Extended MLLT for Gaussian Mixture Models

Peder A. Olsen and Ramesh A. Gopinath
IBM, T. J. Watson Research Center
134 and Taconic Parkway
Yorktown Heights, NY 10598
(914) 945 {3772,2794}, (fax) (914) 945 4490
{*pederao,rameshg*}*@us.ibm.com*

EDICS Category SA 1.6.4

Prior to publication, please maintain the enclosed paper in confidence and use it only for purposes of evaluating the merit of the proposed paper, and other activities reasonably related to the review process, and please do not make it available, in whole or in part, to the public. The authors thanks IEEE Transactions in Speech and Audio Processing for their courtesy and professionalism in this matter.

Abstract

In MLLT the inverse covariance matrix (precision matrix) of Gaussian mixture component j, $j = 1, \ldots, m$ is modeled by $A^T \Lambda_j A$, where $\Lambda_j \in \mathbf{R}_+^{d \times d}$ are diagonal matrices and $A \in \mathbf{R}^{d \times d}$ is a global data transformation matrix. This framework is extended to consider $\Lambda_j \in \mathbf{R}^{D \times D}$ and $A \in \mathbf{R}^{D \times d}$ for $D \geq d$. The model uses a naturally approximating basis expansion for positive definite matrices and yields greater control of the number of parameters used in modeling covariances than methods considered by previous authors. Moreover, the method yields a practical solution to the design of the matrix considered in the recently proposed MLT model. The extended MLLT (EMLLT) model can be viewed as a generalization of several other covariance modeling techniques such as Factor Analysis [26]. Experimental results in a speech recognition task shows a relative gain of 35% over a baseline diagonal covariance model and 28% over the MLLT model corresponding to the baseline when $D = 14d$. The EMLLT model yields a surprising 16% relative gain over a full covariance model. Also EMLLT is invariant to linear transformations of the data.

1 Introduction

The Gaussian mixture model can be written

$$f(x|\Theta) = \sum_{j=1}^{m} \pi_j \mathcal{N}(x; \mu_j, \Sigma_j) \tag{1}$$

where $x = (x_1, \ldots, x_d)^T \in \mathbf{R}^{d\times 1}$, $\Theta = (\bar\pi, \bar\mu, \bar\Sigma)$, $\bar\pi = (\pi_1, \ldots, \pi_m) \in \mathbf{R_+}^{1\times m}$, $1 = \sum_{j=1}^m \pi_j$ $\bar\mu = (\mu_1, \ldots, \mu_m)^T \in \mathbf{R}^{m\times d}$ and $\bar\Sigma = (\Sigma_1, \ldots, \Sigma_m)^T \in \mathbf{R}^{m\times d\times d}$ are symmetric positive definite matrices. In speech recognition d is typically 39 and the total number of Gaussian mixture components in the system is such that the memory requirements and computational cost renders the full covariance model impractical. The simplest and most widely used model considers covariance constraints of the form

$$\Sigma_j^{-1} = \Lambda_j = \mathrm{diag}(\lambda_1^j, \lambda_2^j, \ldots, \lambda_d^j), \quad j = 1, 2, \ldots, m,$$

where the parameters $\lambda_i^j > 0$ are the precisions for mixture component j and dimension i. Another more sophisticated model considers precision matrices of the form

$$\Sigma_j^{-1} = A^T \Lambda_j A, \quad j = 1, 2, \ldots, m,$$

where $\Lambda_j \in \mathbf{R}_+^{d\times d}, A \in \mathbf{R}^{d\times d}$. This model is known as the Maximum Likelihood Linear Transform or simply MLLT [13]. MLLT is a special case of Heteroscedastic Discriminant Analysis introduced in [11] and is the similar to "global" semi-tied covariances introduced in [6]. This model is extended by considering $\Lambda_j = \mathrm{diag}(\lambda_1^j, \lambda_2^j, \ldots, \lambda_D^j) \in \mathbf{R}^{D\times D}$, $A \in \mathbf{R}^{D\times d}$. The l'th row of A is refered to as a_l^T, i.e.

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1d} \\ a_{21} & \cdots & a_{2d} \\ \vdots & & \vdots \\ a_{d1} & \cdots & a_{dd} \\ \vdots & & \vdots \\ a_{D1} & \cdots & a_{Dd} \end{pmatrix} = \begin{pmatrix} a_1^T \\ a_2^T \\ \vdots \\ a_d^T \\ \vdots \\ a_D^T \end{pmatrix}. \tag{2}$$

This allows the equivalent representation

$$\Sigma_j^{-1} = \sum_{l=1}^{D} \lambda_l^j a_l a_l^T \ , \tag{3}$$

where $a_l a_l^T \in \mathbf{R}^{d\times d}$. This model is the Extended MLLT (EMLLT) model. The expansion (3) is a natural basis expansion for positive definite matrices. It is also a basis for the collection of all symmetric matrices. Each of the matrices $a_l a_l^T$, $l = 1, \ldots, D$ are positive semidefinite and so must Σ_j^{-1} be if $\lambda_l^j \geq 0$ for $l = 1, \ldots, D$. Moreover, if $\lambda_l^j > 0$ for $l = 1, \ldots, D$ and $\{a_l\}_{l=1}^D$ spans $\mathbf{R}^d$ then Σ_j^{-1} is positive definite in the strict sense. Should some λ's be negative it may of course still be true that Σ_j^{-1} is positive definite, a fact that will be exploited.

EMLLT is a constrained maximum likelihood technique. When constraints are not invariant to linear transforms of data, e. g. diagonal covariance models, Factor Analysis, etc., it has been observed that the likelihood can be improved by a maximum likelihood linear transform, [14]. EMLLT is invariant to linear transforms of the data since any linear transform of the data can be absorbed into A. Other models that are invariant to linear transforms (ILT) are MLLT and FACILT, [15].

Under these constraints the formula for the individual Gaussian components becomes

$$\mathcal{N}(x;\mu_j,\Sigma_j) = \frac{|\det(A^T\Lambda_j A)|^{1/2}}{(2\pi)^{d/2}} \exp\left\{-\frac{1}{2}(x-\mu_j)^T A^T \Lambda_j A(x-\mu_j)\right\} . \qquad (4)$$

It will be convenient to use matrix notation in what follows. The dimensions of the matrices are as noted:

$$\begin{aligned}
\bar{\pi} &= (\pi_1,\ldots,\pi_m) \in \mathbf{R}_+^{1\times m} \\
\bar{\mu} &= (\mu_1,\ldots,\mu_m)^T \in \mathbf{R}^{m\times d} \\
\bar{\Lambda} &= \begin{pmatrix} \lambda_1^1 & \lambda_2^1 & \ldots & \lambda_D^1 \\ \lambda_1^2 & \lambda_2^2 & \ldots & \lambda_D^2 \\ \vdots & \vdots & & \vdots \\ \lambda_1^m & \lambda_2^m & \ldots & \lambda_D^m \end{pmatrix} \in \mathbf{R}^{m\times D} \\
\bar{x} &= (x_1,x_2,\ldots,x_N) \in \mathbf{R}^{d\times N}
\end{aligned}$$

and $A \in \mathbf{R}^{D\times d}$.

This makes it possible to view much of the computation in terms of matrices and matrix multiplication. There are algorithms faster than the simplest method for matrix multiplication, e.g. Strassen's algorithm, [2],[12, p. 31–34]. The benefits of using Strassen's algorithm are largest when d, D, m and N are relatively large.

1.1 EMLLT for Classification

The EMLLT model can be used for classification by modeling each class conditional density using EMLLT. Furthermore, it is straightforward to allow sharing of rows of A between classes. By extension the EMLLT model can be used to model Gaussian mixtures and model continuous output distributions in Hidden Markov Models. All of the experiments reported in this paper are in the context of a speech recognition application where each of the HMM states is modeled by an EMLLT model with the A matrix being shared globally by all the states.

1.2 Relationship to Similar Models

There have been several papers in the recent past on modeling covariance matrices in Gaussian Mixture Models. Since a (single) Gaussian density is in the exponential family with the precision matrix as the canonical parameter, it is natural to try to model the inverse covariances in a tied fashion. This is precisely what the EMLLT model does. Recently an alternative/generalization of the MLLT model called the MLT (for Multiple Linear Transformations) model was proposed in [10] and [7]. The basic idea there is very similar. The precise relationship can be seen as follows:

MLLT: $D = d$.

MLT: $\lambda_k^j = 0$, for $j \notin I_j$, where I_j is an index set, $I_j \subset \{1, \ldots, D\}$ and $|I_j| = d$.

Factor Analysis: $a_k = \delta_k$, $k = 1, 2, \ldots, d$ (δ_k is the vector with all zeros except at the kth position) and λ_k^j is constrained to be negative for $D \geq k > d$. The number of "factors" is $D - d$. This relationship is not as obvious since Factor Analysis is typically viewed as a covariance modeling technique where each covariance Σ_j is of the form $\Psi_j + \Gamma_j \Gamma_j^T$, Ψ_j is a diagonal matrix and Γ_j is a $d \times (D - d)$ matrix. Equivalently, by the Sherman–Morrison–Woodbury formula, [12, p. 50] the precision matrix is of the form $\Psi_j^{-1} - \Delta_j \Delta_j^T$, where $\Delta_j = \Psi_j^{-1} \Gamma_j (I + \Gamma_j^T \Psi_j^{-1} \Gamma_j)^{-1/2}$. In fact EMLLT generalizes FACILT, the Invariant to Linear Transform version of factor analysis, [15].

Full-Covariance: $D = d(d+1)/2$ and $a_k = e_k$, $k = 1, \ldots, d$, and a_k for $D \geq k > d$ is of the form $e_m + e_n$, $m, n \in \{1, 2, \ldots, d\}$ with $m < n$.

