Feature and model space speaker adaptation with full covariance Gaussians

Daniel Povey, George Saon

IBM T.J. Watson Research Center
Yorktown Heights, NY, USA
{dpovey,gsaon} @ us.ibm.com

Abstract

Full covariance models can give better results for speech recognition than diagonal models, yet they introduce complications for standard speaker adaptation techniques such as MLLR and fMLLR. Here we introduce efficient update methods to train adaptation matrices for the full covariance case. We also experiment with a simplified technique in which we pretend that the full covariance Gaussians are diagonal and obtain adaptation matrices under that assumption. We show that this approximate method works almost as well as the exact method.

1. Introduction

Maximum Likelihood Linear Regression (MLLR) and feature space MLLR (fMLLR, also known as constrained MLLR) are commonly used speaker adaptation techniques; however, the convenient and efficient update techniques that are commonly used [1] only work for diagonal covariance Gaussians. Recently there has been some interest in the use of full covariance Gaussians and subspace representations of full covariance precision matrices [4, 5, 2, 10]. However, to date no very convenient and efficient implementations of MLLR and fMLLR adaptation exist for the full covariance case. In [4, 5], general purpose numerical optimization routines were used to optimize the adaptation matrices; however, this is not very convenient if the aim is to produce self-contained software. In [2], elegant row-by-row updates for adaptation matrices were introduced; however, that approach is considerably less efficient than the approach presented here.

In this paper we present an efficient iterative update that optimizes the adaptation matrices in about the same time as diagonal-covariance MLLR and fMLLR. It is applicable in the "time-efficient" (as opposed to memory-efficient) versions of the MLLR and fMLLR computation, where we accumulate mean statistics (in the case of MLLR) or full covariance mean and variance statistics (in the case of fMLLR). We also present experiments comparing the exact implementations of MLLR and fMLLR to approximate versions in which we approximate the covariances (or precisions) with their diagonal. The diagonal-precision approximation was also used in [2]; however, we show here that the diagonal-covariance approximation (previously used by us in [13]) works better.

2. MLLR

Maximum Likelihood Linear Regression (MLLR) [1] is a speaker adaptation technique in which the means of Gaussians in a speech recognition system are adapted so as to maximize the likelihood of the adaptation data for a particular speaker. The means are transformed with

$$\boldsymbol{\mu}^{(sm)} = \boldsymbol{W}^{(s)}\boldsymbol{\xi}^{(m)} \tag{1}$$

where $\boldsymbol{W}^{(s)} = \left[\boldsymbol{A}^{(s)}\boldsymbol{b}^{(s)}\right]$ is a matrix containing a square transform and a bias term and $\boldsymbol{\xi}^{(m)} = \left[\begin{array}{c}\boldsymbol{\mu}^{(m)}\\ 1\end{array}\right]$.

For diagonal systems, the MLLR matrix is estimated as follows. Let $c^{(sm)} = \sum_{t=1}^{T_s}\gamma^{(stm)}$ be the soft count of Gaussian m from the current speaker and let the vector $\mathcal{E}(\boldsymbol{x})^{(sm)} = \frac{\sum_{t=1}^{T_s}\gamma^{(stm)}\boldsymbol{x}^{(st)}}{\sum_{t=1}^{T_s}\gamma^{(stm)}}$ be the average of the features in frames which align to Gaussian m for speaker s, where $\gamma^{(stm)}$ are the Gaussian posteriors.

The part of the auxiliary function that changes with the current transform $\boldsymbol{W}$ is:

$$-0.5\sum_{m=1}^{M}c^{(sm)}\sum_{i=1}^{d}\frac{\mu_i^{(sm)} - \mathcal{E}(\boldsymbol{x})_i^{(sm)} + \mathcal{E}(\boldsymbol{x}\boldsymbol{x}^T)_{ii}^{(sm)}}{\sigma_i^{2\,(m)}} \tag{2}$$

where $\sigma_i^{2\,(m)}$ is the variance for dimension i of mixture m. This is equivalent to:

$$K - 0.5\sum_{m=1}^{M}c^{(sm)}\sum_{i=1}^{d}\frac{(\boldsymbol{w}_i^T\boldsymbol{\xi}^{(m)})^2 - 2(\boldsymbol{w}_i^T\boldsymbol{\xi}^{(m)})\mathcal{E}(\boldsymbol{x})^{(sm)}(d)}{\sigma_i^{2\,(m)}}, \tag{3}$$

