CAMBRIDGE UNIVERSITY

ENGINEERING DEPARTMENT

PRECISION MATRIX MODELLING FOR LARGE VOCABULARY CONTINUOUS SPEECH RECOGNITION

K.C. Sim and M.J.F. Gales

CUED/F-INFENG/TR485

June 2004

Cambridge University Engineering Department

Trumpington Street

Cambridge. CB2 1PZ

England

E-mail: {kcs23,mjfg}@eng.cam.ac.uk

http://mi.eng.cam.ac.uk/~mjfg

Abstract

Gaussian Mixture Models (GMMs) are commonly used as the output density function for Large Vocabulary Continuous Speech Recognition (LVCSR) systems based on the Continuous Density Hidden Markov Models (CDHMMs). A standard problem when using multivariate GMMs to classify data is how to accurately represent the correlation in its feature. Full covariance matrices yield a good model, but dramatically increase the number of model parameters. Commonly, a diagonal covariance matrix approximation is employed together with some kind of frontend decorrelation techniques such as Semi-tied covariance matrices (STC) to provide the "uncorrelated" feature. Recently, more advanced structured precision matrix approximations have been found to outperform these decorrelation approach, including Extended Maximum Likelihood Linear Transform (EMLLT) and Subspace for Precision and Mean (SPAM) models. Besides, Maximum Mutual Information (MMI) based discriminative training of the STC and SPAM models has been found to yield good improvement over the conventional Maximum Likelihood (ML) training. The recent success of the Minimum Phone Error (MPE) training consistently outperforms the MMI criterion on diagonal covariance matrix systems. This report emphasises on discriminative training of these state-of-the-art precision matrix models using the MPE criterion. The model update procedure is based on maximising the weak-sense auxiliary function. Furthermore, implementation issues associated with building LVCSR systems are also addressed. Finally, these models are evaluated using the large vocabulary continuous telephone speech (CTS) and broadcast news (BN) English tasks.

Contents

1 Introduction

Speech recognition is a pattern classification task that maps a speech waveform into a sequence of words. Most state-of-the-art systems are based on Hidden Markov Models (HMMs) [3]. The HMM is a finite state transducer consisting of a variable number of emitting states connected by arcs describing the transition probabilities (a first order Markov process). The observation emission probability depends only on the current state and is usually represented by a Gaussian Mixture Model (GMM).

A large vocabulary continuous speech recognition (LVCSR) [29] system comprises many Gaussian components (greater than 100,000) with high dimensional data (typically 39 or 52). The use of a full covariance matrix for each Gaussian component dominates the total number of model parameters. Furthermore, it is computationally expensive to train and perform recognition with these models. Also, a large number of training data is required to ensure robust model estimation. The conventional approach to address these problems is to use a diagonal covariance approximation. This assumes that the feature dimensions are uncorrelated given a particular component. Several methods have been employed to improve the validity of this assumption. For example, the use of Mel Frequency Cepstral Coefficients (MFCC) [4] and Perceptual Linear Prediction (PLP) [13] coefficients provides data with minimum correlations. Further decorrelation can also be achieved using feature transformation techniques such as Linear Discriminant Analysis (LDA) [23], Heteroscedastic LDA (HLDA) [15] and Heteroscedastic Discriminant Analysis (HDA) [11].

Recently, more advanced covariance modelling techniques have been found to give improvements over the conventional scheme. For example, Factor-analysed HMMs (FAHMMs) [22] are a covariance modelling scheme that are based on factor analysis (FA) within the HMM framework. Unfortunately, a complex covariance structure results in expensive likelihood computation due to the need to invert the covariance matrices. This motivates modelling of the inverse covariance (precision) matrices instead. This form of modelling is becoming increasingly popular. Models that have been successfully applied to speech recognition include Semi-tied Covariance (STC) [10], Extended MLLT (EMLLT) [18] and Subspace for Precision and Mean (SPAM) [2].

The precision matrix models mentioned above can be described within a generic framework of *basis superposition* [24]. Within this framework, precision matrices are expressed as a linear superposition of a set of symmetric matrices (known as *basis matrices*) weighted by a set of corresponding coefficients (known as *basis coefficients*). Typically, the basis matrices are *globally* tied to yield a compact model representation. The number of basis coefficients, n, can then be adjusted to control the model complexity. For example, if the basis matrices are rank-1 symmetric, the model becomes a STC when $n = d$ and EMLLT when $d < n \leq \frac{d}{2}(d+1)$.

Large vocabulary discriminative training has been found to yield promising gain over the ML training [27, 20]. This has motivated the use of discriminative training for many state-of-the-art LVCSR systems [6]. Previous work has obtained good improvement from ML trained precision matrix models on large vocabulary systems, for example STC [10], EMLLT [14] and SPAM [1]. [14, 1] emphasised the use of adaptation technique for EMLLT and SPAM models. However, many other issues arise when implementing these precision matrix models in LVCSR. For example, problems concerning computational load, memory requirement and robustness become apparent if the number of Gaussian components get increasingly large. These issues are addressed in this report. On small or medium vocabulary systems, MMI training has also been successfully implemented for STC [17] and SPAM [12] models. In this report, large vocabulary MPE training of precision matrix models will be discussed, based on the method described in [20].

This report is organised as follows: Section 2 introduces various forms of covariance and precision matrix modelling techniques. The basis superposition framework is then discussed in Section 2.1. Section 3 then addresses implementation issues of the precision matrix models in LVCSR.

Discriminative training based on the Minimum Phone Error criterion is given in Section 4. Next, experimental results are presented in Section 5. Finally, this is followed by conclusions.

2 Covariance vs. Precision Matrix Modelling

There are inherent *uncertainties* associated with speech data. The speech waveform produced by one speaker may differ from that spoken by another (inter-speaker variability), or even from the same speaker at another instance (intra-speaker variability). Speech data may also be corrupted by noise due to the environment, the devices used to record the speech and digitisation of speech signal (sampling and quantisation). These variabilities are modelled by representing the observations using random vectors[1], associated with these a probability density function (PDF). If the human speech production mechanism is assumed to be conditional upon a large number of independent factors that summed together, then according to the Central Limit Theorem, the observation PDF is given by a multivariate Gaussian:

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

where $\boldsymbol{o}_t$ is a $d \times 1$ observation vector at time t. $\boldsymbol{\mu}_m$ and $\boldsymbol{\Sigma}_m$ are the mean vector and covariance matrix of the Gaussian distribution given by

$$\begin{aligned} \boldsymbol{\mu}_m &= E[\boldsymbol{o}_t] & (2)\\ \boldsymbol{\Sigma}_m &= E[(\boldsymbol{o}_t - \boldsymbol{\mu}_m)(\boldsymbol{o}_t - \boldsymbol{\mu}_m)'] \\ &= E[\boldsymbol{o}_t\boldsymbol{o}_t'] - \boldsymbol{\mu}_m\boldsymbol{\mu}_m' & (3) \end{aligned}$$

where $E[\boldsymbol{x}] = \int \boldsymbol{x}p(\boldsymbol{x})d\boldsymbol{x}$ is the expectation of the random vector $\boldsymbol{x}$. $\boldsymbol{\mu}_m$ and $\boldsymbol{\Sigma}_m$ determine the location and shape of a Gaussian distribution respectively. In pattern classification, the form of the Gaussian distributions associated with each class defines the shape of the decision boundaries. If a variance information is assumed the same for all classes, the decision boundary is simply a straight line (or hyper-plane for high dimensional data) bisecting the mean vectors of the two classes. With additional variance information, a non-linear decision boundary can be defined such that the classification error is minimised.

Equation (3) shows that $\boldsymbol{\Sigma}_m$ is a symmetric matrix whose leading diagonal elements are the variances of each dimension and off-diagonal elements are the covariances in between dimensions:

$$\begin{aligned} \mathrm{var}(\boldsymbol{o}_t(i)) &= \boldsymbol{\Sigma}_m(i,i) = \sigma_{ii}^2 & (4)\\ \mathrm{cov}(\boldsymbol{o}_t(i), \boldsymbol{o}_t(j)) &= \boldsymbol{\Sigma}_m(i,j) = \sigma_{ij}^2 & (5) \end{aligned}$$

where the covariance between dimensions i and j, σ_{ij}^2, are related to the correlation coefficients, ρ_{ij} by

$$\rho_{ij} = \frac{\sigma_{ij}^2}{\sigma_{ii}\sigma_{jj}} \tag{6}$$

where $(-1 \leq \rho_{ij} \leq 1)$, $\rho_{ii} = 1$ and $\rho_{ij} = 0$ indicates that $\boldsymbol{o}_t(i)$ and $\boldsymbol{o}_t(j)$ are uncorrelated. Thus, correlation information is important to yield accurate probability distribution.

The inverse of a covariance matrix is known as a 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 B1 for more details)

[1] Multi-dimensional random variables

A typical speech recognition system consists of many Gaussian components. Modelling these components using full covariance matrices results in a large number of free parameters and consequently leads to the issues of data sparseness and computational intractability. Model compactness and computational efficiency may be improved by introducing some kind of approximation schemes. In the following section, a generic basis superposition framework will be introduced to achieve structured approximations for precision matrix modelling. Next, several forms of structured approximations. Some of these methods approximate the covariance matrix directly while the others focus on precision matrix approximation. In this report, approximation of precision matrix structure will be emphasised.

2.1 The Basis Superposition Framework

Basis superposition is a technique whereby a positive-definite matrix is expressed as a linear superposition of a set of symmetric matrices, $\{\boldsymbol{S}_i\}$; weighted by a corresponding set of coefficients $\{\lambda_{ii}^{(m)}\}$. This technique is applicable to both covariance and precision matrix approximations. For clarity, the basis superposition framework will be described based on the precision matrix approximation. Hence, a precision matrix, $\boldsymbol{P}_m$, can be expressed as

$$\boldsymbol{P}_m = \sum_{i=1}^{n} \lambda_{ii}^{(m)} \boldsymbol{S}_i \tag{7}$$

where n is the basis order. The parameters can be divided into two sets:

- **"Global" parameters**: basis matrices $\boldsymbol{S}_i$ or basis vectors $\boldsymbol{a}_i$
- **"Component" parameters**: basis coefficients $\lambda_{ii}^{(m)}$

If the the basis matrices are chosen to be rank-1 symmetric where $\boldsymbol{S}_i = \boldsymbol{a}_i'\boldsymbol{a}_i$, then

$$\boldsymbol{P}_m = \sum_{i=1}^{n} \lambda_{ii}^{(m)} \boldsymbol{a}_i'\boldsymbol{a}_i = \boldsymbol{A}'\boldsymbol{\Lambda}_m\boldsymbol{A} \tag{8}$$

where the basis vector, $\boldsymbol{a}$, is the ith row of $\boldsymbol{A}$. $\boldsymbol{\Lambda}_m$ is a diagonal matrix with diagonal elements $\lambda_{ii}^{(m)}$.

In the case, where there are a large number of precision matrices to be modelled, the above parameterisation leads to a compact model representation through the tying of basis matrices. In other words, the basis matrices forms 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*.

2.1.1 Basis Decomposition

Basis decomposition of a matrix is an operation where the underlying basic structures of the matrix are extracted. These basic structures may be restricted to satisfy certain characteristics such as *orthogonality* and *independence*. For positive-definite symmetric matrices, there are two common basis decomposition techniques: orthogonal basis decomposition and Choleski factorisation. The former may be achieved using either Eigenvalues decomposition or Singular Value Decomposition (SVD). Both methods decompose a positive-definite matrix $\boldsymbol{M}$ as

$$\boldsymbol{M} = \boldsymbol{A}\boldsymbol{\Lambda}\boldsymbol{A}' \tag{9}$$

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	$\frac{d}{r} \leq n \leq \frac{d}{2}(d+1)$
SPAM	full-rank symmetric	$1 \leq n \leq \frac{d}{2}(d+1)$

Table 1: Precision matrix models as basis decomposition techniques

where the columns of $\boldsymbol{A}$ are given by the orthonormal eigenvectors of $\boldsymbol{M}$. The corresponding eigenvalues are given by the leading diagonal elements of the diagonal matrix $\boldsymbol{\Lambda}$. The orthonormal constraint leads to a unique decomposition with a rotational transformation matrix, $\boldsymbol{A}$. Choleski factorisation decomposes a positive-definite matrix $\boldsymbol{M}$ such that

$$\boldsymbol{M} = \boldsymbol{L}\boldsymbol{L}' = \bar{\boldsymbol{L}}\boldsymbol{\Lambda}\bar{\boldsymbol{L}}' \tag{10}$$

where $\boldsymbol{L}$ and $\bar{\boldsymbol{L}}$ are lower triangular matrices. $\bar{\boldsymbol{L}}$ is a normalised version of $\boldsymbol{L}$ such that the leading diagonal elements are unity and $l_n(i,j) = \frac{l(i,j)}{\sqrt{l(i,i)l(j,j)}}$.

Basis decomposition forms the fundamental concept of the basis superposition precision matrix modelling framework. Common basic structures are extracted from a large number of precision matrices to yield a compact model representation. Each kind of model extracts different number of basic structures of different type. Table 1 compares different precision matrix models as basis decomposition methods. STC and EMLLT are rank-1 basis decomposition methods, differed by the basis orders; SPAM is an arbitrary symmetric basis decomposer. Further explanation of these models are given in the Section 2.

2.1.2 Negative 'Variances'

Let $\bar{\boldsymbol{x}}$ be a random vector such that $\bar{\boldsymbol{x}} \sim \mathcal{N}(\bar{\boldsymbol{\mu}}, \bar{\boldsymbol{\Sigma}})$. If a square and invertible transformation matrix $\boldsymbol{A}$ is applied to $\bar{\boldsymbol{x}}$ such that the resulting random vector, $\boldsymbol{x}$, consists of independent random variables:

$$\boldsymbol{x} = \boldsymbol{A}\bar{\boldsymbol{x}} \tag{11}$$

where $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and $\boldsymbol{\Sigma}$ is diagonal, then $\boldsymbol{A}$ is known as a *decorrelating* transform. The covariance matrices are related by

$$\boldsymbol{\Sigma} = \boldsymbol{A}\bar{\boldsymbol{\Sigma}}\boldsymbol{A}' \tag{12}$$

and the corresponding precision matrices given by

$$\bar{\boldsymbol{P}} = \boldsymbol{A}'\boldsymbol{\Lambda}\boldsymbol{A} \tag{13}$$

where $\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1}$ and $\bar{\boldsymbol{P}} = \bar{\boldsymbol{\Sigma}}^{-1}$. $\boldsymbol{\Lambda}$ is also a diagonal matrix whose diagonal elements, λ_{ii}, are given by the inverse variances of $\boldsymbol{x}$. Comparing to equation (24), equation (13) is simply the STC model and the basis coefficients are simply the inverse variances in the transformed space. However, for EMLLT models, basis coefficients are allowed to be negative. Applying the similar analogy as STC leads to negative 'variances'!