2 The Loglikelihood Computation

Consider the evaluation of the likelihood. Precomputation of quantities specific to the Gaussian components or specific to the individual data vectors speeds up the overall computation. Writing $\ell_j(x)$ or $\ell_j(x; \pi_j, \mu_j, \Lambda_j, A)$ for $\log\left(\pi_j \mathcal{N}(x; \mu_j, (A^T \Lambda_j A)^{-1})\right)$ the following computes the log likelihood for a specific Gaussian component

$$
\begin{aligned}
\ell_j(x) &= \log\left(\pi_j \mathcal{N}(x; \mu_j, (A^T \Lambda_j A)^{-1})\right) \\
&= \log((2\pi)^{-\frac{d}{2}} \pi_j) + \frac{1}{2} \log\left(\det(A^T \Lambda_j A)\right) - \frac{1}{2}(x - \mu_j) A^T \Lambda_j A (x - \mu_j) \\
&= \left[\log(\pi_j) + \frac{1}{2} \log\left(\det(A^T \Lambda_j A)\right) - \frac{d}{2} \log(2\pi) - \frac{1}{2} \mu_j^T A^T \Lambda_j A \mu_j\right] \\
&\quad + \frac{1}{2}(\mu_j A^T \Lambda_j A) x + \frac{1}{2} x^T (A^T \Lambda_j A \mu_j) - \frac{1}{2}(Ax)^T \Lambda_j (Ax) \\
&= b_j + (\mu_j A^T \Lambda_j A) x - \frac{1}{2}(Ax)^T \Lambda_j (Ax) \\
&= b_j + \tau_j^T x - \frac{1}{2} y^T \Lambda_j e_D \\
&= \begin{pmatrix} b_j & \tau_j^T & -\frac{1}{2} e_D^T \Lambda_j \end{pmatrix} \begin{pmatrix} 1 \\ x \\ y \end{pmatrix},
\end{aligned}
$$

where

$$
\begin{aligned}
b_j &= \log(\pi_j) + \frac{1}{2} \log\left(\det(A^T \Lambda_j A)\right) - \frac{d}{2} \log(2\pi) - \frac{1}{2} \mu_j^T A^T \Lambda_j A \mu_j && (5) \\
\tau_j &= A^T \Lambda_j A \mu_j \in \mathbf{R}^{d \times 1} && (6) \\
y &= (Ax) \circ (Ax) \in \mathbf{R}_+^{D \times 1} && (7)
\end{aligned}
$$

and $e_D = (1, 1, \ldots, 1) \in \mathbf{R}^{D\times 1}$. Here $\circ$ denotes element wise multiplication, i.e. the Schur product. If the quantities in (5), (6) and (7) have been precomputed, the loglikelihood computation is reduced to a total of $d + D$ additions and multiplications per Gaussian component per vector. This compares to $2d$ additions and multiplications in the diagonal covariance case. If several vectors are supplied at the same time, say $\bar{x} = (x_1, x_2, \ldots, x_N)$, then write the loglikelihoods

$$\ell = \begin{pmatrix} \ell_1(x_1) & \cdots & \ell_1(x_N) \\ \vdots & & \vdots \\ \ell_m(x_1) & \cdots & \ell_m(x_N) \end{pmatrix} \in \mathbf{R}^{m\times N}$$

as $\ell = be_N + \bar{\tau}\bar{x} - \bar{y}\bar{\Lambda}$, where $\bar{\tau} = (\tau_1, \tau_2, \ldots, \tau_m)^T \in \mathbf{R}^{m\times d}$, $\bar{x} \in \mathbf{R}^{d\times N}$, $\bar{y} = (A\bar{x}) \circ (A\bar{x})$ and $b = (b_1, \ldots, b_m)^T \in \mathbf{R}^{m\times 1}$. The explicit computation of the elements of be_N is of course unnecessary and so using Strassen's algorithm, [12, 2], for matrix multiplication allows computing P in less than $\mathcal{O}(Nm(d+D))$ computations. The likelihoods $f(x_i|\Theta)$, $i = 1, 2, \ldots, N$ can now be computed as $f(x_i|\Theta) = \sum_{j=1}^{m} e^{\ell_j(x_i)}$, which requires evaluating an additional $N \times m$ exponentiations.

2.1 Implementation Details for Lazy Programmers

There is an interesting way to use any implementation of a diagonal covariance to evaluate EMLLT models. For the likelihood computation the following gives a recipe requiring minimal code changes. Many speech recognition programs implement either or both of LDA and MLLT. Through any of these methods there should be a simple facility to replace an ordinary acoustic vector x by Ax. If in addition the variances in the normal diagonal covariance Gaussian mixture component j are replaced with Λ_j^{-1}, the means replaced with $A\mu_j$ and the priors replaced with $\pi_j \det(A^T\Lambda_j A)^{1/2} \det(\Lambda_j)^{1/2}$, then the standard Gaussian computation will result in the correct likelihood values corresponding to the extended MLLT model. However, this method of implementation requires more computation and some care should be taken to assure that $\pi_j \det(A^T\Lambda_j A)^{1/2} \det(\Lambda_j)^{1/2}$ does not underflow. Thus $\log(\pi_j)$ was computed and stored in place of π_j. This change only changing 3 lines of code in the IBM research speech recognizer. The transformed parameters can be precomputed in `Matlab` and so the code changes should in general be minimal. The same technique however does not work for training the parameters, as the transformed data vectors are linearly dependent when $D > d$. Also, note that this esoteric trick will not work if any of the λ_k^j values are allowed to be negative or zero.

3 Parameter Estimation

Following the principle of maximum likelihood the parameters Θ are chosen to maximize the loglikelihood $L(\Theta|\bar{x}) = \frac{1}{N}\sum_{i=1}^{N} \log f(x_i|\Theta)$ of the data $\bar{x} = (x_1, x_2, \ldots, x_N) \in \mathbf{R}^{d\times N}$ with respect to Θ and the constraints $\sum_{j=1}^{m} \pi_j = 1$,

$\pi_j \geq 0$. In addition there is the requirement that $A^T \Lambda_j A$ must be positive definite. The optimization strategy will be according to the EM algorithm, [21, 3]. An outline of the derivation for the EM algorithm is presented here. If x is generated by a random variable X with probability density function (1) with parameters Θ^* it will maximize the function $L(\Theta) = \mathrm{E}_{\Theta^*}[\log f(X|\Theta)]$ at the value $\Theta = \Theta^*$. $L(\Theta) = \lim_{N\to\infty} L(\Theta|\bar{x})$ indicates that Θ^* may be close to the global maximum of $L(\Theta|\bar{x})$ for some fixed N. However this function is unbounded as may be seen by letting $\mu_1 = x_1$ and $\Lambda_1 \to \infty$. Therefore it is necessary to seek a local maximum for $L(\Theta|\bar{x})$ for which $\lambda_k^j < 1/\epsilon$ for all $j = 1, \ldots, m$, $k = 1, \ldots, D$ and sufficiently small ϵ. To achieve this goal start at $\hat{\Theta}$ and find a Θ satisfying $L(\Theta|\bar{x}) \geq L(\hat{\Theta}|\bar{x})$. By use of Jensen's inequality, [17], $\log(\sum_{j=1}^m \alpha_j \beta_j) \geq \sum_{j=1}^m \alpha_j \log \beta_j$ for $\alpha_j \geq 0$, $\beta_j > 0$, $\sum_{j=1}^m \alpha_j = 1$, this can be accomplished as follows

$$
\begin{aligned}
L(\Theta) - L(\hat{\Theta}) &= \frac{1}{N}\sum_{i=1}^{N} \log\left(\frac{f(x_i|\Theta)}{f(x_i|\hat{\Theta})}\right) \\
&= \frac{1}{N}\sum_{i=1}^{N} \log\left(\sum_{j=1}^{m} \frac{\exp\left(\ell_j(x_i|\pi_j,\mu_j,\Lambda_j,A)\right)}{\sum_{k=1}^m \exp\left(\ell_k(x_i|\hat{\pi}_k,\hat{\mu}_k,\hat{\Lambda}_k,\hat{A})\right)} \times \right. \\
&\quad \left. \frac{\exp\left(\ell_j(x_i|\hat{\pi}_j,\hat{\mu}_j,\hat{\Lambda}_j,\hat{A})\right)}{\exp\left(\ell_j(x_i|\hat{\pi}_j,\hat{\mu}_j,\hat{\Lambda}_j,\hat{A})\right)}\right) \\
&\geq \frac{1}{N}\sum_{i=1}^{N} \gamma_{ij}\left(\ell_j(x_i|\pi_j,\mu_j,\Lambda_j,A) - \ell_j(x_i|\hat{\pi}_j,\hat{\mu}_j,\hat{\Lambda}_j,\hat{A})\right) \\
&= Q(\Theta|\hat{\Theta}) - Q(\hat{\Theta}|\hat{\Theta}) ,
\end{aligned}
$$

where γ_{ij} are the a posteriori probabilities of Gaussian component j, $P(j|x_i)$, given by

$$
\begin{aligned}
\gamma_{ij} &= \frac{\exp\left(\ell_j(x_i|\hat{\pi}_j,\hat{\mu}_j,\hat{\Lambda}_j,\hat{A})\right)}{\sum_{k=1}^m \exp\left(\ell_k(x_i|\hat{\pi}_k,\hat{\mu}_k,\hat{\Lambda}_k,\hat{A})\right)} \\
&= \frac{\hat{\pi}_j \mathcal{N}(x;\hat{\mu}_j,(\hat{A}^T\hat{\Lambda}_j\hat{A})^{-1})}{\sum_{k=1}^m \hat{\pi}_k \mathcal{N}(x;\hat{\mu}_k,(\hat{A}^T\hat{\Lambda}_k\hat{A})^{-1})}
\end{aligned}
$$

and

$$
\begin{aligned}
Q(\Theta|\hat{\Theta}) &= \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{m} \gamma_{ij}\ell_j(x_i|\pi_j,\mu_j,\Lambda_j,A) \\
&= \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{m} \gamma_{ij}\left\{\log\pi_j - \frac{d}{2}\log(2\pi) + \frac{1}{2}\log\det(A^T\Lambda_j A) - \right. \\
&\quad \left. \frac{1}{2}(x_i-\mu_j)^T A^T \Lambda_j A (x_i - \mu_j)\right\} .
\end{aligned}
$$

Maximizing $Q(\Theta|\hat{\Theta})$ with respect to Θ ensures $L(\Theta) \geq L(\hat{\Theta})$. The maximization with respect to $\bar{\pi}$ and the constraint $\sum_{j=1}^{m} \pi_j = 1$ is straightforward and yields the values

$$\pi_j = \frac{1}{N} \sum_{j=1}^{N} \gamma_{ij} \ .$$

Likewise, assuming $A^T \Lambda_j A$ invertible $Q(\Theta|\hat{\Theta})$ can be maximized with respect to $\bar{\mu}$ resulting in

$$\mu_j = \frac{\sum_{i=1}^{N} \gamma_{ij} x_i}{\sum_{i=1}^{N} \gamma_{ij}} \ .$$

These update rules are the same as for the usual Gaussian mixture case. The update formula for A and $\bar{\Lambda}$ are not as simple and the next few sections consider these. The strategy will be to keep A fixed then optimize with respect to $\bar{\pi}, \bar{\mu}$ and $\bar{\Lambda}$ then keep $\bar{\Lambda}$ fixed and optimize with respect to $\bar{\pi}, \bar{\mu}$ and A. This procedure is repeated until convergence is reached or overtraining start, whichever comes first.