where the column vector $\boldsymbol{w}_i$ is the transpose of the i'th row of $\boldsymbol{W}$. We can solve for each of the $\boldsymbol{w}_i$ separately. Let

$$\boldsymbol{k}_i = \sum_{m=1}^{M}\frac{c^{(sm)}\boldsymbol{\xi}^{(m)}\mathcal{E}(\boldsymbol{x})^{(sm)}(d)}{\sigma_i^{2\,(m)}} \tag{4}$$

$$\boldsymbol{G}_i = \sum_{m=1}^{M}\frac{c^{(sm)}\boldsymbol{\xi}^{(m)}\boldsymbol{\xi}^{(m)\,T}}{\sigma_i^{2\,(m)}}. \tag{5}$$

Then the part of the auxiliary function which depends on $\boldsymbol{w}_i$ is:

$$\boldsymbol{w}_i^T\boldsymbol{k}_i - 0.5\boldsymbol{w}_i^T\boldsymbol{G}_i\boldsymbol{w}_i. \tag{6}$$

This is maximized when

$$\boldsymbol{w}_i = \boldsymbol{G}_i^{-1}\boldsymbol{k}_i. \tag{7}$$

This can be estimated either in a memory efficient way by accumulating $\boldsymbol{k}_i$ and $\boldsymbol{G}_i$ directly from the data, or in a time efficient way by storing mean statistics and computing $\boldsymbol{k}_i$ and $\boldsymbol{G}_i$ from them.

3. fMLLR

fMLLR, also known as constrained MLLR [1], is a feature space transform where we transform the features with

$$\hat{\boldsymbol{x}}^{(t)} = \boldsymbol{W}^{(s)}\boldsymbol{\xi}^{(t)} \tag{8}$$

This work was funded by DARPA contract HR0011-06-2-0001

where again $\boldsymbol{W}^{(s)} = \left[\boldsymbol{A}^{(s)} b^{(s)}\right]$ contains the square matrix and the bias term, and $\boldsymbol{\xi}^{(t)} = \left[\begin{array}{c} \boldsymbol{x}^{(t)} \\ 1 \end{array}\right]$ is the extended feature vector for time t.

The auxiliary function equals the likelihood of the transformed data plus the log determinant $\log|\det(\boldsymbol{A})|$. The requirement for the determinant is most clear if we view fMLLR as a model space transform (constrained MLLR), where $\boldsymbol{A}$ becomes a transform on the variances ($\boldsymbol{A}^T \boldsymbol{\Sigma}^{(m)} \boldsymbol{A}$).

The part of the auxiliary function excluding the determinant equals

$$-0.5 \sum_{m=1}^{M} c^{(sm)} \mathcal{E} \left(\sum_{i=1}^{d} \frac{(\mu_i^{(m)} - \boldsymbol{w}_i^T \boldsymbol{\xi}^{(t)})^2}{\sigma_i^{2(m)}} \right)^{(sm)} \tag{9}$$

where $\mathcal{E}(\cdot)^{(sm)}$ is the average value for speaker s and Gaussian m. This equals

$$-0.5 \sum_{m=1}^{M} c^{(sm)} \sum_{i=1}^{d} \frac{\mu_i^{(m)^2} - 2\mu_i^{(m)} \boldsymbol{w}_i^T \mathcal{E}(\boldsymbol{\xi})^{(sm)} + \boldsymbol{w}_i^T \mathcal{E}(\boldsymbol{\xi}\boldsymbol{\xi}^T)^{(sm)} \boldsymbol{w}_i}{\sigma_i^{2(m)}}. \tag{10}$$

The quantities $\mathcal{E}(\boldsymbol{\xi})^{(sm)}$ and $\mathcal{E}(\boldsymbol{\xi}\boldsymbol{\xi}^T)^{(sm)}$ can be derived from the mean and full variance statistics from the current speaker: $\mathcal{E}(\boldsymbol{\xi})^{(sm)} = \left[\begin{array}{c} \mathcal{E}(\boldsymbol{x})^{(sm)} \\ 1 \end{array}\right]$ and $\mathcal{E}(\boldsymbol{\xi}\boldsymbol{\xi}^T)^{(sm)} = \left[\begin{array}{cc} \mathcal{E}(\boldsymbol{x}\boldsymbol{x}^T)^{(sm)} & \mathcal{E}(\boldsymbol{x})^{(sm)} \\ \mathcal{E}(\boldsymbol{x})^{(sm)^T} & 1 \end{array}\right]$. Again, the linear and quadratic terms in $\boldsymbol{w}_i$ are gathered as $\boldsymbol{k}_i$ and $\boldsymbol{G}_i$:

$$\boldsymbol{k}_i = \sum_{m=1}^{M} \frac{c^{(sm)} \mu_i^{(m)} \mathcal{E}(\boldsymbol{\xi})^{(sm)}}{\sigma_i^{2(m)}} \tag{11}$$

$$\boldsymbol{G}_i = \sum_{m=1}^{M} \frac{c^{(sm)} \mathcal{E}(\boldsymbol{\xi}\boldsymbol{\xi}^T)^{(sm)}}{\sigma_i^{2(m)}}. \tag{12}$$

The auxiliary function is now:

$$\log(|\det(\boldsymbol{A})|) - \sum_{i=1}^{d} \boldsymbol{w}_i^T \boldsymbol{k}_i - 0.5 \boldsymbol{w}_i^T \boldsymbol{G}_i \boldsymbol{w}_i. \tag{13}$$

3.1. Row-by-row iterative fMLLR

The transform $\boldsymbol{W}$ can estimated through maximization of Equation 13 using an iterative update described in [1]. It uses the fact that the determinant of a matrix equals the dot product of any given row of the matrix with the corresponding row of cofactors. If we are updating the i'th row of the transform then we let the column vector $\boldsymbol{c}_i$ equal the transpose of the i'th row of the cofactors of $\boldsymbol{A}$, extended with a zero in the last dimension to make a vector of size $d+1$, so that the determinant $\det(\boldsymbol{A})$ can be represented as a function of $\boldsymbol{w}_i$ by $\boldsymbol{w}_i^T \boldsymbol{c}_i$. Now we can optimize the function

$$\log(|\boldsymbol{w}_i^T \boldsymbol{c}_i|) + \boldsymbol{w}_i^T \boldsymbol{k}_i - 0.5 \boldsymbol{w}_i^T \boldsymbol{G}_i \boldsymbol{w}_i. \tag{14}$$

Since the matrix of cofactors of a matrix $\boldsymbol{M}$ equals $\det(\boldsymbol{M}) \boldsymbol{M}^{-1^T}$, and the value of $\boldsymbol{w}_i$ that maximizes the expression in Equation 14 is not affected by any constant factor in $\boldsymbol{c}_i$, we could also let $\boldsymbol{c}_i$ equal the i'th column of the current value of $\boldsymbol{A}^{-1}$ (extended with a zero to make a $d+1$ dimensional column vector) and thus avoid any numerical problems that could occur if the determinant is very large or small. If we let $f = \boldsymbol{w}_i^T \boldsymbol{c}_i$, the solution to Equation 14 is $\boldsymbol{w}_i = \boldsymbol{G}_i^{-1}(\boldsymbol{c}_i / f + \boldsymbol{k}_i)$. Substituting the solution for $\boldsymbol{w}_i$ into the expression for f and rearranging, we get $f^2 - f \boldsymbol{c}_i^T \boldsymbol{G}_i^{-1} \boldsymbol{k}_i - \boldsymbol{c}_i^T \boldsymbol{G}_i^{-1} \boldsymbol{c}_i = 0$, which we can solve for f, so the final answer is:

$$\begin{aligned}
[a, b, c] &:= \left[1, \quad -\boldsymbol{c}_i^T \boldsymbol{G}_i^{-1} \boldsymbol{k}_i, \quad -\boldsymbol{c}_i^T \boldsymbol{G}_i^{-1} \boldsymbol{c}_i\right] \\
f &:= \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \\
\boldsymbol{w}_i &:= \boldsymbol{G}_i^{-1} \left(\frac{\boldsymbol{c}_i}{f} + \boldsymbol{k}_i\right).
\end{aligned}$$

We can test the value of the auxiliary function in Equation 14 to see which solution to the quadratic equation is the best one. This is an iterative procedure, so starting from the baseline transform where $\boldsymbol{A} = \boldsymbol{I}, \boldsymbol{b} = \boldsymbol{0}$ we apply the update to each row in turn and continue iterating until the change in the auxiliary function is small, or for (say) 20 iterations.

