b^{(i)}_j}$, $\frac{\partial \mu_{smj}}{\partial z^{(i)}_j}$ and $\frac{\partial \sigma^2_{smj}}{\partial z^{(i)}_j}$ depend on the update methods for the static parameters, μ_{smj} and σ^2_{smj}. Ideally, MPE update of the static parameters is preferred. Unfortunately, the use of the D-smoothing and the I-smoothing with dynamic ML (or dynamic MMI) priors in standard MPE training [100] as described in Chapter 5 complicates the calculation of the *indirect* differentials. In the following, two simpler forms of update are described.

7.3.2.1 Interleaved Dynamic-Static Parameters Estimation

Simultaneous update of the static and dynamic parameters does not yield a closed form solution. Instead, a more efficient way of updating these parameters is to adopt an interleaved update where the static and dynamic parameters are updated as described in Sections 7.3.1 and 7.3.2 in an alternating fashion. As previously mentioned in Section 7.3.2, the dynamic parameters are updated using a gradient optimisation approach where the *complete* differential of the auxiliary function with respect to the dynamic parameters needs to be calculated. This requires the differentials $\frac{\partial \mu_{smj}}{\partial b^{(i)}_j}$, $\frac{\partial \sigma^2_{smj}}{\partial b^{(i)}_j}$, $\frac{\partial \mu_{smj}}{\partial z^{(i)}_j}$ and $\frac{\partial \sigma^2_{smj}}{\partial z^{(i)}_j}$ to be defined according to the update formulae for μ_{smj} and σ^2_{smj}. Using the MPE update formulae complicates the calculation of the complete differential. Therefore, the ML update formulae are used, as proposed by *Povey et al.* in [97].

This approach takes a Maximum Likelihood (ML) trained model and applies the dynamic parameter update (Section 7.3.2) to estimate the $b^{(i)}_j$ and $z^{(i)}_j$. Next, the standard HMM parameters are ML estimated using the updated dynamic parameters. Repeating these two steps yields an interleaving update for the dynamic and static parameters. The interleaved parameter estimation procedure is summarised as follows:

```
1.  Start from an ML trained model
2.  Estimate dynamic parameters using MPE criterion
3.  Estimate static parameters using ML criterion
4.  If converged or sufficient iterations performed, go to step 6
5.  Go to step 2
6.  Estimate static parameters using MPE criterion
```

Figure 7.3: *The interleaved dynamic static parameter estimation procedure*

The static parameters are updated using the ML criterion by keeping the dynamic parameters constant, as described in Section 7.3.1. The dynamic model parameters are then estimated using the gradient in equations (7.43) and (7.44). Because the static parameters are to be updated using the ML criterion in the subsequent training iteration, the partial differential of the mean

and variance with respect to the dynamic parameters are evaluated using equations in (7.37) as

$$\frac{\partial \mu_{smj}}{\partial b_j^{(i)}} = -\frac{h_i(t)\gamma_{sm}^{\mathrm{ml}}(t)}{\tilde{\beta}_{smj}^{\mathtt{ml}}} \tag{7.49}$$

$$\frac{\partial \sigma_{smj}^2}{\partial b_j^{(i)}} = -\frac{2h_i(t)z_{tjj}\gamma_{sm}^{\mathrm{ml}}(t)(o_{tj}-\mu_{smtj})}{\beta_{sm}^{\mathtt{ml}}} \tag{7.50}$$

$$\frac{\partial \mu_{smj}}{\partial z_j^{(i)}} = \frac{2z_{tjj}\gamma_{sm}^{\mathrm{ml}}(t)(o_{tj}-\mu_{smtj})}{\tilde{\beta}_{smj}^{\mathtt{ml}}}\left(h_i(t) - \frac{z_{tjj}^2\gamma_{sm}^{\mathrm{ml}}(t)}{\tilde{\beta}_{smj}^{\mathtt{ml}}}\right) \tag{7.51}$$

$$\frac{\partial \sigma_{smj}^2}{\partial z_j^{(i)}} = \frac{2h_i(t)z_{tjj}\gamma_{sm}^{\mathrm{ml}}(t)(o_{tj}-\mu_{smtj})^2}{\beta_{sm}^{\mathtt{ml}}} \tag{7.52}$$

When $z_{tjj}=1$, the equations (7.49) and (7.50) become those of the standard fMPE presented in [97]. Once the gradient information is computed, the dynamic parameters are updated as follows:

$$\hat{b}_j^{(i)} = b_j^{(i)} + \eta_j^{(i)}\frac{d\mathcal{Q}^{\mathrm{mpe}}}{db_j^{(i)}} \quad \text{and} \quad \hat{z}_j^{(i)} = z_j^{(i)} + \nu_j^{(i)}\frac{d\mathcal{Q}^{\mathrm{mpe}}}{dz_j^{(i)}} \tag{7.53}$$

where $\hat{b}_j^{(i)}$ and $\hat{z}_j^{(i)}$ denote the updated parameters for $b_j^{(i)}$ and $z_j^{(i)}$ respectively. $\eta_j^{(i)}$ and $\nu_j^{(i)}$ are the element specific learning rate for $b_j^{(i)}$ and $z_j^{(i)}$ and are defined as

$$\eta_j^{(i)} = \frac{\alpha\bar{\sigma}_j}{\phi_{ij}^{(b)}+\rho_{ij}^{(b)}} \quad \text{and} \quad \nu_j^{(i)} = \frac{\alpha}{\phi_{ij}^{(z)}+\rho_{ij}^{(z)}} \tag{7.54}$$

where α is a scalar parameter for adjusting the learning rate and $\bar{\sigma}_j$ is the average standard deviation of the Gaussian components in the system. $\phi_{ij}^{(b)}$ and $\rho_{ij}^{(b)}$ are the sum of the positive and negative contributions to the gradient of $\mathcal{Q}_{smt}^{\mathrm{mpe}}$ with respect to $b_j^{(i)}$ at each time, t, as presented in [97]. Similar calculation is used for $\phi_{ij}^{(z)}$ and $\rho_{ij}^{(z)}$. Hence,

$$\phi_{ij}^{(b)} = \sum_{t=1}^{T}\max\left\{\sum_{s=1}^{S}\sum_{m=1}^{M}\frac{d\mathcal{Q}^{\mathtt{mpe}}}{db_j^{(i)}},0\right\} \quad \text{and} \quad \rho_{ij}^{(b)} = \sum_{t=1}^{T}\max\left\{-\sum_{s=1}^{S}\sum_{m=1}^{M}\frac{d\mathcal{Q}^{\mathtt{mpe}}}{db_j^{(i)}},0\right\}$$

$$\phi_{ij}^{(z)} = \sum_{t=1}^{T}\max\left\{\sum_{s=1}^{S}\sum_{m=1}^{M}\frac{d\mathcal{Q}^{\mathtt{mpe}}}{dz_j^{(i)}},0\right\} \quad \text{and} \quad \rho_{ij}^{(z)} = \sum_{t=1}^{T}\max\left\{-\sum_{s=1}^{S}\sum_{m=1}^{M}\frac{d\mathcal{Q}^{\mathtt{mpe}}}{dz_j^{(i)}},0\right\}$$

Usually, standard MPE training of the static parameters are performed to complete the discriminative training of all the parameters. This is achieved using the modified update equations in (7.37), replacing the ML statistics with the corresponding MPE statistics. Further MPE training on top of fMPE and pMPE is this way will be referred to as fMPE+MPE and pMPE+MPE respectively.

7.3.2.2 Direct Dynamic Parameters Estimation

The estimation method described in 7.3.2.1 requires the *complete* differential to take into account of the change in the model parameters in the subsequent ML training. If only the *partial*

differential is considered, the gain from fMPE and pMPE disappears as soon as the static model parameters are updated [97]. This is expected because the interleaved estimation alternates between the use of two different objective functions. When the dynamic parameters are updated using the MPE criterion without taking into account of the *complete* differentials, the estimation process tends to drive the dynamic parameters such that the effective mean vectors and precision matrices are closer to the MPE estimates. This tendency is due to the fact that the initial model parameters are ML trained and a large gain may be obtained by adjusting the dynamic parameters to model static aspect of the system. So, the subsequent ML training of the static parameters will virtually unlearn the discriminative power from the dynamic parameters. Therefore, to ensure that the dynamic parameters are in fact modelling the temporal aspects of the system, it is essential to consider the *complete* differentials. However, computing the *complete* differential requires two passes over the training data. The first pass accumulates the normal MPE statistics $\left(x^{\mathtt{n}}_{smj}, x^{\mathtt{d}}_{smj}, w^{\mathtt{n}}_{smj}, w^{\mathtt{d}}_{smj}, \beta^{\mathtt{n}}_{sm} \text{ and } \beta^{\mathtt{d}}_{sm}\right)$ required by equations (7.47) and (7.48). The second pass then computes the gradients in equations (7.43) and (7.44).

The training time can be reduced if the starting HMM is a well trained MPE model. In this case, the differentials in equations (7.47) and (7.48) will have values small enough that can be safely approximated as zero. This conveniently eliminates the need to accumulate the normal MPE statistics. Moreover, no subsequent reestimation of the static parameters is required. Hence, fMPE and pMPE can be estimated with only a single pass over the training data. These systems are referred to as MPE+fMPE and MPE+pMPE respectively.

```
1.  Start from an ML trained model
2.  Estimate static parameters using MPE criterion
3.  Estimate dynamic parameters using MPE criterion
```

Figure 7.4: *The direct dynamic parameter estimation procedure*

Experimental results in Chapter 8 shows that similar gains may be achieved using this quicker training scheme. However, when context expansions (context expansions will be described in greater detail in Section 7.4) are considered, the direct dynamic parameter update approach gave a much smaller gain compared to the interleaved update described earlier. One explanation to this is that the use of context expansion may have altered the system dynamics considerably and the static parameters are drifted further from their optimum values. Consequently, the zero gradient assumptions for equations (7.47) and (7.48) no longer hold.

7.4 Context Expansion for Semi-parametric Trajectory Model

The original formulation of fMPE also allows the features to be modified depending on the posterior vectors from surrounding time frames. This approach is termed context expansion

in [101]. In the semi-parametric trajectory model formulation, context expansion can be viewed as increasing the modelling power of the trajectory. Recall from the earlier discussion on formulating a trajectory model, one could model the output probability distribution such that it is conditional upon the entire observation sequence (see equation (7.6)). So, there is a question of how many observation vectors to be considered at any given time t? All the discussions so far have been considering only the observation vector at the current time, t. It is possible to extend the dependency to a window of observations around t to allow for context expansion. Equations (7.9), (7.10) and (7.11) may be expressed in a more generic form as follows:

$$\boldsymbol{A}_t = \boldsymbol{I} + \sum_{\tau=-C}^{C} w(\tau) \sum_{i=1}^{n} h_i(t+\tau) \boldsymbol{A}_{\tau}^{(\mathtt{i})} \tag{7.55}$$

$$\boldsymbol{b}_t = \sum_{\tau=-C}^{C} w(\tau) \sum_{i=1}^{n} h_i(t+\tau) \boldsymbol{b}_{\tau}^{(\mathtt{i})} \tag{7.56}$$

$$\boldsymbol{Z}_t = \boldsymbol{I} + \sum_{\tau=-C}^{C} w(\tau) \sum_{i=1}^{n} h_i(t+\tau) \boldsymbol{Z}_{\tau}^{(\mathtt{i})} \tag{7.57}$$

where $w(\tau)$ is the window function of length $2C+1$, i.e. considering C frames on either side of the current frame. C can be viewed as the context of the trajectory. The window function used in this work follows is the same as that introduced in [97], where

$$w(\tau) = \begin{cases} 1 & \tau = 0 \\ \frac{1}{2} & \tau = \pm 1, \pm 2 \\ \frac{1}{3} & \tau = \pm 3, \pm 4, \pm 5 \\ \frac{1}{4} & \tau = \pm 6, \pm 7, \pm 8, \pm 9 \\ \vdots & \\ \frac{1}{N} & \tau = \pm \frac{N(N-1)}{2}, \pm \left(\frac{N(N-1)}{2} + 1 \right), \ldots, \pm \left(\frac{N(N-1)}{2} + N - 1 \right) \end{cases} \tag{7.58}$$

and

$$C = \left(\frac{N(N-1)}{2} + N - 1 \right) \tag{7.59}$$

Consequently, when $N = 4$, $C = 9$ and the number of dynamic parameters will be 19 times more than those without context expansion. To avoid over-training, the dynamic parameters are tied across frames [1,2], [3,4,5] and [6,7,8,9] to the left and right of the current frame, according to the partitions shown in equation (7.58). This is equivalent to taking the average posteriors within the partitions so that the true expansion in terms of the dynamic parameters is only ± 3 (7 times more than that without context expansion). In Povey's later work on fMPE [96], a layered transform configuration was investigated where the window function was considered an intermediate transform layer whose parameters were also learned to maximise the MPE criterion. This approach was reported to yield slight performance gain.

One interesting question arises from context expansion is that when no context expansion is used, is the resulting model still a trajectory model? The answer is *yes*. The key point lies

in the fact that the trajectory is modelled in a semi-parametric way by tracking the position of the acoustic vector at each time using a set of centroids representing the acoustic space. Context expansion merely extends the modelling power of the trajectory model at the expense of increased model parameters.

7.5 Jacobian Compensation

The above mentioned semi-parametric trajectory model, fMPE in particular, involves modification to the features to maximise the MPE objective function. When modelling a feature transformation process, a crucial aspect to be considered is the determinant of the Jacobian matrix, which is given by

$$J = \begin{vmatrix} \frac{\partial \hat{o}_{t1}}{\partial o_{t1}} & \frac{\partial \hat{o}_{t1}}{\partial o_{t2}} & \cdots & \frac{\partial \hat{o}_{t1}}{\partial o_{td}} \\ \frac{\partial \hat{o}_{t2}}{\partial o_{t1}} & \frac{\partial \hat{o}_{t2}}{\partial o_{t2}} & \cdots & \frac{\partial \hat{o}_{t2}}{\partial o_{td}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \hat{o}_{td}}{\partial o_{t1}} & \frac{\partial \hat{o}_{td}}{\partial o_{t2}} & \cdots & \frac{\partial \hat{o}_{td}}{\partial o_{td}} \end{vmatrix} \tag{7.60}$$

where o_{tj} and $\hat{o}_{tj}$ are the jth elements of the original and modified observation vectors respectively. The quantity J is often referred to as simply the Jacobian. Roughly speaking, the Jacobian represents the relative change in the new features with respect to the original features. This has to be compensated in the likelihood expression so that the likelihood is computed consistent with respect to the original features. Jacobian compensation may be ignored for the MPE objective function because it is a rational function of two likelihood expressions (numerator and denominator). Since the Jacobian compensation terms of the numerator and denominator are identical, the effect of the Jacobian cancels out. This is not true, however, when ML training is to be used. In [126], fMPE was adopted as an speaker adaptation technique, where the parameters, for reasons of efficiency, are estimated using the ML criterion with Jacobian compensation.

7.6 Relationship with Linear Adaptation

Equations (7.7) and (7.8) resemble the linear adaptation equations given in equations (6.1) and (6.2) respectively. Therefore, the above semi-parametric trajectory model may also be viewed as performing a time-dependent linear adaptation to the model parameters. Associated with each centroids are the adaptation transforms ($\boldsymbol{A}^{(\mathrm{i})}$, $\boldsymbol{Z}^{(\mathrm{i})}$ and $\boldsymbol{b}^{(\mathrm{i})}$). The transformation matrices at each time being constructed from superimposing a set of bases weighted by time-dependent coefficients. If the bases associated with each centroid are speaker dependent, the model may be viewed as an attempt to perform a two-fold adaptation, over both speakers and time. Similar to the constrained MLLR (CMLLR) formulation presented in Chapter 6, the time dependent

transformations for the mean vectors and covariance matrices may also be constrained to be the same. This yields the time varying feature transformation (or fMPE) as described in the previous section.

7.7 Implementation Issues

The implementation issues for fMPE have already been described in [97]. This section will concentrate on the issues relating to the pMPE implementation. First of all, the likelihood computation of fMPE and pMPE models will be examined. Since fMPE is a feature transformation technique, there is no additional cost for the likelihood calculation of fMPE model compared to the standard HMM system. For pMPE there is a slight increase in this cost. The likelihood of the model parameters, $\boldsymbol{\theta}^m$ given the observation vector, $\boldsymbol{o}_t$, is given by

$$\mathcal{L} = \log p(\boldsymbol{o}_t|\boldsymbol{\mu}_{sm}, \boldsymbol{P}_{sm}) = K + \frac{1}{2}\sum_{j=1}^{d}\left\{\log z_{tjj} - \log \sigma^2_{smj} - \frac{z_{tjj}(o_{tj} - \mu_{smj})^2}{\sigma^2_{smj}}\right\} \tag{7.61}$$

This requires an extra d multiplications and 1 addition compared to the standard model. It also requires z_{tjj} and $\sum_{j=1}^{d} \log z_{tjj}$ to be cached for each frame, t.

Unlike fMPE, pMPE parameter estimation is less reliable and is more likely to be over-trained, particularly when a higher learning rate is used ($\alpha > 1.0$). In such a case, the resulting temporally varying scale, z^2_{tjj} may tend to a value close to zero. To prevent this, a minimum value is applied to z_{tjj}, similar to the concept of variance flooring:

$$\tilde{z}_{tjj} = \max\{z_{tjj}, z_{\texttt{min}}\} \tag{7.62}$$

where $\tilde{z}_{tjj}$ is the floored scale factor and $z_{\texttt{min}}$ is the scale floor. In this work, $z_{\texttt{min}}$ has been set to 0.1.

As mentioned in [97] and previously discussed, the update of the dynamic parameters should not result in a *global* shift or scale in the acoustic space. This provides convenient checks against any implementation errors [97]. Similar checks can also be carried out for pMPE implementation by ensuring that the gradient in equation (7.44) equals to zero when there is a global precision scaling. In other words, when there is only one centroid in the system (not modelling the trajectory), the differentials should equal zero and no dynamic parameters update should be performed. So, $h_1(t) = 1$ and

$$0 = \sum_{s=1}^{S}\sum_{m=1}^{M}\sum_{t=1}^{T}\left(\frac{\partial \mathcal{Q}^{\text{mpe}}_{smt}}{\partial z^{(i)}_j}\bigg|_{h_1(t)=1} + \frac{\partial \mathcal{Q}^{\text{mpe}}_{smt}}{\partial \sigma^2_{smj}}\frac{\partial \sigma^2_{smj}}{\partial z^{(i)}_j}\bigg|_{h_1(t)=1}\right) \tag{7.63}$$

$$0 = \sum_{s=1}^{S}\sum_{m=1}^{M}\sum_{t=1}^{T}\frac{\partial \mathcal{Q}^{\text{mpe}}_{smt}}{\partial \mu_{smj}}\frac{\partial \mu_{smj}}{\partial z^{(i)}_j}\bigg|_{h_1(t)=1} \tag{7.64}$$

Equations (7.63) and (7.64) ensure that the dynamic parameters update will not result in a global scaling of the precision matrices. A proof of equations (7.63) and (7.64) is supplied in Appendix G.

7.8 Summary

This chapter has introduced a discriminative semi-parametric trajectory model. In this model, the state output probability density function is represented by a Gaussian Mixture Model (GMM) where the Gaussian mean vector and the diagonal covariance matrix varies with time. The time dependency is modelled as a smoothed function of the observation sequence using a basis superposition formulation. Each basis is associated with a centroid representing a position in the acoustic space. The corresponding basis coefficients are derived from the posterior of its centroid given the observation at time t and the surrounding observations. Hence, the basis coefficients are time dependent which results in a temporally varying model parameters. In this chapter, it was also shown this semi-parametric trajectory model is the same as the fMPE technique if only the mean vectors are being modelled. In addition, a novel approach of pMPE was also introduced where the precision matrix elements are modelled as temporally varying parameters. In chapter 9, the experimental results of the semi-parametric trajectory models will be presented.

8

Experimental Results of Precision Matrix Models

This chapter presents the experimental results of various precision matrix modelling techniques described in Chapters 3 to 7. First, the experimental setups will be described for the Conversational Telephone Speech (CTS) English (`CTS-E`), Broadcast News (BN) English (`BN-E`) and CTS Mandarin (`CTS-M`) tasks. This includes a brief description of these tasks, the training and test corpora used to develop the models, the system configurations used in the experiments as well as the evaluation setups. Next, the experimental results will be presented in two major sections. The first section corresponds to the investigation of various structured precision matrix approximation schemes as detailed in Chapters 3 to 6. In particular, the experimental results for ML and MPE trained STC, EMLLT and SPAM systems will be compared. The second section of the results are based on the semi-parametric trajectory model described in Chapter 7.

8.1 Experimental Setups

Two major transcription tasks were considered in this work, namely, the conservation telephone speech (CTS) and broadcast news (BN). Both CTS and BN speech and text corpora are provided by the LDC. The CTS speech corpora consist of telephone conversations recorded at a sampling frequency of 8 kHz and sample size of 16 bits. The speech data were collected in two different ways. The earlier data collection, known as the `switchboard` data, are based on daily telephone conversations between relatives and friends. A new data collection method called `fisher` was employed several years ago. For each conversation, two random subjects were invited to talk on a specific topic. In general, the `fisher` style data was found to be easier to transcribe. The BN speech data, on the other hand, are collected from several American radio and television shows broadcasting news. These data may originate from two different channel bandwidths: wide band data ($\sim$16 kHz) or narrow band data ($\sim$8 kHz). This work concentrates on building

acoustic models based on the wide band data only. A baseline acoustic model was used to transcribe the narrow band data for all the BN experiments. This work also considered two different languages for the transcription tasks, English and Mandarin. Therefore, three different transcription tasks were used in the experiments described in the rest of this chapter. They were the CTS-E, BN-E and CTS-M tasks.