To account for the *negative* variances, let p_{ii} be the precision (inverse variance) of the projected space which corresponds to the projection vector $\boldsymbol{a}_i$. p_{ii} can be expressed as

$$\begin{aligned} p_{ii} &= \boldsymbol{a}_i\bar{\boldsymbol{P}}\boldsymbol{a}_i' \\ &= \boldsymbol{a}_i'\boldsymbol{A}'\boldsymbol{\Lambda}\boldsymbol{A}\boldsymbol{a}_i' \\ &= \sum_{j=1}^{n} \lambda_{jj}(\boldsymbol{a}_i'\boldsymbol{a}_j)^2 \end{aligned} \tag{14}$$

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

$$(\boldsymbol{a}_i'\boldsymbol{a}_j)^2 = \begin{cases} 1 & i = j \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

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 (14), the projected variance is actually given by the joint contributions from all the basis vectors. Thus, λ_{ii} can be regarded as a *contribution* to the overall projected precision 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 covariance matrix, $\bar{\boldsymbol{\Sigma}}$, is positive-definite.

2.1.3 Product of Gaussians

One interesting attribute of the basis superposition framework is its close relation to the Product-of-Gaussian (PoG) framework [7]. The product of Gaussian distributions is itself a Gaussian distribution. For simplicity, consider the product of P Gaussian components sharing the same mean vector $\boldsymbol{\mu}$:

$$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{P}) \propto \prod_{p=1}^{P} \left(\frac{|\alpha_p \boldsymbol{P}_p|}{(2\pi)^d}\right)^{1/2} \exp\left\{-\frac{1}{2}(\boldsymbol{o}_t - \boldsymbol{\mu})'\alpha_p \boldsymbol{P}_p(\boldsymbol{o}_t - \boldsymbol{\mu})\right\} \tag{16}$$

where α_p is a scaling factor for each precision matrix $\boldsymbol{P}_p$. The resulting precision matrix $\boldsymbol{P}$ is given by the sum of the precision matrices of each individual Gaussian component:

$$\boldsymbol{P} = \sum_{p=1}^{P} \alpha_p \boldsymbol{P}_p \tag{17}$$

Comparing with equation (7), it immediately becomes clear that basis superposition is in fact an example of a Product of Experts (PoE) system. Strictly speaking, each expert is not required to be a valid Gaussian distribution because $\boldsymbol{P}_p$ is not required to be positive-definite and α_p may be negative. The latter yields 'negative' variances. However, as seen before, they are simply negative variance *contributions* as the resultant precision matrix has to be positive-definite.

2.2 Covariance Matrix Approximations

Ideally, it will be useful to incorporate as much correlation information as possible so that the probability distribution can better represent the random data. Unfortunately, speech recognition involves a large number of Gaussian components (greater than 100,000 for LVCSR) and high dimensional data (typically 39 or 52) making it inefficient to use full covariance structure. This is mainly due to the large number of model parameters and high computational cost to perform training and decoding. Consequently, large amount of training data and compute resources are required. This motivates the use of more efficient covariance modelling schemes, which can be defined to satisfy the following criteria:

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

Many forms of covariance structure approximation have been investigated. One simple example is using a diagonal approximation. Block-diagonal approximation may also be used. Alternatively, more advanced technique that incorporate factor analysis into the HMM framework (FAHMM) has also been found to yield improved modelling. These methods will be described briefly in the following sections.

2.2.1 Diagonal/Block-diagonal Covariance Matrix

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. However, is it a reasonable approximation? Many techniques can be used to make the independence assumption safe. The feature vectors are chosen such that the variables are as uncorrelated as possible. For examples, the Mel Frequency Cepstral Coefficients (MFCC) and Perceptual Linear Prediction (PLP) coefficients are only lightly correlated. Further decorrelation can also be achieved using feature transformation techniques such as Linear Discriminant Analysis (LDA) and Heteroscedastic LDA (HLDA). When the block-diagonal approximation is used, it is intuitive to divide the variables into blocks of static, first and second derivatives. Such an approximation is based on prior expert knowledge, not inferred from the training data.

2.2.2 Implicit Covariance Modelling using Gaussian Mixture Models

Real speech data may not be *normally* distributed as predicted by the Central Limit Theorem. The distribution may be skewed, multi-modal, having sharper peak, longer tail and so on. A popular way to circumvent this problem is to model the distribution using a M-component GMM:

$$p(\boldsymbol{o}_t|\{c_m, \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m\}) = \sum_{m=1}^{M} c_m \mathcal{N}(\boldsymbol{o}_t; \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m) \tag{18}$$

where c_m, $\boldsymbol{\mu}_m$ and $\boldsymbol{\Sigma}_m$ are the weight, mean vector and covariance matrix of the mth Gaussian component respectively. Besides, GMM can also be used to *implicitly* model the correlation using only mixtures of diagonal covariance Gaussian distributions. A GMM with M diagonal covariance Gaussian components has Md free variance parameters. Compared to a full covariance matrix with $\frac{d}{2}(d+1)$ free parameters, GMM provides a compact means of correlations modelling.

2.2.3 Factor-analysed HMM (FAHMM)

It is more natural to extract the common independent factors based on the statistics of the data. One suitable technique is factor analysis. Factor analysis is a technique where high dimensional data is modelled using a small set of *latent* variables. It finds n common underlying independent factors that affect the data. Combining with the d independent unique factors associated to the original variables, the data vectors can be expressed as

$$\boldsymbol{x} = \boldsymbol{\mu} + \boldsymbol{C}\boldsymbol{f} + \boldsymbol{v} \tag{19}$$

where $\boldsymbol{\mu}$ is the mean vector of the random vector $\boldsymbol{x}$, $\boldsymbol{f}$ is a $n \times 1$ vector of common factors, $\boldsymbol{v}$ is a $d \times 1$ vector of unique factors associated to the original variables and $\boldsymbol{C}$ is a $d \times n$ matrix of factors loadings. $\boldsymbol{f}$ and $\boldsymbol{v}$ are random vectors with distributions $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and $\mathcal{N}(\boldsymbol{0}, \boldsymbol{V})$ respectively. The resulting covariance matrix can be expressed as

$$\boldsymbol{\Sigma} = \boldsymbol{C}\boldsymbol{C}' + \boldsymbol{V} \tag{20}$$

One issue 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 one as the common factors. The total number of free parameters is given by $d(n+1)$. This generalisation of this technique has been successfully implemented within the HMM framework. This method in known as the Factor-analysed HMM (FAHMM)[22]. FAHMM is a generative model that belongs to the generalised linear Gaussian model. It can be described using the state-space framework:

$$\boldsymbol{x}_t \sim \mathcal{M}^{\text{hmm}} \qquad \mathcal{M}^{\text{hmm}} = \left\{ a_{ij}, c_{jn}^{(x)}, \boldsymbol{\mu}_{jn}^{(x)}, \boldsymbol{\Sigma}_{jn}^{(x)} \right\} \tag{21}$$

$$\boldsymbol{o}_t = \boldsymbol{C}\boldsymbol{x}_t + \boldsymbol{v}_t \qquad \boldsymbol{v}_t \sim \sum_{m=1}^{M} c_{jm}^{(o)} \mathcal{N}\left(\boldsymbol{o}_t; \boldsymbol{\mu}_{jm}^{(o)}, \boldsymbol{\Sigma}_{jm}^{(o)}\right) \tag{22}$$

where $\boldsymbol{x}_t$ and $\boldsymbol{o}_t$ are the state and observation vectors respectively. $\mathcal{M}^{\text{hmm}}$ denotes the set of HMM model parameters with N Gaussian components. a_{ij} is the transition probability from state i to j. The state and observation noise distributions are represented by Gaussian Mixture Models with parameters $\{c_{jn}^{(x)}, \boldsymbol{\mu}_{jn}^{(x)}, \boldsymbol{\Sigma}_{jn}^{(x)}\}$ and $\{c_{jm}^{(o)}, \boldsymbol{\mu}_{jm}^{(o)}, \boldsymbol{\Sigma}_{jm}^{(o)}\}$ respectively. $\boldsymbol{C}$ is the state-observation matrix (factors loadings matrix), shared by all the Gaussian components. This yields the following form of FAHMM covariance structure:

$$\boldsymbol{\Sigma}_{jmn} = \boldsymbol{C}\boldsymbol{\Sigma}_{jn}^{(x)}\boldsymbol{C}' + \boldsymbol{\Sigma}_{jm}^{(o)} \tag{23}$$

where j, n and m represent the state, state space component and observation space component respectively. Although the FAHMM provides a good form of covariance matrix modelling, it is computationally expensive in decoding due to the inversion of the covariance matrix in likelihood calculation. In fact, it will shown in the later section that modelling the precision (inverse covariance) matrices alleviates the inversion problem, resulting in a more efficient modelling technique.

2.3 Precision Matrix Approximations

In recent years, modelling of precision (inverse covariance) matrices has become increasingly popular. Methods such as Semi-tied Covariances (STC) (also known as MLLT), Extended MLLT (EMLLT) and Subspace for Precision and Mean (SPAM) have been successfully implemented within the HMM framework in speech recognition. Furthermore, the HLDA feature projection scheme can also be considered a special form of precision matrix modelling technique. In the following, each of these techniques will be discussed within the ML framework.

2.3.1 Semi-tied Covariance (STC)

Semi-tied covariances [10] are an example of precision matrix model. The precision matrix can be expressed as

$$\boldsymbol{P}_m = \boldsymbol{A}'\boldsymbol{\Lambda}_m\boldsymbol{A} \tag{24}$$

where $\boldsymbol{A}$ is a $d \times d$ global transformation matrix and $\boldsymbol{\Lambda}_m$ is a diagonal matrix whose leading diagonal elements, $\lambda_{ii} = 1/\sigma_i^2$, are the inverse variances in the transformed space. The total number of free parameters in this model is $d(M+d)$. For large systems, $M >> d$, so the total number of parameters is approximately the same as the case of diagonal covariance (Md). In fact, the global STC system can be viewed as a diagonal covariance system with additional feature global transformation. The STC formulation allows the transformation matrix, $\boldsymbol{A}$ and the transformed variances, σ_i^2 to be estimated efficiently using iterative closed-form update formulae. STC model is also known as a Maximum Likelihood Linear Transform (MLLT) model.

Parameters associated to STC models can be estimated efficiently using the following update formulae (see Appendix B2 for derivations)

$$\lambda_{ii}^{(m)} = \frac{1}{\boldsymbol{a}_i \boldsymbol{W}_m \boldsymbol{a}_i'} \tag{25}$$

$$\boldsymbol{a}_i = \boldsymbol{c}_i' \boldsymbol{G}_i^{-1} \sqrt{\frac{\beta}{\boldsymbol{c}_i' \boldsymbol{G}_i^{-1} \boldsymbol{c}_i}} \tag{26}$$

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

$$\boldsymbol{G}_i = \sum_{m=1}^{M} \beta_m \lambda_{ii}^{(m)} \boldsymbol{W}_m \tag{27}$$

$$\beta = \sum_{m=1}^{M} \beta_m \tag{28}$$

Since equations (25) and (26) are closed form solutions, a trivial starting point for a STC model is to set $\boldsymbol{A}$ as an identity matrix and the corresponding basis coefficients as the inverse variances obtained from a diagonal covariance system, $\lambda_{ii}^{(m)} = 1/\sigma_{ii}^{(m)2}$.

2.3.2 Extended MLLT (EMLLT)

As the name suggests, EMLLT[18] is an extension to the STC (MLLT) model where the precision matrix is given by

$$\boldsymbol{P}_m = \boldsymbol{A}' \boldsymbol{\Lambda}_m \boldsymbol{A} = \sum_{i=1}^{n} \lambda_{ii}^{(m)} \boldsymbol{a}_i' \boldsymbol{a}_i \tag{29}$$

where $\boldsymbol{a}_i$ is the ith row of the $n \times d$ matrix $\boldsymbol{A}$ and λ_{ii} is the ith leading diagonal element of the the $n \times n$ diagonal matrix $\boldsymbol{\Lambda_m}$. Thus, 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 at the lower end to a full covariance matrix model at the upper end.

Due to the rectangular EMLLT transformation matrix, $\boldsymbol{A}$, the term $\log|\boldsymbol{A}' \boldsymbol{\Lambda}_m \boldsymbol{A}|$ cannot be easily differentiated with respect to 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 in order to update the basis vectors. Newton optimisation method is a popular example. Given the current estimate of the basis vector, $\hat{\boldsymbol{a}}_i$, the new estimate, $\boldsymbol{a}_i$, can be written as

$$\boldsymbol{a}_i = \hat{\boldsymbol{a}}_i + \eta \bar{\boldsymbol{f}}_i \bar{\boldsymbol{H}}_i^{-1} \tag{30}$$

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

$$\bar{\boldsymbol{f}}_i = \sum_{m=1}^{M} \beta_m \lambda_{ii}^{(m)} \hat{\boldsymbol{a}}_i \left(\hat{\boldsymbol{\Sigma}}_m - \boldsymbol{W}_m \right) \tag{31}$$

$$\bar{\boldsymbol{H}}_i = \sum_{m=1}^{M} \beta_m \lambda_{ii}^{(m)} \left\{ \hat{\boldsymbol{\Sigma}}_m \left(1 - \hat{\boldsymbol{a}}_i \hat{\boldsymbol{\Sigma}}_m \hat{\boldsymbol{a}}_i' \right) - \boldsymbol{W}_m - \lambda_{ii}^{(m)} \hat{\boldsymbol{\Sigma}}_m \hat{\boldsymbol{a}}_i' \hat{\boldsymbol{a}}_i \hat{\boldsymbol{\Sigma}}_m \right\} \tag{32}$$

respectively. The basis coefficients, on the other hand, may be updated using the following formula[18]

$$\lambda_{ii}^{(m)} = \hat{\lambda}_{ii}^{(m)} + \left(\frac{1}{\hat{\boldsymbol{a}}_i \boldsymbol{W}_m \hat{\boldsymbol{a}}_i'} - \frac{1}{\hat{\boldsymbol{a}}_i \boldsymbol{\Sigma}_m \hat{\boldsymbol{a}}_i'} \right) \tag{33}$$

Full derivations of equations (31), (32) and (33) are given in Appendix B3. The update rule given in equation (33) is known as the additive update. This method yields both positive and negative coefficients. Alternatively, a multiplicative update[18] may be used, which is given by

$$\lambda_{ii}^{(m)} = \hat{\lambda}_{ii}^{(m)} \left(\frac{\hat{\boldsymbol{a}}_i \boldsymbol{\Sigma}_m \hat{\boldsymbol{a}}_i'}{\hat{\boldsymbol{a}}_i \boldsymbol{W}_m \hat{\boldsymbol{a}}_i'} \right) \tag{34}$$

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[18].

2.3.3 Subspace for Precision and Mean (SPAM)

A SPAM[2] model constrains the mean vector and the precision matrix to be within a subspace of the space spanned by a symmetric matrix (a space with dimensionality $\frac{d}{2}(d+1)$). A special form of SPAM model, which leaves the mean vectors unconstrained, is simply a precision matrix model. It is a further generalisation of the EMLLT model where instead of rank-1 matrices, any arbitrary symmetric matrices may be used as the bases. The precision matrix can be expressed as

$$\boldsymbol{P}_m = \sum_{i=1}^{n} \lambda_{ii}^{(m)} \boldsymbol{S}_i = \sum_{i=1}^{n} \lambda_{ii}^{(m)} \left(\sum_{j=1}^{p_n} \lambda_{jj} \boldsymbol{a}_{ij}' \boldsymbol{a}_{ij} \right) \tag{35}$$

where $\boldsymbol{S}_i = \sum_{j=1}^{p_n} \lambda_{jj} \boldsymbol{a}_{ij}' \boldsymbol{a}_{ij}$ is any arbitrary symmetric matrix. It can also be viewed as a *nested* sum of N rank-1 matrices, where $N = \sum_{i=1}^{n} p_n$. Provided at least one of the $\boldsymbol{S}_i$ is positive-definite, n can be as small as 1. In general, N is constrained by $d \leq N \leq \frac{d}{2}(d+1)$.