4. Full covariance MLLR

4.1. Baseline approaches

The full covariance case in MLLR has a simple solution, but it is not a practical one ([1], see footnote page 3). The part of the auxiliary function that depends on the transformation matrix is

$$-0.5 \sum_{m=1}^{M} c^{(sm)} (\boldsymbol{W}\boldsymbol{\xi}^{(m)} - \mathcal{E}(\boldsymbol{x})^{(sm)})^T \boldsymbol{\Sigma}^{(m)^{-1}} (\boldsymbol{W}\boldsymbol{\xi}^{(m)} - \mathcal{E}(\boldsymbol{x})^{(sm)}). \tag{15}$$

This is a quadratic function in the elements of $\boldsymbol{W}$. To solve it in closed form we would have to accumulate a matrix of size d(d+1) by d(d+1) and invert it. This problem would take time $O(d^6)$ which theoretically for $d = 40$ might take about 4 seconds at 1GFLOP; however memory access time would slow this down. In addition, although the matrix should be invertible, we might encounter numerical problems inverting a matrix in this very high dimension [2]. Also the accumulation of the large matrix would take time $O(d^4)$ per Gaussian accessed (assuming we did this from mean statistics) which is slower than the $O(d^3)$ time needed to compute the matrices $\boldsymbol{G}_i$ which currently dominates the computation.

We can also compare with the approach taken in [2, 5] in which a general purpose optimization package was used to compute the MLLR and fMLLR transforms. It is difficult to compare with that approach without more details; however, the current approach does have the advantage of being explicitly spelled out and is free of the requirement to incorporate third-party software.

4.2. Proposed approach

The proposed approach to full-covariance MLLR computation has around the same speed as the baseline MLLR computation (assuming we are using the time-efficient version from stored mean statistics), and is numerically stable. It is an iterative approach in which on each iteration we calculate the gradient of the auxiliary function w.r.t. $\boldsymbol{W}$. We assume that the second gradient is the same that it would be in the diagonal case (represented by the matrices $\boldsymbol{G}_i$), and compute the updated value of $\boldsymbol{W}$. We then measure the auxiliary function to see if it has improved. If it has, we continue to the next iteration; if not, we reduce the learning rate by a factor of 2 by doubling all the matrices $\boldsymbol{G}_i$, and continue. When the change in auxiliary function is small (or after, say, 30 iterations) we stop.

In detail, the method as follows. From stored speaker-specific mean statistics, compute: for $i = 1 \dots d$,

$$\boldsymbol{G}_i = \sum_{m=1}^{M} \frac{c^{(sm)} \boldsymbol{\xi}^{(m)} \boldsymbol{\xi}^{(m)^T}}{\Sigma_{ii}^{(m)}}. \tag{16}$$

Set the transformation matrix $\boldsymbol{W} = [\boldsymbol{Ab}]$ to its initial value $[\boldsymbol{I0}]$. Then, on each iteration, compute the d by $(d+1)$ matrix $\boldsymbol{L}$ which is the gradient w.r.t the auxiliary function:

$$\boldsymbol{L} = \sum_{m=1}^{M} c^{(sm)} (\boldsymbol{\Sigma}^{(m)^{-1}} (\boldsymbol{W}\boldsymbol{\xi}^{(m)} - \mathcal{E}(\boldsymbol{x})^{(sm)}(d))) \boldsymbol{\xi}^{(m)^T}. \tag{17}$$

If $\boldsymbol{l}_i$ is the column vector which equals the transpose of the i'th row of $\boldsymbol{L}$, we can compute the vectors $\boldsymbol{k}_i$ which would give the MLLR auxiliary function $\boldsymbol{w}_i^T \boldsymbol{k}_i - 0.5 \boldsymbol{w}_i^T \boldsymbol{G}_i \boldsymbol{w}_i$ a differential w.r.t. $\boldsymbol{k}_i$ equal to $\boldsymbol{l}_i$:

$$\boldsymbol{k}_i = \boldsymbol{l}_i + \boldsymbol{G}_i \boldsymbol{w}_i. \tag{18}$$

