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 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 are shared among groups of Gaussians; together with diagonal covariance matrices per Gaussian; and EMMLT [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 Gaussians: 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 $j$, $P_j = \sum_{k=1}^{D} \lambda_k^j S_k$.

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.

3.1. Initial approximation

The first step in optimizing the SPAM auxiliary function is to get a good initial approximation for the basis matrices $S_k$. 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 $j$:

$$F = \sum_{j=1}^{J} -0.5c_j (\text{tr}(P_j \Sigma_j)) + 0.5 \log \det(P_j)$$

This function has its maximum when for each $j$, $P_j = \Sigma_j^{-1}$; the second gradient arises from the log determinant term $0.5 \log \det(P_j)$. If we change $P_j$ by a small amount $\Delta_j$, the auxiliary function will change by $-0.25c_j tr(\Delta_j \Sigma_j \Delta_j)$. If $\Sigma_j$ happened to be a multiple of the unit matrix $f_j I$, this function would equal $-0.25c_j f_j^2 vec(\Delta_j)^T vec(\Delta_j)$ 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 $\Sigma_{avg} = \frac{\sum_{j=1}^{J} c_j \Sigma_j}{\sum_{j=1}^{J} c_j}$, we compute a symmetric normalizing matrix $N = \Sigma_{avg}^{-1/2}$. Then for all the variances we set $\Sigma_j' = N \Sigma_j N$. We do all computations with the normalized variances $\Sigma_j'$ and then at the end after computing the normalized basis matrices $S_k'$ and the coefficients $\lambda_k^j$ we can do the reverse normalization $S_k = N^{-1} S_k' N^{-1}$. To reduce the computation we vectorize the matrices in a special way taking advantage of the fact that they are symmetric. Let $vec(A)$ be a splicing together of the lower triangle of $A$ where all the off-diagonal elements are first scaled by $\sqrt{2}$; it returns a vector of size $d \times (d+1)/2$ for a $d \times d$ 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. Principle components analysis

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

\[
X = \sum_{j=1}^{J} c_j f_j^2 \text{vec}(\Sigma_j) \text{vec}(\Sigma_j)^T 
\]

(5)

where \(f_j = \frac{te(\Sigma_j)}{\sqrt{\text{det}(\Sigma_j)}}\). The \(k'th\) basis matrix \(S_k\) will now equal \(\text{mat}(v_j)\), if \(v_j\) is the \(k'th\) eigenvector of \(X\). Note that the basis elements \(S_k\) 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 \(\Sigma_j\):

\[
X' = X + 1000 \text{vec}(I) \text{vec}(I)^T. 
\]

(6)

We use the principal components of \(X'\).

3.2. Iterative optimization

Optimization of the SPAM basis is done 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 \(S_k\), given fixed coefficients \(\lambda_k\), 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 precision \(P_j\) 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, first we calculate the gradient of the auxiliary function \(F\) (Equation 2) w.r.t. each matrix \(S_k\):

\[
\frac{\partial F}{\partial S_k} = 0.5 \sum_{j=1}^{J} c_j (P_j^{-1} - \Sigma_j'). 
\]

(7)

In a Taylor expansion of the auxiliary function, the quadratic term arises from expressions of the form \(-0.25 c_j \text{tr}(P_j^{-1} \Delta_j P_j^{-1} \Delta_j)\), if \(\Delta_j\) is the change in the precision \(P_j\). If the changes in the basis matrices \(S_k\) are \(D_k\), the quadratic term in the expansion can be expressed as a function of the matrices \(D_k\) as:

\[
-0.25 \sum_{j=1}^{J} \sum_{k=1}^{D} \sum_{l=1}^{D} c_j \lambda_k^l \text{tr}(P_j^{-1} D_k P_j^{-1} D_l). 
\]

(8)

This introduces dependencies between all matrix elements of all \(S_j\), 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 \(P_j^{-1}\) as \(f_j I\), where \(f_j = \frac{\text{tr}(P_j^{-1} I)}{\text{tr}(P_j^{-1})}\).