Unfortunately, no closed form solutions for both the basis matrix and coefficient updates for the SPAM model. An efficient conjugate gradient descent algorithm may be used to update the basis coefficients[2]. Appendix B4 gives details of a basis coefficient update using the Polak-Ribeire conjugate-gradient method. The final optimisation simplifies to a line search which maximises the following equation with respect to a scalar variable, $\Delta^{(m)}$.

$$\mathcal{Q}^{(m)} - \hat{\mathcal{Q}}^{(m)} = \frac{\beta_m}{2} \left\{ \sum_{j=1}^{d} \log\left(1 + \Delta^{(m)} w_j^{(m)}\right) - \Delta \sum_{i=1}^{n} d^{(m)}(i)\, \mathrm{Tr}(\boldsymbol{W}_m \boldsymbol{S}_i) \right\} \tag{36}$$

where $\mathcal{Q}^{(m)}$ and $\hat{\mathcal{Q}}^{(m)}$ are the component specific auxiliary functions that are based on the new and old model parameters respectively. The general form of these functions is given by

$$\mathcal{Q}^{(m)} = K_m + \frac{1}{2}\beta_m \Big\{ \log|\boldsymbol{P}_m| - \mathrm{Tr}(\boldsymbol{P}_m \boldsymbol{W}_m) \Big\} \tag{37}$$

$w_j^{(m)}$ is the jth eigenvalue of $\hat{\boldsymbol{P}}_m^{-\frac{1}{2}} \boldsymbol{R}_m \hat{\boldsymbol{P}}_m^{-\frac{1}{2}}$ and $d^{(m)}(i)$ is the ith element of the direction vector for component m, $\boldsymbol{d}^{(m)}$.

2.3.4 Heteroscedastic LDA (HLDA)

HLDA is a feature projection scheme that reduces feature dimensionality and correlation. It can also be considered a precision matrix model [24] where the precision matrix is given by

$$\boldsymbol{P}_m = \boldsymbol{A}' \boldsymbol{\Lambda}_m \boldsymbol{A} = \sum_{i=1}^{d} \lambda_{ii}^{(m)} \boldsymbol{a}_i' \boldsymbol{a}_i + \sum_{i=d+1}^{n} \lambda_{ii} \boldsymbol{a}_i' \boldsymbol{a}_i \tag{38}$$

and the transformation matrix $\boldsymbol{A}$ and $\boldsymbol{\Lambda_m}$ can be expressed in terms of the HDLA *useful* and *nuisance* parameters

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{A}_H \\ \boldsymbol{A}_N \end{pmatrix} \quad \boldsymbol{\Lambda_m} = \begin{pmatrix} \boldsymbol{\Lambda}_H^{(m)} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\Lambda}_N \end{pmatrix} \tag{39}$$

where $\boldsymbol{A}_H$ and $\boldsymbol{A}_N$ are the $d \times n$ HLDA projection and $(n-d)\times n$ nuisance projection respectively. $\boldsymbol{\Lambda}_H^{(m)}$ and $\boldsymbol{\Lambda}_N$ are diagonal matrices whose leading diagonal elements correspond to the inverse variances of the useful and nuisance dimensions respectively. The nuisance dimension variances are modelled with global parameters. As a result, the second term in equation (38) is independent of Gaussian components. 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. Rather than a projection scheme, HLDA can be purely used as a precision matrix model by retaining the nuisance dimension parameters. Thus, the mean vectors in the original feature space are modelled. For convenience, the form of model is called the HLDA-PMM model to differentiate it from the conventional HLDA model.

3 Precision Matrix Modelling for LVCSR

In the previous section, several forms of precision matrix modelling schemes were described that have been successfully applied to HMM-based speech recognition. Most of these ML trained models have been applied to LVCSR systems, including STC [10], EMLLT [14] and SPAM [1]. the emphasis in these literatures have been on adapting these models for Speaker Adaptive Training (SAT) or MLLR adaptation. However, there arise several other issues such as memory requirement, computational feasibility and training robustness when porting these techniques to large systems. System efficiency may be advertly affected if these issues are not addressed properly. Here, various implementation issues for LVCSR systems based on the maximum likelihood (ML) training criterion will be considered. In the next section, Minimum Phone Error (MPE) training of precision matrix models will be presented.

3.1 Maximum Likelihood (ML) Training

Within the HMM framework, model parameters can be estimated using the well-known Baum-Welch (Expectation Maximisation) algorithm. Given the alignment between the observation at time t and the component m from the E-step, the auxiliary function to be maximised in the M-step is given by

$$\begin{aligned} \mathcal{Q}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) &= K + \frac{1}{2}\sum_{t=1}^{T}\sum_{m=1}^{M}\gamma_m(t)\Big\{\log|\boldsymbol{P}_m| - (\boldsymbol{o}_t - \boldsymbol{\mu}_m)'\boldsymbol{P}_m(\boldsymbol{o}_t - \boldsymbol{\mu}_m)\Big\} \\ &= K + \frac{1}{2}\sum_{m=1}^{M}\beta_m\Big\{\log|\boldsymbol{P}_m| - \mathrm{Tr}(\boldsymbol{P}_m\boldsymbol{W}_m)\Big\} \end{aligned} \tag{40}$$

where K subsumes terms independent of the model parameters and the required statistics for the estimation of precision matrix parameters are given by

$$\boldsymbol{W}_m = \frac{\sum_{t=1}^{T}\gamma_m(t)(\boldsymbol{o}_t - \boldsymbol{\mu}_m)(\boldsymbol{o}_t - \boldsymbol{\mu}_m)'}{\beta_m} \tag{41}$$

$$\beta_m = \sum_{t=1}^{T}\gamma_m(t) \tag{42}$$

$\boldsymbol{W}_m$ and β_m are the full covariance statistics and the component occupancy counts respectively. For all the forms of precision matrix modelling, the mean vectors are unconstrained. Thus, the following standard update formula may be used

$$\boldsymbol{\mu}_m = \boldsymbol{u}_m / \beta_m \tag{43}$$

where the mean statistics is given by

$$\boldsymbol{u}_m = \sum_{t=1}^{T} \gamma_m(t) \boldsymbol{o}_t \tag{44}$$

In the following, the update formulae for the model parameters of STC, EMLLT and SPAM models will be described for the convenience of the subsequent discussions regarding LVCSR implementation issues. Derivation of these formulae is provided in Appendix B and further details regarding these models may be obtained from the corresponding literatures.

3.2 Efficient Likelihood Calculation

One of the attractive attributes of precision matrix modelling is its efficiency during decoding. This is mainly due to the fact that the likelihood expression is directly related to the precision matrix:

$$\mathcal{L}(\boldsymbol{\mu}_m, \boldsymbol{P}_m | \mathcal{O}) = K + \frac{1}{2} \sum_{t=1}^{T} \Big\{ \log |\boldsymbol{P}_m| - (\boldsymbol{o}_t - \boldsymbol{\mu}_m)' \boldsymbol{P}_m (\boldsymbol{o}_t - \boldsymbol{\mu}_m) \Big\} \tag{45}$$

where $\mathcal{L}(\boldsymbol{\mu}_m, \boldsymbol{P}_m | \mathcal{O})$ is the likelihood of the model parameters given the complete set of observations, $\mathcal{O}$. The efficiency is clearly illustrated using the basis superposition framework. Substituting equation (7) into equation (45) yields

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\mu}_m, \boldsymbol{P}_m | \mathcal{O}) &= K + \frac{1}{2} \sum_{t=1}^{T} \left\{ \log |\boldsymbol{P}_m| - \sum_{i=1}^{n} \lambda_{ii}^{(m)} (\boldsymbol{o}_t - \boldsymbol{\mu}_m)' \boldsymbol{S}_i (\boldsymbol{o}_t - \boldsymbol{\mu}_m) \right\} \\
&= K + \frac{1}{2} \sum_{t=1}^{T} \left\{ \log |\boldsymbol{P}_m| - \sum_{i=1}^{n} \lambda_{ii}^{(m)} \boldsymbol{o}_t' \boldsymbol{S}_i \boldsymbol{o}_t - \boldsymbol{\mu}_m' \boldsymbol{P}_m \boldsymbol{\mu}_m + 2 \sum_{i=1}^{n} \lambda_{ii}^{(m)} \boldsymbol{\mu}_m' \boldsymbol{S}_i \boldsymbol{o}_t \right\}
\end{aligned}$$

The terms in equation (46) can be divided into two:

- **Model dependent**: $\log |\boldsymbol{P}_m|$ and $\boldsymbol{\mu}_m' \boldsymbol{P}_m \boldsymbol{\mu}_m$
- **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, terms dependent on the observation vectors can be cached and reused for all the Gaussian components. This yields a significant reduction in computational cost.

If the basis matrices are rank-1, equation (45) can be rewritten as

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\mu}_m, \boldsymbol{P}_m | \mathcal{O}) &= K + \frac{1}{2} \sum_{t=1}^{T} \Big\{ \log |\boldsymbol{P}_m| - (\boldsymbol{o}_t - \boldsymbol{\mu}_m)' \boldsymbol{A}' \boldsymbol{\Lambda}_m \boldsymbol{A} (\boldsymbol{o}_t - \boldsymbol{\mu}_m) \Big\} \\
&= K + \frac{1}{2} \sum_{t=1}^{T} \Big\{ \log |\boldsymbol{P}_m| - (\boldsymbol{o}_t' \boldsymbol{A}' - \boldsymbol{\mu}_m' \boldsymbol{A}') \boldsymbol{\Lambda}_m (\boldsymbol{A} \boldsymbol{o}_t - \boldsymbol{A} \boldsymbol{\mu}_m) \Big\} \\
&= K + \frac{1}{2} \sum_{t=1}^{T} \Big\{ \log |\boldsymbol{P}_m| - (\tilde{\boldsymbol{o}}_t' - \tilde{\boldsymbol{\mu}}_m') \boldsymbol{\Lambda}_m (\tilde{\boldsymbol{o}}_t - \tilde{\boldsymbol{\mu}}_m) \Big\}
\end{aligned} \tag{46}$$

where $\tilde{\boldsymbol{o}}_t = \boldsymbol{A}\boldsymbol{o}_t$ and $\tilde{\boldsymbol{\mu}}_m = \boldsymbol{A}\boldsymbol{\mu}_m$ are the projected observation and mean vectors respectively. The model dependent terms are $\log|\boldsymbol{P}_m|$ and $\tilde{\boldsymbol{\mu}}_m$ and the observation term to be cached is the transformed observation, $\tilde{\boldsymbol{o}}_t$. Thus, for STC and EMLLT models, the matrix $\boldsymbol{A}$ is 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 given by $1/\lambda_{ii}^{(m)}$. The likelihood computation is thus linearly proportional to the basis order, n.

3.3 Memory Issues

The major issue with implementing precision matrix models on LVCSR systems is the large amount of memory requirement for full covariance statistics accumulation. The update of the *tied* parameters is highly inefficient, especially for the SPAM models where the basis matrices are not rank-1. In general, it is preferable to get a good initial set of basis matrices and concentrate on updating the basis coefficients which is efficient. Moreover, updating basis coefficients does not require the full covariance statistics. The basis coefficient update formulae for STC and EMLLT in equations (25) and (33) may be rewritten as

$$\begin{aligned}\lambda_{ii}^{(m)} &= 1/\tilde{w}_i &(47)\\ \lambda_{ii}^{(m)} &= \hat{\lambda}_{ii}^{(m)} + \left(1/\tilde{w}_i - \frac{1}{\hat{\boldsymbol{a}}_i\boldsymbol{\Sigma}_m\hat{\boldsymbol{a}}_i'}\right) &(48)\end{aligned}$$

where $\tilde{w}_i$ is known as the *projected* statistics which is given by

$$\begin{aligned}\tilde{w}_i &= \hat{\boldsymbol{a}}\boldsymbol{W}_m\hat{\boldsymbol{a}}' &(49)\\ &= \hat{\boldsymbol{a}}\left(\frac{\sum_{t=1}^{T}\gamma_m(t)(\boldsymbol{o}_t-\boldsymbol{\mu}_m)(\boldsymbol{o}_t-\boldsymbol{\mu}_m)'}{\beta_m}\right)\hat{\boldsymbol{a}}'\\ &= \frac{\sum_{t=1}^{T}\gamma_m(t)(\tilde{o}_{ti}-\tilde{\mu}_{mi})^2}{\beta_m}\end{aligned}$$

for $i = 1, 2, \ldots, n$. $\tilde{w}_i$ is a scalar term. $\tilde{o}_{ti} = \boldsymbol{a}_i\boldsymbol{o}_t$ and $\tilde{\mu}_{mi} = \boldsymbol{a}_i\boldsymbol{\mu}_m$ are the projected observation and mean vectors associated with the projection vector, $\boldsymbol{a}_i$. Hence, the total amount of memory required is proportional to n rather than $\frac{d}{2}(d+1)$ for full covariance statistics, $\boldsymbol{W}_m$. This dramatically reduces the total memory requirement. From the previous section, the values of $\tilde{o}_{ti}$ and $\tilde{\mu}_{mi}$ have been pre-computed and cached for efficient likelihood computation. Thus, no extra cost is incurred in computing the projected statistics for STC and EMLLT models.

Likewise, the sufficient statistics required to update the basis coefficients for SPAM models can also be expressed in terms of the projected statistics, $\tilde{w}_i$, which is given by

$$\begin{aligned}\tilde{w}_i &= \mathrm{Tr}\left(\boldsymbol{S}_i\boldsymbol{W}_m\right) &(50)\\ &= \mathrm{Tr}\left(\boldsymbol{S}_i\frac{\sum_{t=1}^{T}\gamma_m(t)(\boldsymbol{o}_t-\boldsymbol{\mu}_m)(\boldsymbol{o}_t-\boldsymbol{\mu}_m)'}{\beta_m}\right)\\ &= \frac{\sum_{t=1}^{T}\gamma_m(t)(\boldsymbol{o}_t-\boldsymbol{\mu}_m)'\boldsymbol{S}_i(\boldsymbol{o}_t-\boldsymbol{\mu}_m)}{\beta_m}\\ &= \frac{\sum_{t=1}^{T}\gamma_m(t)(\boldsymbol{o}_t'\boldsymbol{S}_i\boldsymbol{o}_t - 2\boldsymbol{\mu}_m'\boldsymbol{S}_i\boldsymbol{o}_t + \boldsymbol{\mu}_m'\boldsymbol{S}_i\boldsymbol{\mu}_m)}{\beta_m}\end{aligned}$$

As before, the projected statistics, $\tilde{w}_i$, is a scalar term for each basis matrix, $\boldsymbol{S}_i$. Hence, the required memory is proportional to the basis order, n. Once again, the computation of projected statistics does not require additional cost because 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. Thus, the update formula of basis coefficients for SPAM in equation (36) can be rewritten as

$$\mathcal{Q}^{(m)} - \hat{\mathcal{Q}}^{(m)} = \frac{\beta_m}{2}\left\{\sum_{j=1}^{d}\log\left(1+\Delta^{(m)}w_j^{(m)}\right) - \Delta\sum_{i=1}^{n}d^{(m)}(i)\bar{w}_i\right\} \tag{51}$$

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.