8.1.1 Training and Test Corpora

Here, a brief summary of various speech corpora used in the acoustic model building will be provided. Tables 8.1 shows the training corpora used in various transcription tasks. Four

Task	Corpus	Data source	Transcription Quality	Data size (hours)
CTS-E	h5etrain03sub	switchboard	Carefully annotated	76
	h5etrain03	switchboard		298
	fsh2004sub	fisher		400
	fsh2004h5etrain03b	switchboard+fisher		2180
BN-E	bnac	broadcast news	Carefully annotated	143
	tdt4	broadcast news	Lightly supervised	231
CTS-M	swm03	switchboard	Carefully annotated	32
	ldc04	fisher		40

Table 8.1 *Summary of various speech training corpora for* CTS-E*,* BN-E *and* CTS-M

sets of training corpora were used for the CTS-E task. h5etrain03sub is a 76-hour subset of h5etrain03, both are switchboard style data. fsh2004sub comprises 400 hours of fisher type data. The largest training corpus used in this work was the combination of fisher (1820 hours of fsh2004) and switchboard (360 hours of h5etrain03b) data, which yielded approximately 2180 hours of training data in total. The two training sets used for the BN-E task were the 143 hours of carefully annotated bnac data and 231 hours of lightly supervised tdt4 data. Finally, the 32 hours swm03 and the 40 hours ldc04 were used in the CTS-M task. It is worth noting that these two corpora were considerably different in terms of their sources and data collection environment. swm03 consists of switchboard style telephone conversation within the North America calling their families in the Mainland China (CallHome) and friends within North America itself (CallFriend). On the other hand, ldc04 was a newer set of data collected by the Hong Kong University of Science and Technology (HKUST). Data collection was done in common with the Fisher English data and random subjects were recruited in several cities across mainland China. To encourage more meaningful conversation, predefined topics similar to those in Fisher English were designed to initiate conversations.

The testing data were the development and evaluation data sets provided by the LDC for

Task	Corpus	Data source	Epoch	Data size (hours)
CTS-E	dev01sub	switchboard	—	3
	eval03	switchboard+fisher		6
	dev04	fisher		3
BN-E	dev03	broadcast news	Jan, 2001	3
	eval03	broadcast news	Feb, 2001	3
	dev04	broadcast news	Jan, 2001	3
	dev04f	broadcast news	Nov & Dec, 2003	3
CTS-M	eval03	switchboard	—	1
	dev04	fisher		2
	eval04	fisher		1

Table 8.2 *Summary of various speech testing corpora for* CTS-E*,* BN-E *and* CTS-M

DARPA evaluation programmes between 2001 and 2004. A summary of the testing corpora used in this work is given in Table 8.2. Three test sets were used in the CTS-E experiments. dev01sub, a 3 hours subset of dev01, comprises switchboard style data. eval03 consists of both switchboard and fisher styles data, 3 hours each. dev04 was a 3 hours fisher data. Four different test sets were used in the BN-E task: dev03, eval03, dev04 and dev04f, three hours each. Part of the dev04f test set contains data from different sources and epoch compared to the other three sets and the training corpora. Hence, the performance evaluated on this test set is poorer and adaptively trained systems were found to yield larger gains on this particular test set (see later for more results). Finally, in CTS-M task, the eval03 and dev04 test sets were used. The nature of these two test sets correspond to the swm03 and ldc04 training sets respectively.

8.1.2 System configurations

The experiments conducted for this work were based primarily on the design and configurations of the recent CU-HTK evaluation systems for CTS-E [22, 23, 24, 54], BN-E [65, 66, 67] and CTS-M [37, 38]. The aim was to investigate the performance of various precision matrix model systems on LVCSR systems and compare that with state-of-the-art CU-HTK systems. All the acoustic model training and evaluation were performed using the Hidden Markov Model Toolkit (HTK) [134]. The toolkit has been modified and extended to support various forms of structured precision matrix modelling schemes. In the following, various aspects of the system configurations will be described.

8.1.2.1 System Frontends

All acoustic models in the experiments of this work used the Perceptual Linear Prediction (PLP) coefficients [55] with 12 static coefficients plus the C0 energy. First and second derivatives were also appended to the features to yield a 39 dimensional feature vector. This is the basic feature used to build the initial model set and perform decision tree state-clustered context-dependent models. Almost all the acoustic models used in this work were based on the 39 dimensional HLDA projection features. This is obtained by training a (39×52) HLDA [68] projection matrix using the ML criterion within a 52 dimensional space (including the third derivative coefficients). For the CTS tasks, conversations may be easily partitioned according to the speaker turns (also known as speaker sides). Cepstral Mean Normalisation (CMN) [3], Cepstral Variance Normalisation (CVN) and Vocal Tract Length Normalisation (VTLN) [69, 115] were applied to the each side. In contrast to CTS, speaker turns may not be identified reliably for the BN tasks. System diarisation (this includes advertisement and music removal, clustering speech segments according to speakers and performing gender detection) [114, 120]. Thus, only segment based CMN was employed.

The frontend used in the CTS Mandarin task is slightly different. Unlike English, Mandarin is a tonal language. To improve the recognition performance, pitch information and its derivatives were appended to the feature vector. Thus, for the original 52 dimensional PLP feature (including the first three derivatives), adding pitch information yielded 56 dimensional features. Since the dynamics of the pitch may be different from those of the PLP coefficients, HLDA was only performed on the 52-dimensional PLP coefficients. Pitch information (together with its first and second derivatives) were appended to the HLDA feature to give the final 42 dimensional features for the tonal HLDA systems. Moreover, since the training data size for the CTS-M task is relatively small (see Table 8.1), Gaussianisation technique was also employed to improve the normalisation of the frontend.

8.1.2.2 Acoustic Models (HMMs)

In this work, hidden Markov models (HMMs) were used to model the basic speech units – *phonemes*. Triphone context-dependent models were used with decision tree state clustering [8, 135]. An iterative mixture splitting approach [134] was used to train a Gaussian Mixture Model distribution for each distinct state in the system. Most systems had a variable number of Gaussian mixture components per state. The number of Gaussian components for each state is made proportional to the state occupancy count raised to a power, in this work 0.2. These systems, known as VarMix, was found to yield slight performance gain compared to systems with the same number of Gaussians but using the same number of Gaussian components for all HMM states. In general ML training was performed in four Baum-Welch iterations while MPE training was performed in eight iterations.

8.1.2.3 Language Models

In all the experiments, standard N-gram language models were used. Table 8.3 summarises the set of language models used in various experiments. All the language models were built by

Transcription Task	Language Model	Training data Size (words)	Vocabulary Size (words)	# of N-grams		
				2	3	4
CTS-E	swe03.58k	1044M	58k	6.0M	9.4M	5.9M
	swe04.58k	489M	58k	9.1M	15.5M	7.8M
BN-E	bne59k	1018M	59k	8.8M	12.7M	6.6M
CTS-M	swm03.11k	151.2M	11k	3.9M	6.5M	—
	ldc04.16k	150.8M	16k	6.4M	9.8M	—

Table 8.3 *Summary of language models used in various tasks*

interpolating various components, each was trained from different data sources. For each type of language model, a bigram, trigram or 4-gram language model may be used depending on the evaluation configurations (described later in this section). Two types of language models were used in the CTS-E task. The swe03.58k language models were used for evaluation on the dev01sub test sets while swe04.58k language models were used for evaluation on the eval03 and dev04 test sets. The swe03.58k and swe04.58k language models both had a vocabulary size of 58k words and were used in the DARPA Rich Transcription 2003 (RT03)and 2004 (RT04) evaluations respectively [22, 24]. Multiple pronunciations were used for the CTS-E experiments. For the BN-E task, only one type of language model (bne59k) with 59k vocabulary size was used. The experiments for the CTS-M task also employed two different language models. Both language models were trained on similar data sources with the exception that the ldc04 acoustic training transcription data was excluded from the swm03.11k language model training. These language models also employed different vocabularies. The 11k vocabulary covers all the words that occurred in the swm03 training data. This was used in the RT03 CU-HTK Mandarin system. For the RT04 CU-HTK system [37, 38], a 16k-word vocabulary was used. This was generated by extending the 11k vocabulary to cover the words occurring in the ldc04 training set as well as approximately 5,000 single character words. The swm03.11k language models was used to evaluate the eval03 test set while the ldc04.16k language model was used for dev04 and eval04.

Most of the evaluations were conducted in a single-pass bigram full decoding using the HTK LVCSR decoder, HDecode, to generate lattices of multiple hypotheses. These lattices were rescored with a trigram language model to yield the final performance. For simplicity, this method is referred to as an unadapted single-pass decoding.

8.2 Initial Experiments

Initial experiments were carried out to determine the training configurations for various precision matrix models. A series of investigations were carried out for the EMLLT training configurations. These include initialisation schemes (Section 8.2.1), number of training iterations (Section 8.2.2) and the comparison between additive and multiplicative basis coefficient update schemes (Section 8.2.3). This is followed by the results of the smoothing constant approximation scheme for the SPAM models in Section 8.2.4. Finally, the effectiveness of structured approximation schemes as a compact precision matrix model representation is also investigated in Section 8.2.5, using a smaller training set, `h5etrain03sub`.

8.2.1 Initialisation Schemes for EMLLT models

As discussed in Section 4.3.2.1, initialisation is important to ensure good starting point that leads to faster convergence. Here, several forms of initialisation schemes introduced earlier were compared for EMLLT. Table 8.4 shows the comparison of these initialisation schemes using ML training. The first 5 rows of the table are based on experiments using 39-dimensional feature vectors (comprising the static parameters, log energy, first and second derivatives) and the final two rows are based on those using 52-dimensional feature vectors (with the additional third derivatives). Four initialisation schemes described in Section 4.3.2.1 were investigated. UI, the

Initialisation	Basis Matrix Dimensions	WER (%)
UI	(78×39)	34.0
STC+HLDA	$\begin{pmatrix} 39 \times 39 \\ 13 \times 39 \end{pmatrix}$	33.4
STC+HLDA	$\begin{pmatrix} 39 \times 39 \\ 26 \times 39 \end{pmatrix}$	33.2
STC+EYE	$\begin{pmatrix} 39 \times 39 \\ 39 \times 39 \end{pmatrix}$	33.1
STC(SS)	$\begin{pmatrix} 39 \times 39 \\ 39 \times 39 \end{pmatrix}$	33.2
STC+HLDA	$\begin{pmatrix} 52 \times 52 \\ 26 \times 52 \end{pmatrix}$	32.6
STC+EYE	$\begin{pmatrix} 52 \times 52 \\ 26 \times 52 \end{pmatrix}$	33.0

Table 8.4 *Average log likelihood and WER (%) performance on* `dev01sub` *for various EMLLT initialisation schemes using ML-trained 16-component EMLLT systems*

uninformative initialisation, is clearly the worst scheme. This method randomly chooses a set of vectors that span the space of a symmetric matrix to form the required basis vectors (see Appendix B). This shows that the system performance is rather sensitive to the initialisation schemes. STC+EYE, STC(SS) and STC+HLDA are better initialisation schemes. For the 39 dimensional feature vectors, the STC+EYE method gave the lowest WER of 33.1% on `dev01sub`. However, the STC+EYE and STC(SS) methods require the total number of bases to be multiple of the feature vector dimension. Otherwise, the bases need to be truncated to give the required basis order. Alternatively, the STC+HLDA allows the initialisation of an arbitrary number of basis vectors. The last two rows of Table 8.4 reveal that the STC+HLDA initialisation scheme is superior to simply truncating the identity matrix (STC-EYE) when the number of basis vectors is not multiple of the feature dimension.

8.2.2 Iteration Numbers for EMLLT models

When the number of basis is greater than the feature dimensionality, there is no closed form solution for the basis updates. Hence, an iterative gradient descent optimisation method has to be used. The effect of the number of iterations on the recognition performance is thus investigated. Figure 8.1 shows the change in the auxiliary function values against each basis vector update

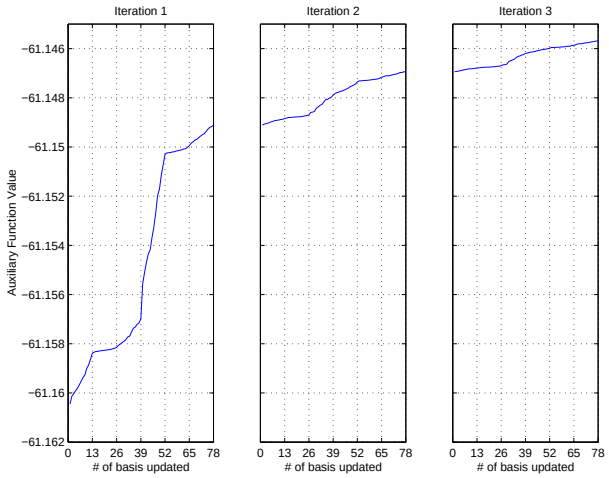

Figure 8.1 *Auxiliary function values vs number of rows updated for 3 basis vector iterations training on* `h5etrain03` *using a 16-component EMLLT system*

for three basis iterations. The basis vectors were initialised using the STC(SS) method where the top and bottom 39 basis vectors where obtained from the STC transform for the speech and silence models respectively. The basis vectors were updated in the *reversed* order. The increase in auxiliary function value on the first iteration is relatively large and rapidly converges on subsequent iterations. Notice the larger increase in the auxiliary function value when updating the

first 13 basis vectors as well as the 40^{th} to 52^{th} basis vectors. Table 8.5 shows the average log

Meta Iterations	Basis Iterations	Average Log Likelihood	WER (%)
1	1	-63.47	33.2
2	1	-63.45	33.3
3	1	-63.43	33.2
1	0	-63.52	33.1
1	2	-63.47	33.3
1	3	-63.46	33.3

Table 8.5 *Average log likelihood and WER (%) on* `dev01sub` *for different number of update iterations using 16-component ML-trained EMLLT systems*

likelihood values and WER for different number of iterations. The meta iteration refers to the outer loop that alternates between the basis vector update and basis coefficient update. The improvement gained from doing multiple iterations is insignificant. This suggest the use of just one meta iteration and one basis iteration for the remaining experiments.

8.2.3 Additive vs. Multiplicative Update for EMLLT models

One powerful feature of the EMLLT model is that it allows the basis coefficients to be negative. To investigate the benefit of having negative basis coefficients, two variants of the basis coefficient update methods were used. The additive update rule allows the coefficients to take any real values while the multiplicative update rule restricts the coefficients take only positive values. Table 8.6 compares the average log likelihood and the WER for these two updates. Clearly, the

Update Method	Average Log likelihood	WER (%)
Additive	-63.47	33.2
Multiplicative	-63.71	33.6

Table 8.6 *Average log likelihood and WER (%) on* `dev01sub` *for additive and multiplicative basis coefficient updates using 16-component ML-trained EMLLT systems*

additive update gave a larger increase in log likelihood value as well as 0.4% absolute reduction in WER compared to the multiplicative update. Thus, the EMLLT models used in subsequent experiments are trained using the additive update.

8.2.4 Smoothing Constant Approximation for SPAM models

In Section 5.3.2, an approximation was used to calculate the D-smoothing constant for MPE training of SPAM models to avoid the need to accumulate the full covariance matrix statistics. To investigate the effect of this approximation, two different SPAM models were trained on `h5etrain03sub` with 4 MPE iterations, using the exact and approximated smoothing constant. The exact method of finding the smoothing constant, D_{sm}, is achieved by starting with a small smoothing constant and gradually increasing its value until the resulting full covariance matrix statistics is positive-definite. On the other hand, the approximated smoothing constant is found by solving the independent quadratic equations correspond to an identity *pseudo* projection matrix. Results from Table 8.7 reveal that the approximation of smoothing constant using the

System	Smoothing Constants	WER (%)				
		0	1	2	3	4
SPAM	Exact	33.3	32.4	31.9	31.9	31.7
	Approximate	33.3	32.3	32.0	31.9	31.7

Table 8.7 *WER (%) performance on* `dev01sub` *of 12-component SPAM systems trained on* `h5etrain03sub` *using exact and approximated smoothing constant*

pseudo projection matrix yields almost similar WER performance compared to exact calculation. In theory, using a STC *pseudo* projection matrix should yield better approximation, but the above results suggest that the use of identity matrix was a sufficient approximation.

8.2.5 Model Training on Small Data Set

To investigate the robustness of the precision matrix models, a smaller training set is used to train the models. Table 8.8 compares the WER performance of 16-component HLDA, EMLLT and

System	Dimension		WER (%)		
	$\boldsymbol{\mu}$	$\boldsymbol{\Sigma}$	ML	MPE(4)	MPE(8)
HLDA	39	39	34.2	32.4	32.2
EMLLT	52	78	33.6	31.9	31.9
SPAM	52	39	33.3	31.7	31.4

Table 8.8 *Comparison of performance for models trained on* `h5etrain03sub`

SPAM models trained on `h5etrain03sub`. The WER of the baseline HLDA ML model is 34.2%. After 4 and 8 MPE training iterations, the WER reduced to 32.4% and 32.2% respectively. The EMLLT and SPAM ML models gave 0.6% and 0.9% absolute WER reduction compared to the baseline. After 4 MPE iterations, the EMLLT model converged at 31.9% while the SPAM model

continued to yield a further 0.3% absolute improvement, giving 31.7% and 31.4% WER after 4 and 8 MPE iterations. Once again, the SPAM model, with more compact model representation, was found to give a more robust performance under the condition of data sparseness.

8.3 Unadapted Experimental Results

This section presents the unadapted experimental results for various precision matrix models. Most of the initial development experiments were based on the CTS-E task. Experimental results on the BN-E task will also be presented to generalise the performance analysis to multiple transcription tasks.

8.3.1 Unadapted Experimental Results on CTS English

For the initial experiments on CTS-E, HMM systems were trained using the h5etrain03 data and evaluated using the dev01sub test set. First, comparison of various precision matrix models will be presented in Section 8.3.1.1. Next, performance of precision matrix models with multiple bases are discussed in Section 8.3.1.2.

8.3.1.1 Comparison of Precision Matrix Models

This section compares various forms of precision matrix models, including the STC, EMLLT, SPAM and HLDA-PMM models. A 16-component HLDA system was chosen as the baseline. The HLDA projection matrix was estimated once and fixed for subsequent training iterations. The EMLLT system consists of 78 basis vectors, initialised using the STC-HLDA method as described in Section 4.3.2.1. Similar to HLDA models, basis matrices for SPAM models were initialised as described in Section 4.3.2.2 and kept constant for subsequent training iterations. In MPE training, only the basis coefficients were updated. All systems were trained using the h5etrain03 training set and evaluated on dev01sub. The word error rate (WER) performance of various precision matrix models is summarised in Table 8.9. The dimension of the mean vectors and basis coefficients as well as the average number of parameters per state are also given in the same table. The baseline HLDA 16-component ML model gave a WER of 33.5% on dev01sub. If the nuisance dimensions are retained, the STC model yields a further 0.2% absolute reduction in WER. By tying the 13 basis coefficients corresponding to the nuisance dimensions, a STC model is transformed into the HLDA-PMM model with effectively 39 basis coefficients per Gaussian component. This model gives another 0.1% absolute improvement over the STC model. The EMLLT model, with 78 basis coefficients (approximately 67% additional number of parameters compared to the HLDA system), reduced the WER to 32.6%. The SPAM model, using only half

System	Average # of parameters per state	Dimensions		WER (%)	
		μ	Σ	ML	MPE
HLDA	1248	39	39	33.5	29.8
STC	1664	52	52	33.3	29.7
HLDA-PMM	1456	52	39+(13)	33.2	29.4
EMLLT	2080	52	78	32.6	29.2
SPAM	1456	52	39	32.8	29.2
HLDA+SPAM	1248	39	39	32.0	*28.5*

Table 8.9 *Comparison of number of parameters and WER (%) performance of ML and MPE trained 16-component precision matrix models*

the number of basis coefficients compared to the EMLLT model, gave similar performance (only 0.2% degradation). Finally, the most compact representation is achieved by combining HLDA and SPAM (HLDA+SPAM) gave the lowest WER of 32.0%, which is 1.5% absolute performance improvement but using the same number of parameters compared to the baseline.

Table 8.9 also shows the MPE performance of various systems. In general, there is an absolute 3.4–3.8% WER reduction from ML to MPE. This shows that the gains from MPE training and precision matrix modelling are *additive*. After MPE training, both the EMLLT and SPAM models yielded the same performance of 29.2%. However, the SPAM model constitutes a more powerful set of basis matrices and hence requires a smaller number of basis matrices for accurate approximation. The best performance was given by the HLDA+SPAM model, which is 1.3% absolute better than the baseline in terms of WER performance. This clearly shows the importance of compact model representation.

