SPAM and full covariance for speech recognition.

Daniel Povey

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

Abstract

The Subspace Precision and Mean model (SPAM) is a way of representing Gaussian precision and mean values in a reduced dimension. This paper presents some large vocabulary experiments with SPAM and introduces an efficient way to optimize the SPAM basis. We present experiments comparing SPAM, diagonal covariance and full covariance models on a large vocabulary task. We also give explicit formulae for an implementation of SPAM.

1. Introduction

Most speech recognition systems use mixtures of diagonal Gaussians, but in in recent years, there have been a number of attempts to improve variance modeling. These include semi-tied covariances [1], in which a number of full-rank matrices each shared among groups of Gaussians, together with diagonal variance matrices per Gaussian; and EMLLT [2], where the full covariance is represented in a reduced dimension as a weighted sum of rank-one matrices. A technique that appears to outperform both of these is the Subspace Precision and Mean model (SPAM) [3], introduced at IBM and used successfully elsewhere [5]. In SPAM, each covariance is represented as a weighted sum over D globally shared full-rank matrices, where D is not necessarily the same as the feature dimension d.

In this paper, we provide explicit formulae for the initialization and optimization of the SPAM basis and the per-Gaussian coefficients, without the requirement for a numerical optimization package as originally used [3]. We also report experiments on a large vocabulary task which shows that SPAM can give about as good performance as a full-covariance model while requiring computation comparable to a diagonal-covariance model.

2. SPAM

SPAM [3] is a way of representing precision matrices in a reduced dimension, so that for Gaussian j,

$$\boldsymbol{P}_j = \sum_{k=1}^{D} \lambda_j^k \boldsymbol{S}_k \tag{1}$$

where λ_j^k are coefficients per Gaussian and $\boldsymbol{S}_k$ are shared basis matrices. D can be less than, equal to or greater than the dimension d of the features.

3. SPAM basis computation

For initialization and optimization of the SPAM basis, for efficiency we use a subset of the Gaussians, selecting the d^2 Gaussians with the largest counts. We use these Gaussians without any weighting by count for basis optimization, i.e. setting their counts c_j to the same value. It is not clear whether this is the best approach.

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

3.1. Initial approximation

The first step in optimizing the SPAM auxiliary function is to get a good initial approximation for the basis matrices $\boldsymbol{S}_k$. As in [3], this is done by means of a quadratic approximation which reduces the problem to a PCA problem in dimension $d(d+1)/2$ where d is the feature dimension. The optimization of the basis requires full covariance statistics. The auxiliary function F is a sum over Gaussians j:

$$F = \sum_{j=1}^{J} -0.5 c_j (tr(\boldsymbol{P}_j \boldsymbol{\Sigma}_j)) + 0.5 \log\det(\boldsymbol{P}_j) \tag{2}$$

This function has its maximum when for each j, $\boldsymbol{P}_j = \boldsymbol{\Sigma}_j^{-1}$; the second gradient arises from the log determinant term $0.5 \log\det(\boldsymbol{P}_j)$. If we change $\boldsymbol{P}_j$ by a small amount $\boldsymbol{\Delta}_j$, the auxiliary function will change by $-0.25 c_j tr(\boldsymbol{\Delta}_j \boldsymbol{\Sigma}_j \boldsymbol{\Delta}_j \boldsymbol{\Sigma}_j)$. If $\boldsymbol{\Sigma}_j$ happened to be a multiple of the unit matrix $f_j I$, this function would equal $-0.25 c_j f^2 vec(\boldsymbol{\Delta}_j)^T vec(\boldsymbol{\Delta}_j)$ where $vec(M)$ means appending the rows of a matrix to form a vector. This is the key to our PCA method of initializing the SPAM basis, and only differs from the one in [3] by the constant factors f_j.

3.1.1. Normalization

Using $\boldsymbol{\Sigma}_{avg} = \frac{\sum_{j=1}^{J} c_j \boldsymbol{\Sigma}_j}{\sum_{j=1}^{J} c_j}$, we compute a symmetric normalizing matrix $\boldsymbol{N} = \boldsymbol{\Sigma}_{avg}^{-1/2}$ Then for all the variances we set

$$\boldsymbol{\Sigma}'_j = \boldsymbol{N} \boldsymbol{\Sigma}_j \boldsymbol{N} \tag{3}$$

We do all computations with the normalized variances $\boldsymbol{\Sigma}'_j$ and then at the end after computing the normalized basis matrices S'_k and the coefficients λ_j^k we can do the reverse normalization

$$\boldsymbol{S}_k = \boldsymbol{N}^{-1} \boldsymbol{S}'_k \boldsymbol{N}^{-1}. \tag{4}$$

The optimization uses the projected-space precisions $\boldsymbol{P}'_j = \sum_{k=1}^{D} \lambda_j^k \boldsymbol{S}'_k$.