3.4 Basis Initialisations

Section 2.3 described several forms of structured precision approximation in terms of basis superposition. It is important to obtain a good initial set of basis to allow fast convergence and avoid 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 identity initialisation may be used, which simply leads to a diagonal covariance matrix system. The initialisation schemes for EMLLT and SPAM will be discussed as follows.

3.4.1 Initialisation Schemes for EMLLT

As discussed in Section 2.3.2, there is no closed form solution to update the basis vectors. Standard optimisation package such as Newton optimisation is used to estimate them. In general, the use of 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 (31) and (32). It is important to note that it is not possible to initialise the basis vectors to be zero vectors as doing so leads to a stationary point[2]. In addition, 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 expressed as

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}^{(1)} \\ \boldsymbol{A}^{(2)} \end{bmatrix} \quad \text{and} \quad \boldsymbol{\Lambda}_m = \begin{bmatrix} \boldsymbol{\Lambda}_m^{(1)} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\Lambda}_m^{(2)} \end{bmatrix} \tag{52}$$

where $\boldsymbol{A}^{(1)}$ and $\boldsymbol{A}^{(2)}$ are $n_1 \times d$ and $n_2 \times d$ matrices and the corresponding coefficient matrices, $\boldsymbol{\Lambda}_m^{(1)}$ and $\boldsymbol{\Lambda}_m^{(2)}$ are $n_1 \times n_1$ and $n_2 \times n_2$ diagonal matrices respectively. Table 2 tabulates several initialisation schemes. The second and third columns describe the properties of each initialisation scheme and the equivalent initial model. 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 arbitrarily 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. It is more intuitive to initialise the basis vectors such that the initial model is a STC model. Thus, the STC basis vectors provide a better set of initial vectors. There are three variants of initialisation that lead to a STC starting point. Firstly, an identity matrix can be 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 yields the STC+EYE initialisation method. Alternatively, the identity matrix may be replaced with a HLDA projection matrix, giving the STC+HLDA

[2]This can be a local maximum or minimum depending on the Hessian matrix

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

Table 2: EMLLT initialisation schemes

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. 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 equivalent initial model.

3.4.2 Initialisation Schemes for SPAM

The update of basis coefficients for SPAM models require a set of matrices, $\boldsymbol{S}_i$ such that the initial precision matrices, $\boldsymbol{P}_m$, are positive definite. In [2], it was suggested that one of the basis matrices is initialised as a positive definite symmetric matrix such that it is the weighted average precision matrix over all the Gaussian components

$$\boldsymbol{S}_n = \frac{\sum_{m=1}^{M} c_m \boldsymbol{P}_m}{\sum_{m=1}^{M} c_m} \tag{53}$$

Positive-definiteness can now be guaranteed by initialising $\lambda_{nn}^{(m)} = 1$ and the rest of the basis coefficients set to zero. The remaining basis matrices can be initialised to any arbitrary values without violating the positive-definiteness constraint. However, according to [2], 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

$$\boldsymbol{V} = \frac{\sum_{m=1}^{M} c_m \boldsymbol{v}_m \boldsymbol{v}_m'}{\sum_{m=1}^{M} c_m} \tag{54}$$

where $\boldsymbol{v}_m = \text{vec}(\boldsymbol{W}_m^{-1})$. On large systems, it was found that the basis matrix initialisation given by equation (54) is not robust due to the robustness of full covariance statistics for each component. Instead, the state-level inverse covariance statistics is used to produce a more reliable set of basis matrices.

$$\boldsymbol{V} = \frac{\sum_{s=1}^{S} c_s \boldsymbol{v}_s \boldsymbol{v}_s'}{\sum_{s=1}^{S} c_s} \tag{55}$$

where

$$\boldsymbol{v}_s = \operatorname{vec}\left(\left\{\sum_{m=1}^{M_s} c_m \boldsymbol{W}_m\right\}^{-1}\right) \tag{56}$$

$$c_s = \sum_{m=1}^{M_s} c_m \tag{57}$$

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 update just the basis coefficients for subsequent training iterations. The exact update procedure for the basis matrices is described in [26].

3.5 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 standard form, for example in HTK [28], of the floored variances, $\bar{\sigma}_{ii}^{(m)2}$, are given by

$$\bar{\sigma}_{ii}^{(m)2} = \max\left(\sigma_{ii}^{(m)2}, \sigma_{ii}^{(vf)2}\right) \tag{58}$$

where $\sigma_{ii}^{(m)2}$ and $\sigma_{ii}^{(vf)2}$ are the ith estimated variance and variance floor respectively. In HTK, the variance floor elements are normally computed as the average within state variances scaled by a predefined factor. Hence,

$$\sigma_{ii}^{(vf)2} = \frac{\alpha \sum_{s=1}^{S} \beta_s \sigma_{ii}^{(s)2}}{\sum_{s=1}^{S} \beta_s} \tag{59}$$

where $\mu_i^{(s)}$, $\sigma_{ii}^{(s)2}$ and β_s are the within state mean, variance and occupancy count respectively. α is a scaling factor which is typically set as 0.1 (10%).

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,

$$1/\hat{\lambda}_{ii}^{(m)} = \max\left(1/\lambda_{ii}^{(m)}, \sigma_{ii}^{(vf)2} a_{ii}^2\right) \tag{60}$$

where $\sigma_{ii}^{(vf)2} a_{ii}^2$ is the ith variance floor element in the transformed space.

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, $\boldsymbol{W}_m$, are accumulated, flooring is done as follows:

$$\bar{w}_{ii} = \max\left(w_{ii}, \sigma_{ii}^{(vf)2}\right) \tag{61}$$

where $\bar{w}_{ii}$ and w_{ii} 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

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

floor is applied to the leading diagonal elements. On contrary, for updating the basis coefficients alone, only the transformed statistics, $\boldsymbol{a}_i \boldsymbol{W}_m \boldsymbol{a}_i'$, 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

$$w_{ii}^* = \max\left(\boldsymbol{a}_i \boldsymbol{W}_m \boldsymbol{a}_i, \sigma_{ii}^{(vf)2} a_{ii}^2\right) \tag{62}$$

where w_{ii}^* floored projected statistics associated with the projection vector $\boldsymbol{a}_i$.

Unfortunately, the use of arbitrary basis matrices for SPAM model does not allow trivial variance flooring when only the projected statistics, $\text{Tr}\left(\boldsymbol{W}_m \boldsymbol{S}_i\right)$, are accumulated. However, if one of the basis matrices is initialised to be positive-definite ($\boldsymbol{S}_1$) [2], 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 $p_{ii}^{(ub)} = 1/\left(\sigma_{ii}^{(vf)2} a_{ii}^2\right)$ be the upper bound for the diagonal elements of the precision matrices. The following satisfies the equivalent variance flooring condition:

$$\bar{p}_{ii} = \max\left(p_{ii}, p_{ii}^{(ub)}\right) \tag{63}$$

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

$$\boldsymbol{P}_m = \lambda_{11}^{(m)} \boldsymbol{S}_1 + \sum_{i=2}^{n} \lambda_{ii}^{(m)} \boldsymbol{S}_i \tag{64}$$

variance floor condition may be achieved by finding the minium value of $\lambda_{11}^{(m)}$ that satisfies

$$\bar{p}_{ii} \geq p_{ii}^{(ub)} \tag{65}$$

for all i. However, this approach is computationally inefficient.

3.6 Multiple Transformations Scheme

The basis superposition framework introduced earlier has an extreme basis tying scheme. A set of *global* basis matrices is used. However, extracting these global basis matrices from a large number of Gaussian component is very difficult and requires a large number of basis for accurate modelling. Alternatively, the components can be partitioned into clusters. Each cluster will then contain a smaller number of components. Extracting basis for each cluster of Gaussian components yields more accurate basis information. Thus, the basis matrices are tied at the cluster level. This leads to a multiple projections scheme where each projection is associated with the set of basis matrices. A good summarisation of multiple projections schemes is given in [9]. Multiple HLDA projections models have been found to lead to good recognition performance [16]. For multiple projections basis superposition models, equation (7) can be rewritten as

$$\boldsymbol{P}_m = \sum_{i=1}^{n} \lambda_{ii}^{(m)} \boldsymbol{S}_i^{g(m)} \tag{66}$$

where $m \in g(m)$. There are many ways to perform Gaussian clustering. One way is to use a regression class tree [8] and the terminal nodes of the tree corresponds to the clusters of Gaussian components.

4 Minimum Phone Error (MPE) Training

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

$$\boldsymbol{\theta}_{\text{MLE}} = \arg\max_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}|\boldsymbol{\mathcal{O}}_T) \tag{67}$$

where $\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{\mathcal{O}}_T)$ is the likelihood of the complete set of observation vectors, $\boldsymbol{\mathcal{O}}_T$, given the model parameters, $\boldsymbol{\theta}$. 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 is in fact generated by the proposed model. In speech recognition, HMMs are commonly employed with the observation density function represented by a GMM. The HMM has many limitations in modelling the human speech production mechanism. Furthermore, the observation distribution may not be appropriately modelled using GMM. So, is there another estimator that is less affected by the invalidity of the model assumptions? 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 kind of objective function is known as *discriminative* training.

Recently, discriminative training has been found to yield improved performance in LVCSR compared to the conventional ML training. Various variants of discriminative objective functions such as Maximum Mutual Information (MMI), Minimum Phone Error (MPE) and Minimum Word Error (MWE) criteria have been introduced. Several MMI trained precision matrix models have been published in recent literatures. *Goel et al.*, 2003 [12] presented the MMI estimation of the SPAM models with small vocabulary system. *McDonough et al.* [17] also employed MMI trained STC models in speaker-adapted training (SAT). Besides, *Tsakalidis et al.* [25] also introduced Discriminative Likelihood Linear Transform (DLLT), a variant of MLLT whose parameters estimation is based on the MMI criterion. Due to the increasing popularity of discriminative training in LVCSR systems, discriminative training will be examined in detail for the precision matrix modelling methods in an LVCSR system. The following discussion is based on the MPE training criterion, although the concept is applicable to MMI and MWE criteria, too.

4.1 The Objective and Weak-sense Auxiliary Functions

As the name suggests, MPE training's objective is to estimate the model parameters such that the resulting phone error with respect to the training data is minimised. Because minimising phone error is equivalent to maximising phone accuracy, the objective function for MPE training may be expressed as maximising

$$\mathcal{F}_{\text{MPE}}(\boldsymbol{\theta}) = \sum_{r}^{R} \frac{\sum_{s_n}^{s_N} p_\theta(\mathcal{O}_r|s_n)^\kappa P(s_n)\text{RawPhoneAccuracy}(s_n, s_r)}{\sum_{s_d}^{s_D} p_\theta(\mathcal{O}_r|s_d)^\kappa P(s_d)} \tag{68}$$

where $P(s)$ is the language model probability for sentence s. This equals to the posterior probability of the correct sentence s_r with the inclusion of a probability scale factor κ which can be adjusted to improve the test-set performance. The $\text{RawPhoneAccuracy}(s_n, s_r)$ and $\text{RawWordAccuracy}(s_n, s_r)$ represent the raw *phone* and *word* accuracies respectively of the sentence s_n given the correct sentence s_r. Equation (68) is also known as the MPE criterion.

As with the ML objective function, the MPE objective function is difficult to optimise directly. The extended Baum-Welch (EBW) algorithm may be used, which requires an auxiliary function to be defined. In ML training, the standard Expectation Maximisation algorithm[5] requires a strong-sense auxiliary function which satisfies the following

$$\mathcal{Q}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) - \mathcal{Q}(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\theta}}) \leq \mathcal{F}(\boldsymbol{\theta}) - \mathcal{F}(\hat{\boldsymbol{\theta}}) \tag{69}$$

where $\mathcal{F}$ and $\mathcal{Q}$ are the objective and auxiliary functions respectively. $\hat{\boldsymbol{\theta}}$ and $\boldsymbol{\theta}$ denote the current and new sets of parameters respectively. Thus, a strong-sense auxiliary function is a strict lower-bound of the objective function such that an increase in the auxiliary function guarantees to increase the objective function, $\mathcal{F}(\boldsymbol{\theta})$, unless a local maximum is reached. Hence, by defining an auxiliary function, $\mathcal{Q}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}})$, which is easy to optimise with respect to the model parameters, the objective function can be maximised by iteratively maximising the auxiliary function. In ML training, the auxiliary function is found by applying the Jensen inequality to the objective function and a strict lower bound function is immediately obtained. Unfortunately, in MPE training, it is not trivial to obtain a strict lower bound function that is easy to optimise with respect to the model parameters due to the denominator terms in equation (68). In [27], a weak-sense auxiliary function is introduced to allow efficient maximisation of the MPE objective function. According to [20], a weak-sense auxiliary function, $\mathcal{Q}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}})$, has the same derivative with respect to the model parameters as the objective function, when evaluated at the current estimates of the model parameters, i.e.,

$$\left.\frac{\partial \mathcal{Q}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}})}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}} = \left.\frac{\partial \mathcal{F}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}} \tag{70}$$

An example weak-sense auxiliary function for discriminative training takes the form

$$\mathcal{Q}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) = \mathcal{Q}^{n}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) - \mathcal{Q}^{d}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) + \mathcal{Q}^{sm}(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\theta}}) \tag{71}$$

where $\mathcal{Q}^{n}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}})$ and $\mathcal{Q}^{d}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}})$ are the auxiliary functions for the numerator and denominator terms respectively in the objective function. $\mathcal{Q}^{sm}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}})$ is a smoothing function to ensure the validity of the weak-sense auxiliary function. This function has a zero differential with respect to the model parameters so that equation (71) satisfies equation (70) as a weak-sense auxiliary function. In the following section, this weak-sense auxiliary function will be used to update the model parameters for the precision matrix models.