4 Update formulas for $\bar{\Lambda}$

To establish an update rule for $\bar{\Lambda}$ first consider A to be fixed, $A = \hat{A}$, as the joint optimization of A and $\bar{\Lambda}$ is quite complicated. Moreover, define $Q_{\bar{\Lambda}}(\Theta|\hat{\Theta})$ such that Q and $Q_{\bar{\Lambda}}$ differs only in a function that is constant with respect to $\bar{\Lambda}$. Using the formula trace$(AB) =$ trace(BA) for matrices A and B the following formula for $Q_{\bar{\Lambda}}$ may be simplified as seen below

$$\begin{aligned}
Q_{\bar{\Lambda}} &\stackrel{def}{=} \frac{1}{2N} \sum_{i=1}^{N} \sum_{j=1}^{m} \gamma_{ij} \left\{ \log\det(A^T \Lambda_j A) - (x_i - \mu_j)^T A^T \Lambda_j A (x_i - \mu_j) \right\} \\
&= \frac{1}{2} \sum_{j=1}^{m} \left(\frac{1}{N} \sum_{i=1}^{N} \gamma_{ij} \right) \log\det(A^T \Lambda_j A) \\
&\quad - \frac{1}{2N} \sum_{i=1}^{N} \sum_{j=1}^{m} \gamma_{ij} \text{trace} \left[(x_i - \mu_j)^T A^T \Lambda_j A (x_i - \mu_j) \right] \\
&= \frac{1}{2} \sum_{j=1}^{m} \pi_j \log\det(A^T \Lambda_j A) \\
&\quad - \frac{1}{2N} \sum_{i=1}^{N} \sum_{j=1}^{m} \gamma_{ij} \text{trace} \left[\Lambda_j A (x_i - \mu_j)(x_i - \mu_j)^T A^T \right] \\
&= \frac{1}{2} \sum_{j=1}^{m} \pi_j \log\det(A^T \Lambda_j A) - \frac{1}{2} \sum_{j=1}^{m} \text{trace} \left[\Lambda_j L_j \right] \ ,
\end{aligned}$$

where

$$L_j = \frac{1}{N}\sum_{i=1}^{N}\gamma_{ij}(Ax_i)(Ax_i)^T - \pi_j(A\mu_j)(A\mu_j)^T ,$$

is the sample covariance of $A\bar{x}$. Since Λ_j is a diagonal matrix only the diagonal elements of L_j are needed to evaluate trace($\Lambda_j L_j$)S. In matrix notation

$$Q_{\bar{\Lambda}}(\Theta|\hat{\Theta}) = \frac{1}{2}\sum_{j=1}^{m}\pi_j \log\det(A^T\Lambda_j A) - \frac{1}{2}e_m\bar{\Lambda}\bar{L} ,$$

where $\bar{L} \in \mathbf{R}^{m\times D}$ is defined by

$$\bar{L}_{jl} = L_{jll} - \pi_j(A\mu_j)_l^2$$

for $j = 1,2,\ldots,m$, $l = 1,2,\ldots,D$ and $(A\mu_j)_l$ denotes the l'th element of the vector $(A\mu_j)$. $\bar{L}$ is computed efficiently using a matrix formulation. The computation described in section 2 allows rapid computation of

$$\Gamma = \begin{pmatrix} \gamma_{11} & \cdots & \gamma_{1N} \\ \vdots & & \vdots \\ \gamma_{m1} & \cdots & \gamma_{mN} \end{pmatrix} \in \mathbf{R}_+^{m\times N} .$$

If in addition $\bar{y} = (A\bar{x}) \circ (A\bar{x}) \in \mathbf{R}_+^{D\times N}$ is introduced $\bar{L}$ is simply

$$\bar{L} = \Gamma\bar{y}^T - (A\bar{\mu})^T \circ (A\bar{\mu})^T \circ (e_D\bar{\pi}) .$$

In other words $\bar{L}$ can be computed using the Schur product and matrix multiplication from Γ and Θ. This allows rapid computation using fast matrix multiplication. To maximize $Q_{\bar{\Lambda}}$ notice that $Q_{\bar{\Lambda}} = \frac{1}{2}\sum_{j=1}^{m} q_j(\Lambda_j|\hat{\Theta})$ where

$$q_j(\Lambda_j|\hat{\Theta}) = \pi_j \log\det(A^T\Lambda_j A) - \sum_{l=1}^{D}\lambda_l^j \bar{L}_{jl} . \tag{8}$$

This means that $q_j(\Lambda_j|\hat{\Theta})$ can be maximized with respect to Λ_j for one value of j at a time.

4.1 A Multiplicative Update of $\bar{\Lambda}$

Each function q_j is a logarithm of a polynomial plus a linear term. This fact will be taken advantage of to maximize the function.

Consider the maximization of a polynomial with positive coefficients damped by an exponential on the region $\mathbf{R}_+^D$. $\exp(q_j(\Lambda_j|\hat{\Theta}))$ is such a function as shown later. The notation $x^\alpha = \prod_{l=1}^{D} x_l^{\alpha_l}$ will be used in what follows.

Theorem 1 *Let $p(x) = \sum_{\alpha\in\mathbf{Z}^D} c_\alpha x^\alpha$, $c_\alpha \geq 0$ be a polynomial (or more generally a once continuously differentially convergent power series) in x and define $f(x) = p(x)e^{-\beta^T x}$ with $\beta = (\beta_1,\beta_2,\ldots,\beta_D)^T \in \mathbf{R}_+^{D\times 1}$, with $\beta_l > 0$ for*

$l = 1, 2, \ldots, D$. *Further assume that*
$x^{(k)} = (x_1^{(k)}, x_2^{(k)}, \ldots, x_D^{(k)})^T$, $x_l^{(0)} \geq 0$ *and*

$$x_l^{(k+1)} = \frac{x_l^{(k)} \frac{\partial}{\partial x_l^{(k)}} p(x^{(k)})}{\beta_l p(x^{(k)})} \tag{9}$$

for $l = 1, 2, \ldots, D$, $k = 0, 1, \ldots$. *Then* $x_l^{(k+1)} \geq 0$ *and* $f(x^{(k+1)}) \geq f(x^{(k)})$. $\square$

As a toy example application of Theorem 1 consider the maximization of the univariate function $f(t) = (t^2 + t)e^{-t}$ for $t \geq 0$. Letting $t^{(0)} = 1$ Theorem 1 gives the update formula $t^{(k+1)} = 2 - (1 + t^{(k)})^{-1}$ giving $x^{(1)} = \frac{3}{2} = 1.5$, $x^{(2)} = \frac{8}{5} = 1.6$, $x^{(3)} = \frac{21}{13} = 1.6154$, $x^{(4)} = \frac{55}{34} = 1.6176$ and $x^{(k)} = \frac{F_{2k+2}}{F_{2k+1}}$, where F_k is the kth Fibonacci number. It follows that $x^{(k)}$ converges to the golden ratio which is the global maximum of f.

Proof of Theorem 1: Starting at the value $x^{(k)} = \hat{x}$ seek a value x satisfying $f(x) \geq f(\hat{x})$ or equivalently $\log f(x) - \log f(\hat{x}) \geq 0$. The following calculation is in the spirit of the EM algorithm and uses Jensen's inequality:

$$\begin{aligned}
\log f(x) - \log f(\hat{x}) &= \log\left(\frac{p(x)e^{-\beta^T x}}{p(\hat{x})e^{-\beta^T \hat{x}}}\right) \\
&= \log\left(\sum_\alpha \frac{c_\alpha x^\alpha}{\sum_\sigma c_\sigma \hat{x}^\sigma} \times \frac{\hat{x}^\alpha}{\hat{x}^\alpha}\right) + \beta^T(\hat{x} - x) \\
&\geq \sum_\alpha \gamma_\alpha \log\left(\frac{x^\alpha}{\hat{x}^\alpha}\right) + \beta^T(\hat{x} - x) \\
&= q(x;\hat{x}) - q(\hat{x};\hat{x}) \ ,
\end{aligned}$$

where $\gamma_\alpha = c_\alpha \hat{x}^\alpha \left(\sum_\sigma c_\sigma \hat{x}^\sigma\right)^{-1}$ and

$$\begin{aligned}
q(x;\hat{x}) &\stackrel{def}{=} \sum_\alpha \gamma_\alpha \log x^\alpha - \beta^T x \\
&= \sum_\alpha \gamma_\alpha \sum_{l=1}^{D} \alpha_l \log x_l - \sum_{l=1}^{D} \beta_l x_l \ .
\end{aligned}$$

Solve $\partial q(x;\hat{x})/\partial x_k = 0$ to maximize $q(x;\hat{x})$ with respect to x. This yields

$$\begin{aligned}
x_k &= \frac{1}{\beta_k} \sum_\alpha \gamma_\alpha \alpha_k \\
&= \frac{\sum_\alpha c_\alpha \alpha_k \hat{x}^\alpha}{\beta_k \sum_\sigma c_\sigma \hat{x}^\sigma} \\
&= \frac{\hat{x}_k \frac{\partial}{\partial x_k} p(\hat{x})}{\beta_k p(\hat{x})} \ .
\end{aligned}$$

Since this value for x maximizes $q(x;\hat{x})$ and $\log f(x) - \log f(\hat{x}) \geq q(x;\hat{x}) - q(\hat{x};\hat{x})$ it follows that $f(x) \geq f(\hat{x})$. As the formula for x also coincides with that of $x^{(k+1)}$ the proof of Theorem 1 is complete. ■

Some further comments regarding Theorem 1 is in place.

- **Remark 1:** Theorem 1 is intimately related to one of Baum–Eagon [1] which states that if

$$x_k = \frac{\hat{x}_k \frac{\partial}{\partial \hat{x}_k} p(\hat{x})}{\sum_{l=1}^{D} \hat{x}_l \frac{\partial}{\partial \hat{x}_l} p(\hat{x})} \quad \text{and} \quad \sum_{l=1}^{D} \hat{x}_l = 1$$

  then $p(x) \geq p(\hat{x})$ and $\sum_{l=1}^{D} x_l = 1$. The parameters β_l, $l = 1, 2, \ldots, D$ plays a role similar to the Lagrange multiplier in the result by Baum and Eagon.

- **Remark 2:** Theorem 1 does not guarantee convergence towards a global or even a local maximum because in general $f(x)$ has no global maximum. For a counter example consider $p(x) = e^{2\beta^T x}$. However in the case when p is a polynomial, as is the case when $p(\Lambda) = \det(A^T \Lambda A)$, remark 3 guarantees convergence.

- **Remark 3:** If $x^{(k)}$ converges to x then $x_l = 0$ or $\partial f / \partial x_l = 0$. This follows directly by setting $x^{(k+1)} = x^{(k)} = x$ in (9).

Returning to the maximization of $\exp(q_j(\Lambda_j|\hat{\Theta})$ it remains to be proven that this function is of the form needed to apply Theorem 1.