To obtain a more complete overview of these models, 28-component HLDA, SPAM and HLDA+SPAM models were trained on the `h5etrain03` data set and evaluated on both `dev01sub` and `eval03` test sets. The results are tabulated in Table 8.10. The SPAM and HLDA+SPAM

System	Effective params per state	dev01sub		eval03	
		ML	MPE	ML	MPE
HLDA	2184	32.3	29.1	31.7	28.4
SPAM	2548	31.5	28.3	30.8	27.6
HLDA+SPAM	2184	31.1	27.9	30.4	27.3

Table 8.10 *WER performance of 28-component precision matrix models on* `dev01sub` *and* `eval03` *for* `CTS` *English task*

systems consistently outperform the baseline HLDA system on both test sets. The gain from mpe training was consistent, approximately 3.1–3.3% absolute WER reduction. The MPE trained SPAM model gave an absolute gain of 0.8% over the baseline on both test sets. As before,

the HLDA+SPAM system gave the best performance of 27.9% and 27.3% on `dev01sub` and `eval03`. These translate to absolute improvements of 1.2% and 1.1% respectively. Two important observations can be made by comparing Tables 8.10 and 8.9. Firstly, as the number of components increases from 16 to 28, the gain from MPE training reduces. This is generally the case when system complexity increases. This is due to the fact that increasing system complexity most certainly improves the model correctness, and hence discrimination power of the system. Consequently, subsequent MPE training will result in a smaller gain. Secondly, and more importantly, increasing the number of Gaussian components from 16 to 28 improved the MPE HLDA system performance from 29.8% to 29.1%. However, this is still 0.6% worse than the 16-component HLDA+SPAM system. Thus, approximating the precision matrix structures using SPAM is superior to implicitly modelling the correlations using a Gaussian Mixture Model (GMM), as described in Section 3.3.

8.3.1.2 Multiple Bases Models

Multiple transformations models, as discussed in Section 3.7, provide a simple and powerful way of improving modelling accuracies without severely increasing the total number of model parameters. Such technique also fits naturally within the basis superposition framework where the basis extraction process is carried out for each cluster of Gaussian components. In this experiment, Gaussian clustering is performed in two different ways. For HLDA and STC models, a regression class tree is used to cluster the Gaussian components with an initial speech-silence split. Splitting criterion is based on the Euclidean distance between Gaussian components. This yields the 65-bases (64 speech, 1 silence) HLDA and STC models[1]. Gaussian clustering for EMLLT model is also achieved using a regression class tree. However, there is no initial speech-silence split and the splitting is based on the Euclidean distance of the vectors of basis coefficients. This results in 64 clusters of Gaussian components. Table 8.11 summarises the WER results for

System	# of transforms	WER (%)		
		ML	MPE(4)	MPE(8)
HLDA	1	33.5	30.8	29.8
	65	32.7	29.7	–
STC	1	33.3	30.3	29.7
	65	32.3	29.7	–
EMLLT	1	32.6	29.8	29.2
	64	32.0	29.0	28.3

Table 8.11 *Comparing WER (%) performance of 16-component precision matrix models with multiple bases on* `dev01sub`

[1] The multiple bases HLDA and STC models were obtained from X. Liu. These models have been trained and decoded using the same setup as described earlier.

multiple bases HLDA, STC and EMLLT models on dev01sub. Compared to single basis models, multiple bases HLDA, STC and EMLLT models gave 0.8%, 1.0% and 0.6% absolute WER reduction respectively. After 4 MPE training iterations, the 65-basis HLDA and STC models gave the same performance of 29.7% WER. On the other hand, the WER performance of the 64-basis EMLLT model was 29.0%. The absolute improvements for these models were observed to be 1.1%, 0.6% and 0.8% respectively compared to their corresponding single-transform models. In general, promising improvements were obtained by using multiple bases, without dramatically increasing the number of model parameters and computational costs. The 64-basis EMLLT system yielded a further 0.7% absolute WER reduction with four additional MPE training. In short, the 64-transform EMLLT model yields absolute improvements of 1.5% and 1.3% for ML and MPE training respectively compared to the baseline single-transform HLDA model. This MPE performance is similar to the 28-component SPAM system as indicated in Table 8.10. However, this exceptionally good performance for the 64-basis EMLLT system did not generalise to eval03, where the performance was found to be 28.1% (0.5% worse than the 28-component SPAM system).

8.3.1.3 Performance of State-of-the-art Systems

Finally, the performance of state-of-the-art 36-component HLDA, HLDA+SPAM and 9k-basis HLDA+STC(9k)[2] systems was evaluated. These state-of-the-art systems were trained using the fsh2004h5etrain03b training data (2180 hours). The system performance was evaluated on eval03 and dev04. Table 8.12 shows the ML, MPE and GD performance of these systems. The

System		eval03			dev04
		s25	fsh	Avg	
HLDA	ML	31.4	23.3	27.5	23.8
	MPE	26.5	18.8	22.8	19.1
	GD	26.0	18.4	22.3	18.7
HLDA+STC (9k)	ML	30.3	22.3	26.5	22.5
	MPE	25.9	18.1	22.1	18.6
	GD	25.4	17.9	21.8	18.3
HLDA+SPAM	ML	30.5	22.7	26.7	23.1
	MPE	25.7	18.1	22.0	18.3
	GD	25.3	17.7	21.6	18.1

Table 8.12 *WER performance of unadapted single-pass decoding for 36-component HLDA, HLDA+STC(9k) and HLDA+SPAM systems on* CTS-E *task*

[2]The 9k bases were chosen such that the Gaussian components in each HMM state share the same basis. The system comprised approximately 9000 distinct states.

baseline MPE trained GD HLDA system gave 22.1% and 23.2% WERs on `eval03` and `dev04` respectively. The gains from MPE and GD training were 4.7% and 0.4–0.5% respectively. The MPE gains for the HDA+STC(9k) system were smaller, only 4.4% and 3.9% on the two test sets. GD training gave a further 0.3% absolute gain on both test sets to yield the final performance of 21.8% and 18.3% on `eval03` and `dev04` respectively. Thus, the final improvement of the HLDA+STC(9k) system was 0.5% and 0.4% absolute on these test sets. The MPE and GD gains for the HLDA+SPAM system were similar to those obtained for the baseline. The final MPE train GD system was 0.2% better than the HLDA+STC(9k) system. It is interesting to note that the total number of model parameters for the HLDA+STC(9k) system in approximately 60% more than the HLDA and HLDA+SPAM system. With the large amount of training data, it was possible to learn a different set of bases for each state in the system. This system was able to reduce the WERs by about half a percent absolute compared to the baseline at the expense of having a larger number of model parameters. However, these parameters were used sub-optimally to model the redundant precision matrix structures in the system. This can be justified clearly from the slightly better performance of the HLDA+SPAM system, using only a set of *global* bases. These bases (which are full-rank) allow the precision matrix structures to be modelled more compactly and avoid any redundancy in modelling the precision matrix structures.

The combination of these systems were also investigated to explore further gains that can be obtained and to examine the differences between the systems using different precision matrix modelling approaches. Confusion network decoding was performed using the lattices produced by the single-pass decoding in Table 8.12. Confusion networks were also generated for subsequent 2-way confusion network combinations (CNCs). The CN decoding and CNC results are summarised in Table 8.13. In general, there was an additional 0.8–0.9% absolute gain from

System	**eval03**			**dev04**
	s25	fsh	Avg	
S1 HLDA	24.9	17.8	21.5	17.9
S2 HLDA+STC(9k)	24.5	17.1	20.9	17.5
S3 HLDA+SPAM	24.2	17.0	20.7	17.3
S1+S2	24.2	17.1	20.8	17.4
S1+S3	24.1	17.1	20.7	17.3
S2+S3	23.8	16.8	20.4	17.1

Table 8.13 *WER performance of unadapted CN decoding and CNC for 36-component HLDA, HLDA+STC(9k) and HLDA+SPAM systems on* `CTS-E` *task*

CN decoding for all the systems. The best individual system was HLDA+SPAM, with 20.7% and 17.3% WERs on `eval03` and `dev04` respectively. Combining HLDA and HLDA+STC(9k) gave a consistent gain of 0.1% over the corresponding individual systems. However, combining HLDA and HLDA+SPAM did not yield further improvement. This is primarily due to the large

performance gap between these systems. Finally, combining HLDA+STC(9k) and HLDA+SPAM yielded a further 0.2–0.3% absolute WER reduction. This suggests that the two systems are relatively different, even though the individual system performance was similar.

8.3.2 Unadapted Experimental Results on BN English

To examine the performance of MPE trained precision matrix models on the BN-E task, a 16-component HLDA+SPAM model was also built to compare with the unadapted HLDA system trained on the bnac+TDT4 (375 hours) data set. These systems were evaluated on the dev03 and eval03 test sets, each consisting of 3 hours data. The results are tabulated in Table 8.14. The ML

System	dev03			eval03		
	ML	MPE	GD	ML	MPE	GD
HLDA	16.3	13.6	13.5	14.6	12.5	12.3
HLDA+SPAM	15.7	13.5	13.2	14.3	12.0	12.0

Table 8.14 *WER performance of 16-component precision matrix models on* dev03 *and* eval03 *for* BN-E *task*

baseline WERs are 16.3% (dev03) and 14.6% (eval03). After MPE training, the WERs reduced to 13.6% and 12.5% respectively. An absolute gain of 0.6% was observed from HLDA+SPAM ML model on dev01sub. The corresponding gain on eval03 was only 0.3%. After MPE training, the gain from HLDA+SPAM was reduced to 0.1% on dev03 but was increased to 0.5% on eval03. Similar to the RT03 setup, gender dependent (GD) models were also built. Starting from the gender-independent (GI) MPE model, GD models were built with 3 MPE+MAP[101] iterations, using the corresponding GI MPE models as the prior. The baseline system gave a further 0.1% and 0.2% WER reductions on dev03 and eval03 respectively. Meanwhile, the HLDA+SPAM model yielded 0.3% improvement on dev03 but no further improvement was obtained on eval03. The final absolute gains of 0.3% on both test sets were found to be statistically significant.

8.3.3 Unadapted Experimental Results on CTS Mandarin

The final set of unadapted evaluation was conducted on the CTS-M task. In contrast to the CTS-E and BN-E tasks, the Gaussianisation technique [15, 116] was employed to normalise the distribution of the features as a normal distribution for each conversation side. This was applied on top of the HLDA projected features. Hence, system using the Gaussianisation frontend will be prefixed by GAUSS instead of HLDA. As before, the best system (GAUSS+SPAM) was selected. The Character Error Rate (CER) performance of GAUSS and GAUSS+SPAM is summarised in Table 8.15. The ML performance of the GAUSS+SPAM model are 1.3% and 1.0% absolute better

System	dev04		eval04	
	ML	MPE	ML	MPE
GAUSS	40.2	36.0	38.2	33.9
GAUSS+SPAM	38.9	35.7	37.2	34.1

Table 8.15 *CER performance of 16-component GAUSS and GAUSS+SPAM models on* dev04 *and* eval04 *for* CTS-M *task*

compared to the baseline GAUSS system. Unlike the results presented so far, the gain from MPE training is much smaller for the GAUSS+SPAM model on the CTS-M task. The baseline GAUSS system gained 4.2–4.3% absolute from MPE training. However, the MPE gain for the GAUSS+SPAM system was only 3.1–3.2%. Thus, the GAUSS+SPAM system gave only 0.3% on dev04 and a degradation of 0.2% on eval04. The poor MPE gain for the GAUSS+SPAM model may be due to the limited amount of training data which has caused an over-training issue.

8.4 Adaptation Experiments of Precision Matrix Models

Adaptation experiments were conducted based on two LVCSR English tasks: BN-E and CTS-E. CMLLR transforms were used for building SAT models. Instead of speaker adaptively train the HLDA+SPAM system to produce a HLDA+SPAM+SAT system, the training approach described in [6] was adopted, where a speaker adaptively trained HLDA baseline system, with diagonal covariance matrices, (HLDA+SAT) was used as an initial model build a HLDA+SAT+SPAM system. In other words, the SPAM precision matrix modelling was performed within the SAT feature space. In testing, MLLR mean transforms for the SPAM models were estimated using two row-by-row iterations as described in Section 6.1.1 (mllr) or simply approximated using the diagonal precision matrix assumption (mllr_approx) (see Section 6.1.2). Similarly, the CMLLR transforms were estimated either using the exact method (cmllr) as described in Section 6.3.1 or approximated using a SAT+DIAGC system (cmllr_approx) (see Section 6.3.2).

Figure 8.2 illustrates the change in the average log likelihood of one speaker with increasing number of iterations for both MLLR mean and CMLLR adaptations. On each iteration, the component alignment was recomputed based on the transforms estimated in the previous iteration. The average log likelihood was found to increase upon every iteration. In Figure 8.2(a), there is very little difference between the mllr and mllr_approx methods for MLLR mean transform estimation. For CMLLR, the log likelihood gain from using the cmllr method is about twice that of the approximated method, cmllr_approx, as depicted in Figure 8.2(b). In spite of this, the WER performance between these two approaches was similar (see the following for experimental results on CTS-E and BN-E tasks).

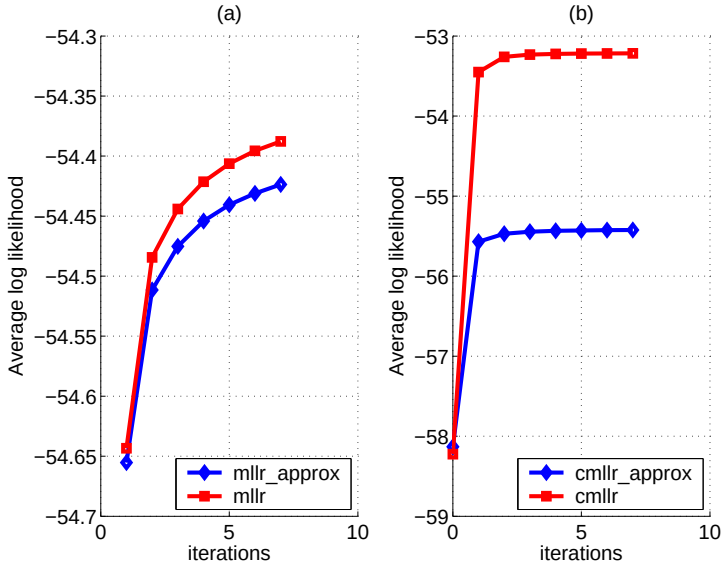

Figure 8.2 *Change in average log likelihood of one speaker on CTS with increasing number of MLLR iterations for (a) MLLR mean and (b) CMLLR, for 28-component SPAM model*

8.4.1 Adaptation Experiments on CTS English Task

First, WER performance was evaluated on the `CTS-E` task. 28-component models were trained using the 400 hours of Fisher data (`fsh2004sub`) and evaluated on two test sets, `eval03` and `dev04` Table 8.16 summarises the results of various adaptation configurations on `CTS-E`. The

System	Adapt Config	eval03			dev04
		s25	fsh	Avg	Avg
HLDA	mllr	26.1	18.1	22.3	18.4
HLDA+SPAM	mllr_approx	25.5	17.9	21.9	17.9
	mllr	25.5	18.0	21.9	18.0
HLDA+SAT	cmllr	25.8	17.8	21.9	17.9
HLDA+SAT+SPAM	cmllr_approx	25.0	17.6	21.4	17.6
	cmllr	24.9	17.5	21.3	17.5

Table 8.16 *Comparisons of MLLR mean and CMLLR adaptations for 28-component HLDA and HLDA+SPAM models, with and without SAT, on* `CTS-E` *task*

WERs of the baseline HLDA system after MLLR adaptation were 22.3% and 18.4% on `eval03` and `dev04` respectively. HLDA+SPAM model with diagonal precision matrix approximated MLLR adaptation (`mllr_approx`) gave 0.4–0.5% gains, although a large proportion of the gain on `eval03` came from `s25` (0.6%). Performing two additional row-by-row iterations, although improved the likelihood, degraded the WER performance by 0.1% on the `fsh` part of `eval03` and `dev04`. For the SAT systems, HLDA+SAT is about 0.3%–0.5% absolute better than the non-SAT baseline on both test sets. Using this model to estimate the CMLLR transforms for the

HLDA+SAT+SPAM system (cmllr_approx) improved the WERs by 0.5% and 0.3% absolute on eval03 and dev04 respectively. Again, the gain on s25 dominated for the gain on the eval03 test set. Exact implementation using the cmllr method gave a consistent improvement of 0.1% on all test sets.

Finally, CMLLR adaptation was performed on the state-of-art systems. The cmllr_approx approximation scheme was used for the HLDA+SPAM system. In addition, the performance of the speaker adaptively train HLDA+SAT and HLDA+SAT+SPAM systems was also evaluated. The summary of the results is shown in Table 8.17. After the CMLLR adaptation, the

System		eval03			dev04
		s25	fsh	Avg	
HLDA	ML	29.0	21.4	25.3	21.6
	MPE	24.8	17.4	21.3	17.8
	GD	24.4	17.1	20.8	17.4
HLDA+STC (9k)	ML	28.2	20.7	24.6	20.8
	MPE	24.2	17.1	20.7	17.4
	GD	23.9	16.7	20.5	17.1
HLDA+SPAM	ML	28.2	20.7	24.6	21.1
	MPE	24.1	16.8	20.5	17.1
	GD	23.7	16.4	20.2	17.1
HLDA+SAT	ML	28.9	21.1	25.1	21.5
	MPE	24.8	17.3	21.2	17.8
HLDA+SAT+SPAM	ML	28.1	20.5	24.5	21.0
	MPE	24.1	16.8	20.5	17.1

Table 8.17 *WER performance of CMLLR adapted single-pass decoding results for state-of-the-art systems on* eval03 *and* dev04 *for* CTS-E *task*

performance gaps between the MPE trained GD precision matrix models and the baseline were marginally smaller. The adapted HLDA+STC(9k) system was 0.3% better than the adapted baseline, compared with 0.4–0.5% on the unadapted configuration. The improvement of the adapted HLDA+SPAM system over its baseline was also reduced by 0.1%. Thus, the adapted HLDA+SPAM system gave an overall improvement of 0.6% and 0.5% on eval03 and dev04 over the baseline. On the other hand, the HLDA+SAT system gave a baseline performance of 21.2% and 17.8% on the two test sets. The HLDA+SAT+SPAM system improved the baseline performance by 0.7% on both test sets. On this particular configuration, SAT did not seem to yield any performance gain compared to the non-SAT MPE systems. In fact, the GD systems outperformed the SAT systems by 0.3–0.4%. However, when more advanced adaptation configurations were used, the SAT systems gave superior performance, as shown later in Section 8.5.1

8.4.2 Adaptation Experiments on BN English Task

Similar comparisons were made for the adapted systems on the BN-E task. 16-component models were trained using 374 hours of bnetrain04sub training data. This consists of 143 hours of carefully annotated data (bnac) and 231 hours of lightly supervised data (tdt4). Adaptation experiments were conducted based on three 3-hour test sets: eval03, dev04 and dev04f. 4-gram rescoring lattices were generated using an adapted HLDA system[3]. Rescoring results are summarised in Table 8.18. For MLLR mean adaptation, a gender dependent (GD) HLDA system

System	Adapt Config	Test Set WER (%)		
		eval03	dev04	dev04f
HLDA	mllr	10.7	13.2	20.0
HLDA+SPAM	mllr_approx	10.6	13.1	19.5
	mllr	10.6	13.1	19.5
HLDA+SAT	cmllr	10.6	13.1	19.5
HLDA+SAT+SPAM	cmllr_approx	10.2	12.7	18.6
	cmllr	10.2	12.8	18.8

Table 8.18 *Comparisons of MLLR mean and CMLLR adaptations for 16-component HLDA and HLDA+SPAM models with and without SAT on* eval03*,* dev04 *and* dev04f *for* BN-E *task*

was chosen as the baseline. This system gave WERs of 10.7%, 13.2% and 20.0% on the three test sets. The exceptionally poor performance on dev04f is due to the large mismatch between the training and the test data. Both mllr_approx and mllr configurations yielded the same performance, which is 0.1% absolute better than the baseline on eval03 and dev04. The gain on dev04f is larger, 0.5% absolute. This shows that MLLR mean adaptation can be efficiently approximated with the diagonal precision matrix assumption for the HLDA+SPAM models and other forms of precision matrix models such as EMLLT.

Also, two forms of CMLLR adaptation for HLDA+SAT+SPAM models were compared using the HLDA+SAT system as the baseline. This system has the same WER performance as the MLLR mean adapted SPAM system. The cmllr_approx configurations gained 0.4% absolute on the first two test sets and 0.9% on dev04f. Again, there is a large gain from the adapted HLDA+SPAM models due to the mismatch between the training and test sets. Similar performance was obtained on eval03 using the exact cmllr configuration. Surprisingly, 0.1% and 0.2% degradations were observed on dev04 and dev04f although the likelihood of the test data given these transforms was higher than those approximated using cmllr_approx. Apart from the gains from the mllr_approx and mllr HLDA+SPAM models on eval03 and dev04, all the gains shown in Table 8.18 were found to be statistically significant[4].

[3] Similar to the P2 stage of the CU-HTK evaluation system [65, 67]

[4] Significance tests were carried out using the NIST Scoring Toolkit.

