MAXIMUM LIKELIHOOD TRAINING OF SUBSPACES FOR INVERSE COVARIANCE MODELING

K. Visweswariah, P. Olsen, R. Gopinath, S. Axelrod

IBM T. J. Watson Research Center, Yorktown Heights, NY - 10598
{kv1, pederao, rameshg, axelrod}@us.ibm.com

ABSTRACT

Speech recognition systems typically use mixtures of diagonal Gaussians to model the acoustics. Using Gaussians with a more general covariance structure can give improved performance; EMLLT [1] and SPAM [2] models give improvements by restricting the inverse covariance to a linear/affine subspace spanned by rank one and full rank matrices respectively. In this paper we consider training these subspaces to maximize likelihood. For EMLLT ML training the subspace results in significant gains over the scheme proposed in [1]. For SPAM ML training of the subspace slightly improves performance over the method reported in [2]. For the same subspace size an EMLLT model is more efficient computationally than a SPAM model, while the SPAM model is more accurate. This paper proposes a hybrid method of structuring the inverse covariances that both has good accuracy and is computationally efficient.

1. INTRODUCTION

State of the art speech recognition systems use Hidden Markov Models with context dependent states and Gaussian Mixture Models for each state. The model for state s is

$$P(x|s) = \sum_{g \in \mathcal{G}(s)} \pi_g N(\mu_g, \Sigma_g),$$

where $x \in \mathbb{R}^d$ and $\mathcal{G}(s)$ is the set of Gaussians modeling state s. The covariances Σ_g are typically assumed diagonal for reasons of efficient Gaussian evaluation, compact storage and robust parameter estimation. Maximum Likelihood Linear Transform (MLLT) [3] or Semi-tied Covariances [4] models improve the performance while maintaining the computational advantages of diagonal covariance modeling. The recently proposed EMLLT and SPAM models restrict the inverse covariances $P_g = \Sigma_g^{-1}$ (precisions) to an affine subspace shared by all Gaussians.

In the EMLLT model [5, 1] the precision matrix is spanned by rank one matrices:

$$P_g = \sum_{k=1}^{D} \lambda_{gk} a_k a_k^T = A \Lambda_g A^T,$$

where A is a $d \times D$ matrix and Λ_j is diagonal of size $D \times D$. In [1] the basis $\{a_k a_k^T\}$ is obtained by training MLLT matrices [3, 4] for various classes of phonemes and then stacking them on top of each other. Note that for the model to be well defined we need P_g to be symmetric positive definite for all Gaussians. In order to ensure that $P_g > 0$ we need at least d linearly independent basis vectors $a_k a_k^T$, i.e., $D \geq d$. Furthermore λ_{gk}'s have to be chosen such that $P_g > 0$. Note that imposing $\lambda_{gk} > 0$ trivially ensures $P_g > 0$. However, allowing λ_{gk} to be negative can provide improved performance [1]. In this paper we impose no artificial constraints on λ_{gj}. Given the basis $\{a_k a_k^T\}$, λ_{gj} can be estimated as in [1, 2].

The SPAM model [2] generalizes the EMLLT model by restricting the means and the precision matrices of the Gaussians to a general affine subspace. In this paper we only consider the case where the precision is constrained to be in a affine subspace of the space of all symmetric matrices and the means are unconstrained. Thus the precision of a Gaussian g is of the form

$$P_g = S_0 + \sum_{j=1}^{D} \lambda_{gj} S_j.$$

In [2]the basis was obtained by optimizing a quadratic approximation to the Q function. Notice that EMLLT is a special case of the SPAM model with $S_0 = 0$ and $S_k = a_k a_k^T$.