We then do the normal MLLR update,

$$\boldsymbol{w}_i = \boldsymbol{G}_i^{-1} \boldsymbol{k}_i. \tag{19}$$

Before and after the update we compute the partial auxiliary function given by:

$$-0.5 \sum_{m=1}^{M} c^{(sm)} (\boldsymbol{W}\boldsymbol{\xi}^{(m)} - \mathcal{E}(\boldsymbol{x})^{(sm)}(d)) \boldsymbol{\Sigma}^{(m)^{-1}} (\boldsymbol{W}\boldsymbol{\xi}^{(m)} - \mathcal{E}(\boldsymbol{x})^{(sm)}(d)). \tag{20}$$

If this has increased we continue to the next iteration; if it has not, we decrease the learning rate by doubling $\boldsymbol{G}_i$. The increased $\boldsymbol{G}_i$ are also used for subsequent iterations. We then set $\boldsymbol{W}$ to its previous value, recompute $\boldsymbol{k}_i$ and update $\boldsymbol{W}$, and retest, continuing until we see an increase. If the change is very small or after a specified number of iterations, we stop.

Note that the extra computation that must be done on each iteration takes $O(d^2)$ time per Gaussian, compared with the $O(d^3)$ time per Gaussian used to compute the matrices $\boldsymbol{G}_i$ in both this and the standard MLLR update. Therefore if the number of iterations is less than the dimension d (which it typically is) the time taken is of the same order as the time taken for the standard MLLR update.

5. Full covariance fMLLR

The full covariance fMLLR update works on the same principle as the full covariance MLLR update. We assume that the second gradient in the data-dependent part of the auxiliary function (i.e., excluding the determinant) is the same as in the diagonal case, and accumulate the matrices $\boldsymbol{G}_i$ the same as in the diagonal case. Then on each iteration of the update we accumulate a d by $d+1$ matrix $\boldsymbol{L}$ which equals the gradient of the data-dependent part of the auxiliary function (i.e., excluding the determinant) w.r.t. the fMLLR transformation, and do a normal fMLLR update using $\boldsymbol{k}_i$ vectors derived from $\boldsymbol{L}$ and $\boldsymbol{G}_i$. We measure the auxiliary function on each iteration and if it fails to increase we double the matrices $\boldsymbol{G}_i$ (to halve the learning rate) and retry.

The matrices $\boldsymbol{G}_i$ are defined as for fMLLR, in Equation 12. The current gradient $\boldsymbol{L}$ is:

$$\boldsymbol{L} = \sum_{m=1}^{M} c^{(sm)} \boldsymbol{\Sigma}^{(m)^{-1}} \mathcal{E}\left((\boldsymbol{\mu}^{(m)} - \boldsymbol{W}^{(s)} \boldsymbol{\xi}) \boldsymbol{\xi}^T \right)^{(sm)}, \tag{21}$$

where $\boldsymbol{\xi}$ is the extended feature vector $\begin{bmatrix} \boldsymbol{x} \\ 1 \end{bmatrix}$. and $\mathcal{E}(\cdot)^{(sm)}$ means the average value of some quantity for frames aligned to the Gaussian m for speaker s. Expressed in terms of the feature statistics, and dropping the superscript ${}^{(sm)}$ in $\mathcal{E}(\cdot)^{(sm)}$, this equals:

$$\sum_{m=1}^{M} c^{(sm)} \boldsymbol{\Sigma}^{(m)^{-1}} \left(\boldsymbol{\mu}^{(m)} \left[\mathcal{E}(\boldsymbol{x})^T \, 1 \right] - \boldsymbol{W}^{(s)^T} \begin{bmatrix} \mathcal{E}(\boldsymbol{x}\boldsymbol{x}^T) & \mathcal{E}(\boldsymbol{x}) \\ \mathcal{E}(\boldsymbol{x})^T & 1 \end{bmatrix} \right). \tag{22}$$

Thus, this implementation of full covariance fMLLR requires us to store full covariance statistics from the data. As for MLLR, on each iteration we set for each column vector $\boldsymbol{l}_i$ corresponding to a row of $\boldsymbol{L}$,

$$\boldsymbol{k}_i = \boldsymbol{l}_i + \boldsymbol{G}_i \boldsymbol{w}_i, \tag{23}$$

and then use the $\boldsymbol{k}_i$ and $\boldsymbol{G}_i$ to estimate the fMLLR matrix $\boldsymbol{W}$ using the iterative row-by-row update as in Section 3.1. As before, we measure the auxiliary function from the data and if it has decreased we double $\boldsymbol{G}_i$, recompute the $\boldsymbol{k}_i$ from the current $\boldsymbol{L}$ and recompute the fMLLR transform $\boldsymbol{W}$.