8.5 Evaluations on Multi-pass and Multi-branch Framework

Previously, the performance of various precision matrix models was compared. In particular, the HLDA+SPAM and HLDA+SAT+SPAM systems were found to be the best individual systems. In this section, the performance of the HLDA+SPAM and HLDA+SAT+SPAM systems will be evaluated based on a multi-pass multi-branch framework similar to the CU-HTK 10xRT Broadcast News transcription system [65, 67]. This evaluation framework incorporates complex adaptation configurations and allows possible improvements from systems combination to be explored. The basic structure of the framework is illustrated in Figure 8.3 Initially, audio data were au-

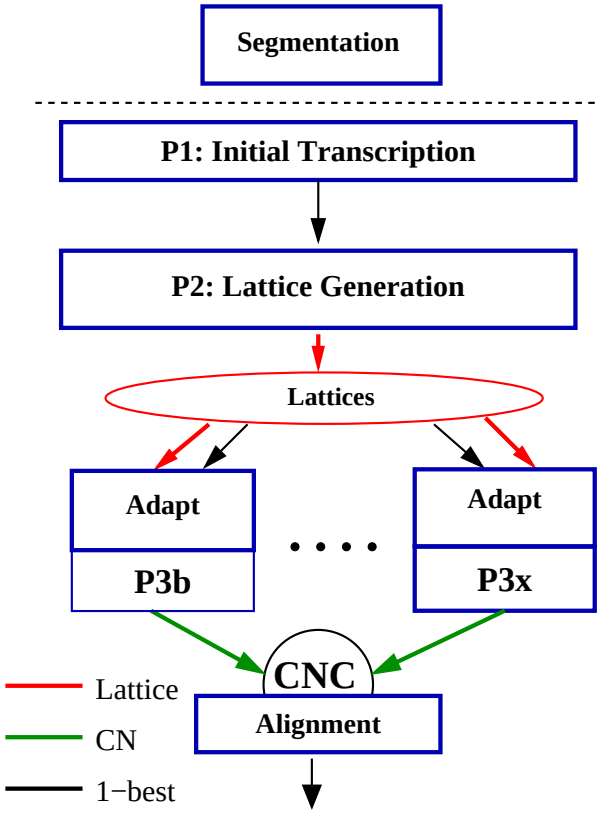

Figure 8.3 *A multi-pass multi-branch evaluation framework*

tomatically segmented to extract only the speech segments. These segments of speech data were recognised using an unadapted non-VTLN model and a trigram language model to provide an initial transcription. This initial transcription was used as a supervision for adaptation in the P2 stage. The adapted model was then used to generate lattices for lattice-based MLLR adaptation and rescoring by subsequent stages. These lattices were generated using a trigram language model, expanded and pruned using a 4-gram category-based language model. Multiple P3 stages may be used in parallel to construct a multi-branch framework. Each of these P3 stages involved complex adaptation configurations as follows. This was performed using an improved supervision generated from P2 to perform a one-best MLLR adaptation followed by a lattice-based MLLR. The final adapted model will then be used to rescore the P2 lattices to yield

the single branch performance. Furthermore, system combination may be performed by generating and combining the confusion networks from each P3 branch using the Confusion Network Combination (CNC) method [25, 76]. The actual forms of adaptation (MLLR mean, variance or constrained MLLR) used in the P3 stages may differ in each individual experiments. As with the other experiments, the CTS-E, BN-E and CTS-M tasks were considered.

8.5.1 Evaluation on CTS English Task

Initially, the development setup was used. 28-component systems were trained using the 400-hour fsh2004sub Fisher conversations, released by the LDC, with a balanced gender and line condition [24]. Quick transcriptions which correspond to these speech were provided by BBN, LDC and another commercial transcription service. The eval03 and dev04 test sets were used for systems evaluation. All system have approximately 6000 physical states after decision tree based tying.

System		eval03			dev04
		s25	fsh	Avg	
P2	HLDA	26.6	18.4	22.6	18.7
P3a	SPron	24.7	17.6	21.3	17.6
P3b	HLDA	24.8	17.7	21.4	17.5
P3c	HLDA+SPAM	23.8	16.5	20.4	16.8
P3d	HLDA+SAT	24.5	17.1	20.9	17.3
P3e	HLDA+SAT+SPAM	23.6	16.4	20.1	16.6
P3a+P3d	CNC	23.9	16.8	20.5	16.9
P3a+P3e		23.6	16.4	20.1	16.6

Table 8.19 *WER performance of various development systems evaluated on* eval03 *and* dev04 *in a multi-pass multi-branch framework for* CTS-E *task*

Table 8.19 shows the performance of various systems evaluated within a multi-pass multi-branch framework. The P2 stage used an adapted HLDA system to generate lattices for subsequent stages. This system gave 22.6% and 18.7% WER performance on eval03 and dev04 respectively. Four P3 branch were used: HLDA, HLDA+SAT, HLDA+SAT+SPAM and SPron. SPron is an HLDA system using only single pronunciation in the lexicon. The best individual system was HLDA+SAT+SPAM with WERs of 20.1% and 16.6% on eval03 and dev04. This gave an absolute improvement of 0.7–0.8% over the HLDA+SAT system. The HLDA+SPAM also gave similar improvements over the HLDA system. The 2-way combination between the SAT and SPron systems was the standard configuration used in the CUHTK CTS-E evaluation system [22, 24]. Significant error rate reduction over individual branches was achieved after system combination. The final error rates were 20.5% on eval03 and 16.9% on dev04. However, this performance is still

about 0.3–0.4% worse than the HLDA+SAT+SPAM single branch performance. Moreover, the single branch HLDA+SAT+SPAM system performance was far better than any other individual system that no possible 2-way combination can yield further improvement.

A second set of evaluation was carried out using the state-of-the-art CTS-E systems trained on 2180 hours of fsh2004h5etrain03b data. The results are tabulated in Table 8.20 For the sys-

System		eval03			dev04
		s25	fsh	Avg	
P3a	HLDA	21.7	14.7	18.3	15.1
P3b	HLDA+SPAM	21.1	14.6	18.0	14.9
P3c	HLDA+STC(9k)	21.6	14.8	18.3	15.3
P3d	HLDA+SAT(QUIN)	21.5	14.8	18.2	15.0
P3e	HLDA+SAT+SPAM	21.0	14.6	17.9	14.7
P3a+P3d	CNC	20.9	14.1	17.6	14.3
P3b+P3d		20.2	13.9	17.2	14.2
P3c+P3d		20.6	14.1	17.4	14.4
P3e+P3d		20.4	14.0	17.3	14.1

Table 8.20 *WER performance of various state-of-the-art systems evaluated on* eval03 *and* dev04 *in a multi-pass multi-branch framework for* CTS-E *task*

tems which were not speaker adaptively trained, gender dependent models were trained. The HLDA, HLDA+SAT(QUIN) and HLDA+SAT+SPAM were the models used in the CUHTK system submitted for the RT04 DARPA evaluation [23, 24]. It is worth noting that the HLDA+SAT(QUIN) system is quite different from the other systems as quinphone context-dependent models were used. The other systems were all triphone models. HLDA+SAT+SPAM is still the best individual system with WERs of 17.9% and 14.7% on eval03 and dev04 respectively. However, in this case, it is only marginally better than the HLDA+SPAM system (0.1–0.2%). Performance of 2-way combinations was examined, too. In general, a better combination performance was obtained by combining a triphone system with a quinphone system. Considering the P3a+P3d combination, which yielded 0.6–0.7% absolute gain over the P3d system, as the baseline. On average, combining the quinphone system with HLDA+SPAM or HLDA+SAT+SPAM gave similar performance. The lowest WER on eval03 was given by the P3b+P3d combination while that on dev04 was given by the P3e+P3d combination.

8.5.2 Evaluation on BN English Task

Similar evaluation was performed for the BN-E task. The BN-E systems were trained on 370 hours of training data. This consists of two parts [66], 140 hours of accurately transcribed broadcast news acoustic training data released by the LDC in 1996 and 1997 and 230 hours of data selected from the tdt4 audio corpora with close-captions based on quick transcriptions. All systems have approximately 6000 physical states after decision tree based tying. The number of components per state was 16 on average. Three BN-E test sets were used, each of them contains six 30 minutes broadcast news shows. These were the eval03, dev04 and dev04f.

System		eval03	dev04	dev04f
P2	HLDA	10.8	13.4	20.1
P3a	SPron	10.2	13.0	19.0
P3b	HLDA	10.5	13.1	19.5
P3c	HLDA+SPAM	9.9	12.5	18.5
P3d	HLDA+SAT	10.3	12.9	18.7
P3e	HLDA+SAT+SPAM	10.0	12.4	18.4
P2+P3a+P3d	CNC	10.1	12.6	18.6
P2+P3a+P3e		10.0	12.4	18.4

Table 8.21 *WER performance of various systems evaluated on* eval03*,* dev04 *and* dev04f *in a multi-pass multi-branch framework for* BN-E *task*

Table 8.21 shows the performance of various systems evaluated within the multi-pass multi-branch framework. A performance pattern similar to the CTS-E results was observed. The best single-branch system was HLDA+SAT+SPAM with WER performance of 10.0%, 12.4% and 18.4% on eval03, dev04 and dev04f respectively. The gains from HLDA+SPAM were 0.6% on eval03 and dev04. A larger gain of 1.0% was obtained on dev04f. This shows that the SPAM precision matrix modelling technique, to some extends, is able to compensate the mismatch between the training and test data. The HLDA+SAT+SPAM, on the other hand, yielded a smaller improvement of 0.3–0.5% over the baseline HLDA+SAT system. In contrast to the CTS-E system, a 3-way combination between the P2, P3a (SAT) and P3c (SPron) branches was the standard configuration used in CUHTK BN-E evaluation system. The final numbers for each of the tasks were 10.1%, 12.6% and 18.6%, with gains of 0.1–0.3% being obtained from system combination. However, this is still about 0.1–0.2% behind the single-branch HLDA+SAT+SPAM system. As with the CTS-E system, no further gain was obtained from any 3-way combination over the individual HLDA+SAT+SPAM performance on all test sets.

8.5.3 Evaluation on CTS Mandarin Task

The final set of multi-pass multi-branch evaluation was conducted on the CTS-M task. The STC, EMLLT and SPAM precision matrix models were considered, using the Gaussianisation frontend. In addition, speaker adaptively trained diagonal covariance matrix and SPAM systems were also evaluated. The CER performance of these systems is given in Table 8.22. The baseline Gaus-

System		CER (%)		
		dev04	eval03	eval04
P3a	GAUSS	34.6	43.3	32.3
P3b	GAUSS+STC	34.4	43.0	31.8
P3c	GAUSS+EMLLT	34.1	42.8	31.4
P3d	GAUSS+SPAM	33.5	42.4	31.0
P3e	GAUSS+SAT	33.7	42.7	31.7
P3f	GAUSS+SAT+SPAM	33.2	41.8	30.5
P3a+P3d	CNC	33.3	42.3	30.9
P3e+P3f		32.8	41.4	30.2

Table 8.22 *CER performance of various systems evaluated on* dev04 *and* eval03 *in a multi-pass multi-branch framework for* CTS-M *task*

sianisation system gave 34.6% and 43.3% CERs on dev04 and eval03 respectively. Modelling the precision matrices using STC improved the baseline performance by 0.2–0.3%. Using an EMLLT model with 84 basis order further improved the performance by 0.2–0.3%. Finally, the performance of SPAM precision matrix modelling gave another 0.4–0.6% absolute CER reduction, yielding a total reduction of 1.1% and 0.9% over the GAUSS system. Speaker adaptively trained systems were also examined. The GAUSS+SAT system is 0.9% and 0.6% absolute better than the GAUSS system. The GAUSS+SAT+SPAM further improved the performance by 0.5% and 0.9% on dev04 and eval03 respectively. The gain from SPAM modelling in the SAT feature space compared to that in the GAUSS feature space was similar on eval03, but almost halved on dev04. The smaller gain of GAUSS+SAT+SPAM on dev04 is due to the larger gain obtained by the GAUSS+SAT system. The overall gain of the GAUSS+SAT+SPAM system over the GAUSS system was similar on both test sets, 1.4–1.5% absolute CER reduction. Combining the GAUSS and GAUSS+SPAM systems using CNC only marginally improved the performance by 0.1–0.2% absolute. On the other hand, combining the speaker adaptively trained systems (GAUSS+SAT and GAUSS+SAT+SPAM) gave a larger absolute improvement of 0.4% on both test sets.

9

Experimental Results of Semi-parametric Trajectory Models

This chapter presents the experimental results of the semi-parametric trajectory models described in Chapter 7. Specifically, the performance of the fMPE and pMPE systems will be studied. The first part of the results were preliminary experimental results based on the `CTS-E` task. Later, a more thorough investigation of the fMPE and pMPE techniques and the effect of combining them will be discussed based on the `CTS-M` task.

9.1 Preliminary Experiments on CTS English

This section presents the preliminary experimental results of the semi-parametric trajectory model on the `CTS-E` task. Systems were trained using the 296 hours switchboard data (`h5etrain03`) and evaluated on a 3-hour test set (`dev01sub`). The baseline was a speaker independent HLDA system with about 6000 states and 16 Gaussian components per state ($\sim$ 99k Gaussians in total). The posterior probabilities, $h_i(t)$, as described in Chapter 7, were calculated based 99k centroids. These centroids were obtained from the Gaussian components in the system. These centroids were grouped into 1024 clusters. The posterior probabilities were calculated by evaluating only the centroids in the 5 most likely clusters. Moreover, posterior probabilities below 0.1 were set to zero to yield an average of approximately 2 non-zero posteriors at each time. No context expansion [97] was used. First, the fMPE and pMPE models were built using 4 interleaved updates (Section 7.3.2.1) with the learning rate, $\alpha = 1.0$. 8 MPE iterations were then performed on top of them to give the fMPE+MPE and pMPE+MPE models. The MPE+fMPE and MPE+pMPE systems were also built using the direct estimation (Section 7.3.2.2) with $\alpha = 0.5$.

Figure 9.1 shows the change in the MPE criteria with increasing training iterations for various systems. The MPE criterion of the ML baseline was 0.74. This was improved by about 0.11 after 12 MPE iterations. The criterion gains for fMPE and pMPE were smaller compared to

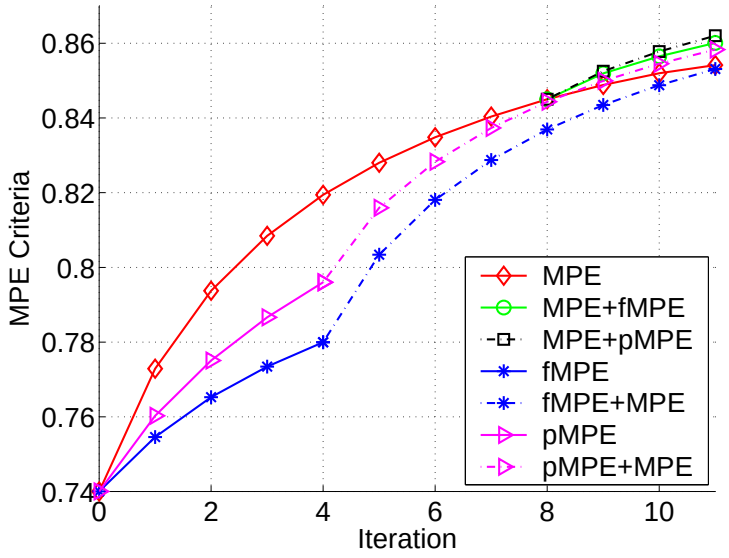

Figure 9.1 *MPE criterion against training iteration*

MPE training. Further MPE training increased the criteria of fMPE+MPE and pMPE+MPE to be similar to the MPE system, with the latter marginally better. The larger criterion gain for pMPE did not generalise to recognition performance (see later). This suggests that pMPE is less robust to overtraining, unlike fMPE [97]. Further criterion gain were obtained with the MPE+fMPE and MPE+pMPE systems.

System	Initial Model	Iter 0	Iter 4	Iter 8
MPE	ML	33.5	30.7	30.2
fMPE+MPE	fMPE	31.9	29.9	29.4
pMPE+MPE	pMPE	32.5	30.4	30.0

Table 9.1 *WER performance on* `dev01sub` *for 16-component models using interleaved parameters estimation*

The Word Error Rate (WER) performance on `dev01sub` for fMPE+MPE and pMPE+MPE systems are shown in Table 9.1. The ML baseline performance was 33.5%. MPE alone reduced the WER by 3.2% absolute. fMPE and pMPE gave 1.6% and 1.0% absolute WER reduction respectively. The WER performance of the pMPE system converged much quicker (after two iterations). MPE training on top of these systems each gained a further 2.5% absolute, which are respectively 0.8% and 0.2% absolute better than the MPE system alone. The performance difference between MPE and pMPE+MPE gradually diminished as the number of MPE training increases. Similar performance was obtained when using the exact pMPE update. However, a slower learning rate ($\alpha = 0.5$) is required to prevent over training. The gains from fMPE and pMPE are not *additive*. Initial experiment of pMPE training on top of the fMPE system (fpMPE) showed 0.5% absolute improvement over the fMPE system. Unfortunately, this gain decreases with increasing MPE training iterations. More investigation is required to study the interaction between the fMPE and pMPE training.

System	Iter 0	Iter 2	Iter 4
MPE	30.2	30.2	30.2
MPE+fMPE	30.2	29.6	29.4
MPE+pMPE	30.2	30.0	29.8

Table 9.2 *WER performance on* `dev01sub` *for 16-component systems using direct parameters estimation*

Table 9.2 compares the WER performance of MPE+fMPE and MPE+pMPE using the direct estimation scheme. The initial model used by all systems was the MPE system trained with 8 iterations. Four additional standard MPE iterations gave no further improvement. The MPE+fMPE system gave similar performance to the fMPE+MPE system, but the training time for the former is more efficient. Also, four additional direct pMPE training is 0.2% better than the pMPE+MPE system. All the gains over standard MPE presented were statistically significant[1], except the 0.2% gain from the pMPE+MPE system.

9.2 Experimental Results on CTS Mandarin

A more detailed investigation of the fMPE and pMPE models was performed on the `CTS-M` task. The acoustic models were trained on 72 hours of `ldc04` and `swm03` data. The systems used in this experiment consist of 4000 distinct states. Gaussianisation frontend was used on top of the HLDA projected feature space. A variety of aspects of the systems were examined as follows.

9.2.1 Choice of the Centroids

This section will investigate two ways of determining the centroids used in the semi-parametric trajectory model. The centroids are typically chosen from the Gaussian components in the system. Typically, the Gaussian components in the systems are clustered into the desired number of groups. The centre of each group is chosen as one of the centroids. In this work, two special cases were considered. The first case simply uses each Gaussian component in the system as the centroids. This yields in a number of centroids which is equivalent to the number of Gaussian components in the system. This approach is feasible is the HMM system is of moderate size. The second approach performs a state-level clustering of the Gaussian components such that the centroids are chosen as the centre of the Gaussian components for each HMM state. Thus, the resulting number of centroids is the same as the number of physical states in the system. Here, the fMPE+MPE systems with 16 component per state will be considered. Hence, there were 64,000 Gaussian components in the system. Therefore, the two methods of selecting the

[1] Significance tests were carried out using the NIST Scoring Toolkit

centroids gave a total of 64k and 4k centroids respectively.

System	Context Expansion	dev04		eval04	
		4k	64k	4k	64k
MPE	–	36.0		33.9	
fMPE+MPE	0	35.6	35.1	33.7	33.1
	± 3	34.4	34.8	32.5	33.4

Table 9.3 *CER performance of fMPE+MPE with different number of centroids on* dev04 *and* eval04 *for* CTS-M *task*

Table 9.3 compares the CER performance of the fMPE+MPE systems with different number of centroids. For the systems without context expansion, the use of 64k centroids gave superior performance compared to the 4k centroids system (0.5–0.6% better). In contrast, the 64k centroids system with ± 3 context expansion was 0.4% and 0.9% worse than the corresponding 4k centroids system. Clearly, the use of 64k centroids with context expansion has dramatically increased the number of model parameters in the system, which has caused the system to be over-trained. From Table 9.3, the 4k centroids with ± 3 context expansion is evidently the preferred configuration. Subsequent experiments will employ the 4k centroids configuration.

9.2.2 The Effect of Context Expansion

From the previous results, the use of context expansion for the fMPE systems gave significant improvements compared to those without context expansion. Table 9.4 summarises the effect

System	dev04		eval04	
	0	± 3	0	± 3
MPE	36.0		33.9	
fMPE+MPE	35.6	34.4	33.7	32.5
pMPE+MPE	35.9	35.4	33.7	33.8
fMPE+pMPE+MPE	35.3	34.7	33.5	33.1

Table 9.4 *CER performance of 4k centroids 16-component fMPE and pMPE systems with 0 and* ± 3 *context expansion on* dev04 *and* eval04 *for* CTS-M *task*

of context expansion for the 4k centroids fMPE+MPE, pMPE+MPE and fMPE+pMPE+MPE systems. The fMPE+MPE systems gave an absolute CER reduction of 0.2–0.4% and 1.4–1.6% using 0 and ± 3 context expansion respectively. Without context expansion, the pMPE+MPE system gave 0.1–0.2% improvements over the MPE alone baseline. However, the ± 3 context expansion did not yield a consistent gain for the pMPE+MPE system. There was a 0.5% gain on

dev04, but a 0.1% degradation in performance on eval04 compared to that without context expansion. When the fMPE and pMPE techniques were combined, there were additional gains of 0.6% and 0.2% on dev04 and eval04 respectively. Unfortunately, with ± 3 expansion, the fMPE+pMPE+MPE system performance was degraded by 0.3% and 0.6% on these test sets. These results indicate that the pMPE technique is highly sensitive to over-fitting issues and does not work very well with complex systems.