To reduce the computation we vectorize the matrices in a special way taking advantage of the fact that they are symmetric. Let $vec'(\boldsymbol{A})$ be a splicing together of the lower triangle of $\boldsymbol{A}$ where all the off-diagonal elements are first scaled by $\sqrt{2}$; it returns a vector of size $d * (d+1)/2$ for a d by d matrix. This preserves the distance measure and can be thought of as a rotation in the space of size d^2, followed by discarding dimensions that are always zero for symmetric matrices. Then we can define the opposite function $mat'(v)$ which splices together a vector into a lower triangular matrix, multiplies the off diagonal elements by $1/\sqrt{2}$ and copies the lower triangle to the upper triangle.

3.1.2. Principal components analysis

The computation of the initial basis involves computing the $d(d+1)/2$ by $d(d+1)/2$ scatter matrix

$$\boldsymbol{X} = \frac{\sum_{j=1}^{J} c_j f_j^2 vec'(\boldsymbol{\Sigma}'_j) vec'(\boldsymbol{\Sigma}'_j)^T}{\sum_{j=1}^{J} c_j} \tag{5}$$

where $f_j = \frac{tr(\boldsymbol{\Sigma}'_j)}{d}$. The $k'th$ basis matrix $\boldsymbol{S}'_k$ will now equal $mat'(v_k)$, if v_k is the k'th eigenvector of $\boldsymbol{X}$. Note that the basis elements $\boldsymbol{S}'_k$ are unit and orthogonal (this is easiest to visualize in their vectorized form). This will be useful when optimizing the coefficients. For convenience in optimizing the coefficients we make a modification to Equation 5 that ensures that the first basis matrix is positive definite and approximately equals the average of $\boldsymbol{\Sigma}'_j$:

$$\boldsymbol{X}' = \boldsymbol{X} + 1000 vec'(\boldsymbol{I}) vec'(\boldsymbol{I})^T. \tag{6}$$

We use the principal components of $\boldsymbol{X}'$.

3.2. Iterative optimization

Optimization of the SPAM basis is done alternately with the optimization of the coefficients (which is described in Section 4) The approach is to find the gradient of the auxiliary function w.r.t. each basis matrix $\boldsymbol{S}'_j$, given fixed coefficients λ_j^k, and find an approximation to the second gradient which allows us to find a reasonable update direction; we then calculate the optimal step size in that direction based on the exact second gradient, which can be computed exactly in an efficient way. But some of the precisions $\boldsymbol{P}'_j$ may no longer be positive definite with the new basis. Rather than limit the update to very small step sizes to prevent this, we recalculate the coefficients and check whether (with the updated coefficients) the auxiliary function has improved. If not, we halve the step size and try again. However, in practice this has never been observed to be necessary. This procedure converges in ten or so iterations. After each update we orthogonalize and normalize the basis matrices (viewed as vectors as described above).

On each iteration, we first calculate the gradient of the auxiliary function F (Equation 2) w.r.t. each matrix $\boldsymbol{S}'_k$,

$$\frac{\partial F}{\partial \boldsymbol{S}'_k} = 0.5 \sum_{j=1}^{J} c_j ({\boldsymbol{P'}_j}^{-1} - \boldsymbol{\Sigma'}_j). \tag{7}$$

In a Taylor expansion of the auxiliary function, the quadratic term arises from expressions of the form $-0.25 c_j tr(\boldsymbol{P}_j^{-1} \boldsymbol{\Delta}_j \boldsymbol{P}_j^{-1} \boldsymbol{\Delta}_j)$, if $\boldsymbol{\Delta}_j$ is the change in the precision $\boldsymbol{P}'_j$. If the changes in the basis matrices $\boldsymbol{S}'_k$ are $\boldsymbol{D}_k$, the quadratic term in the expansion can be expressed as a function of the matrices $\boldsymbol{D}_k$ as:

$$-0.25 \sum_{j=1}^{J} \sum_{k=1}^{D} \sum_{l=1}^{D} c_j \lambda_j^k \lambda_j^l tr({\boldsymbol{P'}_j}^{-1} \boldsymbol{D}_k {\boldsymbol{P'}_j}^{-1} D_l). \tag{8}$$