For a fixed subspace dimension D a SPAM model provide improved performance over an EMLLT model. However, the likelihood computation for the SPAM model is more expensive for the following reason. In both cases the total computation cost can be divided into an up front cost shared by all Gaussians and a per Gaussian cost. The per Gaussian computational cost is the same for both models. The up front computation is the evaluation of $(x^T a_k)^2$ ($O(d)$ operations) for EMLLT as compared to $x^T S_k x$ ($O(d^2)$) for SPAM. If all the Gaussians are evaluated then this additional cost is negligible. However, in most practical systems since only a fraction of the total number of Gaussians are evaluation on each feature vector x, this cost may become significant. To address this issue we propose a *hybrid EMLLT/SPAM* model. The basic idea is to restrict the S_i's themselves to be in a space spanned by K rank one matrices $K > D$:

$$S_i = \sum_{k=1}^{K} u_{ki} a_k a_k^T \tag{1}$$

The contribution of this paper is a set of algorithms for the maximum likelihood training of subspaces for inverse covariance modeling; specifically for EMLLT, SPAM and hybrid EMLLT/SPAM models. To estimate the basis parameters in these models one can proceed as in standard Gaussian Mixture Model estimation via the EM algorithm. The key difference is that in the M-step, the Q function maximization does not have a simple closed form solution. Furthermore, a direct application of a general purpose numerical optimization package is not feasible because the precision matrices have to be positive definite. The positive definiteness constraint for the SPAM model can be handled as in [2]. For the EMLLT model imposing this constraint is more involved because the basis is quadratic in the parameters A. Solving this problem not only allows for the ML training of EMLLT models but also allows for the ML training of hybrid SPAM/EMLLT models defined in (1). The outline of the rest of the paper is as follows. In Section 2 we outline numerical optimization techniques that we use and describe the function we need to optimize. In Sections 3, 4 and 5 we describe our solutions to the problem of ML basis estimation for EMLLT, SPAM and hybrid EMLLT/SPAM models respectively. We present our experimental results in Section 6.

2. NUMERICAL OPTIMIZATION AND ML PARAMETER ESTIMATION

Consider optimization of a real valued function f on $\mathbb{R}^n$. Numerical optimization algorithms (steepest descent, conjugate gradients, BFGS and limited memory BFGS) algorithms typically work as follows: starting at a point x one finds a search direction v and then optimizes $g(t) = f(x + tv)$ over $t > 0$. To find the search direction v the algorithms need evaluation of the gradient of f. In each iteration we need to calculate the function and the gradient at the current point; and then evaluate the function (and possibly its derivative) a few times along v. The efficiency of using generic optimization techniques depends not only on computing the function and it's gradient efficiently but also on being able to evaluate $f(x + tv)$ and its derivative for different t given a fixed x and v. All experiments reported in this paper use an open source optimization package adapted to our needs [6]. The optimization algorithm used is a version of the limited memory BFGS algorithm [7] with the More-Thuente [8] line search algorithm. The problems we deal with here are constrained by the fact that the precision matrices have to be positive definite. As in [2] we ensure that we satisfy the constraints by finding, for a given x and v, a maximum step size B and restricting the line-search algorithm to the interval $0 < t < B$. For all our functions, as a by-product of finding B, we can also quickly calculate the function and its derivative along a line. In each function optimization a special implementation of $f(x + tv)$ and its derivative is provided.

We now introduce the optimization problem that we solve in this paper. Let $\Theta = (S_0, \{S_k\}, \{\lambda_g\}, \{\mu_g\}, \{\pi_g\})$ be the set of parameters in our Gaussian Mixture Model. Given labeled training data (x_t, s_t) our goal is to train *all* parameters to maximize the likelihood of the training data. We use the EM algorithm as direct optimization of the likelihood function would be prohibitively expensive. Given a current set of parameters $\hat{\Theta}$ the E-step of the EM algorithm gives the Q function that we need to optimize over Θ:

$$Q(\Theta, \hat{\Theta}) = \sum_s N(s) \sum_{g \in \mathcal{G}(s)} \tilde{\pi}_g G(P_g, \Sigma_g) + \tilde{\pi}_g \log \pi_g. \quad (2)$$

In the above equation

$$G(P, \Sigma) = \log(\det P) - \operatorname{trace}(\Sigma P),$$

$$\tilde{\pi}_g = \frac{1}{N(s(g))} \sum_{t: s_t = s(g)} \gamma_t(g),$$

$$\gamma_t(g) = \gamma_t(g, \hat{\Theta}) = \frac{\hat{\pi}_g p(x_t | g, s(g), \hat{\Theta})}{p(x_t | s(g), \hat{\Theta})},$$

$$\Sigma_g = \tilde{\Sigma}_g + (\mu_g - \tilde{\mu}_g)(\mu_g - \tilde{\mu}_g)^T,$$

$$\tilde{\mu}_g = \frac{1}{n(g)} \sum_{t: s_t = s(g)} \gamma_t(g) x_t,$$

and

$$\tilde{\Sigma}_g = \frac{1}{n(g)} \sum_{t: s_t = s(g)} \gamma_t(g)(x_t - \tilde{\mu}_g)(x_t - \tilde{\mu}_g)^T.$$