9.2.3 Experiments on Single Component Systems

The previous set of experiments suggests that the pMPE technique is easily over-trained, particularly when used with complex systems. In this section, the modelling power of the pMPE technique is illustrated by considering simple systems with only one Gaussian components per state. Table 9.5 compares the performance of the 4k centroids single component fMPE+MPE,

System	dev04		eval04	
	0	± 3	0	± 3
MPE	44.4		42.2	
fMPE+MPE	42.1	40.1	39.4	37.3
pMPE+MPE	43.3	41.3	40.4	38.6
fMPE+pMPE+MPE	41.6	38.9	39.2	36.6

Table 9.5 *CER performance of 4k centroids 1-component fMPE and pMPE systems with 0 and ± 3 context expansion on* dev04 *and* eval04 *for* CTS-M *task*

pMPE+MPE and fMPE+pMPE+MPE systems. The baseline single component MPE alone system gave 44.4% and 42.2% CER on dev04 and eval04 respectively. The fMPE+MPE system improved the baseline by 2.3–2.8% and 4.3–4.9% absolute using 0 and ± 3 context expansions respectively. The pMPE+MPE system, on the other hand, gave absolute improvements of 1.1–1.8% and 3.1–3.6% with and without context expansions. Combining these techniques yielded further gains of 0.2–0.5% without context expansion and 0.7–1.2% with ± 3 context expansion. Several important conjectures can be made based on these results. Firstly, when a simple acoustic model was used, the gains obtained from the fMPE and pMPE techniques became significantly larger. The loss in the modelling power of the static parameters has been compensated by the dynamic parameters. Moreover, the pMPE+MPE system combined well with context expansion. This provides a clear indication that the pMPE+MPE with ± 3 context expansion sufferred from an over-fitting problem. In addition, promising gains were also obtained by combining the fMPE and pMPE techniques to yield the fMPE+pMPE+MPE system.

As previously mentioned in Chapter 7, the fMPE and pMPE techniques may be viewed as a

semi-parametric trajectory model. From this perspective, the single component fMPE and pMPE systems also provided an interesting account for the trajectory modelling aspects of the system. Because the systems under consideration have only one Gaussian component per state, the observations associated with each HMM state in a standard HMM formulation is thus independent and identically distributed (i.i.d.) with a normal distribution. Thus, the trajectory within each HMM state is *piece-wise constant*. By incorporating the fMPE and pMPE techniques to the single component systems, promising improvements as shown in Table 9.5 were obtained.

9.2.4 Experiments on Multiple Components Systems

From the above results, it is believed that there is a trade off between the system complexity and the gains from fMPE and pMPE modelling. The Gaussian components in a state may be viewed as a pseudo states. Throughout the duration of staying in a particular state, the observation may be associated with one of the pseudo states. Therefore, a non-constant trajectory may, to some extend, be defined *implicitly*. This section investigates the effect of increasing the number of Gaussian components per state in the system. The results are given in Table 9.6. In general,

No. of components	System		dev04		eval04	
			Viterbi	CN	Viterbi	CN
1	S1	MPE	44.4	43.2	42.2	41.2
	S2	fMPE+MPE	40.1	39.3	37.3	36.4
	S3	fMPE+pMPE+MPE	38.9	38.2	36.6	35.6
	S1+S2	CNC	—	39.6	—	37.1
	S2+S3		—	38.4	—	35.7
8	S4	MPE	36.6	35.7	34.7	33.8
	S5	fMPE+MPE	34.6	34.2	32.7	32.3
	S6	fMPE+pMPE+MPE	34.5	33.9	33.0	32.5
	S4+S5	CNC	—	34.4	—	32.7
	S5+S6		—	33.8	—	32.3
16	S7	MPE	36.0	35.0	33.9	33.4
	S8	fMPE+MPE	34.4	33.9	32.5	32.2
	S9	fMPE+pMPE+MPE	34.7	34.0	33.1	32.6
	S7+S8	CNC	—	34.1	—	32.2
	S8+S9		—	33.3	—	31.6

Table 9.6 *Viterbi and CN decoding CER performance of 4k centroids fMPE and pMPE systems with* ± 3 *context expansion for different number of components per state on* `dev04` *and* `eval04` *for* `CTS-M` *task*

the improvements from CN decoding decreases as the number of components in the system

increases. For the single component systems, CN decoding gave 0.8–1.2% absolute gain. As the number of components per state was increased to 8 and 16, the gains from CN decoding decreased to 0.4–0.9% and 0.3–1.0% respectively. Furthermore, the fMPE and pMPE gains also decreased as the number of components per state was increased. In fact, for the 16-component system, the CN-decoding performance of the fMPE+pMPE+MPE system gave a loss of 0.1% and 0.4% on `dev04` and `eval04` respectively compared to the fMPE+MPE system. Moreover, it is also worth pointing out that there is only a marginal performance difference between the 8-component and 16-component systems for fMPE+MPE and fMPE+pMPE+MPE (0.1–0.2%). The corresponding performance difference for the MPE models were 0.7% and 0.4% on `dev04` and `eval04` respectively. This suggests that the 16-component system is probably too complex for the fMPE and pMPE techniques.

Table 9.6 also shows the CNC performance of combining MPE and fMPE+MPE as well as fMPE+MPE and fMPE+pMPE+MPE. Due to the large performance gap between the MPE and fMPE+MPE models, the combination performance was at most the same as the best individual system. The same applied to the single component and 8-component fMPE+pMPE and fMPE+pMPE+MPE (S2+S3 and S5+S6) combinations. However, despite the poorer performance of S9 compared to S8, a further 0.6% was obtained when these two systems are combined. This indicates that the errors made by the two system are considerably different.

9.2.5 Mixing-up the fMPE and pMPE systems

The final experiments on fMPE and pMPE investigates the effects of mixing up (increasing the number of Gaussian components per state) the fMPE+MPE and fMPE+pMPE+MPE systems. The 1-component and 8-component fMPE and fMPE+pMPE systems were mixed up to 16 components per state using an iterative mixture splitting procedure described in [134]. The heaviest components were split in each iteration. Subsequent MPE training was performed on these mixed-up systems. These systems were then compared with the original 16-component fMPE+MPE and fMPE+pMPE+MPE. Table 9.7 compares the effects of mixing up the fMPE and fMPE+pMPE systems from 1-component and 8-component systems. CER performance was obtained using both Viterbi and CN decoding. The purpose of this experiments was to examine the level of system complexity on which the fMPE and pMPE parameters may be learned to yield a better performance. According to the Viterbi decoding results, the gain from mixing up the 8-component fMPE+MPE system to 16-component was minimal. The resulting system is slightly inferior compared to the original 16-component fMPE+MPE system. A performance difference of 0.1–0.3% was observed. However, the CN decoding gave a larger gain on the mixed-up system (0.8–0.9%) compared to the baseline (0.3–0.5%). Consequently, the CN decoding performance of the mixed up system gave a slight improvement of 0.1–0.2%. The trend for the fMPE+pMPE+MPE system was slightly different. Mixing-up the 8-component system seems to

Mix-up from	System		dev04		eval04	
			Viterbi	CN	Viterbi	CN
—	S1	MPE	36.0	35.0	33.9	33.4
	S2	fMPE+MPE	34.4	33.9	32.5	32.2
	S3	fMPE+pMPE+MPE	34.7	34.0	33.1	32.6
	S1+S2	CNC	—	34.1	—	32.2
	S2+S3		—	33.3	—	31.6
8	S4	fMPE+MPE	34.5	33.7	32.8	32.1
	S5	fMPE+pMPE+MPE	34.6	34.1	32.7	32.1
	S1+S4	CNC	—	33.6	—	31.5
	S4+S5		—	33.3	—	31.6
1	S6	fMPE+MPE	35.5	34.7	33.0	31.9
	S7	fMPE+pMPE+MPE	35.6	34.8	32.8	31.8
	S1+S6	CNC	—	33.8	—	31.6
	S6+S7		—	34.3	—	31.3

Table 9.7 *Viterbi and CN decoding CER performance of 4k centroids 16-component fMPE and pMPE systems with* ± 3 *context expansion for mix-up from different number of components per state on* dev04 *and* eval04 *for* CTS-M *task*

yield a more robust system compared to learning the pMPE parameters on the 16-component system. Both Viterbi and CN decoding results indicate small improvements of 0.1–0.5% on both test sets, although the difference is larger on eval04. On the other hand, the performance gains from mixing up the 1-component system is rather inconsistent. There was a huge degradation in CER performance on dev04 (approximately 1.0% absolute). The results on eval04 is more consistent with previous discussion. Larger improvements from CN decoding were observed on the mixed-up systems. These were between 1.0–1.1% absolute gains. It is not obvious, from these results, which is the preferred system. In general, learning the fMPE and pMPE parameters on systems which are too complex suppresses the potential improvements from the trajectory modelling perspective. In contrast, starting from systems which are too simple does not allow the interaction between the dynamic and static parameters to be captured in the estimation process. Therefore, mixing up the fMPE and pMPE systems of medium complexity is probably a better option.

Table 9.7 also shows the CNC system combination results between the MPE and fMPE systems as well as fMPE+MPE and fMPE+pMPE+MPE systems. It is interesting to note that the combination of MPE and fMPE+MPE systems mixed up from 8 components (S1+S4) and 1 component (S1+S6) yielded despite the large performance gap between the individual systems. For example, the S1+S4 system gave an additional absolute 0.6% improvement over S4 even though S1 is 1.3% absolute worse than S4. Furthermore, on dev04, the combination of

S1 (35.0%) and S6 (34.7%) resulted in a remarkable improvement of 0.9% absolute. These results suggest that the iterative mixture splitting process is greatly influenced by the dynamic parameters. This is not surprising as the GMM is capable of modelling trajectory to some extend. Therefore, incorporating explicit trajectory modelling may change the resulting GMM parameter estimate. When combining fMPE+MPE and fMPE+pMPE+MPE systems, the performance gain is not very much affected by the different mixing up configurations. The performance gains were found to be between 0.4–0.6%.

9.3 Multi-pass Multi-branch Evaluation on CTS Mandarin Task

Finally, Table 9.8 compares the performance of SAT and SAT+SPAM systems using several different frontends in a multi-pass multi-branch evaluation framework described in Section 8.5. The basic frontend was the 39-dimensional HLDA projected features plus 3-dimensional pitch features. Side-based Gaussianisation was performed on top of that to yield the GAUSS frontend. Finally, fMPE was used to discriminatively train the GAUSS features to obtain the fMPE frontend. From Table 9.8, the use of Gaussianisation improved the performance by 1.0–1.3% on `dev04` and

System		CER (%)		
		dev04	eval03	eval04
P3a	HLDA	35.8	45.0	33.0
P3b	HLDA+SAT	35.0	44.2	32.4
P3s	HLDA+SAT+SPAM	34.2	43.7	32.1
P3d	GAUSS	34.6	43.3	32.3
P3e	GAUSS+SAT	33.7	42.7	31.7
P3t	GAUSS+SAT+SPAM	33.2	41.8	30.5
P3f	fMPE	33.5	42.0	30.4
P3g	fMPE+SAT	33.0	41.3	30.2
P3u	fMPE+SAT+SPAM	32.7	41.3	29.7
P3b+P3s		33.4	42.9	31.4
P3e+P3t	CNC	32.8	41.4	30.2
P3g+P3u		32.3	40.8	29.3

Table 9.8 *CER performance of various systems evaluated on* `dev04` *and* `eval03` *in a multi-pass multi-branch framework for* `CTS-M` *task*

1.5% and 1.9% on `eval03`. This phenomenon is contrary to the results presented in [39, 74], where Gaussianisation only yielded marginal improvements on both `CTS-E` and `BN-E` tasks. fMPE gave a further improvement of 1.1% and 1.3% on `dev04` and `eval03` respectively. In general, the performance gains obtained from SAT and SAT+SPAM on both HLDA and GAUSS frontends

were similar. SAT improved the CER performance by 0.6–0.9% absolute. SAT+SPAM further reduced the CER by 0.5–0.9%. The improvement from fMPE+SAT and fMPE+SAT+SPAM systems was smaller. The fMPE+SAT system is 0.5–0.7% better than the fMPE system while the fMPE+SAT+SPAM system is only 0.3% better than fMPE+SAT on `dev04` but no gains on `eval03`. Table 9.8 also shows the 2-way combination results between SAT and SAT+SPAM systems with different frontends. With the HLDA frontend, combining SAT and SAT+SPAM gave a further 0.8% improvement compared to the best individual system. However, the combination gains for the GAUSS and fMPE systems were smaller, only 0.3–0.5% absolute CER reduction. The overall combined system performance was improved by 1.0–1.5% when Gaussianised frontend was used. A further improvement of 0.5–0.6% was obtained using the fMPE frontend.

10

Conclusions and Future Work

This thesis has investigated several forms of precision matrix models for continuous density hidden Markov models (CDHMMs) with application to continuous speech recognition. Two major contributions are summarised in Sections 10.1 and 10.2 respectively. The first looks at improved spatial correlation modelling using structured precision matrix approximation schemes to achieve a compact model representation. The second contribution involves the investigation of the time varying precision matrices as a semi-parametric trajectory model. Suggestions for possible future research are also given in Section 10.3.

10.1 Structured Precision Matrix Approximations

Most CDHMM-based speech recognition systems employ multivariate Gaussian mixture models to represent the output probability density functions. A standard problem when modelling multivariate continuous distributions is how to accurately model the correlations between feature elements. Various forms of covariance and precision matrix approximation schemes were reviewed in Chapter 3. In general, approximating the precision matrix structures results in models which are computationally more efficient because the likelihood function is directly expressed in terms of the precision matrices. Approximating the covariance matrix on the other hand requires matrix inversion in likelihood calculation. Specifically, the Semi-Tied Covariances (STC), Extended Maximum Likelihood Linear Transform (EMLLT) and Subspace for Precision and Mean (SPAM) models have been successfully applied to speech recognition tasks with promising improvements over the diagonal covariance matrix approximation approach. Chapter 3 also introduced a generic framework of basis superposition which unifies these precision matrix approximation schemes within a consistent formulation. The goal is to extract the fundamental structures that co-exist within the precision matrices in the system. According to this frame-

work, the precision matrices are modelled by superimposing a set of symmetric matrix bases (*global* parameters), weighted by a corresponding set of basis coefficients which are Gaussian component specific. By compressing as much correlation information as possible into a small set of bases, the effective number of basis coefficients in the system is greatly reduced, thereby yielding a compact model representation. The experimental results presented in Chapter 8 revealed the importance of compact model representation to ensure robust parameters estimation. In particular, the SPAM model, with the most compact precision matrix approximation, gave the best recognition performance.

Previous literatures have presented the STC, EMLLT and SPAM models which are trained using the Maximum Likelihood (LM) criterion. The ML estimation formulae for these models are briefly discussed in Chapter 4. In the same chapter, various implementation issues were studied. These include the computational and memory requirements, initialisation schemes, variance flooring and alternative parameters tying schemes. These aspects were carefully addressed to ensure efficient application of these models in Large Vocabulary Continuous Speech Recognition (LVCSR). Chapter 5 employed the Minimum Phone Error (MPE) training criterion to discriminatively train the precision matrix models. Adaptation and adaptive training of various precision matrix models using the Maximum Likelihood Linear Regression (MLLR) techniques were also discussed in Chapter 6 to produce state-of-the-art performance. As shown in Chapter 8, the gains from precision matrix modelling were almost retained after MPE training and MLLR adaptation.

In general, various precision matrix models were found to yield consistent improvements over several speech transcription tasks, namely the conversational telephone speech English and Mandarin tasks as well as the broadcast news English task. These models have been successfully applied with other techniques such as Heteroscedastic Linear Discriminant Analysis (HLDA), Gaussianisation and Speaker Adaptive Training (SAT). Chapter 8 presented a range of experimental results various precision matrix models evaluated in a number of evaluation setups, including an advanced multi-pass multi-branch evaluation framework. In particular, the speaker adaptively trained SPAM models were shown to improve the standard CUHTK evaluation systems on various transcription tasks.

10.2 Semi-parametric Trajectory Model

The second part of this thesis proposed an alternative form of precision matrix model. Instead of modelling the feature correlations in the acoustic space, a novel approach, called pMPE, attempts to capture the temporally varying attributes in the precision matrix structures. The pMPE model may also be described in the basis superposition framework. In contrast to the previous formulation, the bases now describe the common precision matrix structures over time (*spa-*

tial versus *temporal* superposition). The time dependent basis coefficients are derived based on the posterior probabilities of a set of centroids, motivated by the formulation of the fMPE technique, which has been found to yield improvements over the standard MPE trained systems. The fMPE and pMPE techniques were presented collectively as a semi-parametric trajectory model in Chapter 7. Two parameter estimation procedures and some implementation issues were also discussed. In this work, a trade off between the HMM system complexity and the improvements obtained from fMPE and pMPE was observed. Specifically, the gains from pMPE quickly saturates as the system complexity increases. Nevertheless, the pMPE technique was found to yield small gains over the MPE alone systems. Marginal gains were also obtained when combining the fMPE and pMPE systems using confusion network combination.

10.3 Future Work

There are several possible further research which is beyond the scope of this work, either in terms of improvements to current work or applications in other domains. Several suggestions of these are listed below.

- Most of the research on covariance and precision matrix modelling techniques concern the application of these techniques to HMM-based speech recognition tasks. In general, the basis superposition precision matrix modelling framework may be applied to efficiently approximate a large number of full precision matrices. One of the on-going research along this line being pursued is its application in the area of speaker verification and identification [12, 136].

- This work has emphasised and shown the importance of compact model representation to allow for robust estimation and improved performance. The key element is to find a powerful set of bases which capture as much common structures as possible. This is usually realised at the expanse of increased difficulty in learning these bases and complexity in likelihood computation. This is clearly evident when increasing the modelling power from STC to EMLLT to SPAM. More complicated approximation structures such as the hierarchical correlation compensation (HCC) model [72] has also been reported to yield promising gains. As the model compactness increases, the trainability and its convergence rate reduces dramatically. The performance of these models may be improved by seeking for better initialisation and training schemes [87].

- The semi-parametric trajectory model presented in Chapter 7 of this thesis is a relatively new concept. There are plenty of room for improvements. The research on fMPE technique by itself is actively pursued [59, 96, 126]. The small gains from pMPE may have been due to the simple diagonal basis structure assumed in this work. More general basis structure

may be explored to exploit the full potential of pMPE. A major issue with such generalisation is the dramatical increase in the computational cost in computing the effective full precision matrix, and hence the likelihood at each time. An efficient likelihood computation which takes advantage of the basis superposition structure is required. Finally, to completely model the semi-parametric trajectory, the time varying attributes of the component weights associated with each Gaussian component in a GMM may be modelled. The time dependent weights may be formulated using a basis superposition framework with the superposition weights derived from the centroid posterior probabilities, similar to fMPE and pMPE.

Appendix

A Useful Matrix Algebra

This section provides some useful matrix algebra used in the derivation of the update formulae for various precision matrix models. This includes the matrix inversion and determinant identities as well as some vector and matrix differentiation formulae.

Matrix Inversion and Determinant Identities

This is useful when inverting a matrix of the form $\boldsymbol{A} + \boldsymbol{BDC}$. Its inverse is given by:

$$(\boldsymbol{A} + \boldsymbol{BDC})^{-1} = \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{B}(\boldsymbol{D}^{-1} + \boldsymbol{CA}^{-1}\boldsymbol{B})^{-1}\boldsymbol{CA}^{-1} \tag{A-1}$$

Figure A-1: *Matrix Inversion Lemma*

where $\boldsymbol{A}$ is $n \times n$, $\boldsymbol{B}$ is $n \times m$, $\boldsymbol{C}$ is $m \times n$ and $\boldsymbol{D}$ is $m \times m$. In the case where $m = 1$ ($\boldsymbol{D} = d$), equation (A-1) becomes

$$(\boldsymbol{A} + d\boldsymbol{bc'})^{-1} = \boldsymbol{A}^{-1} - \frac{d\boldsymbol{A}^{-1}\boldsymbol{bc'}\boldsymbol{A}^{-1}}{(1 + d\boldsymbol{c'}\boldsymbol{A}^{-1}\boldsymbol{b})} \tag{A-2}$$

Figure A-2: *Matrix Inversion Lemma with ($\boldsymbol{D} = d$)*

and its determinant can be found using

$$|\boldsymbol{A} + d\boldsymbol{bc'}| = (1 + d\boldsymbol{b}\boldsymbol{A}^{-1}\boldsymbol{c'})|\boldsymbol{A}| \tag{A-3}$$

Figure A-3: *Matrix Determinant Lemma*

Vector and Matrix Differentiation Formulae

First, let us define $\boldsymbol{A}$ and $\boldsymbol{b}$ as $d \times d$ square matrix and $d \times 1$ column vector respectively. x is a scalar variable. The following are commonly used identities:

$$\frac{\partial(\boldsymbol{A}\boldsymbol{b})}{\partial \boldsymbol{b}} = \boldsymbol{A}' \quad \text{(A-4)}$$

$$\frac{\partial(\boldsymbol{b}'\boldsymbol{A}\boldsymbol{b})}{\partial \boldsymbol{b}} = (\boldsymbol{A} + \boldsymbol{A}')\boldsymbol{b} \quad \text{(A-5)}$$

$$\frac{\partial \log |\boldsymbol{A}|}{\partial x} = \mathrm{Tr}\left(\frac{\partial \boldsymbol{A}}{\partial x}\boldsymbol{A}^{-1}\right) \quad \text{(A-6)}$$

Figure A-4: *Matrix differentiation*

B Initialisation for EMLLT transform

This form of initialisation assumes no information about the statistics of the data. A $k \times n$ EMLLT transform can be initialised by condensing a set of $\frac{n}{2}(n+1)$ vectors that spans the symmetric matrix space into the transform. First let $\mathcal{V}$, $\mathcal{I}$ and $\mathcal{J}$ be sets of n dimensional vectors such that vectors in $\mathcal{V}$ span the symmetric matrix space, vectors in $\mathcal{I}$ form an identity matrix, $\mathcal{V} = \mathcal{I} \cup \mathcal{J}$ and $\mathcal{I} \cap \mathcal{J} = O$. So, the EMLLT transform

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}^{(1)} \\ \boldsymbol{A}^{(2)} \end{bmatrix} \quad \text{(B-1)}$$

is initialised with $\boldsymbol{A}^{(1)}$ set to $\boldsymbol{I}_n$ and $\boldsymbol{A}^{(2)}$ is constructed by condensing the vectors in $\mathcal{J}$ into it. There are many ways of defining $\mathcal{V}$ and condensing its vectors into $\boldsymbol{A}^{(2)}$. In the following, I will describe one possible implementation using a simple case where $k = 5$ and $n = 3$. $\mathcal{I}$ and $\mathcal{J}$ can be defined as

$$\mathcal{I} = \left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right\} \quad \text{(B-2)}$$

$$\mathcal{J} = \left\{ \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \right\} \quad \text{(B-3)}$$

$\boldsymbol{A}^{(2)}$ is filled up row by row with the vectors in $\mathcal{J}$. After the final row of $\boldsymbol{A}^{(2)}$ is filled, the next vector in $\mathcal{J}$ is added to the first row and so on until all the vectors in $\mathcal{J}$ are consumed. This gives

$$\boldsymbol{A}^{(2)} = \begin{bmatrix} 2 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} \quad \text{(B-4)}$$

C Precision Matrices and Conditional Independence

Consider d random variables, $\{x_1, x_2, \ldots, x_d\}$ such that x_i is conditionally dependent only on x_j where $j < i$. Thus, these random variables can be written as

$$x_i = \sum_{j=1}^{i-1} c_{ij} x_j + w_i \tag{C-1}$$

where c_{ij} indicates the conditional dependency between x_i and x_j, $w_i \sim \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i^2)$ are d independent random sources. This can be rewritten as

$$\begin{aligned} \boldsymbol{x} &= \boldsymbol{C}\boldsymbol{x} + \boldsymbol{w} \\ &= (\boldsymbol{I} - \boldsymbol{C})^{-1}\boldsymbol{w} \\ &= \boldsymbol{L}^{-1}\boldsymbol{w} \end{aligned} \tag{C-2}$$

where $\boldsymbol{C}$ and $\boldsymbol{L}$ are a lower triangular matrices given by

$$\boldsymbol{C} = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ c_{21} & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ c_{d1} & \cdots & c_{d(d-1)} & 0 \end{pmatrix} \quad \boldsymbol{L} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ -c_{21} & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ -c_{d1} & \cdots & -c_{d(d-1)} & 1 \end{pmatrix} \tag{C-3}$$

The random variables are arranged such that if $c_{ij} = 0$, then $c_{kj} = 0$ for $k > i$. Hence, the precision matrix can then be expressed as

$$\boldsymbol{P} = \boldsymbol{L}'\boldsymbol{\Lambda}\boldsymbol{L} \tag{C-4}$$

where $\boldsymbol{\Lambda}$ is a diagonal matrix whose ith diagonal element is given by $\lambda_{ii} = 1/\sigma_i^2$. Thus, p_{ij} is given by

$$p_{ij} = \sum_{k=i}^{d} \lambda_{kk} c_{ki} c_{kj} \tag{C-5}$$

If x_i and x_j are conditionally independent, $c_ij = 0$. Thus, $p_{ij} = 0$. In other words, a zero off-diagonal precision matrix element indicates conditional independence between the variables corresponding to the row and column indexes.