Lemma 1 *Let* $\Lambda = \operatorname{diag}(\lambda_1, \lambda_2, \ldots, \lambda_D)$ *and* $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_D)$. *Writing* $p(\Lambda)$ *or* $p(\lambda)$ *for the polynomial* $\det(A^T \Lambda A)$ *this polynomial is of the form*

$$p(\lambda) = \sum_{\alpha \in \Delta_d} c_\alpha \lambda^\alpha,$$

where

$$\Delta_d \stackrel{def}{=} \left\{ \alpha : \sum_{l=1}^{D} \alpha_l = d, \alpha_l \in \{0, 1\}, l = 1, 2, \ldots, D \right\}$$

and $c_\alpha \geq 0$, $\alpha \in \Delta_d$. □

Proof of Lemma 1: This is an application of the Cauchy-Binet Theorem. To illustrate the proof the case $d = 2$, $D = 3$ is worked out in detail. Write

$$A = \begin{pmatrix} a & b & c \\ d & e & f \end{pmatrix} \quad \text{and} \quad \Lambda = \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{pmatrix}.$$

By direct computation the lemma is proved in the case $d = 2$, $D = 3$:

$$\begin{aligned}
\det(A^T \Lambda A) &= \lambda_1\lambda_2(ae-bd)^2 + \lambda_1\lambda_3(af-cd)^2 + \lambda_2\lambda_3(bf-ce)^2 \\
&= \lambda_1\lambda_2 \left(\det\begin{pmatrix} a & b \\ d & e \end{pmatrix}\right)^2 + \lambda_1\lambda_3 \left(\det\begin{pmatrix} a & c \\ d & f \end{pmatrix}\right)^2 + \\
&\quad \lambda_2\lambda_3 \left(\det\begin{pmatrix} b & c \\ e & f \end{pmatrix}\right)^2 .
\end{aligned}$$

This is also the strategy for the general case. Write $A(\alpha)$ for the matrix where row l is deleted if and only if $\alpha_l = 0$ then the Cauchy–Binet Theorem, [8, p. 8] or [16, p. 22], says that for matrices $A, B \in \mathbf{R}^{D\times d}$ the relation $\det(A^T B) = \sum_{\alpha\in\Delta_d} \det(A(\alpha))\det(B(\alpha))$ holds. Letting $B = \Lambda A$ and noting that $\det B(\alpha) = \lambda^\alpha \det(A(\alpha))$ shows

$$p(\lambda) = \sum_{\alpha\in\Delta_d} \lambda^\alpha (\det(A(\alpha)))^2 .$$

This completes the proof of the lemma. ■

Lemma 1 and Theorem 1 can now be applied to prescribe an update formula for Λ_j. Compute $\frac{\partial p}{\partial \lambda_k^j}(\Lambda_j)$ by noting by Lemma 1 that $p(\Lambda_j)$ is linear in λ_j^k when all the other variables are considered fixed, i.e. $p(\Lambda_j) = \alpha_k^j \lambda_k^j + \beta_k^j$, where α_k^j, β_k^j depends only on λ_l^j for $l = k$. Thus $\frac{\partial p}{\partial \lambda_k^j}(\Lambda_j) = \alpha_k^j$ and since $p(\Lambda_j|_{\lambda_k^j\to 0}) = \beta_k^j$ it follows that

$$\begin{aligned}
\frac{\partial p}{\partial \lambda_k^j}(\Lambda_j) &= \frac{p(\Lambda_j) - p(\Lambda_j|_{\lambda_k^j\to 0})}{\lambda_k^j} \\
&= \frac{\det(A^T\Lambda_j A) - \det(A^T \Lambda_j|_{\lambda_k^j\to 0} A)}{\lambda_k^j} .
\end{aligned}$$

Substituting this value into Theorem 1 yields the update rule

$$\begin{aligned}
\lambda_k^j &= \frac{\pi_j}{\bar{L}_{jl}} \frac{\det(A^T\hat{\Lambda}_j A) - \det(A^T\hat{\Lambda}_j|_{\hat{\lambda}_k^j\to 0} A)}{\det(A^T\hat{\Lambda}_j A)} \\
&= \frac{\pi_j}{\bar{L}_{jl}} \left(1 - \frac{\det(A^T\hat{\Lambda}_j|_{\hat{\lambda}_k^j\to 0} A)}{\det(A^T\hat{\Lambda}_j A)}\right) .
\end{aligned}$$

For the special case $D = d$, i.e. the regular MLLT model, this yields the formula $\lambda_k^j = \pi_j/\bar{L}_{jl}$, since when $A \in \mathbf{R}^{d\times d}$ by the multiplicative property of the determinant $\det(A^T\hat{\Lambda}_j|_{\hat{\lambda}_k^j\to 0}A) = \det(A)^2 \prod_{l=1}^D \lambda_l^j|_{\lambda_k^j\to 0} = 0$. Thus, no iteration is required and the formula is in accordance with the result in [5]. For large values of d and matrices $\Delta \in \mathbf{R}^{d\times d}$ $\det(\Delta)$ may easily underflow or overflow even in high precision arithmetic. Sophisticated mathematical libraries will therefore directly provide a $\log\det(\Delta)$ function as well as the $\det(\Delta)$ function. The problem then becomes one of avoiding evaluation of $\log(0)$. Special precautions must be taken when $D = d$ or the computation must be arranged differently. By computing $\frac{\partial p}{\partial \lambda_k^j}(\Lambda_j) = \alpha_k^j$ from Λ_j and $\Lambda_j|_{\lambda_j^k \to \lambda_j^k + 1}$ instead the numerical problems above are avoided. This yields the alternate formula

$$\lambda_k^j = \hat{\lambda}_k^j \frac{\pi_j}{\bar{L}_{jk}} \left(\frac{\det(A^T \hat{\Lambda}_j|_{\hat{\lambda}_j^k \to \hat{\lambda}_j^k + 1} A)}{\det(A^T \hat{\Lambda}_j A)} - 1 \right) . \tag{10}$$

This formula will be further simplified in the preceding subsection.

4.1.1 Rank One Updates of Positive Definite Symmetric Matrices

Lemma 2 below gives methods to rapidly compute the determinant and inverse of a rank-one update of a positive definite symmetric matrix. This is very useful in implementing $\bar{\Lambda}$ and A updates efficiently.

Lemma 2 (Rank One Updates) *Let $P \in \mathbf{R}^{d \times d}$ be a strictly positive definite symmetric matrix and let $a, b \in \mathbf{R}^{d \times 1}$ and $\alpha \in \mathbf{R}$. Then*

$$\det(P + \alpha a b^T) = (1 + \alpha a^T P^{-1} b) \det(P) \tag{11}$$

and

$$(P + \alpha a b^T)^{-1} = P^{-1} - \frac{\alpha P^{-1} a b^T P^{-1}}{1 + \alpha a^T P^{-1} b}. \tag{12}$$

Of particular interest are these equations in the symmetric case $a = b$.

$$\det(P + \alpha a a^T) = (1 + \alpha a^T P^{-1} a) \det(P) \tag{13}$$

and

$$(P + \alpha a a^T)^{-1} = P^{-1} - \frac{\alpha P^{-1} a a^T P^{-1}}{1 + \alpha a^T P^{-1} a}. \tag{14}$$

$\square$

Proof of Lemma 2: Noting that

$$\det(P + \alpha a b^T) = \det(P) \det\left(I + (P^{-1/2} a)(b^T P^{-1/2})\right)$$

and substituting $\hat{a} = P^{-1/2} a$ and $\hat{b} = P^{-1/2} b$ it suffices to show $\det(I + \hat{a}\hat{b}^T) = 1 + \hat{a}^T \hat{b}$. This is seen by observing that the eigenvalues of $\hat{a}\hat{b}^T$ are 0 and $\hat{a}^T \hat{b}$ (corresponding to either the right eigenvector $\hat{a}$ or the left eigenvector $\hat{b}$). Adding I to a matrix adds 1 to each of the eigenvalues, making the eigenvalues 1 and $1 + \hat{a}^T \hat{b}$ from which it follows that the determinant is $1 + \hat{a}^T \hat{b}$. This proves (11) and (13). The equation (12) is a special case of the Sherman-Morrison-Woodbury formula, [12, p. 50] and can be verified by straight forward multiplication of $P + \alpha a b^T$ and $P^{-1} - \frac{\alpha P^{-1} a b^T P^{-1}}{1 + \alpha a^T P^{-1} b}$. $\blacksquare$

Applying Lemma 2 to (10) yields the relation

$$\begin{aligned} \det\left(A^T \hat{\Lambda}_j|_{\hat{\lambda}_j^k \to \hat{\lambda}_j^k + 1} A\right) &= \det\left(A^T \hat{\Lambda}_j A + a_k a_k^T\right) \\ &= \det\left(A^T \hat{\Lambda}_j A\right) \left(1 + a_k^T (A^T \hat{\Lambda}_j A)^{-1} a_k\right) . \end{aligned}$$

This yields the conceptually simpler and computationally efficient update rule

$$\lambda_k^j = \hat{\lambda}_k^j \frac{a_k^T (A^T \hat{\Lambda}_j A)^{-1} a_k}{\frac{\bar{L}_{jk}}{\pi_j}} . \tag{15}$$

The evaluation of $\Sigma_j = (A^T \hat{\Lambda}_j A)^{-1}$ needs only be done once for the update of all the values λ_k^j, $k = 1, \ldots, D$ and the computation of $a_k^T (A^T \hat{\Lambda}_j A)^{-1} a_k$ requires only $\mathcal{O}(d^2)$ operations. In contrast the general evaluation of a determinant requires $\mathcal{O}(d^3)$ operations. The update rule (15) can be interpreted as saying

$$\lambda_k^j = \hat{\lambda}_k^j \frac{\text{model variance for projection } a_k}{\text{empirical variance for projection } a_k} .$$

When convergence is reached the empirically measured variance must match the model variance.

4.2 An Additive Update Formula for $\bar{\Lambda}$

Application of (13) yields the relationship

$$q(\Lambda_j|_{\lambda_k^j \to \lambda_k^j + \Delta\lambda_k^j} |\hat{\Theta}) = q(\Lambda_j|\hat{\Theta}) + \pi_j \log \left(1 + \Delta\lambda_k^j a_k^T (A^T \hat{\Lambda}_j A)^{-1} a_k\right) - \Delta\lambda_k^j \bar{L}_{jk} . \tag{16}$$

Equation (16) can be maximized directly with respect to $\Delta\lambda_k^j$ and the global maximum value is attained for

$$\Delta\lambda_k^j = \frac{\pi_j}{\bar{L}_{jk}} - \frac{1}{a_k^T (A^T \hat{\Lambda}_j A)^{-1} a_k} .$$

Computationally this requires changing $(A^T \hat{\Lambda}_j A)^{-1}$ after each λ_k^j has been updated. In contrast the multiplicative update rule allowed all λ_k^j, $k = 1, \ldots, D$ to be changed simultaneously. However, in an implementation a running tab of $(A^T \Lambda_j A)^{-1}$ can be kept at a low cost by use of the Sherman-Morrison-Woodbury formula (14). Letting $\Sigma_j = (A^T \Lambda_j A)^{-1}$, $b_k = \Sigma_j a_k$ gives the identities

$$(A^T \Lambda_j A + \Delta\lambda_k^j a_k a_k^T)^{-1} = \Sigma_j - \frac{\Delta\lambda_k^j b_k b_k^T}{1 + \Delta\lambda_k^j a_k^T b_k}$$

and $a_k^T (A^T \hat{\Lambda}_j A)^{-1} a_k = a_k^T b_k$. This makes the individual updates almost as fast as for the multiplicative update.

Another interesting feature of the additive update formula is that it does not require $\lambda_k^j \geq 0$ and so performes the maximization over a larger domain than was the case for the multiplicative update formula (15). The two algorithms generally converged in the same amount of time in the experiments performed, but the additive update formula achieved better loglikelihood value as expected. In experiments it was seen that λ_k^j became negative for some values of j and k and so implementing the additive update requires caution. Firstly it should be noted that if $A^T\Lambda_j A$ is not positive definite it is possible to have the log likelihood function converge to ∞. This can for example be seen by letting $\lambda_k^j \to -\infty$. Secondly when doing the optimization with respect to A, A can no longer be arbitrarily chosen unless all λ_k^j are positive.