The part of the auxiliary function relating to the fMLLR matrix, which we must measure to determine convergence, is (dropping the superscript ${}^{(sm)}$,

$$-0.5 \sum_{m=1}^{M} c^{(sm)} \left(tr \left(\boldsymbol{\Sigma}^{(m)} \boldsymbol{W}^{(s)} \begin{bmatrix} \mathcal{E}(\boldsymbol{x}\boldsymbol{x}^T) & \mathcal{E}(\boldsymbol{x}) \\ \mathcal{E}(\boldsymbol{x})^T & 1 \end{bmatrix} \boldsymbol{W}^{(s)^T} \right) \right.$$
$$\left. -2 \, \boldsymbol{\mu}^{(m)^T} \boldsymbol{\Sigma}^{(m)} \boldsymbol{W}^{(s)} \begin{bmatrix} \mathcal{E}(\boldsymbol{x}) \\ 1 \end{bmatrix} + \boldsymbol{\mu}^{(m)^T} \boldsymbol{\Sigma}^{(m)} \boldsymbol{\mu}^{(m)} - 2 \log |\det \boldsymbol{A}| \right).$$

6. Approximate MLLR and fMLLR

In additional to exact implementations of MLLR and fMLLR for the full covariance case, we also report experiments with two quite similar approximations. One, which we will call diagonal-precision MLLR and fMLLR (as used for MLLR in [2]), is to perform the diagonal MLLR computation while pretending that the full precision matrix equals a diagonal matrix with the same diagonal elements as the real precision matrix. The other, diagonal-covariance MLLR and fMLLR (as used by us in [13]), assumes that the covariance matrix is diagonal and has the same diagonal elements as the real covariance matrix.

7. Experimental setup

We report experiments on the Mandarin section of the RT'04 test set from the EARS program. The test set is 1 hour long after segmentation. The training data consists of 30 hours of hub4 Mandarin training data, 67.7 hours extracted from TDT-4 data (mainland Chinese only), 42.8h from a new LDC-released database (LDC2005E80) and 50 hours from a private collection of satellite data. The baseline system (similar to that described in [12]) has 6000 cross-word context-dependent states with ±2 phones of context and 100000 Gaussians. The basic features are PLP projected with LDA and MLLT (global semi-tied covariance). Speaker adaptation includes cepstral mean and variance normalization, VTLN, fMLLR and MLLR. The models are trained on VTLN-warped and fMLLR-transformed data.

We report experiments on a baseline diagonal system with 100000 Gaussians, and a full-covariance system with 50000 Gaussians trained for two iterations with full-covariance Gaussians after training a diagonal system. The off-diagonal elements of the full-covariance Gaussians are smoothed as proposed in [6] by multiplying them by $c^{(m)}/(c^{(m)} + \tau)$ where $c^{(m)}$ is the count for the Gaussian and τ is a constant set to 100.

In addition to the standard full-covariance models, in order to have more than one experimental condition we also report results with an extended version of MLLR (XMLLR), described in a companion paper [11]. The technique is very similar to ESAT [8], and involves using mean vectors that are of a higher dimension than the

Speaker adaptation			
None	fMLLR	fMLLR+MLLR	fMLLR+MLLR(rtree)
20.4%	17.9%	17.7%	17.3%

Table 1: Baseline (diagonal, 100k Gaussians).

	Speaker adaptation		
Computation	None	fMLLR	fMLLR+MLLR
Full	19.2%	16.8%	16.6%
Diag-cov	19.2%	16.7%	16.5%
Diag-prec	19.2%	17.0%	16.8%

Table 2: Full covariance ($\tau = 100$), 100k Gaussians

feature vectors and projecting down in a speaker-specific fashion. The MLLR computation is exactly analogous the normal computation, only with different dimensions of some of the quantities involved. For experiments reported here the dimension of means used in XMLLR is 80, compared to a feature dimension of 40.

8. Experimental results

Table 1 shows the baseline performance of the system with and without fMPE [7] and MPE [9]. The last column shows the extra 0.4% to be gained from regression tree MLLR, which has not been implemented in the full covariance case; however, there is no reason why it should not work.