We can also assume that since the SPAM basis was initialized with PCA, the coefficients \(\lambda_j^l\) should be fairly uncorrelated between different dimensions \(k\); assuming that all the variances are all about equal, any cross terms \((k \neq l)\) in Equation 8 will be about zero. The simplified quadratic term is now:

\[
-0.25 \sum_{k=1}^{D} c_j \lambda_k^2 f_j^2 \text{tr}(D_k D_k). 
\]

(9)

This is just a constant times a euclidean distance in the vectorized form of each matrix \(S_k'\), 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 \(F_k = \sum_{j=1}^{J} 0.5 c_j \lambda_j^2 f_j^2\) (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 \(S_k\) become

\[
D_k = \sum_{j=1}^{J} c_j (P_j^{-1} - \Sigma_j'). 
\]

(10)

However, this update amount may not converge because we made some assumptions to get this rule. Instead the change will be \(cD_k\) 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 \(c \sum_{k=1}^{K} D_k \frac{\partial F}{\partial S_k}\), with \(\frac{\partial F}{\partial S_k}\) given in Equation 7. The second order term is \(-c^2 \sum_{j=1}^{J} 0.25 c_j \text{tr}(P_j^{-1} \Delta_j P_j^{-1} \Delta_j)\), where \(\Delta_j = \sum_{k=1}^{K} \lambda_k^j D_k\). The optimal value given the full quadratic approximation to the auxiliary function is:

\[
c = \left( \sum_{j=1}^{J} 0.25 c_j \text{tr}(P_j^{-1} \Delta_j P_j^{-1} \Delta_j) \right)^{-1} \sum_{k=1}^{K} D_k \frac{\partial F}{\partial S_k}. 
\]

(11)

We can now update the basis by setting \(S_k' := S_k + cD_k\) and re-orthogonalize and normalize it by setting, for \(k = 1 \ldots D\), \(S_k := \text{norm}(S_k' - \sum_{l=1}^{D} S_l \text{tr}(S_l S_k'))\), where \(\text{norm}(A) = A / \sqrt{\text{tr}(AA)}\), 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 \(\lambda_k^j\). For efficiency we start the optimization from the previously optimized values \(\lambda_k^j\) for any Gaussian \(j\) for which the old \(\lambda_k^j\) 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 \(\lambda_k^j\) 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 \(\lambda_j^s\) for some \(j\) be \(I_j\). This first step relies on the unit, orthogonal nature of the basis. Let \(M\) be a \(k \times d(d+1)/2\) matrix where each row \(m_k = \text{vec}(S_k')\).

\[
M := \text{vec}(S_k'). 
\]

(12)

The initial estimate of the coefficient vector is \(I_j := M \text{vec}(\Sigma_j^{-1})\). If with these coefficients, \(P_j'\) 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 instead. If the first basis matrix \(S_1\) 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 \(I_j\). If \(S_1\) is not positive definite (and this has not been observed in practice but it is a theoretical possibility) we can find some other set of coefficients \(I_j\) from some other Gaussian \(j\), as optimized on the previous iteration, that gives a positive definite matrix with the current basis; and set \(I_j\) 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

\[ F(I_j) = 0.5 \log \det(P_j') - 0.5 \text{tr}(P_j' \Sigma_j') \]  
(13)

for symmetric \( P_j' \) and \( \Sigma_j' \), the second order term in a quadratic approximation to the auxiliary function around a current value \( Q_j' \) (so \( P_j' = Q_j' + \Delta_j \)) would be:

\[ -0.25 \text{tr}(\Delta_j \Sigma_j'^{-1} \Delta_j Q_j'^{-1}). \]  
(14)

Since we have projected the feature space so that most of the variances are close to unit, the variance \( Q_j'^{-1} \) will be similar to the unit matrix. This makes the second order term approximately equal to \(-0.25 \text{tr}(\Delta_j \Sigma_j')\) which equals \(-0.25 \vec{v}(\Delta_j)^T \vec{v}(\Delta_j)\). 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:

\[ \frac{\partial F}{\partial I_j} = 0.5 M \vec{v}(P_j'^{-1} - \Sigma_j') \]  
(15)

where the basis matrix \( M \) is as defined in Equation 12. It follows that our initial proposed step \( d_j \) for the coefficient vector \( I_j \) on each iteration will equal

\[ d_j = M \vec{v}(P_j'^{-1} - \Sigma_j'). \]  
(16)

Projecting back with the basis matrix \( M \) and converting to a matrix, this will equal a step \( \Delta_k \) in the basis matrix \( S_k \) equal to:

\[ \Delta_k = \text{mat}'(M^T M \vec{v}(P_j'^{-1} - \Sigma_j')). \]  
(17)