5 Accumulating Statistics to Optimize A

When updating A consider $\bar{\Lambda}$ to be fixed. The update formulas for $\bar{\pi}$ and $\bar{\mu}$ are as before $\pi_j = \frac{1}{N}\sum_{j=1}^N \gamma_{ij}$ and $\mu_j = \sum_{i=1}^N x_i\gamma_{ij} / \sum_{i=1}^N \gamma_{ij}$ for $j = 1, 2, \ldots, m$. Define $Q_A(\Theta|\hat{\Theta}) \stackrel{def}{=} Q_\Lambda(\Theta|\hat{\Theta})$. The optimization with respect to A is not quite as straightforward as for $\bar{\Lambda}$, so the use of a generic optimization package will be considered. In this section the focus is on computing $Q_A(\Theta|\hat{\Theta})$ and $\frac{\partial}{\partial a_{kr}} Q_A(\Theta|\hat{\Theta})$ for $k = 1, 2, \ldots, D$, and $s = 1, 2, \ldots, d$. First the expression for $Q_A(\Theta|\hat{\Theta})$ is simplified and analyzed.

$$
\begin{aligned}
Q_A(\Theta|\hat{\Theta}) &= \frac{1}{2N}\sum_{i=1}^N \sum_{j=1}^m \gamma_{ij}\left(\log\det(A^T\Lambda_j A) - (x_i-\mu_j)^T A^T\Lambda_j A(x_i-\mu_j)\right) \\
&= \frac{1}{2}\sum_{j=1}^m \pi_j \log\det(A^T\Lambda_j A) - \\
&\quad \frac{1}{2}\sum_{j=1}^m \operatorname{trace}\left[A^T\Lambda_j A \frac{1}{N}\sum_{i=1}^N (x_i-\mu_j)(x_i-\mu_j)^T\right] \\
&= \frac{1}{2}\sum_{j=1}^m \pi_j \log\det(A^T\Lambda_j A) + \\
&\quad \frac{1}{2}\sum_{j=1}^m \pi_j \operatorname{trace}\left[A^T\Lambda_j A\mu_j\mu_j^T\right] - \\
&\quad \frac{1}{2}\sum_{j=1}^m \operatorname{trace}\left[A^T\Lambda_j A \frac{1}{N}\sum_{i=1}^N \gamma_{ij} x_i x_i^T\right] \\
&= \frac{1}{2}\sum_{j=1}^m \pi_j \log\det(A^T\Lambda_j A) + \\
&\quad \frac{1}{2}\sum_{j=1}^m \pi_j \operatorname{trace}\left[\sum_{k=1}^D \lambda_k^j a_k a_k^T \mu_j\mu_j^T\right] - \\
&\quad \frac{1}{2}\sum_{j=1}^m \operatorname{trace}\left[\sum_{k=1}^D \lambda_k^j a_k a_k^T \frac{1}{N}\sum_{i=1}^N \gamma_{ij} x_i x_i^T\right] \\
&= \frac{1}{2}\sum_{j=1}^m \pi_j \log\det(A^T\Lambda_j A) - \frac{1}{2}\operatorname{trace}\left[\sum_{k=1}^D a_k a_k^T V_k\right]
\end{aligned}
$$

where

$$V_k = \frac{1}{N}\sum_{j=1}^{m}\sum_{i=1}^{N}\lambda_k^j\gamma_{ij}x_ix_i^T - \sum_{j=1}^{m}\pi_j\lambda_k^j\mu_j\mu_j^T \in \mathbf{R}^{d\times d} \quad \text{for} \ \ k=1,\ldots,D \ .$$

Note that this way of computing statistics only requires the storage of D matrices in $\mathbf{R}^{d\times d}$. From the statistics V_k, $Q_A(\Theta|\hat{\Theta})$ can be determined for any value of A. The sum $\sum_{j=1}^{m}\sum_{i=1}^{N}\lambda_k^j\gamma_{ij}x_ix_i^T$ appears to require $NDmd^2$ multiplications, a rather large amount of computation. By taking advantage of the symmetry d^2 may be replaced with $d(d+1)/2$. But even better performance can be obtained by rearranging the terms in the sum

$$\sum_{j=1}^{m}\sum_{i=1}^{N}\lambda_k^j\gamma_{ij}x_ix_i^T = \sum_{i=1}^{N}\left(\sum_{j=1}^{m}\lambda_k^j\gamma_{ij}\right)x_ix_i^T \ .$$

The innermost sum on the right hand side requires a total of Dm multiplications for each value of i and so the total amount of $ND(m+d(d+1)/2)$ multiplications. This is much smaller than $NDmd^2$ when $m>1$ and $d>1$. On the other hand the sum could also be evaluated by summing in the reverse order

$$\sum_{j=1}^{m}\sum_{i=1}^{N}\lambda_k^j\gamma_{ij}x_ix_i^T = \sum_{j=1}^{m}\lambda_k^j\left(\sum_{i=1}^{N}\gamma_{ij}x_ix_i^T\right) \ .$$

Evaluating the innermost sum requires a total of $Nd(d+1)/2$ multiplications per value of j. This yields a total of $Dmd(d+1)/2+Nmd(d+1)/2$ multiplications for the evaluation of V_k, $k=1,2,\ldots,D$. Switching between these two approaches to evaluate V_k according to which of the quantities $ND(m+d(d+1)/2)$ and $Dmd(d+1)/2+Nmd(d+1)/2$ is smaller guarantees the best possible benefit from this strategy. Algorithmically slightly better order of computational complexity can be obtained with the first method by extracting $\sum_{j=1}^{m}\lambda_k^j\gamma_{ij}$ from $\bar{\Lambda}\Gamma$ which can be computed with a fast matrix multiplication algorithm.

5.1 The Derivative of Q_A

V_k, $k=1,2,\ldots,D$ provides all the information needed to compute $Q_A(\Theta|\hat{\Theta})$ for *any* value of the matrix A. Therefore it suffices also to compute the derivatives $\frac{\partial}{\partial a_{kr}}Q_A(\Theta|\hat{\Theta})$. Recall for a general matrix $B\in\mathbf{R}^{d\times d}$ that $\frac{\partial}{\partial b_{rs}}\log\det(B) = (B^{-T})_{rs}$, for $r,s=1,2\ldots,d$, [22, p. 151]. By the chain rule

$$\begin{aligned}
\frac{\partial}{\partial a_{kr}}\log\det(A^T\Lambda_jA) &= \sum_{s=1}^{d}\sum_{t=1}^{d}(A^T\Lambda_jA)_{ts}^{-T}\frac{\partial}{\partial a_{kr}}(A^T\Lambda_jA)_{ts} \\
&= 2\lambda_k^j\sum_{s=1}^{d}(A^T\Lambda_jA)_{rs}^{-1}a_{ks} \\
&= 2\lambda_k^j((A^T\Lambda_jA)^{-1}a_k)_r \ .
\end{aligned}$$

This gives

$$\frac{\partial}{\partial a_{kr}} Q_A(\Theta|\hat{\Theta}) = \sum_{j=1}^{m} \pi_j \lambda_k^j ((A^T \Lambda_j A)^{-1} a_k)_r - (V_k a_k)_r \ .$$

This expression requires as many extractions of determinants as there are Gaussians. This function can be optimized with a generic optimization package, but consider first how a good starting value for A can be found.

5.2 A Good Starting Point for Optimizing Q_A

The cost in optimizing Q_A lies mainly in the expense of evaluating the expression $\frac{1}{2}\sum_{j=1}^{m} \pi_j \log\det(A^T\Lambda_j A)$. This problem disappears entirely in the MLLT case as then $\log\det(A^T\Lambda_j A) = 2\log\det(A) + \log\det(\Lambda_j)$. To take advantage of this trick consider matrices A constrained to be of the form $A = \hat{A}S^T$, where $S \in \mathbf{R}^{d\times d}$ is an invertible matrix.

$$\begin{aligned} Q_A(\Theta|\hat{\Theta}) &= \frac{1}{2}\sum_{j=1}^{m} \pi_j \log\det(S\hat{A}^T\Lambda_j\hat{A}S^T) - \frac{1}{2}\sum_{k=1}^{D} \mathrm{trace}(S\hat{a}_k\hat{a}_k^T S^T V_k) \\ &= Q_A(\hat{\Theta}|\hat{\Theta}) + \log\det(S) - \frac{1}{2}\sum_{k=1}^{D} \mathrm{trace}(S\hat{a}_k\hat{a}_k^T S^T V_k - \hat{a}_k\hat{a}_k^T V_k). \quad (17) \end{aligned}$$

This expression is similar to the formula in the MLLT case. Consider now an optimization strategy invented by M. J. F. Gales, [6] for the optimization of the A matrix in the MLLT case. This strategy involves updating a single row of S while keeping all others fixed and then cycling through all the rows of S until a convergence criteria is reached. The following subsection outlines this strategy.

5.2.1 Row-by-row Update of S

Consider the update of the r^{th} row of S, (say s_r), leaving all other rows fixed. Then, the second term in equation (17) is a quadratic in s_r and can be written in the form

$$\frac{1}{2}\sum_{k=1}^{D} \mathrm{trace}(S\hat{a}_k\hat{a}_k^T S^T V_k) = \frac{1}{2} s_r^T P_r s_r + q_r^T s_r + \text{ const}, \quad (18)$$

where

$$P_r = \sum_{k=1}^{D} (V_k)_{rr} a_k a_k^T,$$

$$Q = (q_1, \ldots, q_r, \ldots, q_d) = \sum_{k=1}^{D} a_k a_k^T S_r^T V_k,$$

and $S_r = S|_{s_r=0}$. The first term in equation (17), namely, $\log\det S$, can be written as $\log(c_r^T s_r)$ where c_r is the cofactor associated with the r^{th} row of S.

Taking the gradient with respect to s_r of $Q_A(\Theta|\hat{\Theta})$ the equation for the critical points becomes

$$\frac{1}{c_r^T s_r} c_r - P_r s_r - q_r = 0, \Leftrightarrow s_r = P_r^{-1}\left(\frac{1}{c_r^T s_r} c_r - q_r\right). \qquad (19)$$

Multiplying both sides with c_r^T gives the following scalar quadratic equation for $x = c_r^T s_r$:

$$x^2 - x c_r^T P_r^{-1} q_r - c_r^T P_r^{-1} c_r = 0,$$

the solution for which is

$$c_r^T s_r = c_r^T P_r^{-1} q_r \pm \sqrt{(c_r^T P_r^{-1} q_r)^2 + 4 c_r^T P_r^{-1} c_r}. \qquad (20)$$

From equation (19) the update formula for s_r follows. Since the equation (20) gives two possible solutions it is prudent to try both solutions and pick the one that yields the larger value. This is equivalent to picking the sign that maximizes the function $\log(x) + \frac{1}{2x^2} c_r^T P_r^{-1} c_r$.