Here $s(g)$ is the HMM state that Gaussian g belongs to, $N(s)$ is the number of samples associated with state s and $n(g) = N(s(g))\tilde{\pi}_g$. Maximizing this function with respect to $(\{\mu_g\}, \{\pi_g\})$ and substituting optimum values back in (2) we get

$$R(S_0, \{S_k\}, \{\lambda_g\}) \triangleq \sum_g n(g) G(P_g, \tilde{\Sigma}_g).$$

For training the subspaces we start with statistics $n(g)$ and $\tilde{\Sigma}_g$ and optimize $R(S_0, \{S_k\}, \{\lambda_g\})$. Since we are hoping to approximate a full covariance model the statistics $n(g)$ and $\tilde{\Sigma}_g$ are collected using a full covariance model. Although we can optimize over S_0, in all cases reported in this paper, we either fix $S_0 = 0$ or $S_0 = (\sum_g n(g)\tilde{\Sigma}_g / N)^{-1}$. To optimize R we alternate between optimizing over $\{\lambda_g\}$ for fixed $\{S_k\}$ and vice versa. Optimizing $R(S_0, \{S_k\}, \{\lambda_g\})$ w.r.t $\{\lambda_g\}$ for fixed $\{S_k\}$ breaks up into independent problems of maximizing $G(P_g, \tilde{\Sigma}_g)$ for each Gaussian g. Although for the rank-one case we could use the coordinate ascent technique described in [1] (which has an elegant closed form solution), we choose to use a technique described in [2]. We also note that in performing the line-search in all our optimizations we have a precision matrix $P(t)$ that is a continuous function of t and $P(0)$ that is guaranteed to be positive definite. Then $P(t)$ is positive definite for all $0 <= t < B$ where B is the smallest positive solution of $\det(P(t)) = 0$. In Sections 3, 4, and 5 we discuss optimizing $R(S_0, \{S_k\}, \{\lambda_g\})$ for different parameterizations of the $\{S_k\}$.

3. ESTIMATION OF RANK-ONE BASES

In this section we describe optimization of $R(S_0, \{a_k a_k^T\}, \{\lambda_g\})$ with respect to A. We note at the outset that the objective function is not concave in A. We consider two different options: One where we optimize over a_j (the j^{th} column of A) fixing all other columns and cycling through the columns and another where we optimize all columns of A jointly. In general we expect the joint optimization to work better than optimizing one element at a time.

3.1. Optimizing one column

Consider optimizing $R(S_0, \{a_k a_k^T\}, \{\lambda_g\})$ w.r.t a particular column a_j. The function we need to maximize is

$$R(a_j) \cong \sum_g n(g)(\log\det(P_g^{'} + \lambda_{gj} a_j a_j^T)) + a_j^T T_j a_j,$$

where

$$T_j = \sum_g n(g) \lambda_{gj} \tilde{\Sigma}_g$$

and $P_g^{'} = S_0 + \sum_{k \neq j} \lambda_{gk} a_k a_k^T$. We use the symbol to $\cong$ to mean that the two quantities are equivalent given the parameters we are optimizing w.r.t. If $P_g^{'}$ were guaranteed to be invertible we could write

$$\det(P_g^{'} + \lambda_{gj} a_j a_j^T) = \det(P_g^{'})(1 + \lambda_{gj} a_j^T P_g^{'-1} a_j),$$

and this would give us an efficient way of restricting the line search and of evaluating the function along a line. Since we cannot assume $P_g^{'}$ is invertible things are slightly more involved. We can assume that we have a set of parameters $(S_0, \{a_k a_k^T\}_{k \neq j}, \{\lambda_g\})$ and an initial point $\hat{a}_j$ such that $\hat{P}_g$ (the current precision matrix) is positive definite for all Gaussians. We can use the Sherman-Morrison-Woodbury formula as follows:

$$\begin{aligned} \det(P_g^{'} + \lambda_{gj} a_j a_j^T) &= \det(\hat{P}_g + \lambda_{gj}(a_j a_j^T - \hat{a}_j \hat{a}_j^T)) \\ &\cong (1 - \lambda_{gj} \hat{a}_j^T \hat{P}_g^{-1} \hat{a}_j)(1 + \lambda_{gj} a_j^T \hat{P}_g^{-1} a_j) \\ &+ \lambda_{gj}^2 (a_j^T \hat{P}_g^{-1} \hat{a}_j)^2. \end{aligned} \quad (3)$$

(3) allows us to perform the optimization efficiently. Both the function $R(a_j)$ and its gradient can be evaluated in $O(d^2)$ operations per Gaussian assuming that $\hat{P}_g^{-1}$ is computed. We calculate $\hat{P}_g^{-1}$ once at the start of the optimization over A. We update P_g^{-1} after optimizing the column a_j. This is achieved efficiently using the Sherman-Morrison-Woodbury formula. Using (3) evaluating $R(a_j + tv)$ and its gradient is extremely efficient. The maximum step size that we can take from the current point $\hat{a}_j$ in a given direction v (and still ensure positive definiteness) can be determined by solving a quadratic equation in one variable for each Gaussian.

3.2. Joint optimization

To optimize $R(S_0, \{a_k a_k^T\}, \{\lambda_g\})$ as a function of A we need to be able to limit the line search to ensure positive definiteness of the precisions i.e given the current point $\hat{A}$ and a search direction B we need to find the smallest positive solution of

$$\det(\hat{P}_g + (B\Lambda_g\hat{A}^T + \hat{A}\Lambda_g B^T)t + B\Lambda_g B^T t^2) = 0. \quad (4)$$

This problem is a well studied problem [9] called the quadratic eigenvalue problem. We solve it by using the fact [9] that the solutions of (4) are the same as the eigenvalues of the following generalized eigen system of size $2d$:

$$\begin{pmatrix} -\hat{P}_g & 0 \\ 0 & I \end{pmatrix} - t \begin{pmatrix} (B\Lambda_g\hat{A}^T + \hat{A}\Lambda_g B^T) & B\Lambda_g B^T \\ I & 0 \end{pmatrix}.$$

Solving this generalized eigenvalue problem allows us to perform evaluations of the function and derivative along the search direction efficiently. The final piece we require is the gradient of $R(S_0, \{a_k a_k^T\}, \{\lambda_g\})$ w.r.t A:

$$\sum_g n(g)(\hat{P}_g^{-1} - \tilde{\Sigma}_g)A\Lambda_g.$$

3.3. Initialization

To initialize our basis we used two methods. One is to start with the EMLLT basis obtained by stacking MLLT matrices as in [1]. We can then take all λ_{gk} to be some small positive number which guarantees a positive definite starting point. The other alternative is to start from an MLLT basis or from a positive definite S_0 and to incrementally add vectors to this starting point. Starting with $S_0 \neq 0$ we get an affine subspace spanned by rank-one matrices which is slightly more general than the standard EMLLT model where $S_0 = 0$. Let us assume we have a basis (and the corresponding λ_{gj}) of size D and we want to increase the size to $D+1$. A closed form solution for $\lambda_{g(D+1)}$ in terms of a_{D+1} is given by:

$$\lambda_{g(D+1)} = \frac{1}{a_{D+1}^T \tilde{\Sigma}_g a_{D+1}} - \frac{1}{a_{D+1}^T \hat{P}_g^{-1} a_{D+1}}.$$

This solution for $\lambda_{g(D+1)}$ results in a positive definite precision for any a_{D+1}. We can substitute this solution back into R and use an unconstrained optimization package to optimize the resulting function.