D STC Parameters Update Formula

The precision matrix expression for a STC model is given by equation (3.23). Substituting this into equation (4.9) yields the following auxiliary function:

$$\begin{aligned}\mathcal{Q}(\boldsymbol{\theta},\hat{\boldsymbol{\theta}}) &= K+\frac{1}{2}\sum_{s=1}^{S}\sum_{m=1}^{M}\beta_{sm}^{\mathtt{ml}}\left\{2\log|\boldsymbol{A}|+\log|\boldsymbol{\Lambda}_{sm}|-\sum_{i=1}^{n}\lambda_{smi}\boldsymbol{a}_i\boldsymbol{W}_{sm}^{\mathtt{ml}}\boldsymbol{a}_i'\right\}\\ &= K+\frac{1}{2}\sum_{s=1}^{S}\sum_{m=1}^{M}\beta_{sm}^{\mathtt{ml}}\left\{2\log|\boldsymbol{A}|+\sum_{i=1}^{n}\log(\lambda_{smi})-\lambda_{smi}\boldsymbol{a}_i\boldsymbol{W}_{sm}^{\mathtt{ml}}\boldsymbol{a}_i'\right\}\\ &= K+\frac{1}{2}\sum_{s=1}^{S}\sum_{m=1}^{M}\beta_{sm}^{\mathtt{ml}}\left\{2\log(\boldsymbol{a}_i\boldsymbol{c}_i)+\sum_{i=1}^{n}\log(\lambda_{smi})-\lambda_{smi}\boldsymbol{a}_i\boldsymbol{W}_{sm}^{\mathtt{ml}}\boldsymbol{a}_i'\right\}\end{aligned} \tag{D-1}$$

where $\boldsymbol{c}_i$ is a column vector of cofactors that correspond to the ith row of $\boldsymbol{A}$. The basis vectors and coefficients can be updated in an alternating fashion, each time keeping the other parameters constant. Thus, an iterative update can be formulated as described in the following.

To update the basis coefficients, simply differentiate equation (D-1) w.r.t. λ_{smi} and equate to zero:

$$\frac{\partial\mathcal{Q}}{\partial\lambda_{smi}}=\frac{\beta_{sm}^{\mathtt{ml}}}{2}\left\{\frac{1}{\lambda_{smi}}-\boldsymbol{a}_i\boldsymbol{W}_{sm}^{\mathtt{ml}}\boldsymbol{a}_i'\right\}=0 \tag{D-2}$$

This yields the ML update formula for λ_{smi} as

$$\lambda_{smi}=\frac{1}{\boldsymbol{a}_i\boldsymbol{W}_{sm}^{\mathtt{ml}}\boldsymbol{a}_i'}=\frac{1}{\mathrm{diag}(\boldsymbol{A}\boldsymbol{W}_{sm}^{\mathtt{ml}}\boldsymbol{A}')} \tag{D-3}$$

Thus, the basis coefficients given by the ML estimator is simple the inverse variance of the projected covariance statistics.

Similarly, the basis vectors can be updated by differentiating equation (D-1) w.r.t. $\boldsymbol{a}_i$ and equate to zero:

$$\begin{aligned}\frac{\partial\mathcal{Q}}{\partial\boldsymbol{a}_i} &= \sum_{s=1}^{S}\sum_{m=1}^{M}\beta_{sm}^{\mathtt{ml}}\left\{\frac{\boldsymbol{c}_i'}{\boldsymbol{a}_i\boldsymbol{c}_i}-\lambda_{smi}\boldsymbol{a}_i\boldsymbol{W}_{sm}^{\mathtt{ml}}\right\}\\ &= \frac{\beta\boldsymbol{c}_i'}{\boldsymbol{a}_i\boldsymbol{c}_i}-\boldsymbol{a}_iG_i\\ &= \frac{\boldsymbol{G}_i(\beta\boldsymbol{c}_i'\boldsymbol{G}_i^{-1}-\boldsymbol{a}_i)}{\boldsymbol{a}_i\boldsymbol{c}_i}=0\end{aligned} \tag{D-4}$$

where the required statistics are given by

$$\boldsymbol{G}_i = \sum_{s=1}^{S}\sum_{m=1}^{M}\beta_{sm}^{\mathtt{ml}}\lambda_{smi}\boldsymbol{W}_{sm}^{\mathtt{ml}} \tag{D-5}$$

$$\beta = \sum_{s=1}^{S}\sum_{m=1}^{M}\beta_{sm}^{\mathtt{ml}} \tag{D-6}$$

It is clear that equation (D-4) is true if and only if $\boldsymbol{c}_i'\boldsymbol{G}_i^{-1}$ is parallel to $\boldsymbol{a}_i$. So,

$$\boldsymbol{a}_i = \alpha \boldsymbol{c}_i'\boldsymbol{G}_i^{-1} \tag{D-7}$$

where α is a scalar. Substituting this back into equation (D-4) yields

$$\alpha = \sqrt{\frac{\beta}{\boldsymbol{c}_i'\boldsymbol{G}_i^{-1}\boldsymbol{c}_i}} \tag{D-8}$$

and hence,

$$\boldsymbol{a}_i = \boldsymbol{c}_i'\boldsymbol{G}_i^{-1}\sqrt{\frac{\beta}{\boldsymbol{c}_i'\boldsymbol{G}_i^{-1}\boldsymbol{c}_i}} \tag{D-9}$$

E EMLLT Parameters Update Formula

Since EMLLT is an extension to STC, similar iterative update formulation is applicable when updating the EMLLT parameters, alternating between the basis vectors and coefficients. However, the rectangular EMLLT transformation matrix $\boldsymbol{A}$ does not allow simplification of the determinant term in equation (4.9). Consequently, a standard optimisation routine is used to optimise the basis vectors. On the other hand, the basis coefficients can be updated using an efficient closed-form solution. In the following, the closed form update formula for λ_{smi} will be derived. Then, the optimisation of basis vectors based on the gradient descent method will be described.

There are two forms of basis coefficient update, namely the *additive* and *multiplicative* updates. The former results in both positive and negative coefficients while the latter restricts the coefficients to be strictly positive. The additive update has been found to yield better results. Hence, only discuss the additive update will be discussed.

Firstly, let the new estimate of λ_{smi} be written as

$$\hat{\lambda}_{smi} = \lambda_{smi} + \Delta_{smi} \tag{E-1}$$

Thus, the auxiliary function in equation (4.9) can be rewritten as

$$\begin{aligned}
\mathcal{Q}_\Delta(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) &= \mathcal{Q}_\Delta(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\theta}}) + \frac{1}{2}\sum_{s=1}^{S}\sum_{m=1}^{M} \beta_{sm}^{\mathtt{ml}} \left\{ \log\left| \frac{\boldsymbol{P}_{sm} + \Delta_{smi}\boldsymbol{a}_i'\boldsymbol{a}_i}{\boldsymbol{P}_{sm}} \right| - \Delta_{smi}\boldsymbol{a}_i\boldsymbol{W}_{sm}^{\mathtt{ml}}\boldsymbol{a}_i' \right\} \\
&= \mathcal{Q}_\Delta(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\theta}}) + \frac{1}{2}\sum_{s=1}^{S}\sum_{m=1}^{M} \beta_{sm}^{\mathtt{ml}} \left\{ \log(1 + \Delta_{smi}\boldsymbol{a}_i\boldsymbol{\Sigma}_{sm}\boldsymbol{a}_i') - \Delta_{smi}\boldsymbol{a}_i\boldsymbol{W}_{sm}^{\mathtt{ml}}\boldsymbol{a}_i' \right\}
\end{aligned} \tag{E-2}$$

Differentiating this w.r.t. Δ_{smi} yields

$$\frac{\partial \mathcal{Q}}{\partial \Delta_{smi}} = \frac{\boldsymbol{a}_i\boldsymbol{\Sigma}_{sm}\boldsymbol{a}_i'}{1 + \Delta_{smi}\boldsymbol{a}_i\boldsymbol{\Sigma}_{sm}\boldsymbol{a}_i'} - \boldsymbol{a}_i\boldsymbol{W}_{sm}^{\mathtt{ml}}\boldsymbol{a}_i' = 0 \tag{E-3}$$

and equating to zero gives the ML solution of Δ_{smi} as

$$\Delta_{smi} = \frac{1}{\boldsymbol{a}_i \boldsymbol{W}_{sm}^{\mathtt{ml}} \boldsymbol{a}_i'} - \frac{1}{\boldsymbol{a}_i \boldsymbol{\Sigma}_{sm} \boldsymbol{a}_i'} \tag{E-4}$$

It is important to update the covariance matrix after the update of λ_{smi} to ensure that the subsequent updates are correct. This can be done easily using the Matrix Inversion Lemma:

$$\boldsymbol{\Sigma}_{sm} = \left(\boldsymbol{\Sigma}_{sm}^{-1} + \Delta_{smi} \boldsymbol{a}_i' \boldsymbol{a}_i\right)^{-1} = \boldsymbol{\Sigma}_{sm} - \frac{\Delta_{smi} \boldsymbol{\Sigma}_{sm} \boldsymbol{a}_i' \boldsymbol{a}_i \boldsymbol{\Sigma}_{sm}}{1 + \Delta_{smi} \boldsymbol{a}_i \boldsymbol{\Sigma}_{sm} \boldsymbol{a}_i'} \tag{E-5}$$

As previously mentioned, the update of basis vectors does not have a closed form solution. In this report, the update of the basis vectors will be described using the gradient descent method. To do this, the gradient vector as well as the Hessian matrix have to be computed. The new basis vector $\boldsymbol{a}_i$ can be expressed in terms of the old one as:

$$\hat{\boldsymbol{a}}_i = \boldsymbol{a}_i + \boldsymbol{\Delta}_i \tag{E-6}$$

So, the resulting precision matrices can be written as

$$\hat{\boldsymbol{P}}_{sm} = \boldsymbol{P}_{sm} + \lambda_{smi} (\hat{\boldsymbol{a}}_i' \hat{\boldsymbol{a}}_i - \boldsymbol{a}_i' \boldsymbol{a}_i) \tag{E-7}$$

Using the Sherman-Morrison-Woodbury formula, the determinant of the new precision matrix is given by

$$\mathcal{G}(\hat{\boldsymbol{a}}_i) = \frac{|\hat{\boldsymbol{P}}_{sm}|}{|\boldsymbol{P}_{sm}|} = (1 - \lambda_{smi} \boldsymbol{a}_i \boldsymbol{\Sigma}_{sm} \boldsymbol{a}_i')(1 + \lambda_{smi} \hat{\boldsymbol{a}}_i \boldsymbol{\Sigma}_{sm} \hat{\boldsymbol{a}}_i') + \left(\lambda_{smi} (\hat{\boldsymbol{a}}_i \boldsymbol{\Sigma}_{sm} \boldsymbol{a}_i')\right)^2 \tag{E-8}$$

Thus, the auxiliary function in equation (4.9) can be expressed in terms of $\hat{\boldsymbol{a}}_i$ and $\boldsymbol{a}_i$ as

$$\mathcal{Q}_A^{(i)}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) = \mathcal{Q}_A^{(i)}(\boldsymbol{\theta}, \boldsymbol{\theta}) + \frac{1}{2} \sum_{s=1}^{S} \sum_{m=1}^{M} \beta_{sm}^{\mathtt{ml}} \left\{ \log \frac{|\hat{\boldsymbol{P}}_{sm}|}{|\boldsymbol{P}_{sm}|} - \lambda_{smi} (\hat{\boldsymbol{a}}_i \boldsymbol{W}_{sm}^{\mathtt{ml}} \hat{\boldsymbol{a}}_i' - \boldsymbol{a}_i \boldsymbol{W}_{sm}^{\mathtt{ml}} \boldsymbol{a}_i') \right\} \tag{E-9}$$

To perform gradient descent optimisation, the gradient vector and Hessian matrix of the auxiliary function are needed:

$$\boldsymbol{f}_i = \frac{\partial \mathcal{Q}}{\partial \hat{\boldsymbol{a}}_i} = \frac{1}{2} \sum_{s=1}^{S} \sum_{m=1}^{M} \beta_{sm}^{\mathtt{ml}} \left\{ \frac{\mathcal{G}'(\hat{\boldsymbol{a}}_i)}{\mathcal{G}(\hat{\boldsymbol{a}}_i)} - 2\lambda_{smi} \hat{\boldsymbol{a}}_i \boldsymbol{W}_{sm}^{\mathtt{ml}} \right\} \tag{E-10}$$

$$\boldsymbol{H}_i = \frac{\partial^2 \mathcal{Q}}{\partial \hat{\boldsymbol{a}}_i' \partial \hat{\boldsymbol{a}}_i} = \frac{1}{2} \sum_{s=1}^{S} \sum_{m=1}^{M} \beta_{sm}^{\mathtt{ml}} \left\{ \frac{\mathcal{G}(\hat{\boldsymbol{a}}_i) \mathcal{G}''(\hat{\boldsymbol{a}}_i) - [\mathcal{G}'(\hat{\boldsymbol{a}}_i)]' [\mathcal{G}'(\hat{\boldsymbol{a}}_i)]}{\mathcal{G}(\hat{\boldsymbol{a}}_i)^2} - 2\lambda_{smi} \boldsymbol{W}_{sm}^{\mathtt{ml}} \right\} \tag{E-11}$$

where

$$\mathcal{G}'(\hat{\boldsymbol{a}}_i) = 2 \left\{ \lambda_{smi} \hat{\boldsymbol{a}}_i \boldsymbol{\Sigma}_{sm} + \lambda_{smi}^2 \left(\boldsymbol{a}_i \boldsymbol{\Sigma}_{sm} \boldsymbol{a}_i' \boldsymbol{a}_i \boldsymbol{\Sigma}_{sm} - \boldsymbol{a}_i \boldsymbol{\Sigma}_{sm} \boldsymbol{a}_i' \boldsymbol{a}_i \boldsymbol{\Sigma}_{sm} \right) \right\} \tag{E-12}$$

$$\mathcal{G}''(\hat{\boldsymbol{a}}_i) = 2 \left\{ \lambda_{smi} \boldsymbol{\Sigma}_{sm} + \lambda_{smi}^2 \left(\boldsymbol{\Sigma}_{sm} \boldsymbol{a}_i' \boldsymbol{a}_i \boldsymbol{\Sigma}_{sm} - \boldsymbol{a}_i \boldsymbol{\Sigma}_{sm} \boldsymbol{a}_i' \boldsymbol{\Sigma}_{sm} \right) \right\} \tag{E-13}$$

Given the current estimate of the basis vector, $\hat{\boldsymbol{a}}_i$

$$\mathcal{G}(\boldsymbol{a}_i) = 1 \tag{E-14}$$

$$\mathcal{G}'(\boldsymbol{a}_i) = 2\lambda_{smi} \boldsymbol{a}_i \boldsymbol{\Sigma}_{sm} \tag{E-15}$$

$$\mathcal{G}''(\boldsymbol{a}_i) = 2\lambda_{smi} \boldsymbol{\Sigma}_{sm} \left(1 - \boldsymbol{a}_i \boldsymbol{\Sigma}_{sm} \boldsymbol{a}_i' + \lambda_{smi} \boldsymbol{a}_i' \boldsymbol{a}_i \boldsymbol{\Sigma}_{sm}\right) \tag{E-16}$$

Hence, the gradient vector and Hessian matrix at the current estimate is given by

$$\bar{\boldsymbol{f}}_i = \boldsymbol{f}_i\Big|_{\hat{\boldsymbol{a}}_i=\boldsymbol{a}_i} = \sum_{s=1}^{S}\sum_{m=1}^{M} \beta_{sm}^{\mathtt{ml}}\lambda_{smi}\boldsymbol{a}_i\left(\boldsymbol{\Sigma}_{sm} - \boldsymbol{W}_{sm}^{\mathtt{ml}}\right) \tag{E-17}$$

$$\bar{\boldsymbol{H}}_i = \boldsymbol{H}_i\Big|_{\hat{\boldsymbol{a}}_i=\boldsymbol{a}_i} = \sum_{s=1}^{S}\sum_{m=1}^{M} \beta_{sm}^{\mathtt{ml}}\lambda_{smi}\left\{\boldsymbol{\Sigma}_{sm}\left(1-\boldsymbol{a}_i\boldsymbol{\Sigma}_{sm}\boldsymbol{a}_i'\right) - \boldsymbol{W}_{sm}^{\mathtt{ml}} - \lambda_{smi}\boldsymbol{\Sigma}_{sm}\boldsymbol{a}_i'\boldsymbol{a}_i\boldsymbol{\Sigma}_{sm}\right\} \tag{E-18}$$

and the new estimate of the basis vector is given by

$$\hat{\boldsymbol{a}}_i = \boldsymbol{a}_i + \eta\bar{\boldsymbol{f}}_i\bar{\boldsymbol{H}}_i^{-1} \tag{E-19}$$

where η is a scalar that determines the step size of each iteration. This is usually initialised as unity and gradually decreasing its value until the new estimate of the basis vector yields an increase in the auxiliary function.