4.2 MPE Training of Precision Matrix Model Parameters

First, let us recall that the ML auxiliary function is given by

$$\mathcal{Q}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) = K + \frac{1}{2}\sum_{m=1}^{M} \beta_m \Big\{\log|\boldsymbol{P}_m| - \mathrm{Tr}(\boldsymbol{P}_m \boldsymbol{W}_m)\Big\} \tag{72}$$

The full covariance statistics, $\boldsymbol{W}_m$, can be rewritten in terms of the sufficient statistics such that

$$\boldsymbol{W}_m = \frac{1}{\beta_m}\left(\tilde{\boldsymbol{\Sigma}}_m - \tilde{\boldsymbol{\mu}}_m \boldsymbol{\mu}'_m - \boldsymbol{\mu}_m \tilde{\boldsymbol{\mu}}_m + \beta_m \boldsymbol{\mu}_m \boldsymbol{\mu}'_m\right) \tag{73}$$

where

$$\tilde{\boldsymbol{\mu}}_m = \sum_{t=1}^{T} \gamma_m(t) \boldsymbol{o}_t \tag{74}$$

$$\tilde{\boldsymbol{\Sigma}}_m = \sum_{t=1}^{T} \gamma_m(t) \boldsymbol{o}_t \boldsymbol{o}'_t \tag{75}$$

Thus, equation (72) can be rewritten in terms of the sufficient statistics, $\tilde{\boldsymbol{X}} = \{\beta_m, \tilde{\boldsymbol{\mu}}_m, \tilde{\boldsymbol{\Sigma}}_m\}$, given the set of parameters, $\boldsymbol{\theta}$, such that

$$\mathcal{G}_{\boldsymbol{\theta}}(\tilde{\boldsymbol{X}}) = K + \frac{1}{2}\sum_{m=1}^{M} \beta_m \left\{\log|\boldsymbol{P}_m| - trace\left(\boldsymbol{P}_m\left(\frac{\tilde{\boldsymbol{\Sigma}}_m - \tilde{\boldsymbol{\mu}}_m \boldsymbol{\mu}'_m - \boldsymbol{\mu}_m \tilde{\boldsymbol{\mu}}_m + \beta_m \boldsymbol{\mu}_m \boldsymbol{\mu}'_m}{\beta_m}\right)\right)\right\}$$

and equation (71) becomes

$$\mathcal{G}_{\boldsymbol{\theta}}(\tilde{\boldsymbol{X}}) = \mathcal{G}_{\boldsymbol{\theta}}\left(\tilde{\boldsymbol{X}}^{(n)}\right) - \mathcal{G}_{\boldsymbol{\theta}}\left(\tilde{\boldsymbol{X}}^{(d)}\right) + \mathcal{G}_{\boldsymbol{\theta}}\left(\tilde{\boldsymbol{X}}^{(sm)}\right) \tag{76}$$

where $\tilde{\boldsymbol{X}}^{(n)} = \left\{\beta_m^{(n)}, \tilde{\boldsymbol{\mu}}_m^{(n)}, \tilde{\boldsymbol{\Sigma}}_m^{(n)}\right\}$ and $\tilde{\boldsymbol{X}}^{(d)} = \left\{\beta_m^{(d)}, \tilde{\boldsymbol{\mu}}_m^{(d)}, \tilde{\boldsymbol{\Sigma}}_m^{(d)}\right\}$ denote the numerator and denominator statistics respectively. As previously mentioned, the smoothing function has a zero differential w.r.t the model parameters to ensure that the overall function is still a weak-sense auxiliary function. One form of such smoothing function (suggested in [27]) is given by:

$$\mathcal{Q}^{sm}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) = K + \frac{1}{2}\sum_{m=1}^{M} D_m \left\{\log|\boldsymbol{P}_m| - trace(\boldsymbol{P}_m(\hat{\boldsymbol{\Sigma}}_m))\right\} \tag{77}$$

The set of parameters, $\tilde{\boldsymbol{X}}^{(sm)}$, which correspond to the smoothing function are given by

$$\beta_m^{(sm)} = D_m \tag{78}$$

$$\tilde{\boldsymbol{\mu}}_m^{(sm)} = D_m \hat{\boldsymbol{\mu}}_m \tag{79}$$

$$\tilde{\boldsymbol{\Sigma}}_m^{(sm)} = D_m(\hat{\boldsymbol{\Sigma}}_m + \hat{\boldsymbol{\mu}}_m \hat{\boldsymbol{\mu}}_m') \tag{80}$$

where D_m is a smoothing constant to be adjusted to ensure that the resulting weak-sense auxiliary function is convex. Maximising this auxiliary function with respect to the mean vector and covariance matrix parameters yields the following

$$\boldsymbol{\mu}_m = \frac{\tilde{\boldsymbol{\mu}}_m^{(c)} + D\hat{\boldsymbol{\mu}}_m}{\beta_m^{(c)} + D_m} \tag{81}$$

$$\boldsymbol{W}_m = \frac{\tilde{\boldsymbol{\Sigma}}_m^{(c)} + D_m(\hat{\boldsymbol{\Sigma}}_m + \hat{\boldsymbol{\mu}}_m \hat{\boldsymbol{\mu}}_m')}{\beta_m^{(c)} + D_m} - \boldsymbol{\mu}_m \boldsymbol{\mu}_m' \tag{82}$$

where

$$\tilde{\boldsymbol{\mu}}_m^{(c)} = \tilde{\boldsymbol{\mu}}_m^{(n)} - \tilde{\boldsymbol{\mu}}_m^{(d)} \tag{83}$$

$$\tilde{\boldsymbol{\Sigma}}_m^{(c)} = \tilde{\boldsymbol{\Sigma}}_m^{(n)} - \tilde{\boldsymbol{\Sigma}}_m^{(d)} \tag{84}$$

$$\beta_m^{(c)} = \beta_m^{(n)} - \beta_m^{(d)} \tag{85}$$

The set of parameters, $\tilde{\boldsymbol{X}}^{(c)} = \left\{\beta_m^{(c)}, \tilde{\boldsymbol{\mu}}_m^{(c)}, \tilde{\boldsymbol{\Sigma}}_m^{(c)}\right\}$ are known as the combined statistics. We shall see in the next section that this combined statistics can be easily altered to incorporate smoothing techniques such as I-smoothing [20] and Maximum a-Posteriori (MAP) [21] smoothing. To ensure that the auxiliary function is valid, $\boldsymbol{W}_m$ is required to be positive-definite. Combining equations (81) and (82) gives the full covariance statistics in terms of D_m,

$$\boldsymbol{W}_m = \frac{\boldsymbol{B}_2 D_m^2 + \boldsymbol{B}_1 D_m + \boldsymbol{B}_0}{\beta_m^{(c)} + D_m} \tag{86}$$

where

$$\boldsymbol{B}_2 = \hat{\boldsymbol{\Sigma}}_m \tag{87}$$

$$\boldsymbol{B}_1 = \tilde{\boldsymbol{\Sigma}}_m^{(c)} + \beta_m^{(c)}\left(\hat{\boldsymbol{\Sigma}}_m + \hat{\boldsymbol{\mu}}_m \hat{\boldsymbol{\mu}}_m'\right) - \left(\hat{\boldsymbol{\mu}}_m \tilde{\boldsymbol{\mu}}_m^{(c)\prime} + \tilde{\boldsymbol{\mu}}_m^{(c)} \hat{\boldsymbol{\mu}}_m'\right) \tag{88}$$

$$\boldsymbol{B}_0 = \beta_m^{(c)} \tilde{\boldsymbol{\Sigma}}_m^{(c)} - \tilde{\boldsymbol{\mu}}_m^{(c)} \tilde{\boldsymbol{\mu}}_m^{(c)\prime} \tag{89}$$

The smoothing constant, D_m, is thus given by the largest positive eigenvalues of the Quadratic Eigenvalue Problem (QEP) [12] of equation (86). In practice, a lower bound is applied to the smoothing constant value such that the actual smoothing constant value, $\hat{D}$, is given by

$$\hat{D} = \max\left(2D, E\beta_m^d\right) \tag{90}$$

where the lower bound, $E\beta_m^d$ is applied to ensure that the combined occupancy count, $\beta_m^{(c)}$, is greater than zero. $E = 2$ is empirically found to lead to good test-set performance[20].

Once the overall statistics in equations (81) and (82) are found, the auxiliary function in equation (71) can be maximised to discriminatively train the model parameters using the ML techniques described in Section 3.

4.3 I-Smoothing

Previously, a smoothing function was used to ensure that the weak-sense auxiliary function is convex. For highly non-convex functions, a large smoothing constant, D_m, is required. As D_m tends to infinity, equations (81) and (82) tend to the original mean vector and full covariance matrix respectively. Thus, for large values of D_m, the convergence rate is low. Instead of smoothing onto the old parameters set, a better model set may be used to achieve more reliable smoothing. In [20], *Povey .et .al* introduced the I-smoothing technique which was found to improve robustness in MPE training. Instead of using the current estimate of the parameters as the smoothing values, I-smoothing employs dynamic smoothing values obtained from the ML estimated parameters. Incorporation of I-smoothing method into MPE training of precision matrix models can be realised by simply adopting the technique described in [20]. The I-smoothing aspect is represented by the I-smoothing function, $\mathcal{Q}^{ml}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}})$, such that equation (71) is redefined as

$$\mathcal{Q}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) = \mathcal{Q}^{n}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) - \mathcal{Q}^{d}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) + \mathcal{Q}^{sm}(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\theta}}) + \mathcal{Q}^{ml}(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\theta}}) \tag{91}$$

where

$$\mathcal{Q}^{ml}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) = K + \frac{1}{2}\sum_{m=1}^{M} \tau^I \left\{ \log|\boldsymbol{P}_m| - \mathrm{Tr}\left(\boldsymbol{P}_m \left(\frac{\tilde{\boldsymbol{\Sigma}}_m^{(ml)} - \tilde{\boldsymbol{\mu}}_m^{(ml)}\tilde{\boldsymbol{\mu}}_m^{(ml)\prime}/\beta_m^{(ml)}}{\beta_m^{(ml)}}\right)\right)\right\} \tag{92}$$

τ^I is the I-smoothing constant and $\beta_m^{(ml)}$, $\tilde{\boldsymbol{\mu}}_m^{(ml)}$ and $\tilde{\boldsymbol{\Sigma}}_m^{(ml)}$ are the required ML statistics. Incorporating I-smoothing is easy by rewriting the combined statistics in equations (83), (84) and (85) as

$$\begin{aligned}
\tilde{\boldsymbol{\mu}}_m^{(c)} &= \tilde{\boldsymbol{\mu}}_m^{(n)} - \tilde{\boldsymbol{\mu}}_m^{(d)} + \frac{\tau^I}{\beta_m^{(ml)}}\tilde{\boldsymbol{\mu}}_m^{(ml)} && (93)\\
\tilde{\boldsymbol{\Sigma}}_m^{(c)} &= \tilde{\boldsymbol{\Sigma}}_m^{(n)} - \tilde{\boldsymbol{\Sigma}}_m^{(d)} + \frac{\tau^I}{\beta_m^{(ml)}}\tilde{\boldsymbol{\Sigma}}_m^{(ml)} && (94)\\
\beta_m^{(c)} &= \beta_m^{(n)} - \beta_m^{(d)} + \tau^I && (95)
\end{aligned}$$

It is easy to see that as the I-smoothing constant, τ^I, tends to infinity, the resulting estimation formulae tend to those of the ML training.

4.4 Approximating the Smoothing Constant, D_m

Determination of the smoothing constant value as described earlier is memory inefficient because solving the quadratic eigenvalue problem for equation (86) requires storing the full covariance statistics. In Section 3.3, it was mentioned that storing of full covariance statistics for large systems results in intensive memory requirement.

For STC and EMLLT models, the smoothing constant can be determined more efficiently by considering the transformed mean and variance vectors. By applying the ith projection vector to equation (86), the transformed variance statistics is thus given by

$$\boldsymbol{a}_i \boldsymbol{W}_m \boldsymbol{a}_i' = \frac{b_2^{(i)} D_m^2 + b_1^{(i)} D_m + b_0^{(i)}}{\beta_m^{(c)} + D_m} \tag{96}$$

where

$$b_2^{(i)} = \boldsymbol{a}_i \hat{\boldsymbol{\Sigma}}_m \boldsymbol{a}_i' \tag{97}$$

$$b_1^{(i)} = \boldsymbol{a}_i (\tilde{\boldsymbol{\Sigma}}_m^{(c)} + \beta_m^{(c)} \hat{\boldsymbol{\Sigma}}_m) \boldsymbol{a}_i' + \beta_m^{(c)} (\boldsymbol{a}_i \hat{\boldsymbol{\mu}}_m)^2 - 2 (\boldsymbol{a}_i \hat{\boldsymbol{\mu}}_m) \left(\boldsymbol{a}_i \tilde{\boldsymbol{\mu}}_m^{(c)}\right) \tag{98}$$

$$b_0^{(i)} = \beta_m^{(c)} \boldsymbol{a}_i \tilde{\boldsymbol{\Sigma}}_m^{(c)} \boldsymbol{a}_i' - \left(\boldsymbol{a}_i \tilde{\boldsymbol{\mu}}_m^{(c)}\right)^2 \tag{99}$$

Hence, the QEP is simplified to solving n independent quadratic equations given by equation (96) using only the transformed statistics, $\boldsymbol{a}_i \tilde{\boldsymbol{\mu}}_m^{(c)}$ and $\boldsymbol{a}_i \tilde{\boldsymbol{\Sigma}}_m^{(c)} \boldsymbol{a}_i'$. Thus, the same set of statistics is used to determine the smoothing constant, D_m, and to estimate the model parameters.

Unlike STC and EMLLT models where the basis matrices are rank-1, the 'projected' statistics, $\mathrm{Tr}(\boldsymbol{W}_m \boldsymbol{S}_i)$, associated to the basis matrices, $\boldsymbol{S}_i$ of the SPAM model can not be used to determine the smoothing constant. There is no way to infer the positive-definiteness of the full covariance statistics, $\boldsymbol{W}_m$, from the 'projected' statistics, $\mathrm{Tr}(\boldsymbol{W}_m \boldsymbol{S}_i)$. Instead of obtaining the exact smoothing constant value by solving the QEP for equation (86), this value may 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 broken down into 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_m ($\boldsymbol{a}_i^* \boldsymbol{W}_m \boldsymbol{a}_i^{*'}$) and the other one for estimating the model parameters ($\mathrm{Tr}(\boldsymbol{W}_m \boldsymbol{S}_i)$). To yield a good approximation for the smoothing constant, $\boldsymbol{A}^*$ should be chosen such that the transformed space is as uncorrelated as possible. So, it is intuitive to select the STC transform as the *pseudo* transformation matrix. In the case where STC transform is unavailable, an identity matrix may be used. In fact, in the next section, it will be shown that the use of identity matrix as the *pseudo* transformation matrix yields very good approximation.

4.5 Building Tied-Variance Systems

Although the structured precision matrix approximation provides compact model representation, the complete update of the model parameters (tied and untied) requires accumulation of full covariance statistics. This is highly inefficient for large systems with a large amount of variance components. Instead of just updating the basis coefficients to keep the memory requirement low, tied-variance systems can also be built to limit the total number of variance components. For systems which employ GMM as the observation pdf, the Gaussian components may be training by iteratively splitting the components. This process is known as centroid splitting or *mix-up*. This provides a convenient way of building tied-variance systems. Consider a system with m_1 components is mixed-up to m_2 components. On each split, the resulting two components will have distinct mean components, but a *shared* variance component. Thus, the final system will have effectively m_2 mean components but only m_1 variance components.