This introduces dependencies between all matrix elements of all $\boldsymbol{S}_j$, which makes the problem intractable. However, we can put to good use the fact that the typical variance is close to the unit matrix, and approximate $\boldsymbol{P}_j^{-1}$ as $f_j \boldsymbol{I}$, where $f_j = \frac{tr({\boldsymbol{P'}_j}^{-1}\boldsymbol{I})}{d}$. We can also assume that since the SPAM basis was initialized with PCA, the coefficients λ_j^k should be fairly uncorrelated between different dimensions k; assuming that all the variances are all about equal, any cross terms ($k \neq l$) in in Equation 8 will be about zero. The simplified quadratic term is now:

$$-0.25 \sum_{k=1}^{D} c_j {\lambda_j^k}^2 f_j^2 tr(\boldsymbol{D}_k \boldsymbol{D}_k) \tag{9}$$

This is just a constant times a euclidean distance in the vectorized form of each matrix ${\boldsymbol{S'}}_k$, and the update rule becomes gradient descent with a different speed $1/F_k$ for each value of k, where the factors F_k are computed as $F_k = \sum_{j=1}^{J} 0.5 c_j {\lambda_j^k}^2 f_j^2$ (this includes a factor of 2 because we want the second gradient, not the coefficient of the quadratic term), and using the expression for the gradients in Equation 7 the proposed changes to $\boldsymbol{S}_k$ become

$$\boldsymbol{D}_k = \frac{\sum_{j=1}^{J} c_j ({\boldsymbol{P'}_j}^{-1} - \boldsymbol{\Sigma'}_j)}{F_k}. \tag{10}$$

However, this update amount may not converge because we made some assumptions to get this rule. Instead the change will be $c\boldsymbol{D}_k$ for a shared constant c, where we work out c for optimum improvement as follows. The first-order term in c in the auxiliary function is $c \sum_{k=1}^{K} \boldsymbol{D}_k \frac{\partial F}{\partial \boldsymbol{S}_k}$, with $\frac{\partial F}{\partial \boldsymbol{S}_k}$ given in Equation 7. The second order term is $-c^2 \sum_{j=1}^{J} 0.25 c_j tr({\boldsymbol{P'}_j}^{-1} \boldsymbol{\Delta}_j {\boldsymbol{P'}_j}^{-1} \boldsymbol{\Delta}_j)$, where $\boldsymbol{\Delta}_j = \sum_{k=1}^{D} \lambda_j^k \boldsymbol{D}_k$. The optimal value given the full quadratic approximation to the auxiliary function is:

$$c = \frac{\sum_{k=1}^{K} \boldsymbol{D}_k \frac{\partial F}{\partial \boldsymbol{S'}_k}}{\sum_{j=1}^{J} 0.5 c_j tr({\boldsymbol{P'}_j}^{-1} \boldsymbol{\Delta}_j {\boldsymbol{P'}_j}^{-1} \boldsymbol{\Delta}_j)}. \tag{11}$$

We can now update the basis by setting $\boldsymbol{S}'_k := \boldsymbol{S}'_k + c\boldsymbol{D}_k$ and re-orthogonalize and normalize it by setting, for $k = 1 \ldots D$, $\boldsymbol{S}'_k := norm(\boldsymbol{S}'_k - \sum_{l=1}^{k-1} \boldsymbol{S}'_l tr(\boldsymbol{S}'_l \boldsymbol{S}'_k))$, where $norm(\boldsymbol{A}) = \boldsymbol{A}/\sqrt{tr(\boldsymbol{A}\boldsymbol{A})}$, i.e. ensuring that the vectorized form of the matrix has unit length and that they are all orthogonal.

After each update of the basis matrices we re-optimize the coefficients λ_j^k. For efficiency we start the optimization from the previously optimized values λ_j^k for any Gaussian j for which the old λ_j^k gives a positive definite matrix with the new basis. After optimizing the coefficients we check that the auxiliary function has improved compared to its value before optimizing the basis; if it has not, as noted above, we could reduce the update amount by half and try again but this does not happen in practice.

4. Coefficients computation

Computing the coefficients λ_j^k is the most computationally expensive part of the procedure and for optimizing all the coefficients in the system (as opposed to the d^2 largest-count Gaussians used to optimize the basis) we parallelize the computation.