5.3 Computational Considerations

The cost of evaluating $\sum_{j=1}^m \pi_j \log\det(A^T \Lambda_j A)$ is $\mathcal{O}(mDd^2)$. For values of m in the range 10^4 to 10^6 this becomes rather significant. Consider therefore breaking the problem up into subproblems as follows. By Lemma 2

$$\log\det(A^T\Lambda_j A) = \log\det(A^T\Lambda_j|_{\lambda_k^j\to 0}A) + \log\left(1 + \lambda_k^j a_k^T (A^T\Lambda_j|_{\lambda_k^j\to 0}A)^{-1} a_k\right) .$$

Q_A can then be written on the form

$$Q_A(\Theta|\hat{\Theta}) = \frac{1}{2}\sum_{j=1}^m \pi_j \log\left(1 + \lambda_k^j a_k^T (A^T\Lambda_j|_{\lambda_k^j\to 0}A)^{-1} a_k\right) + a_k^T V_k a_k + \text{const} ,$$

where const refers to the remaining terms that are independent of a_k. Once the constant and $(A^T\Lambda_j|_{\lambda_k^j\to 0}A)^{-1}$ has been precomputed the computational cost of each function evaluation is down to $\mathcal{O}(md^2)$. Furthermore to compute $Q_A(\Theta|\hat{\Theta})$ for $a_k = u + tv$, $u, v \in \mathbf{R}^d$, for any $t \in \mathbf{R}$ can be accomplished with only $\mathcal{O}(m)$ operations after some precomputation has been performed. Each row a_k can now be optimized separately. Each of these optimization steps can be performed with a generic function optimizer, and inbetween the steps the multiplicative S update can be considered since the cost is only of the order of $\mathcal{O}(Dd^3)$ operations.

6 MAP and Smoothing Using a Bayesian Prior

This section considers repeating the theory in the preceeding sections using a Bayesian prior for the parameters $\Theta = (\bar{\pi}, \bar{\mu}, \bar{\Sigma})$. This prior shall be denoted

$P_B(\Theta)$. Instead of maximizing $\frac{1}{N}\sum_{i=1}^N \log f(x_i|\Theta)$ or equivalently $\prod_{i=1}^N f(x_i|\Theta)$

$$P_B(\Theta)\prod_{i=1}^N f(x_i|\Theta)$$

is maximized instead. There are two reasons to consider using a Bayesian prior. Firstly, a known problem for the EM algorithm is that the determinant of the covariance for a Gaussian component may converge to zero. Imposing a prior with the property that covariances with small determinant values are extremely unlikely prevents this phenomenon from happening, [24].

Secondly, imposing an appropriate prior will allow us to perform maximum a posteriori adaptation (MAP adaptation) as developed in [9]. The prior distribution imposed is in the spirit of [9, 24]. Let

$$P_B(\Theta) = P_B(\bar\pi|\underline{\bar\pi},\eta^0)\prod_{j=0}^m P_B(\mu_j|\underline{\mu}_j,\Sigma_j,\eta_j^1)P_B(\Sigma_j|\underline{\Sigma}_j,\eta_j^2,\alpha_j)\ ,$$

where it should be understood that $\underline{\bar\pi}$, $\underline{\bar\mu}$ and $\underline{\bar\Sigma}$ serves as centroids for the distributions of $\bar\pi$, $\bar\mu$ and $\bar\Sigma$. Furthermore η^0, $\bar\eta^1$ and $\bar\eta^2$ could be thought of as fractional counts analogous to $\sum_{i=1}^N \gamma_{ij}$.

The distributions for the parameters $\bar\pi$, $\bar\mu$ and $\bar\Sigma^{-1}$ are respectively the Dirichlet distribution [23, p. 44], the Gaussian distribution and the Wishart distribution [23, p. 85]. Precise definitions of the priors and the update formulas will be given in what follows.

For the mixture weights the prior distribution is

$$\begin{aligned} P_B(\bar\pi|\underline{\bar\pi},\eta^0) &\stackrel{def}{=} \frac{\Gamma\left(\eta^0(m+\sum_{j=1}^m \underline{\pi}_j)\right)}{\prod_{j=1}^m \Gamma\left(\eta^0(\underline{\pi}_j+1)\right)}\prod_{j=1}^m \pi_j^{\eta^0\underline{\pi}_j} \\ &\propto \prod_{j=1}^m \pi_j^{\eta^0\underline{\pi}_j}\ , \end{aligned}$$

where $\pi_j,\underline{\pi}_j > 0$, $1 = \sum_{j=1}^m \pi_j = \sum_{j=1}^m \underline{\pi}_j$ and $\eta^0\pi_j > -1$. This yields the following update rule for π_j

$$\pi_j = \frac{\sum_{i=1}^N \gamma_{ij} + \eta^0\underline{\pi}_j}{N+\eta^0}\ . \tag{21}$$

Note that either $\eta^0 = 0$ or $\underline{\pi}_j = \pi_j$ yields the same update rule as before.

For the means impose the prior:

$$\begin{aligned} P_B(\mu_j|\underline{\mu}_j,\Sigma_j,\eta_j^1) &\stackrel{def}{=} \mathcal{N}\left(\mu_j;\underline{\mu}_j,\frac{1}{\eta_j^1}\Sigma_j\right) \\ &= (2\pi)^{-\frac{d}{2}}\left|\det\left(\frac{1}{\eta_j^1}\Sigma_j\right)\right|^{-\frac{1}{2}} e^{-\frac{\eta_j^1}{2}(\mu_j-\underline{\mu}_j)^T\Sigma_j^{-1}(\mu_j-\underline{\mu}_j)} \end{aligned}$$

for $j = 1, 2, \ldots, m$. The update rule corresponding to this prior is

$$\mu_j = \frac{\sum_{i=1}^N \gamma_{ij} x_i + \eta_j^1 \underline{\mu}_j}{\sum_{i=1}^N \gamma_{ij} + \eta_j^1}, j = 1, 2, \ldots, m. \tag{22}$$

Again either of the choices $\eta_j^1 = 0$ or $\underline{\mu}_j = \mu_j$ leads to the same update rule as before.

Finally the prior for Σ_j is:

$$\begin{aligned}
P_B(\Sigma_j | \underline{\Sigma}_j, \eta_j^2, \alpha_j) &\stackrel{def}{=} W_{\alpha_j}\left(\Sigma_j^{-1}; \frac{1}{\eta_j^2}\underline{\Sigma}_j^{-1}\right) \\
&= \frac{\left|\det\left(\frac{1}{\eta_j^2}\underline{\Sigma}_j^{-1}\right)\right|^{(d-\alpha_j-1)/2} \left|\det \underline{\Sigma}_j\right|^{d/2}}{2^{d\alpha_j/2}\pi^{\alpha_j(\alpha_j-1)/4}\prod_{i=1}^{\alpha_j}\Gamma\left(\frac{d+1-i}{2}\right)} e^{-\frac{\eta_j^2}{2}\mathrm{trace}\left(\underline{\Sigma}_j \Sigma_j^{-1}\right)} \\
&\propto \left|\det\left(\frac{1}{\eta_j^2}\underline{\Sigma}_j^{-1}\right)\right|^{(d-\alpha_j-1)/2} e^{\frac{-\eta_j^2}{2}\mathrm{trace}\left(\underline{\Sigma}_j \Sigma_j^{-1}\right)} .
\end{aligned}$$

Applying these priors to $Q(\Theta|\hat{\Theta})$ gives the following formula

$$\begin{aligned}
Q(\Theta|\hat{\Theta}) &= \frac{1}{2N}\sum_{j=1}^m \left(d - \alpha_j + \sum_{i=1}^n \gamma_{ij}\right) \log\det\left(\Sigma_j^{-1}\right) \\
&\quad -\frac{1}{2N}\sum_{j=1}^m \mathrm{trace}\left[\Sigma_j^{-1}\left(\eta_j^1(\underline{\mu}_j - \mu_j)(\underline{\mu}_j - \mu_j)^T\right.\right. \\
&\qquad \left.\left. +\eta_j^2\underline{\Sigma}_j + \sum_{i=1}^N \gamma_{ij}(x_i - \mu_j)(x_i - \mu_j)^T\right)\right] \\
&\quad +\mathrm{const} .
\end{aligned}$$

It is clear that this Q function is still of the same form as before (a sum of log determinants plus a linear form in Σ_j^{-1}). Therefore the same update rules as before works as long as the appropriate Bayesian "penalty" terms is added to the statistics. The new statistics are as follows. For the mixture weights that are used in the formula, π_j must be replaced with

$$\frac{d - \alpha_j}{N} + \frac{1}{N}\sum_{i=1}^N \gamma_{ij} . \tag{23}$$

Similarly L_j is computed by

$$\begin{aligned}
L_j &= \frac{1}{N}\sum_{i=1}^N \gamma_{ij}(Ax_i)(Ax_i)^T - \pi_j(A\mu_j)(A\mu_j)^T \\
&\quad +\eta_j^1\left(A(\underline{\mu}_j - \mu_j)\right)\left(A(\underline{\mu}_j - \mu_j)\right)^T + \eta_j^2 A\underline{\Sigma}_j A^T
\end{aligned} \tag{24}$$

for $j = 1, 2, \ldots, m$. The iteration procedure for Λ_j is then as before. For A the statistics is given by the formula

$$V_k = \frac{1}{N}\sum_{j=1}^{m}\sum_{i=1}^{N}\lambda_k^j\gamma_{ij}x_ix_i^T - \sum_{j=1}^{m}\pi_j\lambda_k^j\mu_j\mu_j^T + \sum_{j=1}^{m}\lambda_k^j\eta_j^1(\underline{\mu}_j - \mu_j)(\underline{\mu}_j - \mu_j)^T + \sum_{j=1}^{m}\lambda_k^j\eta_j^2\underline{\Sigma}_j \,. \tag{25}$$

The Q function is optimized as in the previous section by using a generic function optimizer.

6.1 Smoothing EM Update Using Bayesian Priors

Mimicking [24] set $\eta^0 = \eta_1^1 = \cdots = \eta_m^1 = 0$, $\eta_1^2 = \cdots = \eta_m^2 = n$, $\alpha_1 = \cdots = \alpha_m = d$ and $\underline{\Sigma}_1 = \cdots = \underline{\Sigma}_m = I$. As noted before this will cause no change in the update rules for $\bar{\pi}$ and $\bar{\mu}$, whereas the new formulas for L_j and V_k becomes

$$L_j = \frac{1}{N}\sum_{i=1}^{N}\gamma_{ij}(Ax_i)(Ax_i)^T - \pi_j(A\mu_j)(A\mu_j)^T + nAA^T, \quad j = 1, \ldots, m$$

and

$$V_k = \frac{1}{N}\sum_{j=1}^{m}\sum_{i=1}^{N}\lambda_k^j\gamma_{ij}x_ix_i^T - \sum_{j=1}^{m}\pi_j\lambda_k^j\mu_j\mu_j^T + n\sum_{j=1}^{m}\lambda_k^jI, \quad k = 1, \ldots, D.$$

A typical value envisioned for n is 2.

6.2 MAP Adaptation with EMLLT

Suppose that a model $\underline{\Theta} = (\underline{\bar{\pi}}, \underline{\bar{\mu}}, \underline{\bar{\Sigma}})$ was previously trained on a big training corpus. To retrain the parameters on a new smaller training corpus the MAP technique is used, ensuring that none of the counts become zero. Choosing a minimal count n ensures a minimum points necessary for an EM update. Choose the parameters $\eta^0 = \eta_1^1 = \cdots = \eta_m^1 = \eta_1^2 = \cdots = \eta_m^2 = n$, $\alpha_1 = \cdots = \alpha_m = d$, and let $\underline{\pi}_j, \underline{\mu}_j$ and $\underline{\Sigma}_j$ be given by the model trained on the big training corpus then perform the EM updates according to (21), (22), (23), (24) and (25).