Since the variances are tied over multiple Gaussian components, it is more convenient to modify equations (87), (88) and (89) such that they are not dependent on the mean statistics, $\tilde{\boldsymbol{\mu}}_m^{(c)}$. To do this, let us redefine the mean and variance statistics such that

$$\begin{aligned}
\bar{\boldsymbol{\mu}}_m &= \sum_{t=1}^{T} \gamma_m(t)(\boldsymbol{o}_t - \hat{\boldsymbol{\mu}}_m) \\
&= \tilde{\boldsymbol{\mu}}_m - \beta_m \hat{\boldsymbol{\mu}}_m && (100) \\
\bar{\boldsymbol{\Sigma}}_m &= \sum_{t=1}^{T} \gamma_m(t)(\boldsymbol{o}_t - \hat{\boldsymbol{\mu}}_m)(\boldsymbol{o}_t - \hat{\boldsymbol{\mu}}_m)' \\
&= \tilde{\boldsymbol{\Sigma}}_m - \bar{\boldsymbol{\mu}}_m \hat{\boldsymbol{\mu}}_m' - \hat{\boldsymbol{\mu}}_m \bar{\boldsymbol{\mu}}_m' - \beta_m \hat{\boldsymbol{\mu}}_m \hat{\boldsymbol{\mu}}_m' && (101)
\end{aligned}$$

Then, the mean and variance updates in equations (81) and (82) may be rewritten as

$$\begin{aligned}
\boldsymbol{\mu}_m &= \hat{\boldsymbol{\mu}}_m + \frac{\bar{\boldsymbol{\mu}}_m^{(c)}}{\beta_m^{(c)} + D_m} && (102) \\
\boldsymbol{W}_m &= \frac{\bar{\boldsymbol{\Sigma}}_m^{(c)} + \bar{\boldsymbol{\mu}}_m \hat{\boldsymbol{\mu}}_m' + \hat{\boldsymbol{\mu}}_m \bar{\boldsymbol{\mu}}_m' + D_m \hat{\boldsymbol{\Sigma}}_m}{\beta_m^{(c)} + D_m} + \hat{\boldsymbol{\mu}}_m \hat{\boldsymbol{\mu}}_m' - \boldsymbol{\mu}_m \boldsymbol{\mu}_m' \\
&= \frac{(\beta_m^{(c)} + D_m)(\bar{\boldsymbol{\Sigma}}_m^{(c)} + D_m \hat{\boldsymbol{\Sigma}}_m) + \bar{\boldsymbol{\mu}}_m \bar{\boldsymbol{\mu}}_m'}{\beta_m^{(c)} + D_m} && (103)
\end{aligned}$$

This simplifies equations (87), (88) and (106) as

$$\begin{aligned}
\bar{\boldsymbol{B}}_2 &= \hat{\boldsymbol{\Sigma}}_m && (104) \\
\bar{\boldsymbol{B}}_1 &= \bar{\boldsymbol{\Sigma}}_m^{(c)} + \beta_m^{(c)} \hat{\boldsymbol{\Sigma}}_m && (105) \\
\bar{\boldsymbol{B}}_0 &= \beta_m^{(c)} \bar{\boldsymbol{\Sigma}}_m^{(c)} + \left\{ \bar{\boldsymbol{\mu}}_m^{(c)} \bar{\boldsymbol{\mu}}_m^{(c)\prime} \right\} && (106)
\end{aligned}$$

Setting $\bar{\boldsymbol{\mu}}_m = 0$ leaves the mean parameters unchanged as given by equation (102). This is a reasonable assumption provided the mean is well trained. Thus, the term in the curly brackets in equation (106) may be dropped for tied-variance systems.

5 Experimental Results

Discriminative training of precision matrices was evaluated on a conversational telephone speech (CTS) task, which consists of multi-speaker spontaneous telephone conversational speech, provided by the Linguistic Data Consortium (LDC). Data was coded into 12 PLP coefficients at a frame rate of 10ms with a frame size of 25ms, together with the log energy term, first, second and third derivatives to form 52-dimensional feature vectors. Two training sets were used. The first set is the `h5etrain03` training data, comprising 296 hours of speech. The second set is the subset of the first one, `h5etrain03sub`, with 76 hours of data. Acoustic models are represented by decision tree state-clustered triphone models with 6189 distinct states. Side-based Ceptral Mean Normalisation (CMN), Cepstral Variance Normalisation (CVN) and Vocal Tract Length Normalisation (VTLN) are used in all the systems. System performance is evaluated on the 2.97 hours `dev01sub` test set. The decoding dictionary consists of 58231 words with multiple pronunciation probabilities.

The models used in all the experiments were built using HTK[28]. Initial experiments were based on the `h5etrain03` training set and the `dev01sub` test set. The first set of experiments were conducted to investigate various initialisation schemes for EMLLT models. Next, various forms of precision matrix models were compared. The performance of multiple transforms systems were also evaluated. The robustness of these models were investigated by using a smaller training set(`h5etrain03sub`). Finally, selected systems were evaluated on the full CTS and Broadcast News (BN) English tasks.

5.1 Comparing different EMLLT initialisation Schemes

As discussed in Section 3.4.1, initialisation is important to ensure good starting point that leads to faster convergence. Here, several forms of initialisation schemes introduced earlier were compared. Table 3 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 additional third derivatives). UI, the uninformative

Initialisation	Basis Matrix Dimensions	Average Log Likelihood	WER (%)
UI	78 x 39	-64.562	34.0
STC+HLDA	(39 x 39)+(13 x 39)	-63.621	33.4
STC+HLDA	(39 x 39)+(26 x 39)	-63.508	33.2
STC+EYE	(39 x 39)+(39 x 39)	-63.472	33.1
STC (SS)	(39 x 39)+(39 x 39)	-63.468	33.2
STC+HLDA	(52 x 52)+(39 x 52)	-53.513	32.6
STC+EYE	(52 x 52)+(39 x 52)	-53.534	33.0

Table 3: Average log likelihood and WER (%) performance for various EMLLT initialisation schemes

initialisation is clearly the worst scheme. This method simply distributes a set of vectors that span the complete space of a symmetric matrix to form the required basis vectors. This also clearly illustrates the sensitivity of EMLLT model performance toward the initialisation schemes. On the other hand, the STC-EYE and STC(SS) methods provide much more sensible initialisations, giving WERs of 33.1% and 33.2% respectively. The STC-HLDA method, on the other hand, provides a convenient way of initialising a variable number of basis vectors. In fact, the last two rows of Table 3 illustrate that STC-HLDA is preferred to simply truncating the identity matrix (STC-EYE) when the number of basis vectors is not multiple of the feature dimensionality, d.

5.2 Comparing various 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. This baseline system uses a 39×52 HLDA frontend projection with 39-dimensional mean vectors and diagonal covariance matrices. ML training was conducted with four Baum-Welch iterations while MPE models were trained with 8 iterations. 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. Similar to HLDA models, basis matrices for SPAM models were initialised once 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) numbers are summarised in Table 4. 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, an STC model is transformed into the HLDA-PMM model with effectively 39 coefficients per Gaussian component. This model gives another 0.1% absolute improvement over the STC model. In MPE training, 8 iterations were performed. The 16-component HLDA model gave 3.7% absolute gain over the ML model. Similar gains were obtained from EMLLT and HLDA-PMM models, 3.4% and 3.8% absolute reduction in WER respectively. Hence, after 8 MPE iterations, the EMLLT system gave 0.6% absolute improvement over the

System	# of Gaussian components	Effective parameters per state	Dimensions		WER (%)	
			$\boldsymbol{\mu}$	$\boldsymbol{\Sigma}$	ML	MPE
Baseline HLDA	16	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 4: Comparison of number of parameters and WER (%) performance of ML and MPE trained precision matrix models

baseline. Meanwhile, the HLDA-PMM model is 0.3% absolute better than the STC model, using fewer number of free parameters. The SPAM model, with approximately 14% fewer number of parameters, yields similar results as EMLLT. This is because the SPAM model constitutes a more powerful set of basis matrices and hence requires a smaller number of basis matrices for accurate approximation. Furthermore, building a SPAM model within the HLDA subspace (HLDA+SPAM) gave a further 0.7% absolute WER reduction compared to SPAM after MPE training. This results in a total absolute improvement of 1.3% over the baseline. This clearly shows the importance of compact model representation.

5.3 Multiple Transformations Schemes

Multiple transformations models provides a simple and powerful way of improving modelling accuracies without severely increasing the total number of model parameters. Such technique also naturally fits within the basis superposition framework where the basis extraction process is carried out for each cluster of Gaussian components. 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-transform (64 speech, 1 silence) HLDA and STC models[4]. 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 5 summarises the WER results for multiple projections HLDA, STC and EMLLT models.

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 5: Comparing WER (%) performance of precision matrix models with multiple transformations

Compared to single transform models, multiple transform HLDA, STC and EMLLT models gave 0.8%, 1.0% and 1.5% absolute WER reduction respectively. This translate to 2.4%, 3.0% and 4.6% relative reductions respectively. This trend is expected because the number of model parameters

[4]The multiple transforms HLDA and STC models were obtained from X. Liu. These models have been trained and decoded using the same setup as described earlier.

associated to the transform (basis vectors) increases in the same order. After 4 MPE training iterations, the relative improvements for HLDA, STC and EMLLT were observed to be 3.6%, 2.0% and 2.7% respectively compared to their corresponding single-transform models. No particular trend is observed in this case. Since only the basis coefficients are discriminatively trained, the model with the least number of model parameters is expected to yield the greatest gain. This is evidently shown by the large gain from the 65-transform HLDA model. Surprisingly, the 64-transform EMLLT model gave a larger gain compared to the STC model. In fact, this EMLLT model gave a further 0.5% absolute WER reduction after 4 addition MPE training iterations. 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.

5.4 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 6 compares the WER performance of 16-component HLDA, EMLLT and SPAM

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 6: Comparison of performance for models trained on `h5etrain03sub`

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.

Besides, two different SPAM models were also trained with 4 MPE iterations, using the exact and approximated smoothing constant as described in Section 4. Instead of finding the exact solution by solving the quadratic eigenvalue problem, a quicker solution is to start with a small smoothing constant and gradually increases its value until the resulting full covariance 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

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 7: Comparing exact and approximated smoothing constant

from Table 7 reveals that the approximation of smoothing constant using the *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.

5.5 Precision Matrix Models on LVCSR Systems

So far, the performance of various precision matrix models was presented based on the `dev01sub` test set for CTS task. To obtain a more complete overview of these models, three models were selected to be trained on the `h5etrain03` data set and evaluated on the 6 hours `eval03` test set. These models are the 28-component HLDA, 16-component 64-transform EMLLT, 28-component SPAM and 28-component HLDA+SPAM models. As before, the models were trained with 4 ML iterations and 8 MPE iterations. Due to memory constraint, is it not possible to accumulate full covariance statistics for the 28-component systems. Hence, the basis vectors(matrices) for EMLLT(SPAM) models were initialised using the 16-component systems. The results are tabulated in Table 8. The WERs of the baseline HLDA system on `dev01sub` are 32.3% and 29.1% respectively

System	Dim	# of xforms	# of comp	Effective params per state	dev01sub		eval03	
					MLE	MPE	MLE	MPE
Baseline HLDA	39	1	28	2184	32.3	29.1	31.7	28.4
EMLLT	78	64	16	2080	32.0	28.3	31.7	28.1
SPAM	39	1	28	2548	31.5	28.3	30.8	27.6
HLDA+SPAM	39	1	28	2184	31.1	27.9	30.4	27.3

Table 8: WER performance of precision matrix models on `dev01sub` and `eval03` for `CTS` task

for ML and MPE trained models. On `eval03`, the WERs are 31.7% and 28.4% respectively. 3.2% and 3.3% absolute improvements were obtained on `dev01sub` and `eval03` respectively after MPE training on HLDA system. The 64-transform 16-component EMLLT model has approximately 100 parameters per state fewer than the baseline. After ML and MPE training, the improvements obtained were 0.3% and 0.8% respectively on `dev01sub`. The gain from MPE training is surprisingly large. However, this gain does not generalise to `eval03` test set. The gain of the 64-transform EMLLT model was rather disappointing. No gain was observed after ML training and only 0.3% difference was obtained after MPE training. On the other hand, 28-component SPAM model, with about 400 parameters per state more than the HLDA system (due to the *unprojected* mean vectors), gave a consistent absolute WER reduction of 0.8% after both ML and MPE training on `dev01sub`. On `eval03`, similar gains were observed: 0.9% and 0.8% WER reductions after ML and MPE training respectively. Finally, the HLDA+SPAM model, with similar number of model parameters as the baseline, gave the best performance of all. It gave a further 0.4% and 0.3% WER reduction over the SPAM model on `dev01sub` and `eval03` respectively. Overall, the 28-comp HLDA+SPAM model is 1.2% and 1.1% absolute better than the baseline on `dev01sub` and `eval03` respectively.

To generalise the performance of the precision matrix models, several systems were also built for the Broadcast News (BN) English tasks. The training data set, `bnac`, consists of 144 hours training samples. All the systems have 16 Gaussian components per state. These systems were evaluated on `dev03` and `eval03` test sets, each consists of 3 hours training samples. The preformance of several precision matrix models are tabulated in Table 9. First, an approach similar

System	# of xforms	dev03			eval03		
		MLE	MPE	GD	MLE	MPE	GD
Baseline HLDA	1	17.7	15.2	14.9	15.6	13.7	13.6
EMLLT	64	17.2	15.1	–	15.6	13.9	–
SPAM	1	17.0	15.1	–	15.4	13.7	–
HLDA-PMM	1	17.6	15.7	–	15.8	14.2	–
HLDA+SPAM	1	16.9	14.9	14.6	15.1	13.4	13.4

Table 9: WER performance of precision matrix models on `dev03` and `eval03` for `BN` tasks

to the CTS systems is adopted. The single-transform HLDA, 64-transform EMLLT and single-transform SPAM models were trained. The baseline HLDA system gave 17.7% and 15.6% WERs on `dev03` and `eval03` respectively after ML and 15.2% and 13.7% respectively after MPE training. Due to the large number of free parameters for the 64-transform EMLLT model, only 0.1% gain was obtained on `dev03` after MPE training. Furthermore, this model is 0.2% worse than the baseline on `eval03`. The gains observed for SPAM models were also small: 0.1% on `dev03` and no improvement on `eval03`. It was suspected that modelling the 52-dimensional mean vectors in this case is sub-optimal. Thus, the HLDA-PMM model was built to verify this. The results indeed revealed that the HLDA-PMM models performed worse than the HLDA models. Hence, the HLDA+SPAM model was built. This model gave 0.8% and 0.5% absolute WER reduction on `dev03` and `eval03` respectively after ML training. The gain from ML training is slightly higher than expected on `dev03`. This may be due to the quality of the `dev03` transcriptions. After MPE training, HLDA+SPAM model is 0.3% absolute better than the baseline on both test sets. In addition, gender dependent (GD) systems were also built for HLDA and HLDA+SPAM systems. Starting from the models after 8 MPE iterations, gender-dependent models were built with 3 MPE+MAP[21] iterations, using the gender-independent MPE models as the prior. The HLDA system gave a further 0.3% and 0.1% 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`.

6 Conclusions

This report has presented the implementation of various existing precision matrix models in LVCSR systems, including the semi-tied covariance (STC), extended MLLT (EMLLT) and subspace for precision and mean (SPAM) models. These precision matrix models were generalised using a basis superposition framework, where the parameters comprises the basis matrices (tied parameters) and the basis coefficients (untied parameters). Models belonging to this generic framework allow efficient likelihood calculation which leads to efficient decoding. In addition, this report also presented HLDA as a precision matrix model with the nuisance dimension basis coefficients globally tied (HLDA-PMM). Various issues concerning large system implementation were addressed. In particular, computational tractability and memory requirement are two important factors that affect the efficiency of the systems. Besides the efficient likelihood calculation, it was also shown that with the basis superposition formulation, compact model representation can be achieved which enhances the robustness in parameters estimation. High computational cost and slow convergence of the basis matrix update motivate the use of good initialisation schemes. Several EMLLT basis vectors initialisation methods were introduced. With good initialisation, these models can be trained by updating only the basis coefficients, which is more efficient and requires significantly less memory.