F SPAM Parameters Update Formula

SPAM model[7] is a generalisation to EMLLT models where the rank one constraint on the basis matrices is removed. Thus, the precision matrix of a SPAM model can be written as

$$\boldsymbol{P}_{sm} = \sum_{i=1}^{n}\lambda_{smi}\boldsymbol{S}_i \tag{F-1}$$

where $\boldsymbol{S}_i$, is an arbitrary symmetric matrix provided the final precision matrix is positive definite. Substitute the expression of precision matrix into equation (4.9) leads to the following modified auxiliary function

$$\mathcal{Q}(\boldsymbol{\theta},\hat{\boldsymbol{\theta}}) = K + \frac{1}{2}\sum_{s=1}^{S}\sum_{m=1}^{M}\beta_{sm}^{\mathtt{ml}}\left\{\log\left(|\boldsymbol{P}_{sm}|\right) - \sum_{i=1}^{n}\lambda_{smi}\,\mathrm{Tr}(\boldsymbol{W}_{sm}^{\mathtt{ml}}\boldsymbol{S}_i)\right\} \tag{F-2}$$

There is no closed form solution to optimise equation (F-2) w.r.t. the model parameters. However, the basis coefficients, λ_{smi} can be updated using an efficient conjugate gradient algorithm[7]. The following describes the coefficient update using the Polak-Ribeire conjugate-gradient method[94]. First, let

$$\hat{\boldsymbol{P}}_{sm} = \boldsymbol{P}_{sm} + \Delta_{smi}\boldsymbol{S}_i \tag{F-3}$$

where $\boldsymbol{P}_{sm}$ and $\hat{\boldsymbol{P}}_{sm}$ denote the current and new estimate of the precision matrix. Thus, the auxiliary function with respect to the mth component, sth state and ith basis matrix can be expressed as

$$\mathcal{Q}_{sm}(\boldsymbol{\theta},\hat{\boldsymbol{\theta}}) = K + \frac{\beta_{sm}^{\mathtt{ml}}}{2}\Big\{\log\left(|\boldsymbol{P}_{sm}+\Delta_{smi}\boldsymbol{S}_i|\right) - \mathrm{Tr}(\boldsymbol{W}_{sm}^{\mathtt{ml}}\boldsymbol{P}_{sm}) - \Delta_{smi}\,\mathrm{Tr}(\boldsymbol{W}_{sm}^{\mathtt{ml}}\boldsymbol{S}_i)\Big\} \tag{F-4}$$

Differentiate w.r.t. Δ_{smi} about the point $\Delta_{smi} = 0$ gives

$$\bar{f}_{smi} = \frac{\partial \mathcal{Q}^{(i)}}{\partial \Delta_{smi}}\Big|_{\Delta_{smi}=0} = \frac{\beta^{\texttt{ml}}_{sm}}{2}\Big\{\mathrm{Tr}(\boldsymbol{P}^{-1}_{sm}\boldsymbol{S}_i) - \mathrm{Tr}(\boldsymbol{W}^{\texttt{ml}}_{sm}\boldsymbol{S}_i)\Big\} \quad \text{(F-5)}$$

$$= \frac{\beta^{\texttt{ml}}_{sm}}{2}\Big\{\mathrm{Tr}\left((\boldsymbol{P}^{-1}_{sm} - \boldsymbol{W}^{\texttt{ml}}_{sm})\boldsymbol{S}_i\right)\Big\} \quad \text{(F-6)}$$

This gives gradient of the auxiliary function w.r.t., λ_{smi}. Given the gradient vector, $\bar{\boldsymbol{f}}^{(k)}_{sm}$, at the kth iteration, the conjugate gradient direction vector, $\boldsymbol{d}^{(k)}_{sm}$, is then given by

$$\boldsymbol{d}^{(k)}_{sm} = \bar{\boldsymbol{f}}^{(k)}_{sm} + \eta_k \boldsymbol{d}^{(k-1)}_{sm} \quad \text{(F-7)}$$

where

$$\eta_k = \frac{\bar{\boldsymbol{f}}^{(k)'}_{sm}\left(\bar{\boldsymbol{f}}^{(k)}_{sm} - \bar{\boldsymbol{f}}^{(k-1)}_{sm}\right)}{\bar{\boldsymbol{f}}^{(k-1)'}_{sm}\bar{\boldsymbol{f}}^{(k-1)}_{sm}} \quad \text{(F-8)}$$

and $\boldsymbol{d}^{(0)}_{sm} = \boldsymbol{0}$. Once the direction vector is found, the optimisation simplifies to a one dimensional line search problem where

$$\hat{\boldsymbol{P}}_{sm} = \boldsymbol{P}_{sm} + \Delta_{sm}\boldsymbol{R}_m \quad \text{(F-9)}$$

$$\boldsymbol{R}_m = \sum_{i=1}^{n} d_{smi}\boldsymbol{S}_i \quad \text{(F-10)}$$

and d_{smi} is the i^{th} element of the direction vector, $\boldsymbol{d}_{sm}$, given by equation (F-7). Thus, the change in the auxiliary function for component m in state s can be written as

$$\mathcal{Q}_{sm} - \hat{\mathcal{Q}}_s = \frac{\beta^{\texttt{ml}}_{sm}}{2}\left\{\log\left(\left|\frac{\boldsymbol{P}_{sm} + \Delta_{sm}\boldsymbol{R}_m}{\boldsymbol{P}_{sm}}\right|\right) - \Delta_{sm}\,\mathrm{Tr}(\boldsymbol{W}^{\texttt{ml}}_{sm}\boldsymbol{R}_m)\right\} \quad \text{(F-11)}$$

$$= \frac{\beta^{\texttt{ml}}_{sm}}{2}\left\{\log\left(\left|\boldsymbol{I} + \Delta_{sm}\boldsymbol{P}^{-\frac{1}{2}}_{sm}\boldsymbol{R}_m\boldsymbol{P}^{-\frac{1}{2}}_{sm}\right|\right) - \Delta_{sm}\,\mathrm{Tr}(\boldsymbol{W}^{\texttt{ml}}_{sm}\boldsymbol{R}_m)\right\} \quad \text{(F-12)}$$

$$= \frac{\beta^{\texttt{ml}}_{sm}}{2}\left\{\sum_{j=1}^{d}\log\left(1 + \Delta_{sm}w_{smj}\right) - \Delta_{sm}\,\mathrm{Tr}(\boldsymbol{W}^{\texttt{ml}}_{sm}\boldsymbol{R}_m)\right\} \quad \text{(F-13)}$$

where w_{smj} is the j^{th} eigenvalue of $\boldsymbol{P}^{-\frac{1}{2}}_{sm}\boldsymbol{R}_m\boldsymbol{P}^{-\frac{1}{2}}_{sm}$. The line search can be further constrained to ensure that $1 + \Delta_{sm}w_{smj} > 0$. Hence, the line search is confined to be within the range $-1/w^{(max)}_{sm} < \Delta_{sm} < -1/w^{(min)}_{sm}$, where $w^{(max)}_{sm}$ and $w^{(min)}_{sm}$ are the largest positive and negative eigenvalues. If $w^{(max)}_{sm}$ ($w^{(min)}_{sm}$) does not exist, the range of Δ_{sm} is then unbounded from below (above).

G Checks for fMPE and pMPE Differentials

A crucial aspect of training the fMPE and pMPE models is the determination of the differentials used in the gradient-based optimisation (see Chapter 7). It is important to ensure that the

fMPE and pMPE parameters are used to model the dynamic (temporally varying) aspects of the system. For the interleaved update presented in Section 7.3.2.1, this is achieved by using the *complete* differentials so that the static aspects are accounted for by the static parameters in the subsequent ML training iteration. In the following, the fMPE and pMPE differentials will be shown to learn only the temporally varying attributes of the system. This is realised by showing that the differentials are zero when only one centroid is used for fMPE and pMPE, because more than one centroid is required to yield a trajectory model. This also provides convenient checks for the fMPE and pMPE implementations. The derivation will proceed as follows. By setting the number of centroids, $n = 1$, the corresponding posterior probability will always be $h_1(t) = 1$.

Checks for fMPE Differentials

The checks for fMPE differentials are given by [97]:

$$0 = \sum_{s=1}^{S}\sum_{m=1}^{M}\sum_{t=1}^{T}\left(\left.\frac{\partial \mathcal{Q}_{smt}^{\text{mpe}}}{\partial b_j^{(i)}}\right|_{h_1(t)=1} + \left.\frac{\partial \mathcal{Q}_{smt}^{\text{mpe}}}{\partial \mu_{smj}}\frac{\partial \mu_{smj}}{\partial b_j^{(i)}}\right|_{h_1(t)=1}\right) \tag{G-1}$$

$$0 = \sum_{s=1}^{S}\sum_{m=1}^{M}\sum_{t=1}^{T}\left.\frac{\partial \mathcal{Q}_{smt}^{\text{mpe}}}{\partial \sigma_{smj}^2}\frac{\partial \sigma_{smj}^2}{\partial b_j^{(i)}}\right|_{h_1(t)=1} \tag{G-2}$$

The complete fMPE differential is given by the sum of the RHS of equations (G-1) and (G-2). Hence, by proofing these equation shows that the differential becomes zero when there is only one centroid in the system. By using equations (7.45), (7.47) and (7.49), the RHS of equation (G-1) may be expressed as

$$\begin{aligned}
&\sum_{s=1}^{S}\sum_{m=1}^{M}\sum_{t=1}^{T}\left(\left.\frac{\partial \mathcal{Q}_{smt}^{\text{mpe}}}{\partial b_j^{(i)}}\right|_{h_1(t)=1} + \left.\frac{\partial \mathcal{Q}_{smt}^{\text{mpe}}}{\partial \mu_{smj}}\frac{\partial \mu_{smj}}{\partial b_j^{(i)}}\right|_{h_1(t)=1}\right) \\
&= \sum_{s=1}^{S}\sum_{m=1}^{M}\sum_{t=1}^{T}\left(\frac{\gamma_{sm}^{\text{mpe}}(t)(\boldsymbol{o}_t - \mu_{smtj})}{\sigma_{smj}^2} - \frac{\gamma_{sm}^{\text{ml}}(t)}{\tilde{\beta}_{smj}^{\texttt{ml}}}\frac{\left(x_{smj}^{\texttt{n}} - x_{smj}^{\texttt{d}}\right)}{\sigma_{smj}^2}\right) \\
&= \sum_{s=1}^{S}\sum_{m=1}^{M}\left(\frac{\left(x_{smj}^{\texttt{n}} - x_{smj}^{\texttt{d}}\right)}{\sigma_{smj}^2} - \frac{\left(x_{smj}^{\texttt{n}} - x_{smj}^{\texttt{d}}\right)}{\sigma_{smj}^2}\right) = 0
\end{aligned} \tag{G-3}$$

using the fact that

$$\sum_{s=1}^{S}\sum_{m=1}^{M}\sum_{t=1}^{T}\gamma_{sm}^{\text{mpe}}(t) = \sum_{s=1}^{S}\sum_{m=1}^{M}\beta_{sm}^{\texttt{mpe}} = 0 \tag{G-4}$$

$$\sum_{t=1}^{T}\gamma_{sm}^{\text{mpe}}(t)\boldsymbol{o}_t = \left(x_{smj}^{\texttt{n}} - x_{smj}^{\texttt{d}}\right) \tag{G-5}$$

$$\sum_{t=1}^{T}\gamma_{sm}^{\text{ml}}(t) = \tilde{\beta}_{smj}^{\texttt{ml}} \tag{G-6}$$

Similarly, by using equations (7.48) and (7.50), the RHS of equation (G-2) may be expressed as

$$\begin{aligned}
&\sum_{s=1}^{S}\sum_{m=1}^{M}\sum_{t=1}^{T}\frac{\partial \mathcal{Q}_{smt}^{\text{mpe}}}{\partial \sigma_{smj}^2}\frac{\partial \sigma_{smj}^2}{\partial b_j^{(i)}}\bigg|_{h_1(t)=1} \\
= &-\sum_{s=1}^{S}\sum_{m=1}^{M}\sum_{t=1}^{T}\frac{z_{tjj}\gamma_{sm}^{\text{ml}}(t)(o_{tj}-\mu_{smtj})}{\beta_{sm}^{\texttt{ml}}}\left(\frac{(w_{smj}^{\texttt{n}}-w_{smj}^{\texttt{d}})/\sigma_{smj}^2-\beta_{sm}^{\texttt{mpe}}}{\sigma_{smj}^2}\right) \\
= &\ 0\times\sum_{s=1}^{S}\sum_{m=1}^{M}\left(\frac{(w_{smj}^{\texttt{n}}-w_{smj}^{\texttt{d}})/\sigma_{smj}^2-\beta_{sm}^{\texttt{mpe}}}{\sigma_{smj}^2}\right)=0
\end{aligned} \tag{G-7}$$

with the help of the update formula for the static mean

$$\mu_{smj}=\frac{\sum_{t=1}^{T}z_{tjj}\gamma_{sm}^{\text{ml}}(t)o_{tj}}{\sum_{t=1}^{T}z_{tjj}\gamma_{sm}^{\text{ml}}(t)} \tag{G-8}$$

Checks for pMPE Differentials

The checks for pMPE differentials are given by [112]:

$$0=\sum_{s=1}^{S}\sum_{m=1}^{M}\sum_{t=1}^{T}\left(\frac{\partial \mathcal{Q}_{smt}^{\text{mpe}}}{\partial z_j^{(i)}}\bigg|_{h_1(t)=1}+\frac{\partial \mathcal{Q}_{smt}^{\text{mpe}}}{\partial \sigma_{smj}^2}\frac{\partial \sigma_{smj}^2}{\partial z_j^{(i)}}\bigg|_{h_1(t)=1}\right) \tag{G-9}$$

$$0=\sum_{s=1}^{S}\sum_{m=1}^{M}\sum_{t=1}^{T}\frac{\partial \mathcal{Q}_{smt}^{\text{mpe}}}{\partial \mu_{smj}}\frac{\partial \mu_{smj}}{\partial z_j^{(i)}}\bigg|_{h_1(t)=1} \tag{G-10}$$

The complete pMPE differential is given by the sum of the RHS of equations (G-9) and (G-10). By using equations (7.46), (7.48) and (7.52), the RHS of equation (G-9) may be expressed as

$$\begin{aligned}
&\sum_{s=1}^{S}\sum_{m=1}^{M}\sum_{t=1}^{T}\left(\frac{\partial \mathcal{Q}_{smt}^{\text{mpe}}}{\partial z_j^{(i)}}\bigg|_{h_1(t)=1}+\frac{\partial \mathcal{Q}_{smt}^{\text{mpe}}}{\partial \sigma_{smj}^2}\frac{\partial \sigma_{smj}^2}{\partial z_j^{(i)}}\bigg|_{h_1(t)=1}\right) \\
= &\sum_{s=1}^{S}\sum_{m=1}^{M}\sum_{t=1}^{T}\left(\frac{\gamma_{sm}^{\text{mpe}}(t)(1-\psi_{smtj}(o_{tj}-\mu_{smtj})^2)}{z_{tjj}}\right. \\
&\left.+\frac{(w_{smj}^{\texttt{n}}-w_{smj}^{\texttt{d}})/\sigma_{smj}^2-\beta_{sm}^{\texttt{mpe}}}{\sigma_{smj}^2}\times\frac{z_{tjj}\gamma_{sm}^{\text{ml}}(t)(o_{tj}-\mu_{smtj})^2}{\beta_{sm}^{\texttt{ml}}}\right) \\
= &\sum_{s=1}^{S}\sum_{m=1}^{M}\left(-\frac{(w_{smj}^{\texttt{n}}-w_{smj}^{\texttt{d}})}{\sigma_{smj}^2}+\frac{(w_{smj}^{\texttt{n}}-w_{smj}^{\texttt{d}})}{\sigma_{smj}^2}-\beta_{sm}^{\texttt{mpe}}\right)=0
\end{aligned} \tag{G-11}$$

by making use of equations (G-4), (G-5) and (G-6) and the definition of the static variance of the jth dimension

$$\sigma_{smj}^2=\frac{\sum_{t=1}^{T}z_{tjj}\gamma_{sm}^{\text{ml}}(t)(o_{tj}-\mu_{smtj})^2}{\beta_{sm}^{\texttt{ml}}} \tag{G-12}$$

(G-13)

Bibliography

[1] F. Agakov. Investigations of Gaussian product-of-experts models. Master's thesis, Division of Informatics, The University of Edinburgh, 2000. Available at http://www.dai.ed.ac.uk/homes/felixa/all.ps.gz. 3.8.2

[2] T. Anastasakos, J. McDonough, R. Schwartz, and J. Makhoul. A compact model for speaker-adaptive training. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 1999. 2.2.3

[3] B. S. Atal. Effectiveness of linear prediction characteristics of the speech wave for automatic speaker identification and verification. *Journal of the Acoustical Society of America*, 55(6):1304–1312, 1974. 8.1.2.1

[4] S. Axelrod, V. Goel, R. A. Gopinath, P. A. Olsen, and K. Visweswariah. Discriminative training of subspace constrained Gaussian mixture models for speech recognition. *Submitted to IEEE Trans. Speech and Audio Processing*, March 2004. 5.3

[5] S. Axelrod, V. Goel, R. A. Gopinath, P. A. Olsen, and K. Visweswariah. Subspace constrained Gaussian mixture models for speech recognition. *IEEE Transactions on Speech and Audio Processing*, 13(6):1144–1160, November 2005. 3.6.3, 3.8, 3.8.1, 3.8.1, 4.1.1

[6] S. Axelrod, V. Goel, B. Kingsbury, K. Visweswariah, and R. A. Gopinath. Large vocabulary conversational speech recognition with a subspace constraint on inverse covariance matrices. In *Proc. Eur. Conf. Speech Commun. Technol.*, 2003. 4.3, 6, 6.3, 8.4

[7] S. Axelrod, R. Gopinath, and P. Olsen. Modeling with a subspace constraint on inverse covariance matrices. In *Proc. Int. Conf. Spoken Lang. Process.*, 2002. 3, 1.2, 3.5, 3.6, 3.6.3, 4.1.1, 4.3.2.2, 4.3.2.2, 17, F, F

[8] L. R. Bahl, P. V. de Souza, P. S. Gopalkrishnan, D. Nahamoo, and M. A. Picheny. Context dependent modelling of phones in continuous speech using decision trees. In *Proceedings*

DARPA Speech and Natural Language Processing Workshop, pages 264–270, 1991. 9, 8.1.2.2

[9] Lalit R. Bahl, Peter F. Brown, Peter V. de Souza, and Robert L. Mercer. Maximum mutual information estimation of hidden Markov model parameters for speech recognition. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, pages 49–52, 1986. 2.2.2

[10] J. K. Baker. The DRAGON-System – an overview. *IEEE Transactions on Acoustic, Speech and Signal Processing*, 23:24–29, 1975. 1.1

[11] L. E. Baum and J. A. Eagon. An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology. *Bull. Amer. Math. Soc.*, 73:360–363, 1967. 7, 5.2

[12] J. Bilmes. Factored sparse inverse covariance matrices. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2000. 10.3

[13] J. A. Bilmes. Buried Markov models for speech recognition. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, pages 713–716, March 1999. 3, 2.3, 2.3.1, 2.3.1, 3, 7.1

[14] P. Brown. *The Acoustic-Modelling Problem in Automatic Speech Recognition*. PhD thesis, Carnegie-Mellon University, 1987. 2.2.2

[15] S. S. Chen and R. A. Gopinath. Gaussianization. In *NIPS*, pages 423–429, 2000. 8.3.3

[16] C. Chesta, O. Siohan, and C. Lee. Maximum a posteriori linear regression for hidden Markov model adaptation. In *Proc. Eur. Conf. Speech Commun. Technol.*, volume 1, pages 211–214, 1999. 2.2.3

[17] W. Chou. Maximum a posteriori linear regression with elliptically symmetric matrix priors. In *Proc. Eur. Conf. Speech Commun. Technol.*, volume 1, pages 1–4, 1999. 2.2.3

[18] S. B. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustic, Speech and Signal Processing*, 28(4):357–366, 1980. 5, 3.4.1

[19] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39:1–39, 1977. 2, 7, 8, 4

[20] Y. Ephraim, A. Demdo, and L. R. Rabiner. A minimum discrimination approach to hidden Markov modelling. *IEEE Transactions on Information Theory*, September 1989. 2.2.2

[21] Y. Ephraim and L. R. Rabiner. On the relations between modelling approaches for speech recognition. *IEEE Transactions on Information Theory*, March 1990. 2.2.2

[22] G. Evermann, H. Y. Chan, M. J. F. Gales, T. Hain, X. Liu, D. Mrva, L. Wang, and P. C. Woodland. Development of the 2003 CU-HTK conversational telephone speech transcription system. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2004. 9, 8.1.2, 8.1.2.3, 8.5.1

[23] G. Evermann, H. Y. Chan, M. J. F. Gales, B. Jia, X. Liu, D. Mrva, K. C. Sim, L. Wang, P. C. Woodland, and K. Yu. Development of the 2004 CU-HTK english CTS systems using more than two thousand hours of data. In *Proc. Fall 2004 Rich Transcription Workshop (RT-04f)*, November 2004. 9, 8.1.2, 8.5.1

[24] G. Evermann, H. Y. Chan, M. J. F. Gales, B. Jia, D. Mrva, P. C. Woodland, and K. Yu. Training lvcsr systems on thousands of hours of data. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, pages 849–852, March 2005. 9, 8.1.2, 8.1.2.3, 8.5.1, 8.5.1, 8.5.1

[25] G. Evermann and P. C. Woodland. Posterior probability decoding, confidence estimation and system combination. In *Proc. Speech Transcription Workshop*, 2000. 2.2.4, 8.5

[26] J. G. Fiscus. A post-processing system to yield reduced word error rates: Recogniser Output Voting Error Reduction (ROVER). In *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 1997. 2.2.4

[27] H. Franco and A. Serralheiro. Training HMMs using a minimum error approach. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 1991. 2.2.2

[28] S. Furui. Speaker independent isolated word recognition using dynamic features of speech spectrum. *IEEE Trans. Acoustics, Speech and Signal Processing*, ASSP-34:52–59, 1986. 5

[29] M. J. F. Gales. The generation and the use of regression class trees for MLLR adaptation. Technical Report CUED/F-INFENG/TR263, Cambridge University, 1996. (via anonymous) ftp://svr-www.eng.cam.ac.uk. 2.2.3, 3.7

[30] M. J. F. Gales. Adapting semi-tied full-covariance HMMs. Technical Report CUED/F-INFENG/TR298, Cambridge University, 1997. (via anonymous) ftp://svr-www.eng.cam.ac.uk. 4.2.1

[31] M. J. F. Gales. Maximum likelihood linear transformations for HMM-based speech recognition. Technical Report CUED/F-INFENG/TR291, Cambridge University, 1997. (via anonymous) ftp://svr-www.eng.cam.ac.uk. 2.2.3, 6

[32] M. J. F. Gales. Semi-tied full-covariance matrices for hidden Markov models. Technical Report CUED/F-INFENG/TR287, Cambridge University, 1997. (via anonymous) ftp://svr-www.eng.cam.ac.uk. 3, 1.2, 3.5, 4.2