4.1. Initial estimate of coefficients

For each Gaussian we first obtain an initial estimate of the coefficients. Let the vector of coefficients λ_j^k for some j be $\boldsymbol{l}_j$. This first step relies on the unit, orthogonal nature of the basis. Let $\boldsymbol{M}$ be a k by $d(d+1)/2$ matrix where each row

$$\boldsymbol{m}_k = vec'(\boldsymbol{S}'_k)^T. \tag{12}$$

The initial estimate of the coeffiecient vector is $\boldsymbol{l}_j := \boldsymbol{M} vec'({\boldsymbol{\Sigma'}_j}^{-1})$. If with these coefficients, $\boldsymbol{P'}_j$ is not positive definite (as will occasionally happen), we must find some other coefficients that give a positive definite precision matrix and start with them intead. If the first basis matrix $\boldsymbol{S}_1$ is positive definite (as it will definitely be if this is the first iteration of optimizing the SPAM basis and this is the initial estimate obtained as in Section 3.1) we do this by setting to zero all but the first element of $\boldsymbol{l}_j$. If $\boldsymbol{S}_1$ is not positive definite (and this has not been observed in practice but it is a theoretical possibilitiy) we can find some other set of coefficients $\boldsymbol{l}_{j'}$ from some other Gaussian j', as optimized on the previous iteration, that gives a positive definite matrix with the current basis; and set $\boldsymbol{l}_j$ to that.

4.2. Iterative update of coefficients

The iterative part of the coefficients optimization approach relies on the fact that the basis is unit and orthogonal and that the average variance in our projected space is the unit matrix (so hopefully all variances are close to the unit matrix). When optimizing the auxiliary function

$$F(\boldsymbol{l}_j) = 0.5 \log\det(\boldsymbol{P}'_j) - 0.5 tr(\boldsymbol{P}'_j \boldsymbol{\Sigma}'_j) \quad (13)$$

for symmetric $\boldsymbol{P}'_j$ and $\boldsymbol{\Sigma}'_j$, the second order term in a quadratic approximation to the auxiliary function around a current value $\boldsymbol{Q}'_j$ (so $\boldsymbol{P}'_j = \boldsymbol{Q}'_j + \boldsymbol{\Delta}_j$) would be:

$$-0.25 tr(\boldsymbol{\Delta}_j \boldsymbol{Q}'^{-1}_j \boldsymbol{\Delta}_j \boldsymbol{Q}'^{-1}_j). \quad (14)$$

Since we have projected the feature space so that most of the variances are close to unit, the variance $\boldsymbol{Q}'^{-1}_j$ will be similar to the unit matrix. This makes the second order term approximately equal to $-0.25 tr(\boldsymbol{\Delta}_j \boldsymbol{\Delta}_j)$ which equals $-0.25 vec'(\boldsymbol{\Delta}_j)^T vec'(\boldsymbol{\Delta}_j)$. Thus means that we can do simple gradient descent in the vectorized space of covariance matrices with a learning rate of $1/(-2 \times -0.25) = 2$ and the update should be a reasonable starting point. Since the basis has been arranged to be an orthonormal subspace of the vectorized space of covariance matrices, we can just go in the direction of the gradient of the coefficients with learning rate of 2. The gradient w.r.t. the coefficients is:

$$\frac{\partial F}{\partial \boldsymbol{l}_j} = 0.5 \boldsymbol{M} vec'(\boldsymbol{P}'^{-1}_j - \boldsymbol{\Sigma}'_j) \quad (15)$$

where the basis matrix $\boldsymbol{M}$ is as defined in Equation 12. It follows that our initial proposed step $\boldsymbol{d}_j$ for the coefficient vector $\boldsymbol{l}_j$ on each iteration will equal

$$\boldsymbol{d}_j = \boldsymbol{M} vec'(\boldsymbol{P}'^{-1}_j - \boldsymbol{\Sigma}'_j). \quad (16)$$

Projecting back with the basis matrix $\boldsymbol{M}$ and converting to a matrix, this will equal a step $\boldsymbol{\Delta}_k$ in the basis matrix $\boldsymbol{S}_k$ equal to:

$$\boldsymbol{\Delta}_k = mat'(\boldsymbol{M}^T \boldsymbol{M} vec'(\boldsymbol{P}'^{-1}_j - \boldsymbol{\Sigma}'_j)). \quad (17)$$