Discriminative training of various precision matrix models was presented based on the minimum phone error (MPE) criterion. The use of a weak-sense auxiliary function allows various required statistics (numerator, denominator and smoothing) to be combined to yield a similar auxiliary function as for the maximum likelihood training. Thus, once the combined statistics are computed, the update formulae derived for ML training may be directly applied to discriminatively train the model parameters. A smoothing constant needs to be determined to ensure validity of the weak-sense auxiliary function. The inefficiency in solving the QEP for SPAM models can be alleviated by using a *pseudo* transformation matrix to mimic the smoothing constant determination process for STC or EMLLT models. This approximation was found to yield similar WER performance compared to using the exact solutions obtained from QEP.

Finally, the experimental results reveal that various precision matrix models outperform the standard HLDA diagonal covariance matrix system on CTS English Task. The improvement obtained from the EMLLT MPE model was 0.6% absolute at the expense of using greater model parameters. Meanwhile, the SPAM models provide a more powerful set of basis matrices, which improved the baseline by 0.8% absolute. When combined with HLDA, the HLDA+SPAM model was found to yield the best performance, giving 1.1% absolute WER reduction. Similarly, the best system for BN English task was found to be the HLDA+SPAM, with 0.2% absolute improvement. Without dramatical increase in the total number of model parameters, multiple transformation models were found to yield between 2% to 5% relative reduction in word error rate compared to single-transform models.

In short, it can be concluded that various precision matrix models have been successfully implemented in LVCSR discriminatively trained systems and several implementation issues were addressed to yield robust training and efficient decoding. The robustness of SPAM and HLDA-PMM models motivates the research in more compact model structures for improved performance, particularly for LVCSR.

Acknowledgements

The authors would like to thank X. Liu, Cambridge University, for his multiple transforms HLDA and STC models. This work was supported by DARPA grant MDA972-02-1-0013. The paper does not necessarily reflect the position or the policy of the US Government and no official endorsement should be inferred.

References

[1] 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. Eurospeech*, 2003.

[2] S. Axelrod, R. Gopinath, and P Olsen. Modeling with a subspace constraint on inverse covariance matrices. In *Proc. ICSLP*, 2002.

[3] 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.

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

[5] 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.

[6] 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. ICASSP*, 2004.

[7] 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.

[8] 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. Available via anonymous ftp from: svr-www.eng.cam.ac.uk.

[9] M.J.F. Gales. Maximum likelihood multiple projection schemes for hidden Markov models. Technical Report CUED/F-INFENG/TR365, Cambridge University, 1999. Available via anonymous ftp from: svr-www.eng.cam.ac.uk.

[10] M.J.F. Gales. Semi-tied covariance matrices for hidden Markov models. *IEEE Transactions on Speech and Audio Processing*, 7(3):272–281, 1999.

[11] N.K. Goel and A.G. Andreou. Heteroscedastic discriminant analysis and reduced-rank HMMs for improved speech recognition. *Speech Communications*, 26:283–297, 1998.

[12] V. Goel, S. Axelrod, R. Gopinath, P Olsen, and K. Visweswariah. Discriminative estimation of Subspace Precision and Mean (SPAM) models. In *EUROSPEECH*, 2003.

[13] H. Hermansky. Perceptual Linear Predictive (PLP) analysis of speech. *Journal of the Acoustic Society of America*, 87(4):1738–1752, 1990.

[14] 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. ICSLP*, 2002.

[15] N. Kumar. *Investigation of Silicon-Auditory Models and Generalization of Linear Discriminant Analysis for Improved Speech Recognition*. PhD thesis, Johns Hopkins University, 1997.

[16] 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.

[17] J. McDonough and A. Waibel. Maximum mutual information speaker adapted training with semi-tied covariance matrices. In *Proc. ICASSP*, 2003.

[18] P. Olsen and R.A. Gopinath. Modelling inverse covariance matrices by basis expansion. In *Proc. ICASSP*, 2002.

[19] E. Polak. *Computational Methods in Optimization: A Unified Approach*. Academic Press, 1971.

[20] D. Povey and P.C. Woodland. Minimum Phone Error and I-smoothing for improved discriminative training. In *Proc.ICASSP*, 2002.

[21] D. Povey, P.C. Woodland, and M.J.F. Gales. Discriminative MAP for acoustic model adaptation. In *ICASSP*, 2003.

[22] 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. Available via anonymous ftp from: svr-www.eng.cam.ac.uk.

[23] G. Saon, M. padmanabhan, R. Gopinath, and S. Chen. Maximum likelihood discriminant feature spaces. In *Proc. ICASSP*, 2000.

[24] K.C. Sim and M.J.F. Gales. Basis superposition precision matrix modelling for large vocabulary continuous speech recognition. In *Proc. ICASSP*, 2004.

[25] S. Tsakalidis, V. Doumpiotis, and W. Byrne. Discriminative linear transforms for feature normalisation and speaker adaptation in hmm estimation. In *Proc. ICSLP*, 2002.

[26] K. Visweswariah, P. Olsen, R. Gopinath, and S. Axelrod. Maximum likelihood training of subspaces for inverse covariance modeling. In *Proc. ICASSP*, 2003.

[27] P.C. Woodland and D. Povey. Large scale discriminative training of hidden Markov models in speech recognition. *Computer Speech and Language*, 16(1):25–48, Jan 2002.

[28] S. Young, D. Kershaw, J. Odell, D. Ollason, V. Valtchev, and P. Woodland. *The HTK Book (for HTK version 3.0)*. Cambridge University, 1997.

[29] 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.

Appendix

A Useful Matrix Algebra

A1 Matrix Determinant and Inversion Identities

This is useful when inverting a matrix of the form $\boldsymbol{A} + \boldsymbol{BDC}$. Its inverse is given by:

$$\boxed{(\boldsymbol{A} + \boldsymbol{BDC})^{-1} = \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{B}(\boldsymbol{D}^{-1} + \boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B})^{-1}\boldsymbol{C}\boldsymbol{A}^{-1} \qquad \text{(A1)}}$$

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 $m = 1$, equation (A1) becomes

$$\boxed{(\boldsymbol{A} + d\boldsymbol{b}\boldsymbol{c}')^{-1} = \boldsymbol{A}^{-1} - \frac{d\boldsymbol{A}^{-1}\boldsymbol{b}\boldsymbol{c}'\boldsymbol{A}^{-1}}{(1 + d\boldsymbol{c}'\boldsymbol{A}^{-1}\boldsymbol{b})} \qquad \text{(A2)}}$$

and its determinant can be found using

$$\boxed{|\boldsymbol{A} + d\boldsymbol{b}\boldsymbol{c}'| = (1 + d\boldsymbol{c}'\boldsymbol{A}^{-1}\boldsymbol{b})|\boldsymbol{A}| \qquad \text{(A3)}}$$

A2 Vector and Matrix Differentiations

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:

$$\boxed{\begin{aligned}
\frac{\partial(\boldsymbol{A}\boldsymbol{b})}{\partial\boldsymbol{b}} &= \boldsymbol{A}' && \text{(A4)}\\
\frac{\partial(\boldsymbol{b}'\boldsymbol{A}\boldsymbol{b})}{\partial\boldsymbol{b}} &= (\boldsymbol{A} + \boldsymbol{A}')\boldsymbol{b} && \text{(A5)}\\
\frac{\partial|\boldsymbol{A}|}{\partial x} &= \mathrm{Tr}\left(\boldsymbol{A}^{-1}\frac{\partial\boldsymbol{A}}{\partial x}\right) && \text{(A6)}
\end{aligned}}$$

B Concepts and Derivations

B1 Precision Matrix 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

$$x_i = \sum_{j=1}^{i-1} c_{ij}x_j + w_i \qquad \text{(B1)}$$

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

$$\begin{aligned} \boldsymbol{x} &= \boldsymbol{C}\boldsymbol{x} + \boldsymbol{w} && \text{(B2)} \\ \boldsymbol{x} &= (\boldsymbol{I} - \boldsymbol{C})^{-1}\boldsymbol{w} && \text{(B3)} \\ &= \boldsymbol{L}^{-1}\boldsymbol{w} && \text{(B4)} \end{aligned}$$

where $\boldsymbol{C}$ and $\boldsymbol{L}$ are a lower triangular matrices given by

$$\boldsymbol{C} = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ c_{21} & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ c_{d1} & \cdots & c_{d(d-1)} & 0 \end{pmatrix} \quad \boldsymbol{L} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ -c_{21} & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ -c_{d1} & \cdots & -c_{d(d-1)} & 1 \end{pmatrix} \tag{B5}$$

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

$$\boldsymbol{P} = \boldsymbol{L}'\boldsymbol{\Lambda}\boldsymbol{L} \tag{B6}$$

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

$$p_{ij} = \sum_{k=i}^{d} \lambda_{kk} c_{ki} c_{kj} \tag{B7}$$

If x_i and x_j are conditionally independent, $c_i j = 0$. 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.

B2 STC Parameters Update

The precision matrix expression for a STC model is given by equation (24). Substituting this into equation (40) yields the following auxiliary function:

$$\begin{aligned} \mathcal{Q}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) &= K + \frac{1}{2}\sum_{m=1}^{M} \beta_m \left\{ 2\log|\boldsymbol{A}| + \log|\boldsymbol{\Lambda}_m| - \sum_{i=1}^{n} \lambda_{ii}^{(m)} \boldsymbol{a}_i \boldsymbol{W}_m \boldsymbol{a}_i' \right\} \\ &= K + \frac{1}{2}\sum_{m=1}^{M} \beta_m \left\{ 2\log|\boldsymbol{A}| + \sum_{i=1}^{n} \log(\lambda_{ii}^{(m)}) - \lambda_{ii}^{(m)} \boldsymbol{a}_i \boldsymbol{W}_m \boldsymbol{a}_i' \right\} \\ &= K + \frac{1}{2}\sum_{m=1}^{M} \beta_m \left\{ 2\log(\boldsymbol{a}_i \boldsymbol{c}_i) + \sum_{i=1}^{n} \log(\lambda_{ii}^{(m)}) - \lambda_{ii}^{(m)} \boldsymbol{a}_i \boldsymbol{W}_m \boldsymbol{a}_i' \right\} && \text{(B8)} \end{aligned}$$

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 (B8) w.r.t. $\lambda_{ii}^{(m)}$ and equate to zero:

$$\frac{\partial \mathcal{Q}}{\partial \lambda_{ii}^{(m)}} = \frac{\beta_m}{2} \left\{ \frac{1}{\lambda_{ii}^{(m)}} - \boldsymbol{a}_i \boldsymbol{W}_m \boldsymbol{a}_i' \right\} = 0 \tag{B9}$$

This yields the ML update formula for $\lambda_{ii}^{(m)}$ as

$$\lambda_{ii}^{(m)} = \frac{1}{\boldsymbol{a}_i \boldsymbol{W}_m \boldsymbol{a}_i'} = \frac{1}{diag(\boldsymbol{A}\boldsymbol{W}_m \boldsymbol{A}')} \tag{B10}$$

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 (B8) w.r.t. $\boldsymbol{a}_i$ and equate to zero:

$$\begin{aligned} \frac{\partial \mathcal{Q}}{\partial \boldsymbol{a}_i} &= \sum_{m=1}^{M} \beta_m \left\{ \frac{\boldsymbol{c}_i'}{\boldsymbol{a}_i \boldsymbol{c}_i} - \lambda_{ii}^{(m)} \boldsymbol{a}_i \boldsymbol{W}_m \right\} \\ &= \frac{\beta \boldsymbol{c}_i'}{\boldsymbol{a}_i \boldsymbol{c}_i} - \boldsymbol{a}_i G_i \\ &= \frac{\boldsymbol{G}_i (\beta \boldsymbol{c}_i' \boldsymbol{G}_i^{-1} - \boldsymbol{a}_i)}{\boldsymbol{a}_i \boldsymbol{c}_i} = 0 \end{aligned} \tag{B11}$$

where the required statistics are given by

$$\boldsymbol{G}_i = \sum_{m=1}^{M} \beta_m \lambda_{ii}^{(m)} \boldsymbol{W}_m \tag{B12}$$

$$\beta = \sum_{m=1}^{M} \beta_m \tag{B13}$$

It is clear that equation (B11) is true if and only if $\boldsymbol{c}_i' \boldsymbol{G}_i^{-1}$ is parallel to $\boldsymbol{a}_i$. So,

$$\boldsymbol{a}_i = \alpha \boldsymbol{c}_i' \boldsymbol{G}_i^{-1} \tag{B14}$$

where α is a scalar. Substituting this back into equation (B11) yields

$$\alpha = \sqrt{\frac{\beta}{\boldsymbol{c}_i' \boldsymbol{G}_i^{-1} \boldsymbol{c}_i}} \tag{B15}$$

and hence,

$$\boldsymbol{a}_i = \boldsymbol{c}_i' \boldsymbol{G}_i^{-1} \sqrt{\frac{\beta}{\boldsymbol{c}_i' \boldsymbol{G}_i^{-1} \boldsymbol{c}_i}} \tag{B16}$$

B3 EMLLT Parameters Update

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 (40). 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 $\lambda_{ii}^{(m)}$ 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 $\lambda_{ii}^{(m)}$ be written as

$$\lambda_{ii}^{(m)} = \hat{\lambda}_{ii}^{(m)} + \Delta_{ii}^{(m)} \tag{B17}$$

Thus, the auxiliary function in equation (40) can be rewritten as

$$\begin{aligned}
\mathcal{Q}_\Delta(\boldsymbol{\theta},\hat{\boldsymbol{\theta}}) &= \mathcal{Q}_\Delta(\hat{\boldsymbol{\theta}},\hat{\boldsymbol{\theta}}) + \frac{1}{2}\sum_{m=1}^{M}\beta_m\left\{\log\left|\frac{\boldsymbol{P}_m + \Delta_{ii}^{(m)}\boldsymbol{a}_i'\boldsymbol{a}_i}{\boldsymbol{P}_m}\right| - \Delta_{ii}^{(m)}\boldsymbol{a}_i\boldsymbol{W}_m\boldsymbol{a}_i'\right\} \\
&= \mathcal{Q}_\Delta(\hat{\boldsymbol{\theta}},\hat{\boldsymbol{\theta}}) + \frac{1}{2}\sum_{m=1}^{M}\beta_m\left\{\log(1 + \Delta_{ii}^{(m)}\boldsymbol{a}_i\boldsymbol{\Sigma}_m\boldsymbol{a}_i') - \Delta_{ii}^{(m)}\boldsymbol{a}_i\boldsymbol{W}_m\boldsymbol{a}_i'\right\} \qquad \text{(B18)}
\end{aligned}$$