7 Experiments and Results

Speech Recognition experiments in a car environment were performed on a corpus considered in the related paper [10]. The corpus used is larger both in terms of training and test data. The relative improvements of MLT reported in [10] is not as good as relative improvements of EMLLT reported in this paper. The baseline system explored in [10] had 5261 Gaussians. This is about half the size of the model used in this paper, and so the corresponding overall error rates are therefore appropriately lower for the EMLLT model.

7.1 The Test Set

The speech recognition system and test set used in these experiments are described in [4]. The test set was recorded in a car environment and consists of a total of 73743 words. It is divided into the tasks (A) Addresses, (C) Command and Control, (D) Digits and (R) Radio Control. Each of these tasks were recorded in a car at 3 different velocities: 0 mph (0 kph), 30 mph (50kph) and 60 mph (100 kph). Table 1 breaks down the subtasks by number of words.

Number of words in test set				
task/speed	00mph	30mph	60mph	all
A	5249	5249	5249	15747
C	4413	4418	4418	13249
D	10086	10086	10086	30258
R	4677	4906	4906	14489
all	24425	24659	24659	73743

Table 1: Number of words in the test script broken down by tasks.

7.2 Recognition Results and Baseline Acoustic Models

The baseline system used 39 dimensional feature vectors consisting of 13 dimensional cepstral vectors with corresponding first and second order derivatives. The baseline system had 89 phonemes, 37 of which were word model phonemes specific to the digit task. In total there were 680 context dependent phonemes of which necessarily 37 were digit specific. The system had a total of 10253 Gaussians. The performance of the baseline system is given in Table 2. Another essential comparison necessary is that of the MLLT system ($D = d$). Using an MLLT matrix lowers the error rate by 9.5%, see Table 3.

Error rate for baseline acoustic model				
task/speed	00mph	30mph	60mph	all
A	2.69%	3.43%	5.73%	3.95%
C	0.79%	1.54%	3.62%	1.99%
D	0.76%	2.27%	8.57%	3.87%
R	0.83%	1.20%	3.32%	1.80%
all	1.20%	2.17%	6.03%	**3.14%**

Table 2: Baseline Acoustic Model Recognition results.

Error rate for MLLT acoustic model				
task/speed	00mph	30mph	60mph	all
A	2.84%	3.71%	6.19%	4.25%
C	0.61%	1.67%	2.99%	1.76%
D	0.74%	1.92%	6.86%	3.18%
R	0.79%	1.02%	2.87%	1.57%
all	1.18%	2.08%	5.23%	**2.84%**

Table 3: Recognition results for baseline system retrained with an MLLT transformation.

7.3 Recognition Results with the EMLLT Model

Following the descriptions for estimating the parameters $\bar{\pi}, \bar{\mu}, \bar{\Lambda}$ and A models with $D = 2d$, $D = 4d$, $D = 8d$ and $D = 14d$ were built. The largest model yields a 35% relative improvement over the baseline model and 28% over the MLLT model. The smaller models yields smaller improvements. The precise breakdown of the error rates is given in Table 4, 5, 6 and 7.

Error rate for $D = 2d$ acoustic model				
task/speed	00mph	30mph	60mph	all
A	2.51%	3.39%	5.87%	3.92%
C	0.61%	1.29%	2.58%	1.49%
D	0.76%	1.73%	5.92%	2.80%
R	0.49%	1.04%	2.71%	1.43%
all	1.06%	1.87%	4.67%	**2.54%**

Table 4: Recognition results for $D = 2d$ system.

Error rate for $D = 4d$ acoustic model				
task/speed	00mph	30mph	60mph	all
A	2.21%	3.45%	5.81%	3.82%
C	0.70%	1.24%	2.63%	1.52%
D	0.52%	1.49%	5.13%	2.38%
R	0.53%	1.10%	2.55%	1.41%
all	0.92%	1.78%	4.31%	**2.34%**

Table 5: Recognition results for $D = 4d$ system.

The direct comparison between the $2d \times d$, $4d \times d$, $8d \times d$ and $14d \times d$ EMLLT models and the MLLT model with the same number of Gaussian components shows an improvement in the error rate. While this certainly reflects better

Error rate for $D = 8d$ acoustic model				
task/speed	00mph	30mph	60mph	all
A	1.91%	3.18%	4.88%	3.32%
C	0.70%	1.09%	2.17%	1.32%
D	0.47%	1.32%	4.96%	2.25%
R	0.45%	0.88%	2.14%	1.17%
all	0.81%	1.59%	3.88%	**2.10%**

Table 6: Recognition results for $D = 8d$ system.

Error rate for $D = 14d$ acoustic model				
task/speed	00mph	30mph	60mph	all
A	1.91%	2.88%	4.55%	3.11%
C	0.68%	1.02%	2.26%	1.32%
D	0.46%	1.33%	4.85%	2.21%
R	0.47%	0.84%	2.24%	1.19%
all	0.81%	1.50%	3.80%	**2.04%**

Table 7: Recognition results for $D = 14d$ system.

acoustic modeling no claim can be made as to EMLLT being a better model than MLLT from this alone. The number of parameters in the EMLLT model is considerably larger than that of the MLLT model and so a fair comparison requires an MLLT model with a large enough number of Gaussians to match the number of parameters in the EMLLT model. The parameter count for a diagonal covariance model is $m_{\mathrm{Diag}}(2d+1)$, $m_{\mathrm{MLLT}}(2d+1)+d^2$ for the MLLT model and $m_{\mathrm{EMLLT}}(1+d+D)+Dd$ for the EMLLT model. This means that to make a mixture of diagonal covariance Gaussians with the same number of parameters as the EMLLT model

$$m_{\mathrm{Diag}} \approx \frac{m_{\mathrm{EMLLT}}(1+d+D)+Dd}{2d+1} .$$

The experiments used $m_{\mathrm{EMLLT}} = 10253$, $d = 39$ and $D = 2d$, $4d$, $8d$ and $14d$. The corresponding numbers for m_{Diag} are 15353, 25515 and 45838 respectively for the $2d$, $4d$ and $8d$ EMLLT systems. As choosing the number of Gaussian components with absolute precision in a full fledged state of the art speech recognition system is somewhat difficult, slightly larger systems were built. The actual values used for m_{Diag} was 17028, 26460 and 46500. The word error rates for the diagonal covariance models are displayed in Table 8, 9 and 10. The corresponding recognition results for the MLLT models with the same number of Gaussian components are displayed in Table 11, 12 and 13. These results compare favorably with the diagonal covariance results, thus reaffirming that MLLT indeed provides improved modeling even when used for larger mixtures. Also observe that EMLLT indeed yields better results than the corresponding size MLLT model. The improved acoustic modeling captured by modeling structure in the precision matrix may not fully be explaining the improvements gained by increasing the number of Gaussian components in the MLLT model.

Error rate for diagonal covariance system, $m_{\text{Diag}} = 17028$				
task/speed	00mph	30mph	60mph	all
A	2.34%	3.33%	5.89%	3.85%
C	0.61%	1.58%	3.19%	1.80%
D	0.70%	2.04%	8.71%	3.82%
R	0.75%	1.30%	3.57%	1.89%
all	1.05%	2.09%	6.10%	**3.08**%

Table 8: Recognition results for diagonal system with 17028 Gaussian components.

Error rate for diagonal covariance system, $m_{\text{Diag}} = 26460$				
task/speed	00mph	30mph	60mph	all
A	2.25%	3.37%	5.92%	3.85%
C	0.66%	1.47%	3.19%	1.77%
D	0.66%	1.99%	8.53%	3.73%
R	0.58%	1.20%	3.30%	1.71%
all	0.99%	2.04%	5.98%	**3.01**%

Table 9: Recognition results for diagonal system with 26460 Gaussian components.

The readers attention should be drawn to the following two implementation specific details that contributed towards giving the best results. Firstly the results shown used the additive algorithm for the update of $\bar{\Lambda}$. The additive algorithm gave consistently better results than the multiplicative algorithm. The overall error rates for $D = 2d$, $4d$, $8d$ and $14d$ with the multiplicative update rule were 2.65%, 2.45%, 2.26% and 2.19%. With the additive update rule the corresponding error rates were 2.54%, 2.34%, 2.10% and 2.04%. This gain is consistent and substantial enough that it is worthwhile spending the effort to implement the additive update.

Secondly, the best result used the following method to initialize the starting value for A. For comparison the result from an experiment involving an $8d \times d$ matrix is shown. The $8d \times d$ matrix was initialized from an identity matrix and $7d$ rows chosen randomly according to a probability density with uniform density on the unit sphere in $\mathbf{R}^d$. This system achieved an error rate of 2.32% with the multiplicative algorithm after 8 iterations of $\bar{\Lambda}$ and A updates. This should be compared to the error rate 2.26% reached with only 4 iterations when the matrix was initialized in the manner described below.

Error rate for diagonal covariance system, $m_{\text{Diag}} = 46500$				
task/speed	00mph	30mph	60mph	all
A	2.21%	3.37%	5.79%	3.79%
C	0.57%	1.43%	2.90%	1.63%
D	0.63%	2.12%	7.78%	3.51%
R	0.58%	1.10%	2.85%	1.53%
all	0.95%	2.06%	5.50%	**2.84%**

Table 10: Recognition results for diagonal system with 46500 Gaussian components.

Error rate for MLLT system, $m_{\text{MLLT}} = 17028$				
task/speed	00mph	30mph	60mph	all
A	2.55%	3.56%	5.77%	3.96%
C	0.77%	1.58%	3.21%	1.86%
D	0.72%	1.79%	6.60%	3.04%
R	0.75%	1.02%	2.94%	1.58%
all	1.13%	1.98%	5.09%	**2.74%**

Table 11: Recognition results for MLLT system with 17028 Gaussian components.

The initialization process is illustrated here for the $D = 14d$ EMLLT model. To initialize this system a $d \times d$ MLLT matrix was estimated for data corresponding to phonemes from 14 different phonetic groupings. Phonetic groupings for American-English as suggested by [18, 20, 25] was used. The 14 phonetic groupings used for the 14xMLLT system is shown in Table 14.

In view of these results it is natural to ask how this compares with a full covariance model. After all a full covariance model is far easier to train and an EMLLT model is just an approximation to a full covariance model. When building a full covariance system with $m_{\texttt{fullCov}} = 10253$ and using the corresponding diagonal covariance system as a starting point for the parameter estimation the overall error rate became 2.52%. Speculating that the initial value given for the covariance is of significance a full covariance system was initialized with the multiplicatively update $14d \times d$ system whose performance was 2.19%. The fully trained full covariance system gave an error rate of 2.44%, better than when initialized with a diagonal covariance system, yet performing far worse than the EMLLT system that was used to initialize the full covariance model.

7.4 Sparse Matrix Representations of $\bar{\Lambda}$

However nice the performance of the 14xMLLT system, the model is roughly 7 times larger than the baseline model. This hurts the computational perfor-

Error rate for MLLT system, $m_{\mathrm{MLLT}} = 26460$				
task/speed	00mph	30mph	60mph	all
A	2.50%	3.26%	5.72%	3.82%
C	0.59%	1.61%	3.08%	1.76%
D	0.80%	1.62%	6.20%	2.87%
R	0.49%	0.98%	2.61%	1.37%
all	1.07%	1.84%	4.82%	**2.58%**

Table 12: Recognition results for MLLT system with 26460 Gaussian components.