However, this step size may not be optimal even with the quadratic assumption because $\boldsymbol{P}' \neq \boldsymbol{I}$. Instead we decide to add some constant k times the proposed step. The quadratic approximation to the change in auxiliary function can be computed as a function of k as:

$$0.5k\, tr(\boldsymbol{\Delta}_j(\boldsymbol{P}'^{-1}_j - \boldsymbol{\Sigma}'_j)) - 0.25k^2 tr(\boldsymbol{\Delta}_j \boldsymbol{P}'^{-1}_j \boldsymbol{\Delta}_j \boldsymbol{P}'^{-1}_j). \quad (18)$$

The optimal value of k (according to the quadratic approximation) will thus be:

$$k = tr(\boldsymbol{\Delta}_j(\boldsymbol{P}'^{-1}_j - \boldsymbol{\Sigma}'_j))/tr(\boldsymbol{\Delta}_j \boldsymbol{P}'^{-1}_j \boldsymbol{\Delta}_j \boldsymbol{P}'^{-1}_j). \quad (19)$$

Due to the quadratic approximation there is still a possibility with this update rule that we can overshoot, and either fail to improve the auxiliary function or enter the region where $\boldsymbol{P}'_j$ is not positive definite. Therefore after each update we compute the eigenvalues of $\boldsymbol{P}'_j$ to make sure that they are all positive, and compute the auxiliary function (Equation 13). If it has not increased, we repeatedly halve k until it increases. However, close to convergence this time-consuming check can be eliminated if

$$0.25k^2 tr(\boldsymbol{\Delta}_j \boldsymbol{P}'^{-1}_j \boldsymbol{\Delta}_j \boldsymbol{P}'^{-1}_j) < 0.12, \quad (20)$$

i.e. the quadratic term in k in the auxiliary function is less than 0.12. The justification is beyond the scope of this paper but is based on reducing the auxiliary function to a form $\alpha + \beta k + 0.5 \sum_{d=1}^{D} log(1 + k\gamma_d)$ for $\beta > 0$ and taking the worst-case scenario which occurs when all but one of the γ_d are zero and the nonzero γ_d is negative (it also relies on the fact that k has been computed as the optimal value according to a quadratic approximation to the auxiliary function).

The update of the coefficients must be continued for typically tens of iterations for good convergence; we continue until the change in auxiliary function per iteration is small.

	Speaker adaptation	
	fMLLR	fMLLR+MLLR
Baseline ML	15.2%	14.6%
fMPE+MPE	13.9%	13.4%
fMPE+rebuild	14.4%	13.8%

Table 1: Baseline system performance: English, TC-STAR setup

5. Full covariance setup

Our full covariance estimation incorporates smoothing as introduced in [9] and as used in previous full covariance systems at IBM, e.g. [10]. This consists of scaling the off-diagonal elements of the covariance by a scale $\frac{c}{\tau+c}$ where c is the count of the data assigned to the Gaussian and τ is a smoothing constant (100 in this case).

5.1. Full covariance fMPE

We also report full covariance experiments with fMPE. This involves some fairly straightforward matrix calculus where we compute the direct and indirect gradients [6] with respect to the data. Most of the equations are exactly analogous to the diagonal case, an exception being that we have to take into account the scaling of the off-diagonal described above. This turns out to involve an analogous scaling on a matrix representing a gradient w.r.t. full-covariance statistics.

6. Experimental setup

We report experiments on data from the English portion of the European TC-STAR project [11], which consists of European parliamentary speeches in (accented) English. After segmentation and silence removal the training data is 80 hours long. We test on the 2006 English development data, which is 3 hours long. The baseline system has 6000 cross-word context-dependent states with ± 2 phones of context and 150000 Gaussians. The basic features are PLP+LDA+MLLT. Speaker adaptation includes cepstral mean and variance normalization, VTLN, fMLLR and MLLR. The models are trained on VTLN-warped and fMLLR-transformed data. In addition we train fMPE [6, 7] and MPE [8]. All results in this paper are given without language model rescoring. The baseline results are in Table 1. This last number serves as the baseline for our experiments; all systems are built from scratch on top of fMPE features.

We also report some 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 system is as for TC-STAR, but with 100000 Gaussians, and we are not rebuilding any systems on top of fMPE features (any fMPE training is done in the normal way, from an existing trained system).