However, this step size may not be optimal even with the quadratic assumption because \( P_j' \neq I \). 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:

\[ 0.5k \text{tr}(\Delta_j (P_j'^{-1} - \Sigma_j')) - 0.25k^2 \text{tr}(\Delta_j P_j'^{-1} \Delta_j P_j'^{-1}). \]  
(18)

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

\[ k = \text{tr}(\Delta_j (P_j'^{-1} - \Sigma_j'))/\text{tr}(\Delta_j P_j'^{-1} \Delta_j P_j'^{-1}). \]  
(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 \( P_j' \) is not positive definite. Therefore after each update we compute the eigenvalues of \( P_j' \) 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

\[ 0.25k^2 \text{tr}(\Delta_j P_j'^{-1} \Delta_j P_j'^{-1}) < 0.12, \]  
(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 \( \alpha + 3k + 0.5 \sum_{d=1}^{D_{\text{diag}}} \log(1 + k_\gamma d) \) for \( \beta > 0 \) and taking the worst-case scenario which occurs when all but one of the \( \gamma_d \) are zero and the nonzero \( \gamma_d \) 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.

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 \( \tau \). Thus means that we can do simple gradient descent 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 with 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 \( \tau \) from 0 to 100 gives us 0.4% im-

<table>
<thead>
<tr>
<th>Speaker adaptation</th>
<th>fMLLR</th>
<th>fMLLR+MPE</th>
<th>fMPE+rebuild</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline ML</td>
<td>15.2%</td>
<td>14.6%</td>
<td>13.9%</td>
</tr>
<tr>
<td>fMPE+MPE</td>
<td>13.9%</td>
<td>13.4%</td>
<td>14.8%</td>
</tr>
<tr>
<td>fMPE+rebuild</td>
<td>14.4%</td>
<td>13.8%</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Baseline system performance: English, TC-STAR setup
| #Gauss | Baseline ML | fMPE+MPE | Fullcov, \(\tau = 100\) | Fullcov, \(\tau = 0\) | Fullcov, \(\tau = 100 + \text{fMPE}\) |
|--------|-------------|----------|-----------------|-----------------|-----------------
| 100k   | 17.5%       | 16.8%    | 17.1%           | 15.5%           |
| 50k    | 16.7%       | 14.7%    | 17.6%           | 15.0%           |

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

<table>
<thead>
<tr>
<th>#Gauss</th>
<th>Diagonal</th>
<th>Fullcov, (\tau = 100)</th>
<th>SPAM, (D = 80)</th>
<th>SPAM, (D = 160)</th>
</tr>
</thead>
<tbody>
<tr>
<td>300k</td>
<td>14.5%</td>
<td>14.0%</td>
<td>14.8%</td>
<td>14.6%</td>
</tr>
<tr>
<td>250k</td>
<td>14.6%</td>
<td>14.1%</td>
<td>14.4%</td>
<td>14.0%</td>
</tr>
<tr>
<td>200k</td>
<td>14.8%</td>
<td>13.9%</td>
<td>14.3%</td>
<td>14.2%</td>
</tr>
<tr>
<td>150k</td>
<td>15.4%</td>
<td>14.4%</td>
<td>14.8%</td>
<td>14.0%</td>
</tr>
<tr>
<td>125k</td>
<td>15.0%</td>
<td>14.1%</td>
<td>14.3%</td>
<td>14.1%</td>
</tr>
<tr>
<td>100k</td>
<td>15.4%</td>
<td>14.0%</td>
<td>14.3%</td>
<td>14.0%</td>
</tr>
<tr>
<td>75k</td>
<td>15.7%</td>
<td>13.9%</td>
<td>14.4%</td>
<td>14.0%</td>
</tr>
<tr>
<td>50k</td>
<td>17.0%</td>
<td>14.1%</td>
<td>14.3%</td>
<td>14.0%</td>
</tr>
<tr>
<td>40k</td>
<td>16.3%</td>
<td>14.1%</td>
<td>14.5%</td>
<td>14.3%</td>
</tr>
<tr>
<td>30k</td>
<td>17.0%</td>
<td>14.2%</td>
<td>14.8%</td>
<td>14.4%</td>
</tr>
<tr>
<td>20k</td>
<td>17.8%</td>
<td>14.4%</td>
<td>15.1%</td>
<td>14.8%</td>
</tr>
</tbody>
</table>

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

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