Error rate for MLLT system, $m_{\mathrm{MLLT}} = 46500$				
task/speed	00mph	30mph	60mph	all
A	2.17%	3.12%	5.73%	3.68%
C	0.66%	1.45%	3.08%	1.73%
D	0.66%	1.73%	5.89%	2.76%
R	0.62%	1.00%	2.47%	1.37%
all	0.98%	1.83%	4.67%	**2.50%**

Table 13: Recognition results for MLLT system with 46500 Gaussian components.

mance and causes recognition to be approximately 7 times slower than for the baseline model. This paper does not aim to study this computational problem in detail, but a few suggestions as to how to improve the speed with little or no degradation in the error rate are shown. For these purposes the 8xMLLT model that was trained using the multiplicative update rule is used. The matrix $\bar{\Lambda}$ contains many small values and so it seems natural to prune these away. In other words a sparsity structure on $\bar{\Lambda}$ will be sought. Forcing such sparsity structure and retraining the model yields some of the computational advantages of the MLT model in [10]. Since the multiplicative algorithm never changes the value of λ_k^j once it is set to zero any such model can easily be retrained using the multiplicative update rule. Figure 1 shows the λ_k^j values for the $D = 8d$ EMLLT model for $k = 1$ in ascending order. Note that almost half of the values are very small. The figure also contains the same graph on a log scale.

The difficulty becomes now how to decide the sparsity structure. force to zero.

7.4.1 Sparsity Structure by Kullback Leibler Distance

The basic idea for choosing which λ_k^j's to zero out is to pick the values that change the estimated distribution the least in the Kullback Leibler distance sense. The Kullback Leibler distance between two full covariance gaussians is,

Phoneme group	List of phonemes
front vowels	IX EH IY IH
mid vowels	AE AA AH AX AO AXR ER
back vowels	UH UW
diphtongs	AW AY EY OW OY
liquids	W L
glides	Y R
nasals	M N NG
voiced stops	B BD D DD G GD DX
unvoiced stops	K KD P PD T TD DX
voiced fricatives	V DH Z ZH
unvoiced fricatives	F TH S SH TS
whisper	HH
affricates	CH JH
silence phonemes	D$ X

Table 14: Groupings of American-English phonemes used in initializing A for $D = 14d$.

[19, p. 89-90]

$$\begin{aligned} D(\mathcal{N}(\cdot;\mu_1,\Sigma_1)|\mathcal{N}(\cdot;\mu_2,\Sigma_2) &= \frac{1}{2}\log\frac{\det\Sigma_2}{\det\Sigma_1} + \frac{1}{2}\mathrm{trace}(\Sigma_1(\Sigma_2^{-1}-\Sigma_1)) + \\ & \frac{1}{2}\mathrm{trace}(\sigma_2^{-1}(\mu_1-\mu_2)(\mu_1-\mu_2)^T) . \end{aligned} \tag{26}$$

Since only changes in $\bar{\Lambda}$ into a sparse matrix $\bar{\Lambda}^0$ is considered, the last term in (26) can be ignored. Letting $\Sigma_1 = (A^T\Lambda_jA)^{-1}$ and $\Sigma_2 = (A^T\Lambda_j^0A)^{-1}$ the Kullback Leibler distance becomes

$$\frac{1}{2}\left\{\log\det(A^T\Lambda_jA) - \log\det(A^T\Lambda_j^0A) + \mathrm{trace}((A^T\Lambda_jA)^{-1}(A^T\Lambda_j^0A)) - d\right\} .$$

Letting $B = A(A^T\Lambda_jA)^{-1}A^T$ the above expression simplifies to

$$\frac{1}{2}\left\{\mathrm{trace}(\Lambda_j^0B) - \log\det(A^T\Lambda_j^0A) + \mathrm{const}\right\} . \tag{27}$$

Λ_j^0 was chosen in a greedy fashion by selectively setting λ_j^k's to zero and choosing the λ_k^j that minimizes the Kullback Leibler distance from the original distribution. Notice that the only computation needed for computing the Kullback Leibler distance when setting a single additional λ_k^j to zero is to re-evaluate $-\log\det(A^T\Lambda_j^0A)$ and subtract $B_{kk}\lambda_k^j$. And as noted in Lemma 2 the rank one update of $\det(A^T\Lambda_j^0A)$ can be performed in $\mathcal{O}(d^2)$ operations.

Following this procedure and picking md, $2md$ and $4md$ values in $\bar{\Lambda}$ to be kept, the resulting models were generated and $\bar{\Lambda}$ retrained with the multiplicative update rule. The resulting error rate were 2.98%, 2.31% and 2.24% as

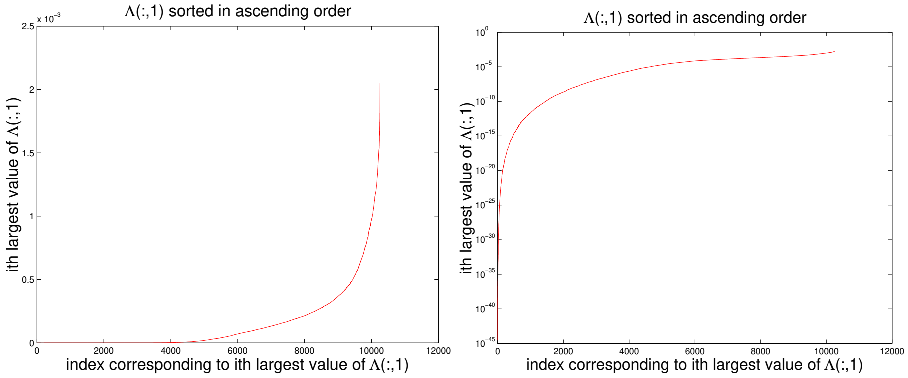

Figure 1: The values of $\bar{\Lambda}$ sorted in ascending order for $k = 1$ for the $D = 8d$ EMLLT system.

compared to the original model that had an error rate of 2.26%. Evidently this intuitive technique does not perform well in the case of aggressive pruning. E.g. the case of md nonzero entries in $\bar{\Lambda}$ is worse than the MLLT model. Notice however that Theorem 1 provides a method to dynamically estimate the best choice for the remaining nonzero entries in $\bar{\Lambda}$ that minimizes the KL distance. This allows the remaining nonzero values to compensate for the values that were zeroed out. Also observe the similarity between (8) and (27) that suggests optimizing

$$\frac{1}{2}\{\mathrm{trace}(\Lambda_j^0 B) - \pi_j \log\det(A^T \lambda_j^0 A)\}$$

instead. Moreover, if access to the statistics $\bar{L}$ is given it may be even better to directly optimize (8) when finding a sparsity structure for $\bar{\Lambda}$.

References

[1] 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," Bulletin of the American Mathematical Society, (**73**), pp. 360–363, 1967.

[2] A. Borodin and I. Munro "The Computational Complexity of Algebraic and Numeric Problems," American Elsevier, New York, pp. 45-46 , 1975.

[3] 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**(B), pp. 1-38, 1977.

[4] S. Deligne, E. Eide, R. Gopinath, D. Kanevsky, B. Maison, P. Olsen, H. Printz and J. Sedivy, "Low–resource speech recognition of 500 word vocabularies," *Proceedings of Eurospeech*, 2001.

[5] M. J. F. Gales, “Maximum likelihood linear transformations for HMM-based speech recognition,” Technical Report 291, Cambridge University, England, May 1997.

[6] M. J. F. Gales, “Semi-tied Covariance Matrices for Hidden Markov Models”, IEEE Trans. in Speech and Audio Processing, Vol. 7, pp. 272-281, 1999.

[7] M. J. F. Gales “Factored Semi-Tied Covariance Matrices,” *Proceedings of NIPS, Denver, USA*, 2000.

[8] F. R. Gantmacher, “Theory of Matrices,” Vol. 1, Chelsea, New York, 1960.

[9] J.-L. Gauvain and C.-H. Lee, “Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains,” *IEEE Transactions on Speech and Audio Processing*, **2** (2), pp. 291–298, 1994.

[10] N. K. Goel and R. A. Gopinath, “Multiple Linear Transforms,” *Proceedings of ICASSP Salt Lake City, USA*, 2001.

[11] N. K. Goel and A. G. Andreou, “Heteroscedastic Discriminant Analysis and Reduced-Rank HMMs for Improved Speech Recognition”, *Speech Communications*, **26**, pp. 283–297, 1998.

[12] G. H. Golub and C. F. Van Loan, “Matrix Computations,” *The Johns Hopkins University Press*, Third Ed., Baltimore, 1996.

[13] R. A. Gopinath, “Maxmimum Likelihood Modeling with Gaussian Distributions for Classification,” *ICASSP, Seattle, USA*, 1998.

[14] R. A. Gopinath, “Constrained Maximum Likelihood Modeling with Gaussian Distributions,” *Proceedings of ARPA Workshop on Human Language Understanding,* January 1998.

[15] R. A. Gopinath, B Ramabhadran and S Dharanipragada, “Factor Analysis Invariant to Linear Transformations of Data,” *Proceedings International Conference on Spoken Language Processing,* Sydney Australia, 1998.

[16] R. A. Horn and C. R. Johnson, “Matrix Analysis,” Cambridge University Press, Cambridge, 1985.

[17] J. L. W. V. Jensen, “Sur les fonctions convexes et les inégalités entre les valeurs moyennes,” *Acta Mathematica*, pp. 175–193, Uppsala, Sweden, 1906.

[18] D. Jurafsky and J. H. Martin, “An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition,” Prentice Hall, Upper Saddle River, New Jersey, pp. 91–103, 2000.

[19] S. Kullback, “Information Theory and Statistics,” Dover Publications Inc., Mineola, New York, 1968.

[20] P. Ladefoged, "A Course in Phonetics," Harcourt Brace Jovanovich Inc., 32d Ed., 1993.

[21] L. A. Liporace, "Maximum Likelihood Estimation for Multivariate Observation of Markov Sources," *IEEE Transactions of Information Theory* **5**, pp. 729–734, 1982.

[22] J. R. Magnus and H. Neudecker, "Matrix differential calculus with applications in statistics and econometrics," John Wiley, New York, 1999.

[23] K. V. Mardia, J. T. Kent and J. M. Bibby, "Multivariate Analysis," Academic Press, New York, 2000.

[24] D. Ormoneit and V. Tresp, "Improved Gaussian mixture density estimates using Bayesian penalty terms and network averaging," *Advances in Neural Information Processing*, (**8**), MIT Press, pp 542–548, 1995.

[25] L. R. Rabiner and R. W. Schafer, "Digital Processing of Speech Signals," Prentice Hall, Englewood Cliffs, New Jersey, pp. 43–45, 1993.

[26] D. B. Rubin and D. T. Thayer, "EM algorithms for ML factor analysis," *Psychometrika*, **47**, pp. 69–76.

[27] G. Saon, M. Padmanabhan, R. A. Gopinath and S. Chen. "Maximum Likelihood Discriminant Feature Spaces," *Proceedings of ICASSP, Istanbul, Turkey*, 2000.