7. Experimental results

7.1. Smoothing in full covariance systems

The results on Mandarin data in Table 2 are presented mainly to show the importance of smoothing the off-diagonal in a full covariance system: changing τ from 0 to 100 gives us 0.4% im-

	# Gauss	Speaker adaptation fMLLR+MLLR
Baseline ML	100k	17.5%
fMPE+MPE	100k	16.8%
Fullcov, $\tau = 100$	50k	16.7%
Fullcov, $\tau = 0$	50k	17.1%
Fullcov, $\tau = 100$ +fMPE	50k	15.5%

Table 2: Diagonal vs. full covariance: Mandarin RT'04 setup

	System type			
	Diagonal	Fullcov	SPAM	
#Gauss		$\tau = 100$	$D = 80$	$D = 160$
300k	14.5%			
250k	14.6%			
200k	14.8%	14.2%	14.0%	14.0%
150k	15.4%	14.1%	14.1%	14.3%
125k	15.0%	13.8%	14.0%	14.0%
100k	15.4%	14.0%	14.1%	14.0%
75k	15.7%	13.9%	14.3%	14.0%
50k	17.0%	14.1%	14.4%	14.0%
40k	16.3%	14.1%	14.5%	14.3%
30k	17.0%	14.2%	14.8%	14.4%
20k	17.8%	14.4%	15.1%	14.8%

Table 3: Diagonal, Fullcov vs. SPAM, TC-STAR setup. fMLLR adaptation only.

provement. It also demonstrates that fMPE can work with full covariance Gaussians, with more than 1% absolute improvement from our best diagonal fMPE+MPE result. We do not yet have results with MPE but it has previously been shown, at least in the absence of MPE, full covariance Gaussians can be trained with MPE [9]. Further experiments with full covariance use smoothing with $\tau = 100$. This smoothing does not appear to help with SPAM.

7.2. Full covariance vs. SPAM

Table 3 compares diagonal vs. full-covariance vs. SPAM systems on TC-STAR data with varying numbers of Gaussians. Systems were built from scratch based on fixed state alignments, with sizes 300k and 150k. The experiments down to 150k Gaussians inclusive are based on merging Gaussians in a maximum likelihood fashion in steps from the 300k system, with one pass of re-estimation between each step (with the number of Gaussians per state based on a power rule count$^{0.2}$) Below 150k, the experiments are based on merging Gaussians in the same way starting from the 150k system. Full-covariance and SPAM systems are trained in two E-M steps in each case, starting from the same-sized diagonal system. In SPAM training, the basis is trained on each iteration based on stored full covariance statistics.

The best absolute results are acheived with full covariance (13.8%), followed closely by SPAM (14.0%), with the best diagonal system at 14.5%.

Table 4 shows the effect of optimizing the SPAM basis, versus leaving it at the initial PCA-estimated state. It appears to help somewhat when the number of Gaussians is small. Note that Figure 2 in [4] shows that basis optimization also helps more when the dimension D is smaller.

	SPAM, $D = 80$	
#Gauss	Optimized	Not-optimized
125k	14.0%	14.1%
100k	14.1%	14.1%
75k	14.3%	14.3%
50k	14.4%	14.5%
40k	14.5%	14.7%
30k	14.8%	14.9%
20k	15.1%	15.4%

Table 4: Effect of SPAM basis optimization, TC-STAR setup. fMLLR adaptation only.

8. Conclusions

In this paper we have for the first time presented complete and explicit formulas for reasonably efficient SPAM basis and coefficients optimization. We have also presented experiments on a large vocabulary task which show that SPAM models give better absolute results than diagonal models and nearly as good as smoothed full covariance models.

9. References

[1] M.J.F. Gales, "Semi-tied covariance matrices for hidden Markov models," IEEE Transactions on Speech and Audio Processing, vol. 7, no. 3, pp. 272–281, 1999.

[2] 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.

[3] 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.

[4] S. Axelrod, V. Goel, R. A. Gopinath, P. A. Olsen & K. Visweswariah. "Subspace Constrained Gaussian Mixture Models for Speech Recognition." Submitted to IEEE Trans. Speech & Audio Processing, September 2003.

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

[6] D. Povey, B. Kingsbury, L. Mangu, G. Saon, H. Soltau, G. Zweig, "fMPE: Discriminatively trained features for speech recognition," *ICASSP*, 2005.

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

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

[9] D. Povey, "Discriminative Training for Large Vocabulary. Speech Recognition." PhD thesis, Cambridge University,. 2003.

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

[11] B. Ramabhadran, O. Siohan, L. Mangu, G. Zweig, M. Westphal. H. Schulz & A. Soneiro, "The IBM 2006 Speech Transcription System for European Parliamentary Speeches," Submitted to *Interspeech*, 2006.