[33] M. J. F. Gales. Maximum likelihood multiple projection schemes for hidden Markov models. Technical Report CUED/F-INFENG/TR365, Cambridge University, 1999. (via anonymous) ftp://svr-www.eng.cam.ac.uk. 3.7, 6, 6.2, 6.3, 6.3, 6.3.1

[34] M. J. F. Gales. Semi-tied covariance matrices for hidden Markov models. *IEEE Transactions on Speech and Audio Processing*, 7(3):272–281, 1999. 3, 1.2, 3.5, 3.6, 3.6.1, 3.6.5, 4.2, 4.2.1, 4.3

[35] M. J. F. Gales. Factored semi-tied covariances. In *Proc. NIPS*, 2000. 3.7

[36] M. J. F. Gales and S. S. Airey. Product of Gaussians for speech recognition. Technical Report CUED/F-INFENG/TR.458, Cambridge University, 2003. Available from: svr-www.eng.cam.ac.uk/˜mjfg. 3.8, 3.8.2, 3.8.2

[37] M. J. F. Gales, B. Jia, X. Liu, K. C. Sim, P. C. Woodland, and K. Yu. Development of the CUHTK 2004 RT04f mandarin conversational telephone speech transcription system. In *Proc. Fall 2004 Rich Transcription Workshop (RT-04f)*, November 2004. 9, 8.1.2, 8.1.2.3

[38] M. J. F. Gales, B. Jia, X. Liu, K. C. Sim, P. C. Woodland, and K. Yu. Development of the CUHTK 2004 RT04f mandarin conversational telephone speech transcription system. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, pages 861–864, March 2005. 9, 8.1.2, 8.1.2.3

[39] M. J. F. Gales, X. Liu, K. C. Sim, and K. Yu. Investigation of acoustic modeling techniques for LVCSR systems. In *Proc. Fall 2004 Rich Transcription Workshop (RT-04f)*, November 2004. 9.3

[40] M. J. F. Gales and P. C. Woodland. Mean and variance adaptation within the MLLR framework. *Computer Speech and Languages*, 10:249–264, 1996. 1.2, 2.2.3, 6, 6.1

[41] M. J. F. Gales and S. J. Young. Segmental hidden Markov models. In *Proc. Eur. Conf. Speech Commun. Technol.*, pages 1579–1582, 1993. 3, 3, 7.1

[42] M. J. F. Gales and S. J. Young. The theory of segmental hidden Markov models. Technical Report CUED/F-INFENG/TR133, Cambridge University, 1993. (via anonymous) ftp://svr-www.eng.cam.ac.uk. 3, 3, 7.1

[43] H. Gish and K. Ng. Parametric trajectory models for speech recognition. In *Proc. Int. Conf. Spoken Lang. Process.*, volume 1, pages 466–469, Philadelphia, PA, 1996. 3

[44] N. K. Goel and A. G. Andreou. Heteroscedastic discriminant analysis and reduced-rank HMMs for improved speech recognition. *Speech Communication*, 26:283–297, 1998. 5, 3.6.5

[45] N. K. Goel and R. A. Gopinath. Multiple linear transforms. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2001. 3.7

[46] V. Goel, S. Axelrod, R. Gopinath, P Olsen, and K. Visweswariah. Discriminative estimation of Subspace Precision and Mean (SPAM) models. In *Proc. Eur. Conf. Speech Commun. Technol.*, 2003. 5.3, 5.3.2

[47] W. D. Goldenthal and J. R. Glass. Statistical trajectory models for phonetic recognition. In *Proc. Int. Conf. Spoken Lang. Process.*, pages 1871–1874, Yokohama, Japan, 1994. 3

[48] Y. Gong and J. P. Haton. Stochastic trajectory modeling for speech recognition. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, volume I, pages 57–60, 1994. 3

[49] P. Gopalakrishnan, D. Kanevsky, A. N*á*das, and D. Nahamoo. Decoder selection based on cross-Entropies. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 1988. 2.2.2

[50] P. Gopalakrishnan, D. Kanevsky, A. N*á*das, and D. Nahamoo. A generalisation of the baum algorithm in rational objective functions. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 1989. 5.2

[51] P. Gopalakrishnan, D. Kanevsky, A. N*á*das, and D. Nahamoo. An inequality for rational functions with applications to some statistical estimation problems. *IEEE Trans. Info. Theory*, 37:107–113, 1991. 5.2

[52] R. A. Gopinath. Constrained maximum likelihood modeling with Gaussian distributions. In *Proceeding of ARPA Workshop on Human Language Understanding*, January 1998. 3

[53] R. A. Gopinath. Maximum likelihood modeling with Gaussian distributions for classification. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, volume II, pages II–661–II–664, 1998. 3, 3.6.1

[54] T. Hain, P. C. Woodland, G. Evermann, X. Liu, G. L. Moore, D. Povey, and L. Wang. Automatic transcription of conversational telephone speech. development of the CU-HTK 2002 system. Technical Report CUED/F-INFENG/TR465, Cambridge University, 2003. (via anonymous) ftp://svr-www.eng.cam.ac.uk. 8.1.2

[55] H. Hermansky. Perceptual Linear Predictive (PLP) analysis of speech. *Journal of the Acoustical Society of America*, 87(4):1738–1752, 1990. 5, 3.4.1, 8.1.2.1

[56] G. E. Hinton. Products of experts. In *Proceedings of the Ninth International Conference on Artificial Neural Networks (ICANN 99)*, pages 1–6, 1999. 3.8.2

[57] B. J. Huang, S. E. Levinson, and M. M. Sondhi. Maximum likelihood estimation for multivariate mixture observations of Markov chains. *IEEE Trans. Information Theory*, IT-32:307–309, March 1986. 3

[58] J. Huang, V. Goel, R. A. Gopinath, B. Kingsbury, P. Olsen, and K. Visweswariah. Large vocabulary conversational speech recognition with the extended maximum likelihood linear transformation (EMLLT) model. In *Proc. Int. Conf. Spoken Lang. Process.*, 2002. 4.3, 6, 6.3

[59] J. Huang and D. Povey. Discriminatively trained features using fMPE for multi-stream audio-visual speech recognition. In *Proc. Interspeech*, September 2005. 10.3

[60] Xuedong Huang, Alex Acero, Hsiao-Wuen Hon, and Raj Reddy. *Spoken Language Processing: A Guide to Theory, Algorithm and System Development*. Prentice Hall PTR, 1st edition, 2001. 1.1.1

[61] F. Jelinek. Continuous speech recognition by statistical methods. In *Proc. IEEE*, volume 64, pages 532–556, 1976. 1.1

[62] B. H. Juang, W. Chou, and C. H Lee. Minimum classification error rate methods for speech recognition. *IEEE Transactions on Speech and Audio Processing*, May 1997. 2.2.2

[63] J. Kaiser, B. Horvat, and Z. Kacic. A novel loss function for the overall risk criterion based discriminative training of HMM models. In *Proc. Int. Conf. Spoken Lang. Process.*, 2000. 5

[64] S. Kapadia, V. Valtchev, and S. J. Young. MMI training for continuous phoneme recognition on the TIMIT database. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 1993. 2.2.2

[65] D. Y. Kim, H. Y. Chan, G. Evermann, M. J. F. Gales, D. Mrva, K. C. Sim, and P. C. Woodland. Development of the CU-HTK 2004 broadcast news transcription systems. In *Proc. Fall 2004 Rich Transcription Workshop (RT-04f)*, November 2004. 9, 8.1.2, 3, 8.5

[66] D. Y. Kim, H. Y. Chan, G. Evermann, M. J. F. Gales, D. Mrva, K. C. Sim, and P. C. Woodland. Development of the CU-HTK 2004 broadcast news transcription systems. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, pages 861–864, March 2005. 9, 8.1.2, 8.5.2

[67] D. Y. Kim, G. Evermann, T. Hain, D. Mrva, S. E. Tranter, L. Wang, and P. C. Woodland. Recent advances in broadcast news transcription. In *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 105–110, 2003. 9, 8.1.2, 3, 8.5

[68] N. Kumar. *Investigation of Silicon-Auditory Models and Generalization of Linear Discriminant Analysis for Improved Speech Recognition*. PhD thesis, Johns Hopkins University, 1997. 5, 3.4.1, 3.5, 3.6.5, 8.1.2.1

[69] L. Lee and R. Rose. Speaker normalisation using efficient frequency warping procedures. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, pages 353–356, 1996. 8.1.2.1

[70] C. J. Legetter and P. C. Woodland. Maximum likelihood linear regression speaker adaptation of continuous density HMMs. *Computer Speech and Languages*, 1997. 1.2, 2.2.3, 2.2.3, 6

[71] S. E. Levinson, L. R. Rabiner, and M. M. Sondhi. An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition. Technical report, Bell System Tech. J., 1983. 1.1

[72] Hui Lin, Ye Tian, JianLai Zhou, and Hui Jiang. Hierarchical correlation compensation for hidden Markov models. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2005. 10.3

[73] X. Liu and M. J. F. Gales. Automatic model complexity control using marginalized discriminative growth functions. In *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2003. 3.7

[74] X. Liu, M. J. F. Gales, K. C. Sim, and K. Yu. Investigation of acoustic modeling techniques for LVCSR systems. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, pages 849–852, March 2005. 9.3

[75] A. Ljolje, Y. Ephraim, and L. R. Rabiner. Estimation of hidden Markov model parameters by minimising empirical error rate. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 1990. 2.2.2

[76] L. Mangu, E. Brill, and A. Stolcke. Finding consensus among words: Lattice-based word error minimization. In *Proc. Eur. Conf. Speech Commun. Technol.*, 1999. 2.2.4, 8.5

[77] J. McDonough and A. Waibel. Maximum mutual information speaker adapted training with semi-tied covariance matrices. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2003. 5.3

[78] B. Merialdo. Phonetic recognition using hidden Markov models and maximum mutual information training. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, volume 1, pages 111–114, 1988. 2.2.2

[79] A. Nádas. A decision theoretic formulation of a training problem in speech recognition and a comparison of training by unconditional and conditional maximum likelihood. *IEEE Transactions on Acoustic, Speech and Signal Processing*, August 1983. 2.2.2

[80] A. Nádas, D. Nahamoo, and M. A. Picheny. On a model-robust training method for speech recognition. *IEEE Transactions on Acoustic, Speech and Signal Processing*, ASSP-36:1432–1436, 1988. 2.2.2

[81] NIST. Benchmark tests : Rich transcription (RT). http://www.nist.gov/speech/tests/rt. 1.1.3

[82] Y. Normandin. *Hidden Markov models, maximum mutual information estimation and the speech recognition problem*. PhD thesis, McGill University, Montreal, 1991. 2.2.2

[83] Y. Normandin, R. Lacouture, and R. Cardin. MMIE training for large vocabulary continuous speech recognition. In *Proc. Int. Conf. Spoken Lang. Process.*, pages 1367–1371, 1994. 2.2.2

[84] Y. Normandin and D. Morgera. An improved MMIE training algorithm for speaker-independent small vocabulary, continuous speech recognition. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, pages 537–540, 1991. 2.2.2

[85] P. Olsen and R. A. Gopinath. Modelling inverse covariance matrices by basis expansion. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2002. 3, 1.2, 3.5, 3.6, 3.6.2, 4.2, 4.2.2, 4.2.2, 4.2.2, 5.3.2

[86] P. Olsen and R. A. Gopinath. Modelling inverse covariance matrices by basis expansion. *IEEE Transactions on Speech and Audio Processing*, 12(1):37–46, January 2004. 3, 1.2, 3.6, 4.2

[87] Peder Olsen, Karthik Visweswariah, and Ramesh Gopinath. Initializing subspace constrained Gaussian mixture models. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2005. 4.3.2.2, 10.3

[88] M. Ostendorf, V. Digalakis, and O. Kimball. From HMM's to segment models: A unified view of stochastic modeling for speech recognition. *IEEE Transactions on Speech and Audio Processing*, 4(5):360–378, 1996. 3, 2.3, 2.3.2, 3, 7.1

[89] M. Ostendorf and S. Roukos. A stochastic segment model for phoneme-based continuous speech recognition. *IEEE Trans on Acoustics, Speech and Signal Processing*, 37(12):1857–1869, 1989. 3, 2.3, 2.3.2, 3, 7.1

[90] D. S. Pallett. Speech results on resource management task. http://www.nist.gov/speech/history/pdf/rm89_task.pdf, 1989. 4

[91] D. S. Pallett. A look at NIST's benchmark ASR tests: Past, present and future. http://www.nist.gov/speech/history/pdf/NIST_benchmark_ASRtests_2003.pdf, 2003. 4

[92] D. S. Pallett, N. L. Dahlgren, J. G. Fiscus W. M. Fisher, J. S. Garofolo, and B. C. Tjaden. DARPA february 1992 ATIS benchmark test results. http://www.nist.gov/speech/history/pdf/darpa92_atis.pdf, 1992. 4

[93] D. S. Pallett, J. G. Fiscus, and J. S. Garofolo. Resource management corpus: September 1992 benchmark test results. http://www.nist.gov/speech/history/pdf/resource_management_92eval.pdf, 1992. 4

[94] E. Polak. *Computational Methods in Optimization: A Unified Approach*. Academic Press, 1971. 4.1.2, F

[95] D. Povey. *Discriminative Training for Large Vocabulary Speech Recognition*. PhD thesis, Cambridge University, 2003. 3, 2.2.2, 2.3.3, 19, 1, 5.2, 5.3.1, 5.4, 7.3

[96] D. Povey. Improvements to fMPE for discriminative training of features. In *Proc. Interspeech*, September 2005. 3, 2.3.3, 21, 7.4, 10.3

[97] D. Povey, B. Kingsbury, L. Mangu, G. Saon, H. Soltau, and G. Zweig. fMPE: Discriminatively trained features for speech recognition. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2005. 3, 1.2, 2.3.3, 7.2.2, 21, 7.3.2, 7.3.2, 7.3.2.1, 7.3.2.1, 7.3.2.1, 7.3.2.2, 7.4, 7.7, 7.7, 9.1, 9.1, G

[98] D. Povey and P. C. Woodland. Frame discrimination training of HMMs for large vocabulary speech recognition. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, March 1999. 2.2.2

[99] D. Povey and P. C. Woodland. An investigation of frame discrimination for continuous speech recognition. Technical Report CUED/F-INFENG/TR420, Cambridge University, May 2000. 2.2.2

[100] D. Povey and P. C. Woodland. Minimum Phone Error and I-smoothing for improved discriminative training. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2002. 3, 2.2.2, 5.2, 5.2, 5.2, 5.3, 5.3.2, 5.4, 7.3.2

[101] D. Povey, P. C. Woodland, and M. J. F. Gales. Discriminative MAP for acoustic model adaptation. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2003. 5.2, 5.4, 7.4, 8.3.2

[102] L. A. Rabiner. A tutorial on hidden Markov models and selective applications in speech recognition. In *Proc. of the IEEE*, volume 77, pages 257–286, February 1989. 2.1

[103] A-V. I. Rosti and M. J. F. Gales. Factor analysed hidden Markov models for speech recognition. Technical Report CUED/F-INFENG/TR453, Cambridge University, 2003. (via anonymous) ftp://svr-www.eng.cam.ac.uk. 12, 3.4.2

[104] A-V. I. Rosti and M. J. F. Gales. Switching linear dynamical systems for speech recognition. Technical Report CUED/F-INFENG/TR461, Cambridge University, 2003. (via anonymous) ftp://svr-www.eng.cam.ac.uk. 3, 2.3, 2.3.2, 2.3.3, 3, 7.1

[105] G. Saon, M. Padmanabhan, R. Gopinath, and S. Chen. Maximum likelihood discriminant feature spaces. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2000. 5, 3.4.1, 3.6.5

[106] G. Saon, D. Povey, and G. Zweig. CTS decoding improvements at IBM. In *Presented at EARS STT Workshop*, 2003. 5.4

[107] G. Saon, G. Zweig, and M. Padmanabhan. Linear feature space projections for speaker adaptation. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2001. 2.2.3

[108] L. Saul and M. Rahim. Modeling acoustic correlations by factor analysis. *NIPS '97*, 1997. 3.4.2

[109] L. Saul and M. Rahim. Maximum likelihood and minimum classification error factor analysis for automatic speech recognition. *IEEE Transactions on Speech and Audio Processing*, 1999. 3.4.2

[110] K. C. Sim and M. J. F. Gales. Basis superposition precision matrix modelling for large vocabulary continuous speech recognition. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2004. 3.5, 3.6.5

[111] K C Sim and M J F Gales. Precision matrix modelling for large vocabulary continuous speech recognition. Technical Report CUED/F-INFENG/TR485, Cambridge University, 2004. (via anonymous) ftp://svr-www.eng.cam.ac.uk. 5.3.2

[112] K. C. Sim and M. J. F. Gales. Temporally varying model parameters for large vocabulary continuous speech recognition. In *Proc. Interspeech*, September 2005. 3, 1.2, 2.3.3, 22, G

[113] K. C. Sim and M. J. F. Gales. Minimum phone error training of precision matrix models. *to appear in IEEE Transactions on Speech and Audio Processing*, July 2006. 5.3

[114] R. Sinha, S. E. Tranter, M. J. F. Gales, and P. C. Woodland. The cambridge university march 2005 speaker diarisation system. In *Proc. Interspeech*, September 2005. 8.1.2.1

[115] A. Sixtus, S. Molau, S. Kanthak, R. Schluter, and H. Ney. Recent improvements of the RWTH large vocabulary speech recognition system on spontaneous speech. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, pages 1671–1674, June 2000. 8.1.2.1

[116] G. Soan, A. Dharanipragada, and D. Povey. Feature-space Gaussianisation. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, May 2004. 8.3.3

[117] H. Soltau, B. Kingsbury, L. Mangu, D. Povey, G. Saon, and G. Zweig. The IBM 2004 conversational telephony system for rich transcription. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2005. 3

[118] K. Tokuda, H. Zen, and T. Kitamura. Trajectory modeling based on HMMs with the explicit relationship between static and dynamic features. In *Proc. Eur. Conf. Speech Commun. Technol.*, pages 865–868, 2003. 3, 2.3, 3, 7.1

[119] K. Tokuda, H. Zen, and T. Kitamura. Reformulating the HMM as a trajectory model. In *Proc. of Beyond HMM – Workshop on statistical modeling approach for speech recognition*, December 2004. 3, 2.3, 3, 7.1

[120] S. E. Tranter. Two-way cluster voting to improve speaker diarisation performance. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, pages 753–756, March 2005. 8.1.2.1

[121] S. Tsakalidis, V. Doumpiotis, and W. Byrne. Discriminative linear transforms for feature normalisation and speaker adaptation in HMM estimation. In *Proc. Int. Conf. Spoken Lang. Process.*, 2002. 5.3

[122] V. Valtchev. *Discriminative Methods in HMM-based Speech Recognition*. PhD thesis, University of Cambridge, UK, March 1995. 2.2.2

[123] V. Valtchev, J. J. Odell, P. C. Woodland, and S. J. Young. Lattice-based discriminative training for large vocabulary speech recognition. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 1996. 2.2.2

[124] V. Vanhoucke. *Mixture of Inverse Covariances: Covariance Modeling for Gaussian Mixtures with Applications to Automatic Speech Recognition*. PhD thesis, Stanford University, July 2003. 3.5, 3.6.3

[125] V. Vanhoucke and A. Sankar. Mixture of inverse covariances. *IEEE Transactions on Speech and Audio Processing*, 12(3):250–264, May 2004. 3.5, 3.6.3, 4.1.1

[126] K. Visweswariah and P. Olsen. Feature adaptation using projection of Gaussian posteriors. In *Proc. Interspeech*, September 2005. 7.5, 10.3

[127] K. Visweswariah, P. Olsen, R. Gopinath, and S. Axelrod. Maximum likelihood training of subspaces for inverse covariance modeling. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2003. 3.6, 3.6.4, 3.6.4, 4.2.2, 4.3.2.2

[128] C. J. Wellekens. Explicit time correlation in hidden Markov models for speech recognition. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, pages 384–386, 1987. 3, 2.3.1, 3

[129] C. K. I. Williams, F. V. Agakov, and S. N. Felderhof. Products of Gaussians. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2001. MIT Press. 3.8.2, 3.8.2

[130] P. C. Woodland. Hidden Markov models using vector linear prediction and discriminative output distributions. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, volume 1, pages 509–512, 1992. 3, 2.3.1, 3, 7.1

[131] P. C. Woodland and D. Povey. Large scale discriminative training for speech recognition. In *Proc. ASR*, September 2000. 2.2.2

[132] P. C. Woodland and D. Povey. Large scale MMIE training for conversational telephone speech recognition. In *Proc. Speech Transcription Workshop*, May 2000. 2.2.2

[133] S. J. Young. Large vocabulary continuous speech recognition: A review. In *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 3–28, Snowbird, Utah, December 1995. 3

[134] S. J. Young, D. Kershaw, J. J. Odell, D. Ollason, V. Valtchev, and P. C. Woodland. *The HTK Book (for HTK version 3.0)*. Cambridge University, 1997. 9, 4.3.3, 8.1.2, 8.1.2.2, 9.2.5

[135] S. J. Young, J. J. Odell, and P. C. Woodland. Tree-based state tying for high accuracy acoustic modelling. In *Proceedings ARPA Workshop on Human Language Technology*, pages 307–312, 1994. 9, 8.1.2.2

[136] X. Zhou, Z. Yao, and B. Dai. Improved covariance modeling for GMM in speaker identification. In *Proc. Interspeech*, September 2005. 10.3