Differentiating this w.r.t. $\Delta_{ii}^{(m)}$ yields

$$\frac{\partial\mathcal{Q}}{\partial\Delta_{ii}^{(m)}} = \frac{\boldsymbol{a}_i\boldsymbol{\Sigma}_m\boldsymbol{a}_i'}{1 + \Delta_{ii}^{(m)}\boldsymbol{a}_i\boldsymbol{\Sigma}_m\boldsymbol{a}_i'} - \boldsymbol{a}_i\boldsymbol{W}_m\boldsymbol{a}_i' = 0 \qquad \text{(B19)}$$

and equating to zero gives the ML solution of $\Delta_{ii}^{(m)}$ as

$$\Delta_{ii}^{(m)} = \frac{1}{\boldsymbol{a}_i\boldsymbol{W}_m\boldsymbol{a}_i'} - \frac{1}{\boldsymbol{a}_i\boldsymbol{\Sigma}_m\boldsymbol{a}_i'} \qquad \text{(B20)}$$

It is important to update the covariance matrix after the update of $\lambda_{ii}^{(m)}$ to ensure that the subsequent updates are correct. This can be done easily using the Matrix Inversion Lemma:

$$\boldsymbol{\Sigma}_m = \left(\hat{\boldsymbol{\Sigma}}_m^{-1} + \Delta_{ii}^{(m)}\boldsymbol{a}_i'\boldsymbol{a}_i\right)^{-1} = \hat{\boldsymbol{\Sigma}}_m - \frac{\Delta_{ii}^{(m)}\hat{\boldsymbol{\Sigma}}_m\boldsymbol{a}_i'\boldsymbol{a}_i\hat{\boldsymbol{\Sigma}}_m}{1 + \Delta_{ii}^{(m)}\boldsymbol{a}_i\hat{\boldsymbol{\Sigma}}_m\boldsymbol{a}_i'} \qquad \text{(B21)}$$

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:

$$\boldsymbol{a}_i = \hat{\boldsymbol{a}}_i + \boldsymbol{\Delta}_i \qquad \text{(B22)}$$

So, the resulting precision matrices can be written as

$$\boldsymbol{P}_m = \hat{\boldsymbol{P}}_m + \lambda_{ii}^{(m)}(\boldsymbol{a}_i'\boldsymbol{a}_i - \hat{\boldsymbol{a}}_i'\hat{\boldsymbol{a}}_i) \qquad \text{(B23)}$$

Using the Sherman-Morrison-Woodbury formula, the determinant of the new precision matrix is given by

$$\mathcal{G}(\boldsymbol{a}_i) = \frac{|\boldsymbol{P}_m|}{|\hat{\boldsymbol{P}}_m|} = (1 - \lambda_{ii}^{(m)}\hat{\boldsymbol{a}}_i\hat{\boldsymbol{\Sigma}}_m\hat{\boldsymbol{a}}_i')(1 + \lambda_{ii}^{(m)}\boldsymbol{a}_i\hat{\boldsymbol{\Sigma}}_m\boldsymbol{a}_i') + \left(\lambda_{ii}^{(m)}(\boldsymbol{a}_i\hat{\boldsymbol{\Sigma}}_m\hat{\boldsymbol{a}}_i')\right)^2 \qquad \text{(B24)}$$

Thus, the auxiliary function in equation (40) can be expressed in terms of $\boldsymbol{a}_i$ and $\hat{\boldsymbol{a}}_i$ as

$$\mathcal{Q}_A^{(i)}(\boldsymbol{\theta},\hat{\boldsymbol{\theta}}) = \mathcal{Q}_A^{(i)}(\hat{\boldsymbol{\theta}},\hat{\boldsymbol{\theta}}) + \frac{1}{2}\sum_{m=1}^{M}\beta_m\left\{\log\frac{|\boldsymbol{P}_m|}{|\hat{\boldsymbol{P}}_m|} - \lambda_{ii}^{(m)}(\boldsymbol{a}\boldsymbol{W}_m\boldsymbol{a}' - \hat{\boldsymbol{a}}\boldsymbol{W}_n\hat{\boldsymbol{a}}')\right\} \qquad \text{(B25)}$$

To perform gradient descent optimisation, the gradient vector and Hessian matrix of the auxiliary function are needed:

$$\begin{aligned}
\boldsymbol{f}_i &= \frac{\partial\mathcal{Q}}{\partial\boldsymbol{a}_i} = \frac{1}{2}\sum_{m=1}^{M}\beta_m\left\{\frac{\mathcal{G}'(\boldsymbol{a}_i)}{\mathcal{G}(\boldsymbol{a}_i)} - 2\lambda_{ii}^{(m)}\boldsymbol{a}\boldsymbol{W}_m\right\} \qquad \text{(B26)} \\
\boldsymbol{H}_i &= \frac{\partial^2\mathcal{Q}}{\partial\boldsymbol{a}_i'\partial\boldsymbol{a}_i} = \frac{1}{2}\sum_{m=1}^{M}\beta_m\left\{\frac{\mathcal{G}(\boldsymbol{a}_i)\mathcal{G}''(\boldsymbol{a}_i) - [\mathcal{G}'(\boldsymbol{a}_i)]'[\mathcal{G}'(\boldsymbol{a}_i)]}{\mathcal{G}(\boldsymbol{a}_i)^2} - 2\lambda_{ii}^{(m)}\boldsymbol{W}_m\right\} \qquad \text{(B27)}
\end{aligned}$$

where

$$\mathcal{G}'(\boldsymbol{a}_i) = 2\left\{\lambda_{ii}^{(m)}\boldsymbol{a}_i\hat{\boldsymbol{\Sigma}}_m + \lambda_{ii}^{(m)2}\left(\boldsymbol{a}_i\hat{\boldsymbol{\Sigma}}_m\hat{\boldsymbol{a}}_i'\hat{\boldsymbol{a}}_i\hat{\boldsymbol{\Sigma}}_m - \hat{\boldsymbol{a}}_i\hat{\boldsymbol{\Sigma}}_m\hat{\boldsymbol{a}}_i'\boldsymbol{a}_i\hat{\boldsymbol{\Sigma}}_m\right)\right\} \quad \text{(B28)}$$

$$\mathcal{G}''(\boldsymbol{a}_i) = 2\left\{\lambda_{ii}^{(m)}\hat{\boldsymbol{\Sigma}}_m + \lambda_{ii}^{(m)2}\left(\hat{\boldsymbol{\Sigma}}_m\hat{\boldsymbol{a}}_i'\hat{\boldsymbol{a}}_i\hat{\boldsymbol{\Sigma}}_m - \hat{\boldsymbol{a}}_i\hat{\boldsymbol{\Sigma}}_m\hat{\boldsymbol{a}}_i'\hat{\boldsymbol{\Sigma}}_m\right)\right\} \quad \text{(B29)}$$

Given the current estimate of the basis vector, $\hat{\boldsymbol{a}}_i$

$$\mathcal{G}(\hat{\boldsymbol{a}}_i) = 1 \quad \text{(B30)}$$

$$\mathcal{G}'(\hat{\boldsymbol{a}}_i) = 2\lambda_{ii}^{(m)}\hat{\boldsymbol{a}}_i\hat{\boldsymbol{\Sigma}}_m \quad \text{(B31)}$$

$$\mathcal{G}''(\hat{\boldsymbol{a}}_i) = 2\lambda_{ii}^{(m)}\hat{\boldsymbol{\Sigma}}_m\left(1 - \hat{\boldsymbol{a}}_i\hat{\boldsymbol{\Sigma}}_m\hat{\boldsymbol{a}}_i' + \lambda_{ii}^{(m)}\hat{\boldsymbol{a}}_i'\hat{\boldsymbol{a}}_i\hat{\boldsymbol{\Sigma}}_m\right) \quad \text{(B32)}$$

Hence, the gradient vector and Hessian matrix at the current estimate is given by

$$\bar{\boldsymbol{f}}_i = \boldsymbol{f}_i\Big|_{\boldsymbol{a}_i=\hat{\boldsymbol{a}}_i} = \sum_{m=1}^{M}\beta_m\lambda_{ii}^{(m)}\hat{\boldsymbol{a}}_i\left(\hat{\boldsymbol{\Sigma}}_m - \boldsymbol{W}_m\right) \quad \text{(B33)}$$

$$\bar{\boldsymbol{H}}_i = \boldsymbol{H}_i\Big|_{\boldsymbol{a}_i=\hat{\boldsymbol{a}}_i} = \sum_{m=1}^{M}\beta_m\lambda_{ii}^{(m)}\left\{\hat{\boldsymbol{\Sigma}}_m\left(1 - \hat{\boldsymbol{a}}_i\hat{\boldsymbol{\Sigma}}_m\hat{\boldsymbol{a}}_i'\right) - \boldsymbol{W}_m - \lambda_{ii}^{(m)}\hat{\boldsymbol{\Sigma}}_m\hat{\boldsymbol{a}}_i'\hat{\boldsymbol{a}}_i\hat{\boldsymbol{\Sigma}}_m\right\} \quad \text{(B34)}$$

and the new estimate of the basis vector is given by

$$\boldsymbol{a}_i = \hat{\boldsymbol{a}}_i + \eta\bar{\boldsymbol{f}}_i\bar{\boldsymbol{H}}_i^{-1} \quad \text{(B35)}$$

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.

B4 SPAM Parameters Update

SPAM model[2] 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

$$\boldsymbol{P}_m = \sum_{i=1}^{n}\lambda_{ii}^{(m)}\boldsymbol{S}_i \quad \text{(B36)}$$

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 (40) leads to the following modified auxiliary function

$$\mathcal{Q}(\boldsymbol{\theta},\hat{\boldsymbol{\theta}}) = K + \frac{1}{2}\sum_{m=1}^{M}\beta_m\left\{\log\left(|\boldsymbol{P}_m|\right) - \sum_{i=1}^{n}\lambda_{ii}^{(m)}\,\mathrm{Tr}(\boldsymbol{W}_m\boldsymbol{S}_i)\right\} \quad \text{(B37)}$$

There is no closed form solution to optimise equation (B37) w.r.t. the model parameters. However, the basis coefficients, $\lambda_{ii}^{(m)}$ can be updated using an efficient conjugate gradient algorithm[2]. The following describes the coefficient update using the Polak-Ribeire conjugate-gradient method[19]. First, let

$$\boldsymbol{P}_m = \hat{\boldsymbol{P}}_m + \Delta_i^{(m)}\boldsymbol{S}_i \quad \text{(B38)}$$

where $\hat{\boldsymbol{P}}_m$ and $\boldsymbol{P}_m$ denote the current and new estimate of the precision matrix. Thus, the auxiliary function w.r.t. the mth component and ith basis matrix can be expressed as

$$\mathcal{Q}^{(mi)}(\boldsymbol{\theta},\hat{\boldsymbol{\theta}}) = K + \frac{\beta_m}{2}\left\{\log\left(\left|\boldsymbol{P}_m + \Delta_i^{(m)}\boldsymbol{S}_i\right|\right) - \mathrm{Tr}(\boldsymbol{W}_m\boldsymbol{P}_m) - \Delta_i^{(m)}\,\mathrm{Tr}(\boldsymbol{W}_m\boldsymbol{S}_i)\right\} \quad \text{(B39)}$$

Differentiate w.r.t. $\Delta_i^{(m)}$ about the point $\Delta_i^{(m)} = 0$ gives

$$f_i^{(m)} = \frac{\partial \mathcal{Q}^{(i)}}{\partial \Delta_i^{(m)}}\Big|_{\Delta_i^{(m)}=0} = \frac{\beta_m}{2}\Big\{\mathrm{Tr}(\boldsymbol{P}_m^{-1}\boldsymbol{S}_i) - \mathrm{Tr}(\boldsymbol{W}_m\boldsymbol{S}_i)\Big\} \tag{B40}$$

$$= \frac{\beta_m}{2}\Big\{\mathrm{Tr}\left((\boldsymbol{P}_m^{-1} - \boldsymbol{W}_m)\boldsymbol{S}_i\right)\Big\} \tag{B41}$$

This gives gradient of the auxiliary function w.r.t., $\lambda_{ii}^{(m)}$. Given the gradient vector, $\boldsymbol{f}_k^{(m)}$, at the kth iteration, the conjugate gradient direction vector, $\boldsymbol{d}_k^{(m)}$, is then given by

$$\boldsymbol{d}_k^{(m)} = \boldsymbol{f}_k + \eta_k \boldsymbol{d}_{k-1}^{(m)} \tag{B42}$$

where

$$\eta_k = \frac{\boldsymbol{f}_k'(\boldsymbol{f}_k - \boldsymbol{f}_{k-1})}{\boldsymbol{f}_{k-1}'\boldsymbol{f}_{k-1}} \tag{B43}$$

and $\boldsymbol{d}_0^{(m)} = \boldsymbol{0}$. Once the direction vector is found, the optimisation simplifies to a one dimensional line search problem where

$$\boldsymbol{P}_m = \hat{\boldsymbol{P}}_m + \Delta^{(m)}\boldsymbol{R}_m \tag{B44}$$

$$\boldsymbol{R}_m = \sum_{i=1}^{n} d^{(m)}(i)\boldsymbol{S}_i \tag{B45}$$

and $d(i)$ is the i^{th} element of the direction vector given by equation (B42). Thus, the auxiliary function can be written as

$$\mathcal{Q}^{(m)} - \hat{\mathcal{Q}}^{(m)} = \frac{\beta_m}{2}\left\{\log\left(\left|\frac{\hat{\boldsymbol{P}}_m + \Delta^{(m)}\boldsymbol{R}_m}{\hat{\boldsymbol{P}}_m}\right|\right) - \Delta^{(m)}\,\mathrm{Tr}(\boldsymbol{W}_m\boldsymbol{R}_m)\right\} \tag{B46}$$

$$= \frac{\beta_m}{2}\left\{\log\left(\left|\boldsymbol{I} + \Delta^{(m)}\hat{\boldsymbol{P}}_m^{-\frac{1}{2}}\boldsymbol{R}_m\hat{\boldsymbol{P}}_m^{-\frac{1}{2}}\right|\right) - \Delta^{(m)}\,\mathrm{Tr}(\boldsymbol{W}_m\boldsymbol{R}_m)\right\} \tag{B47}$$

$$= \frac{\beta_m}{2}\left\{\sum_{j=1}^{d}\log\left(1 + \Delta^{(m)}w_j^{(m)}\right) - \Delta^{(m)}\,\mathrm{Tr}(\boldsymbol{W}_m\boldsymbol{R}_m)\right\} \tag{B48}$$

where $w_j^{(m)}$ is the j^{th} eigenvalue of $\hat{\boldsymbol{P}}_m^{-\frac{1}{2}}\boldsymbol{R}_m\hat{\boldsymbol{P}}_m^{-\frac{1}{2}}$. The line search can be further constrained to ensure that $1 + \Delta^{(m)}w_j^{(m)} > 0$. Hence, the line search is confined to be within the range $-1/w_{max}^{(m)} < \Delta^{(m)} < -1/w_{min}^{(m)}$, where $w_{max}^{(m)}$ and $w_{min}^{(m)}$ are the largest positive and negative eigenvalues. If $w_{max}^{(m)}$ ($w_{min}^{(m)}$) does not exist, the range of $\Delta^{(m)}$ is then unbounded from below (above).