Table 2 shows the effect of the different kinds of fMLLR and MLLR computation (exact; pretending the variances are diagonal; pretending the precisions are diagonal) on a full covariance system. The differences are quite small, so in order to get a better idea whether there are any consistent differences we also test on two different setups. Table 3 is the result on a full covariance system with no smoothing of the variances ($\tau = 0$) and decoding without word-boundary information (the result of an error). The third setup, Table 4, is with XMLLR [11] in which the mean vectors have a larger dimension than the feature vectors.

Looking over all three setups, we find that in general the exact computation is best, and that the computation where we pretend the precisions are diagonal (as done for MLLR in [2]) is always the worst. This effect seems to appear at the fMLLR level. We were told [3] that the approach of pretending the precisions are diagonal was attempted for the fMLLR case in work reported in [2] but led to poor results. We do notice that in the diagonal-covariance case the log determinant of the fMLLR matrix $\mathbf{A}^{(\mathbf{s})}$ is somewhat larger than the exact case (e.g., a difference of 2) but in the diagonal-precision case the log determinant is much smaller (e.g. a difference of -6). It may be possible to devise some approach that overcomes these systematic biases.

	Speaker adaptation	
Computation	fMLLR	fMLLR+MLLR
Full	18.6%	18.2%
Diag-cov	18.5%	18.4%
Diag-prec	18.7%	18.4%

Table 3: Full covariance ($\tau = 0$), no word-boundary, 100k Gaussians

	Speaker adaptation/training iteration			
	fMLLR		fMLLR+MLLR	
Computation	1	2	1	2
Full	16.9%	16.8%	16.4%	16.1%
Diag-cov	16.9%	16.9%	16.4%	16.3%
Diag-prec	17.3%	17.1%	16.8%	16.6%

Table 4: Full covariance ($\tau = 100$), XMLLR ($D = 80$), 50k Gaussians, no word-boundary, 100k Gaussians

9. Conclusions

We have presented a reasonably efficient exact method to compute fMLLR and MLLR adaptation matrices for full covariance Gaussians, and have compared it with some approximate approaches. We have demonstrated experimentally that our exact method gives reasonable improvements, and have shown that we can generally get most of the improvement by using the diagonal of the Gaussian covariances to reduce it to the diagonal case.

10. References

[1] M.J.F. Gales, “Maximum Likelihood Linear Transformations for HMM-based Speech Recognition.,” *Computer Speech and Language Volume 12*, 1998.

[2] K.C. Sim & M.J.F. Gales, “Adaptation of Precision Matrix Models on Large Vocabulary Continuous Speech Recognition”, *ICASSP*, 2005.

[3] Personal communication from Mark Gales, March 2006.

[4] J. Huang, V. Goel, R. Gopinath, B. Kingsbury, P. Olsen & K. Visweswariah, “Large vocabulary conversational speech recoginition with the extended maximum likelihood linear trnasformation (EMLLT) modcel,” *ICSLP,* 2002.

[5] S. Axelrod, V. Goel, B. Kingsbury, K. Visweswariah & R.A. Gopinath, “Large vocabulary conversational speech recognition with a subspace constraint on inverse covariance matrices,” *Eurospeech,* 2003.

[6] D. Povey, “Discriminative Training for Large Vocabulary. Speech Recognition.” PhD thesis, Cambridge University,. 2003.

[7] D. Povey, B. Kingsbury, L. Mangu, G. Saon, H. Soltau, G. Zweig, “Improvements to fMPE for Discriminative Training of Features,” *Interspeech,* 2005.

[8] M.J.F. Gales, “Multiple-cluster adaptive training schemes,” *ICASSP*, 2001.

[9] D. Povey and P. C. Woodland, “Minimum Phone Error and I-smoothing for Improved Discriminative Training,” *ICASSP*, 2002.

[10] D. Povey, “SPAM and full covariance for speech recognition,” submitted to: *Interspeech,* 2006.

[11] D. Povey, “Extended MLLR for improved speaker adaptation,” submitted to: *Interspeech,* 2006.

[12] H. Soltau, B. Kingsbury, L. Mangu, D. Povey, G. Saon & G. Zweig, “The IBM 2004 Conversational Telephony System for Rich Transcription,” *ICASSP*, 2005.

[13] “Acoustic modeling with full-covariance Gaussians,” G. Saon, B. Kingsbury, L. Mangu, D. Povey, H. Soltau & G. Zweig In *EARS STT Worshop*, 2004.