4. ESTIMATION OF SPAM BASES

The problem of optimizing SPAM bases given fixed coefficients is a concave problem and is formally equivalent to the problem of solving for the coefficients λ_g given the basis. Given a search direction $\{B_k\}$, to limit the line search, we need to find the smallest positive root of

$$\det(\hat{P}_g + t\sum_k \lambda_{gk} B_k) = 0$$

for each Gaussian g. This requires the solution of a generalized real symmetric eigenvalue problem for each Gaussian. Once the eigenvalue problem is solved line-searches can be performed efficiently. To initialize the training of SPAM bases we start with the basis obtained using the technique in [2]. There an approximation to the function R is found that results in an eigenvalue problem (of size $d * (d+1)/2$) to be solved to find the basis.

5. HYBRID BASES

To optimize $R(S_0, \{S_k\}, \{\lambda_g\})$ when S_k are given as in (1), we alternate between optimizing the A, U and $\{\lambda_g\}$. U is a $K \times D$ matrix with elements u_{ki}. For fixed U and $\{\lambda_g\}$ A can be optimized by the methods described in Section 3. Optimization over U is essentially the same as the problem of optimizing a SPAM basis and is handled as described in Section 4. We initialize the basis by starting with $S_0 = (\sum_g n(g)\tilde{\Sigma}_g/N)^{-1}$ and growing the rank-one basis to the desired size K as described in Section 3.3. We then estimate the coefficients in this rank one basis of size K and initialize U to be the D principal components of the K dimensional coefficient vectors.

6. EXPERIMENTAL RESULTS

All experiments reported on in this paper were conducted on a test database collected in a car [1]. We report word error rates on a test set comprised of small vocabulary grammar based tasks (addresses, digits, command and control) and consists of 73743 words. Data for each task was collected at 3 speeds: idling, 30mph and 60mph. All acoustic models reported on here were built on $d = 52$ dimensional feature vectors. The 52 dimensional vector was obtained by projecting down from nine cepstral vectors (each thirteen dimensional) spliced together. The projection was obtained using LDA. The size $d = 52$ was chosen because the best full-covariance model performance was obtained at this size [10]. The acoustic model used separate digit phones with a total of 89 phones. All the acoustic models had a total of 10253 Gaussians distributed across 680 context dependent states using BIC based on a diagonal covariance system.

For all experiments acoustic models were built from a fixed Viterbi alignment of the training data using a baseline diagonal covariance model. To train the basis, we first collected the statistics $\{n(g), \Sigma_g\}$ using a full covariance model. Once the basis is trained we then train models in that fixed basis via the EM algorithm from the aligned data using the methods described in [2] and [5].

To just train the basis efficiently we could use statistics $\{n(g), \Sigma_g\}$ not for all Gaussians but for a smaller set that is representative of the covariances that we need to model. We experimented with using one Gaussian per state by combining the statistics for all Gaussians belonging to that state and using statistics $\{n(s), \Sigma_s\}$ where $n(s) = \sum_{g \in \mathcal{G}(s)} n(g)$ and $\Sigma_s = \sum_{g \in \mathcal{G}(s)} n(g)\Sigma_g$. We trained an EMLLT basis with $D = 2d$ using each of these sets of statistics. The two bases, trained on state level statistics and the full statistics, resulted in models with word error rates of 2.11 and 2.04 respectively. We felt this difference was enough to merit training all bases on the full statistics since this was not prohibitively slow.

For the training of the EMLLT basis we next compared the one-column-at-a-time approach described in Section 3.1 with the joint optimization method described in 3.2: the two models had WER's of 2.04 and 2.03 respectively. Both bases were trained from the same initial point and for the same number of iterations. For a fixed number of iterations the joint method gives a better value of likelihood than the one-column-at-a-time approach. In our implementation the joint method was slower than the one-column-at a-time approach. Therefore, we used the latter approach in all our other experiments.

Table 1 compares EMLLT models with ML trained bases to those obtained by stacking MLLT matrices for various phone classes. The latter method is restricted to generating basis sizes D that are multiples of the feature space size d. Preliminary experiments on the method used to generate the phone classes (manual vs data-driven) showed no significant differences in performance. We see that ML training of bases essentially reduces the number of parameters required to model the covariance by half. The last row in the table is the performance of a full covariance model. The first row is an MLLT model so both methods of training are the same.

D	Stacking	ML training
d	2.67	2.67
2d	2.35	2.04
4d	2.01	1.81
8d	1.82	1.65
15d	1.64	-
26.5d	1.58	-

Table 1. WER comparison of EMLLT bases obtained by stacking to ML trained ones for various basis sizes

The next table shows the flexibility (we can now have $D < d$) and benefits of using an affine subspace in the rank-one case. With $D = d$ the standard MLLT system in Table 1 is about 10% worse than the $D = d$ affine rank-one subspace system in Table 2. Using an affine subspace incurs no extra cost in the evaluation of Gaussians.

D = d	D = .5d	D = 0.25d
2.42	2.89	3.05

Table 2. WER of models with precisions modeled in affine rank one spaces

Table 3 compares ML training of SPAM bases to bases trained via the modified Frobenius norm as outlined in [2]. We see that the ML training method improves over the modified Frobenius method only for very small D. The last row reports results by gathering statistics with the models reported on in the last but one row and performing a further round of basis training. The eigenvalue problem to be solved for the modified Frobenius method can be solved much faster than the ML training procedure and hence it is to be preferred unless D is quite small.

D	Modified Frobenius	ML training
0.75d	2.05	1.99
0.5d	2.25	2.23
0.25d	2.70	2.50
0.25d (iter 2)	2.64	2.47

Table 3. WER comparison of SPAM bases obtained by using the modified Frobenius norm to ML trained ones

Finally Table 4 reports on a hybrid basis that we built using the technique described in Section 5. We see that this basis achieves the performance of a SPAM basis of the same size. The advantage of using a hybrid basis is one of computational cost. With $d = 52$ and $D = 39$ as in our case the cost of evaluating about 1500 Gaussians (1500×91) is comparable to the up front cost of evaluating $x^T S_k x$ (approx. 1400×39). Using a hybrid basis with $K = 208$ cuts the up front cost by a factor of 3.

D	ML trained SPAM	Hybrid (K=4d)
0.75d	1.99	2.01

Table 4. WER comparison of a SPAM basis to a hybrid SPAM/EMLLT basis

7. CONCLUSIONS

EMLLT and SPAM models allow for flexible sharing of parameters for modeling the covariance of Gaussians. These models improve over diagonal MLLT models with similar number of parameters ([2, 5]). In this paper we presented techniques for Maximum-Likelihood training of EMLLT and SPAM bases. For EMLLT models we report significant improvement in performance over EMLLT models trained using the techniques in [5]. For the case of SPAM ML training only gives gains at small basis sizes over the method presented in [2]. In this paper we also proposed a hybrid EMLLT/SPAM model that gives the performance benefits of SPAM models while significantly lowering the up front computation involved in using SPAM models.

8. REFERENCES

[1] P. Olsen, R. Gopinath, "Modeling inverse covariance matrices by basis expansion," *Proceedings of ICASSP*, 2002.

[2] S. Axelrod, R. Gopinath, P. Olsen, "Modeling with a subspace constraint on inverse covariance matrices," *Proceedings of ICSLP*, 2002.

[3] R. Gopinath, "Maximum likelihood modeling with gaussian distributions for classification," *Proceedings of ICASSP*, 1998.

[4] M. J. F. Gales, "Semi-tied covariance matrices for hidden markov models," *IEEE Transactions in Speech and Audio Processing*, 1999.

[5] P. Olsen, R. Gopinath, "Modeling inverse covariance matrices by basis expansion," *IEEE Transactions in Speech and Audio Processing*, Submitted.

[6] M. S. Gockenbach, W. W. Symes , "The Hilbert Class Library," http://www.trip.caam.rice.edu/txt/hcldoc/html/.

[7] D. C. Liu, J. Nocedal, "On the limited memory bfgs method for large scale optimization problems," *Mathematical Programming*, vol. 45, pp. 503–528, 1989.

[8] More, Thuente, "Line search algorithms with guaranteed sufficient decrease," *ACM TOMS*, vol. 20, no. 3, pp. 286–307, 1994.

[9] F. Tisseur, K. Meerbergen, "The quadratic eigenvalue problem," *Society for industrial and applied mathematics*, vol. 43, no. 2, pp. 235–286, 2001.

[10] S. Axelrod, R. Gopinath, P. Olsen, K. Visweswariah, "Dimensional reduction, covariance modeling and computational complexity in asr systems," *Submitted ICASSP*, 2003.